# Co-lexicographically Ordering Automata and Regular Languages - Part I

NICOLA COTUMACCIO, Gran Sasso Science Institute, Italy and Dalhousie University, Canada
GIOVANNA D'AGOSTINO and ALBERTO POLICRITI, University of Udine, Italy
NICOLA PREZZA, Ca' Foscari University of Venice, Italy

The states of a finite-state automaton $\mathcal{N}$ can be identified with collections of words in the prefix closure of the regular language accepted by $\mathcal{N}$. But words can be ordered, and among the many possible orders a very natural one is the co-lexicographic order. Such naturalness stems from the fact that it suggests a transfer of the order from words to the automaton's states. This suggestion is, in fact, concrete and in a number of articles automata admitting a *total* co-lexicographic (*co-lex* for brevity) ordering of states have been proposed and studied. Such class of ordered automata — *Wheeler automata* — turned out to require just a constant number of bits per transition to be represented and enable regular expression matching queries in constant time per matched character.

Unfortunately, not all automata can be totally ordered as previously outlined. In the present work, we lay out a new theory showing that all automata can always be *partially* ordered, and an intrinsic measure of their complexity can be defined and effectively determined, namely, the minimum width $p$ of one of their admissible *co-lex partial orders*–dubbed here the automaton's *co-lex width*. We first show that this new measure captures *at once* the complexity of several seemingly-unrelated hard problems on automata. Any NFA of co-lex width $p$: (i) has an equivalent powerset DFA whose size is exponential in $p$ rather than (as a classic analysis shows) in the NFA's size; (ii) can be encoded using just $\Theta(\log p)$ bits per transition; (iii) admits a linear-space data structure solving regular expression matching queries in time proportional to $p^2$ per matched character. Some consequences of this new parameterization of automata are that PSPACE-hard problems such as NFA equivalence are FPT in $p$, and quadratic lower bounds for the regular expression matching problem do not hold for sufficiently small $p$.



A preliminary version of this work appeared in the Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA) [32].

N. Prezza was funded by the European Union (ERC, REGINDEX, 101039208). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

A. Policriti was partially supported by project funded under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.4 - Call for tender No. 3138 of 16 December 2021, rectified by Decree n.3175 of 18 December 2021 of Italian Ministry of University and Research funded by the European Union – NextGenerationEU; Project code CN_00000033, Concession Decree No. 1034 of 17 June 2022 adopted by the Italian Ministry of University and Research, CUP G23C22001110007, Project title "National Biodiversity Future Center - NBFC".

Authors' addresses: N. Cotumaccio, Gran Sasso Science Institute, viale Francesco Crispi 7, 67100 L'Aquila (AQ), Italy and Dalhousie University, Faculty of Computer Science, 6050 University Avenue, Halifax, NS B3H 4R2, Canada; emails: nicola.cotumaccio@gssi.it, nicola.cotumaccio@dal.ca; G. D'Agostino and A. Policriti, University of Udine, Department of Mathematics, Computer Science and Physics, Via delle Scienze 206, 33100 Udine (UD), Italy; emails: {giovanna.dagostino, alberto.policriti}@uniud.it; N. Prezza, Ca' Foscari University of Venice, Department of Environmental Sciences, Informatics and Statistics, Via Torino, 155, 30170 Mestre, Venezia (VE), Italy; email: nicola.prezza@unive.it.



0004-5411/2023/08-ART27 $15.00
https://doi.org/10.1145/3607471

Having established that the co-lex width of an automaton is a fundamental complexity measure, we proceed by (i) determining its computational complexity and (ii) extending this notion from automata to regular languages by studying their smallest-width accepting NFAs and DFAs. In this work we focus on the deterministic case and prove that a canonical minimum-width DFA accepting a language $\mathcal{L}$–dubbed the Hasse automaton $\mathcal{H}$ of $\mathcal{L}$–can be exhibited. $\mathcal{H}$ provides, in a precise sense, the best possible way to (partially) order the states of any DFA accepting $\mathcal{L}$, as long as we want to maintain an operational link with the (co-lexicographic) order of $\mathcal{L}$'s prefixes. Finally, we explore the relationship between two conflicting objectives: minimizing the width and minimizing the number of states of a DFA. In this context, we provide an analogue of the Myhill-Nerode Theorem for co-lexicographically ordered regular languages.

CCS Concepts: • **Theory of computation → Fixed parameter tractability**; **Data compression**; **Pattern matching**; **Sorting and searching**; **Regular languages**;

Additional Key Words and Phrases: Graph compression, powerset construction, FPT, Wheeler graphs, FM-index, Burrows-Wheeler Transform, Myhill-Nerode Theorem

**ACM Reference format:**
Nicola Cotumaccio, Giovanna D'Agostino, Alberto Policriti, and Nicola Prezza. 2023. Co-lexicographically Ordering Automata and Regular Languages - Part I. *J. ACM* 70, 4, Article 27 (August 2023), 73 pages.
https://doi.org/10.1145/3607471

## 1 INTRODUCTION

Equipping the domain of a structure with some kind of order is often a fruitful move performed in both Computer Science and Mathematics. An *order* provides direct access to data or domain elements and sometimes allows to tackle problems otherwise too computationally difficult to cope with. For example, in descriptive complexity it is not known how to logically capture the class $P$ in general, while this can be done on ordered structures (see [61]). In general, the price to be paid when requiring/imposing an order, is a — sometimes significant — restriction of the class of structures to which subsequent results refer. If we do not wish to pay such a price, a *partial* order can be a natural alternative. Then, the "farther" the partial order is from a total order, the less powerful will be the applications of the established results. In other words, the "distance" from a total order of the partial order at hand becomes a *measure* of the extent to which we have been able to "tame" the class of structures under consideration.

In this article, inspired by the above — somehow foggy — considerations, we focus on the class of finite automata. Partial orders and automata have already met and attracted attention because of their relation with logical, combinatorial, and algebraic characterization of languages. In the literature (see, among many others, [22, 66, 75]) a partially-ordered NFA is an automaton where the transition relation induces a partial order on the set of its states. Here we pursue a different approach, closer to the one given in [77]. Our starting point is a work by Gagie et al. [47], presenting a simple and unified perspective on several algorithmic techniques related to *suffix sorting* (in particular, to the Burrows-Wheeler transform [23], an ubiquitous string permutation in text compression and indexing — see also [65]). The general idea is to enforce and exploit a total order among the states of a given automaton, induced by an *a priori* fixed order of its underlying alphabet which propagates through the automaton's transition relation.[1] The resulting automata, called *Wheeler automata*, admit efficient data structures for solving string matching on the automaton's paths and enable a representation of the automaton in a space proportional to that of the edges' labels — as well as enabling more advanced compression mechanisms, see [4, 71]. This is in contrast with the fact that general graphs require a logarithmic (in the graph's size) number of bits per

[1]The dependence on a fixed order of the alphabet marks the difference between this approach and the one of [77].

edge to be represented, as well as with recent results showing that in general, regular expression matching and string matching on labeled graphs can not be solved in subquadratic time, unless the strong exponential time hypothesis is false [6, 40, 41, 50, 70].

Wheeler languages — i.e., languages accepted by Wheeler automata — form an interesting class of subregular languages, where determinization becomes computationally easy (polynomial). As was to be expected, however, requiring the existence of a *total* Wheeler order over an automaton comes with a price. Not all automata enjoy the Wheeler property, and languages recognized by Wheeler automata constitute a relatively small class: a subclass of star-free languages [3].

Our results here show that, as a matter of fact, Wheeler (automata and) languages can be seen as a first level — in some sense the optimal level — of two hierarchies (deterministic and non-deterministic) encompassing all regular languages. A level of each such hierarchy is based on a *partial* order of minimum *width*[2] defined over the states of an automaton accepting the language. More precisely, in Definition 2.1 we identify a class of *partial* orders — the *co-lexicographic* ones, or *co-lex* for brevity — among the states of an automaton $\mathcal{N}$, reflecting naturally the (*total*) co-lexicographic order among the prefixes of the language $\mathcal{L}(\mathcal{N})$ accepted by $\mathcal{N}$: we require that (1) states with different incoming labels are sorted according to the underlying alphabet's order and (2) the co-lex order propagates through pairs of equally-labeled edges. The minimum *width* of a co-lex order over $\mathcal{N}$ results in a measure — dubbed width($\mathcal{N}$) — that we prove being a fundamental parameter for classifying finite automata as well as regular languages. Letting $\mathcal{L}$ be a regular language, denote by $\text{width}^N(\mathcal{L})$ the smallest integer $p$ such that there exists an NFA $\mathcal{N}$ such that $\mathcal{L} = \mathcal{L}(\mathcal{N})$ and width($\mathcal{N}$) = $p$, and by $\text{width}^D(\mathcal{L})$ the notion similarly defined using DFAs. These two non-deterministic/deterministic widths define two corresponding hierarchies of regular languages. The overall goal of our work is to show that these hierarchies shed new light on the relations existing between the sizes of DFAs and NFAs recognizing a given regular language, as well as classify languages by their propensity to be *searched*, with important applications to regular expression matching and string matching on labeled graphs.

In this article, we mostly focus on the deterministic case and study in detail the hierarchy based on $\text{width}^D(\mathcal{L})$ while, in a companion complementary article (see [31]), we complete the picture by studying the non-deterministic case, proving (among other results) that the two hierarchies are strict and do not collapse, and that in general they are exponentially separated, save for level one — the Wheeler languages — where allowing nondeterminism does not add expressive power. The present work and the companion article [31] are an extension of the SODA'21 article [32].

Consider an NFA $\mathcal{N}$ of co-lex width $p$. As far as motivations for introducing the new concept of co-lex orders are concerned, in this article we show that:

(1) the well-known explosion in the number of states occurring when computing the *powerset* DFA Pow($\mathcal{N}$) is exponential in $p$, rather than (as a classic analysis shows) in the size of $\mathcal{N}$. We prove that a similar exponential bound holds for width(Pow($\mathcal{N}$)).
(2) $\mathcal{N}$ can be encoded using just $\log(p\sigma) + O(1)$ bits per transition on DFAs, where $\sigma$ is the alphabet's size. NFAs can be encoded using additional $\log p$ bits per transition.
(3) String matching on $\mathcal{N}$'s paths and testing membership in the automaton's accepted language can be solved in $O(p^2 \log\log(p\sigma))$ time per matched character.

Result (1) provides one of the few known parameterizations of NFAs and immediately implies that hard problems such as NFA equivalence and universality are actually easy on small-width NFAs (for example Wheeler NFAs [47], for which width($\mathcal{N}$) = 1 holds). The result allows us to conclude also that the two deterministic and non-deterministic hierarchies of regular languages are

[2]The width of a partial order is the size of its largest antichain.

Table 1. Known and New Lower and Upper Bounds for Computing the Width of an Automaton (DFA/NFA) and the Deterministic/non-deterministic Widths of a Regular Language (Encoded as a DFA/NFA)

| compute \ given | $\mathcal{A}$ : DFA | $\mathcal{A}$ : NFA |
|---|---|---|
| $\text{width}(\mathcal{A})$ | $O(\min(\lvert Q\rvert^2, \lvert\delta\rvert \log \lvert Q\rvert))$ [9, 30] | NP-HARD [53, Theorem 2] |
| $p = \text{width}^D(\mathcal{L}(\mathcal{A}))$ | $\lvert\delta\rvert^{O(p)}$ [Theorem 4.27]<br>$\Theta(\lvert\delta\rvert^2)$ for $p = 1$ [8] | PSPACE-HARD [35, Theorem 10] |
| $p = \text{width}^N(\mathcal{L}(\mathcal{A}))$ | $\Theta(\lvert\delta\rvert^2)$ for $p = 1$ [8] | PSPACE-HARD [35, Theorem 10] |

In this table, $\lvert Q\rvert$ is the number of states and $\lvert\delta\rvert$ is the number of transitions of the automaton $\mathcal{A} = (Q, s, \delta, F)$. All hardness bounds follow from recent works dealing with the Wheeler case (automata of co-lex width equal to 1 and languages of deterministic and non-deterministic widths equal to 1). Computing the non-deterministic width in the case $p = 1$ is a polynomial problem if the input is a DFA because, only for this level of the two hierarchies, the deterministic and non-deterministic widths coincide [3] (they are both equal to 1). In this case, a quadratic algorithm for recognizing languages of deterministic (thus also non-deterministic) width equal to 1 has recently been proposed in [8], together with a matching conditional quadratic lower bound based on SETH.

exponentially related. Result (2) provides a new compression paradigm for labeled graphs. Result (3) breaks existing lower bounds for regular expression matching [6] and for string matching on labeled graphs. More details on these connections, themes and trends in the literature, are discussed in Section 1.1.

Having established that the co-lex width of a language/automaton is a fundamental complexity measure, we address the problem of the effectiveness of such a measure in the deterministic case: are the width of a DFA and the deterministic width of a language $\mathcal{L}$ (presented by an automaton) computable and, if so, at which cost? We observe that the width of a language is not in general equal to the width of, say, its minimum DFA, since already at level one of the deterministic hierarchy (i.e., Wheeler languages) there are languages whose minimum DFA has deterministic width larger than one [2]. This makes the language-width problem non-trivial.

Table 1 reports our complexity results, as well as hardness results that follow from recent works dealing with the Wheeler case. We prove that the width of a DFA can be computed in polynomial time. This is in contrast with a recent result showing that Wheeler NFAs are NP-complete to recognize [53], which implies that deciding whether the width of a NFA is smaller than a given value is NP-hard. We then show that, although the deterministic width of a language, $\text{width}^D(\mathcal{L})$, differs in general from the width of its (unique) minimum DFA, $\text{width}^D(\mathcal{L})$ can be computed in polynomial time for constant values of $\text{width}^D(\mathcal{L})$ given a DFA for $\mathcal{L}$, and a canonical automaton realizing this minimum width can be exhibited. Again, this is in contrast with the fact that recognizing Wheeler languages ($\text{width}^D(\mathcal{L}) = \text{width}^N(\mathcal{L}) = 1$) is a PSPACE-complete problem when the input language $\mathcal{L}$ is provided as an NFA [35]. The key observation for our result is a combinatorial property of automata that we called the *entanglement* number of a DFA, a quantity measuring the intrinsic co-lex incomparability of the automaton's states. The entanglement of the minimum DFA for a regular language turns out to exactly correspond both to the deterministic width of the language and to the width of the above mentioned canonical automaton, dubbed the *Hasse* automaton for the language. As we shall prove, these results imply that $\text{width}^D(\mathcal{L})$ can be computed from the minimum deterministic automaton recognizing $\mathcal{L}$.

A further contribution of this article is to explore the relationship between two (conflicting) objectives on deterministic automata: minimizing the co-lex width and minimizing the number of states. In this context, we provide an analogue of the Myhill-Nerode Theorem for regular languages applied to the concept of co-lex ordered automata.

### 1.1 Our Work in Context

Our proposal in this article aims at proving that, by pairing automata with co-lex orders, we can classify regular languages by their *propensity to be sorted*. Our classification represent a useful parameterization *simultaneously* for diverse automata-related measures: (1) the complexity of NFA determinization by the powerset-construction algorithm, (2) the encoding bit-complexity of automata, and (3) the complexity of operations on regular languages (e.g., membership) and on labeled graphs (e.g., pattern matching). As we discuss below, previous works focused on studying the complexity of points (1–3) *separately* and by cases, i.e., by studying notable classes of automata, regular languages, and graphs on which these problems are easy. To the best of our knowledge, ours is the only parameterization of automata/labeled graphs capturing *simultaneously* all these aspects.

*NFA determinization and existing subregular classifications.* An extensive and detailed classification of the complexity of the powerset construction algorithm on families of subregular languages is carried out in [16]. That study proves that for the most popular and studied classes of subregular languages — including (but not limited to) star-free [67], ordered [77], comet [21] and suffix/prefix/infix-closed languages — the output of the powerset construction is exponential in the size of the input NFA: for all the mentioned families, the resulting DFA may have at least $2^{n-1}$ states in the worst case, where $n$ is the number of states of the input NFA. Previously-known families with a sub-exponential upper bound include unary regular languages, with a bound of $e^{\Theta(\sqrt{n \ln n})}$ states [27], and the family of finite languages over alphabet of size $\sigma$, with a bound of $O(\sigma^{\frac{n}{\log_2 \sigma + 1}})$ states [74]. In this context, our nondeterministic hierarchy of subregular languages — classifying languages by the width $p$ of their smallest-width accepting NFA and guaranteeing an equivalent DFA of size at most $2^p(n - p + 1) - 1$ (see Theorem 3.2) — represents a more complete classification than the above-mentioned classes (since it captures all regular languages), even though our deterministic and nondeterministic hierarchies are orthogonal to these classes in some cases (see Section 4.4 for a study of the relations existing between our proposal and the class of star-free languages).

Interestingly, [16] shows that the class of ordered automata [77] — automata admitting a total states' order that must propagate through pairs of equally-labeled edges — does have a worst-case exponential-output powerset construction. Since in our work, we show that the powerset construction builds a small-size DFA on bounded-width automata, this fact shows that the small difference between simply imposing an order on the states which is consistent with the transition relation (ordered automata) and linking this property with a fixed order of the underlying alphabet (our framework), does have significant practical consequences in terms of deterministic state complexity.

As far as other parameterizations of powerset construction are involved, we are aware of only one previous attempt in the literature: the notion of *automata width* introduced in [62]. Intuitively, given an NFA $\mathcal{N}$ the width of $\mathcal{N}$ as defined in [62] is the maximum number of $\mathcal{N}$'s states one needs to keep track of simultaneously while looking for an accepting path for some input word (for the word maximizing such quantity). By its very definition, this quantity is directly linked to the output's size of powerset construction (while for our co-lex width, establishing such a connection will be more involved).

Further notable classifications of subregular languages include the star-height hierarchy [38] (capturing all regular languages) and the Straubing-Thérien hierarchy [80, 81] (capturing the star-free languages). To the best of our knowledge, these classifications do not lead to useful parameterizations for the automata/graph problems considered in this article.

*Graph compression.* Graph compression is a vast topic that has been extensively studied in the literature (see for example the survey [13]). Most solutions discussed below consider the unlabeled and undirected case; in those cases, a compressor for labeled graphs can be obtained by just storing the labels and the edges' directions separately using $\lceil \log \sigma \rceil + 1$ additional bits per edge ($\sigma$ is the alphabet's size).

Existing results studying worst-case information-theoretic lower bounds of graph encodings can be used as the reference base for the compression methods discussed below. First of all, note that the worst-case information-theoretic number of bits needed to represent a directed graph with $m$ edges and $n$ vertices is $\log \binom{n^2}{m} = m \log(n^2/m) + \Theta(m)$, that is, $\log(n^2/m) + \Theta(1)$ bits per edge. The same lower bound holds on undirected graphs up to a constant additive number of bits per edge. Other useful bounds (on automata) are studied in the recent work of Chakraborty et al. [25]. In that article, the authors present a succinct encoding for DFAs using $\log \sigma + \log n + 1.45$ bits per transition ($n$ is the number of states) and provide worst-case lower bounds as a function of the number of states: in the worst case, DFAs cannot be encoded using less than $(\sigma - 1) \log n + O(1)$ bits per state and NFAs cannot be encoded in less than $\sigma n + 1$ bits per state. The same article provides encodings matching these lower bounds up to low-order terms.

In order to compare our encodings with the state-of-the-art, we anticipate that our solutions store DFAs within $\log(p\sigma) + O(1)$ bits per transition (Corollary 5.21) and NFAs within $\log(p^2\sigma) + O(1)$ bits per transition (Corollary 5.34). Most of the parameterized graph encodings discussed in the literature (read below) provide a space bound *per vertex*. In Corollary 5.21 we show that our DFA encoding uses no more than $\sigma \log(p\sigma) + O(\sigma)$ bits per state. In Corollary 5.34 we show that our NFA encoding uses no more than $2p\sigma \log(p^2\sigma) + O(p\sigma)$ bits per state. Note that the former bound asymptotically matches Chakraborty et al.'s lower bound for DFAs (since that $p \leq n$), while the latter matches Chakraborty et al.'s lower bound for NFAs up to a logarithmic multiplicative factor.

For our purposes, it is useful to divide graph compression strategies into *general graph compressors* and *compact encodings* for particular graph classes. The former compressors work on arbitrary graphs and exploit sources of redundancy in the graph's topology in order to achieve a compact representation. Compressors falling into this category include (this list is by no means complete, see [13] for further references) $K^2$ trees on the graph's adjacency matrix [20], straight-line programs on the graph's adjacency list representation [28], and context-free graph grammars [39]. A shared feature of these compressors is that, in general, they do not provide guarantees on the number of bits per edge that will be used to encode an arbitrary graph (for example, a guarantee linked with a particular topology or graph parameter such as the ones discussed below); the compression parameter associated with the graph is simply the size of the compressed representation itself. This makes these techniques not directly comparable with our approach (if not experimentally).

Techniques exploiting particular graph topologies or structural parameters of the graph to achieve more compact encodings are closer to our parameterized approach, bearing in mind that also, in this case, a direct comparison is not always possible in the absence of known relations between our parameter $p$ and the graph parameters mentioned below (the investigation of such relations represents an interesting future research direction). A first example of such a parameter (on undirected graphs) is represented by *boxicity* [73], that is, the minimum number $b$ of dimensions such that the graph's edges correspond to the intersections of $b$-dimensional axes-parallel boxes (the case $b = 1$ corresponds to interval graphs). Any graph with boxicity $b$ can be represented naively using $O(b)$ words per vertex (that is, storing each vertex as a $b$-dimensional box), regardless of the fact that its number of edges could be quadratic in the number of vertices (even in the interval graph case). Similar results are known for graphs of small clique-width/bandwidth/treedepth/treewidth [42, 58, 59] and bounded genus [36]; any graph from these graph families can be encoded in $O(k)$ bits per vertex, where $k$ is the graph parameter under

consideration. Similarly, posets (transitively-closed DAGs) of width $w$ can be encoded succinctly using $2w + o(w)$ bits per vertex [83]. While the above-mentioned methods focus on particular graph parameters, another popular approach is to develop ad-hoc compact encodings for particular graph topologies. Separable graphs (graphs admitting a separator of size $O(n^c)$ breaking the graph into components of size $\alpha n$ for some $c < 1$ and $\alpha < 1$) allow for an encoding using $O(1)$ bits per vertex [15]. This class includes planar graphs — admitting also an encoding of 4 bits per vertex [46] — and trees — admitting an encoding of 2 bits per vertex (e.g., a simple balanced-parenthesis representation) and an encoding of $1 + o(1)$ bits per vertex when every internal node has exactly two children [57]. Circular-arc graphs (a class including interval graphs) of maximum degree $\Delta$ can be encoded in $\log \Delta + O(1)$ bits per vertex, and this bound is asymptotically tight [26]; in the same article, the authors show that circular-arc graphs with chromatic number $\chi$ admit an encoding using $\chi + o(\chi)$ bits per vertex. As mentioned above, on NFAs our proposed encoding uses a space per state that can be expressed as a function of $p$ and $\sigma$, thereby fitting with previous research on compact parameterized representations of graphs.

*Regular expression matching and string matching on labeled graphs.* "Regular expression matching" (REM) refers to the problem of determining whether there exist substrings of an input string that can be derived from an input regular expression. This problem generalizes that of determining membership of a string to a regular language, and it finds important applications which include text processing utilities (where regular expressions are used to define search patterns), computer networks (see [82]), and databases (see [33]). A closely-related problem is that of "exact **string matching on labeled graphs**" (**SMLG**): find which paths of an edge-labeled graph match (without edits) a given string (see [40]). This problem arises naturally in several fields, such as bioinformatics [7, 78], where the *pan-genome* is a labeled graph capturing the genetic variation within a species (and pattern matching queries are used to match an individual's genome on this graph), and graph databases [5]. Since NFAs can be viewed as labeled graphs, it is not surprising that existing lower and upper bounds for both problems have been derived using the same set of techniques.

In order to compare our approach with the state of the art, we anticipate that we describe labeled graph indexes solving SMLG (and thus REM by constructing the index on an NFA derived from the regular expression) in $O(p^2 \log\log(p\sigma))$ time per matched character, where $p$ is the co-lex width of the graph/automaton under consideration (Theorem 5.29). The general idea behind our approach is that co-lexicographically ordering the states of the underlying automaton accepting a language is a way of *structuring* the search-space where functionalities will eventually operate. In this sense, the co-lex width of a labeled graph (or that of a language) is a measure capturing the intrinsic complexity of the *entire collection* of strings that we aim at representing.

Backurs and Indyk in [6] carry out a detailed study of the complexity of the REM problem as a function of the expression's structure for all regular expressions of depth up to 3. For each case (there are in total 36 ways of combining the regular operators "|", Kleene plus, Kleene star, and concatenation up to depth 3), they either derive a sub-quadratic upper bound (where *quadratic* means the string's length multiplied by the regular expression's size) or a quadratic lower bound conditioned on the Strong Exponential Time Hypothesis [56] or on the Orthogonal Vectors conjecture [19]. Note that this classification does not capture regular expressions of arbitrary depth. Similarly, Equi et al. [40] establish lower and upper bounds for the SMLG problem, even in the scenario where one is allowed to pre-process the graph in polynomial time [41] (that is, building a graph index); their work represents a *complete* classification of the graph topologies admitting either sub-quadratic pattern matching algorithms or quadratic lower bounds (obtained assuming the Orthogonal Vectors conjecture); in this context, *quadratic* means proportional to the string's length times the graph's size. In all these works, as well as in further articles refining these analyses by providing finer lower bounds or better upper bounds for particular cases [12, 18, 49, 51, 52],

the problem's complexity is studied by cases and does not depend on a parameter of the language or the graph.

Techniques parameterizing the problem's complexity on a graph parameter do exist in the literature, and are closer to the spirit of our work. These parameters include (these works consider the SMLG problem) the size of the labeled direct product [72], the output size of powerset construction [69], and a generalization of DAGs called $k$-funnels [24]. Like in our setting, in all these cases quadratic query complexity is obtained in the worst case (on graphs maximizing the parameter under consideration).

### 1.2 Organization of the Article

The article is organized as follows. In Section 2 we give some preliminary definitions, we formally introduce the width-based deterministic and non-deterministic hierarchies, prove some preliminary results related to them, and state all the problems to be considered in the rest of the article. Problems are classified as *automata*, *language*, and *compression/indexing* related. In Section 3 we prove one of the strongest properties of our notion of co-lex width: this parameter yields a new upper bound to the size of the smallest DFA equivalent to a given NFA and implies new FPT analyses for hard problems on NFAs such as universality, equivalence, and membership. In the central part of the article, we discuss and propose solutions to the problems defined in Section 2. In the first paragraphs of Section 4 we give a simple polynomial solution to the DFA-width problem, thus solving Problem 1 (NFA width problem) when the input is a DFA. In the following three subsections, we introduce tools that allow us to compute the deterministic width of a language: in particular, Section 4.1 introduces the concept of *entanglement* of a DFA, a measure able to capture the language's deterministic width on the minimum DFA; Section 4.2 exhibits a canonical DFA (dubbed the *Hasse automaton* of a language) of minimum width, thus solving Problem 3 (Minimum-width DFA), and gives an automata-free characterization for each level of the deterministic hierarchy, thus solving Problem 4 (Automata-free characterization); Section 4.3 puts together the notions developed in the two previous subsections to derive an algorithm computing the deterministic width of a language, which solves the deterministic side of Problem 2 (Language width problem). In Section 4.4 we compare our notion of deterministic width with an important class of subregular languages: the star-free languages. In Section 4.5 we prove a Myhill-Nerode theorem for each level of the deterministic hierarchy, thereby providing an alternative solution to Problem 4 (Automata-free characterization). In Section 5, we consider compression and indexing problems over finite state automata. More in detail, in Section 5.1 we establish the necessary tools at the core of our data structures (in particular, the *path coherence* property); in Section 5.2 we present a space-efficient representation for automata (the *automata BWT*, or *aBWT*) preserving the accepted regular language (for any input NFA) and the automaton's topology for a strict superclass of the DFAs; in Section 5.3 we augment the aBWT to obtain an index solving subpath queries on languages and NFAs; finally, in Section 5.4 we augment the aBWT to make it faithful (i.e., preserving the automaton's topology) also on NFAs. These last contributions solve Problems 5 (Compressing automata) and 6 (Indexing automata). In Section 6 we draw our conclusions.

After the bibliography, an index gathers the main mathematical symbols and definitions used throughout the article, linking them to the location where they are defined. The article is concluded with Appendix A, where we prove general results related to partitions and orders which are needed in some of our results in Section 4.2 and could be of independent interest.

Notice that in Section 2 we present the main idea of the article: how to (partially) order the sets of states of an automaton by lifting the co-lex order on strings. Then, Sections 3–5 can be read independently from one another: the reader interested in automata theory can focus on Sections 3 and 4, and the reader interested in graph compression can focus on Section 5.

## 2 DEFINITIONS, FIRST RESULTS, AND PROBLEMS

In this section, we first present all basic definitions required in order to follow the article: sequences, finite-state automata, orders, and model of computation used in our data-structures results. After giving all the necessary definitions, in Section 2.2 we introduce the core concept of our article: the co-lex width of a finite-state automaton. Section 2.3 extends the notion of co-lex width to regular languages. These two sections also formally introduce the problems, tackled in the next sections, related to the notion of co-lex width, and prove some preliminary results. To conclude, Section 2.4 formally defines the problems of automata compression and indexing.

### 2.1 Basics

*Sequences.* Let $\Sigma$ be a finite alphabet and let $\Sigma^*$ be the set of all finite sequences (also called *words* or *strings*) on $\Sigma$, with $\varepsilon$ being the empty sequence. We write $\beta \dashv \alpha$ if $\alpha, \beta \in \Sigma^*$ and $\beta$ is a suffix of $\alpha$.

Throughout the article, we assume that there is a fixed total order $\preceq$ on $\Sigma$ (in our examples, the alphabetical order). When using an integer alphabet in Section 5, the total order will be the standard order on integers. The special symbol $\# \notin \Sigma$ is considered smaller than any element in $\Sigma$. We extend $\preceq$ to words in $\Sigma^*$ *co-lexicographically*, that is, for $\alpha, \beta \in \Sigma^*$, we have $\alpha \preceq \beta$ if and only if either $\alpha$ is a suffix of $\beta$, 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 $a \prec b$.

*Finite-state automata.* A *non-deterministic* finite automaton (an NFA) accepting strings in $\Sigma^*$ is a tuple $\mathcal{N} = (Q, s, \delta, F)$ where $Q$ is a finite set of states, $s$ is a *unique* initial state, $\delta(\cdot,\cdot) : Q \times \Sigma \to \mathcal{P}ow(Q)$ is the transition function (where $\mathcal{P}ow(Q)$ is the set of all subsets of $Q$), and $F \subseteq Q$ is the set of final states. We write $Q_{\mathcal{N}}, s_{\mathcal{N}}, \delta_{\mathcal{N}}, F_{\mathcal{N}}$ when the automaton $\mathcal{N}$ is not clear from the context and, conversely, if the automaton is clear from the context we will not explicitly say that $s$ is the initial state, $\delta$ is the transition function and $F$ is the set of final states.

With $|\delta|$ we denote the cardinality of $\delta$ when seen as a set of triples over $Q \times Q \times \Sigma$. In other words, $|\delta| = |\{(u, v, a) \mid v \in \delta(u, a),\ u, v \in Q, a \in \Sigma\}$. In fact, in our results (especially the data-structure related ones) we will often treat NFAs as edge-labeled graphs having as set of nodes $Q$ and set of edges $\{(u, v, a) \mid v \in \delta(u, a),\ u, v \in Q, a \in \Sigma\}$.

As customary, we extend $\delta$ to operate on strings as follows: for all $u \in Q, a \in \Sigma$, and $\alpha \in \Sigma^*$:

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

We denote by $\mathcal{L}(\mathcal{N}) = \{\alpha \in \Sigma^* : \delta(s, \alpha) \cap F \neq \emptyset\}$ the language accepted by the automaton $\mathcal{N}$. We say that two automata are *equivalent* if they accept the same language.

We assume, without loss of generality, that all states in our automata are *useful*, that is, from each state one can reach a final state (possibly, the state itself). We assume also that every state is reachable from the (unique) initial state. Hence, the collection of prefixes of words accepted by $\mathcal{N}$, $\mathrm{Pref}(\mathcal{L}(\mathcal{N}))$, will consist of the set of words that can be read on $\mathcal{N}$ starting from the initial state.

Following [3], we adopt a specific notation to denote the set of words reaching a given state and to denote the set of states reached by a given word:

— if $u \in Q$, let $I_u$ be the set of words *reaching* $u$ from the initial state:

$$I_u = \{\alpha \in \mathrm{Pref}(\mathcal{L}(\mathcal{N})) : u \in \delta(s, \alpha)\};$$

— if $\alpha \in \mathrm{Pref}(\mathcal{L}(\mathcal{N}))$, let $I_\alpha$ be the set $\delta(s, \alpha)$ of all states *reached* from the initial state by $\alpha$.

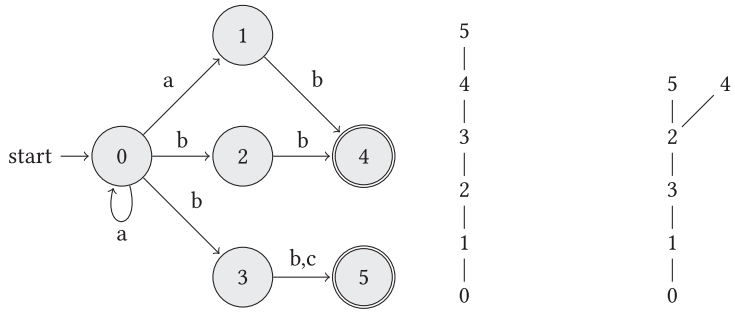



Fig. 1. An NFA and the Hasse diagrams — that is, graphs depicting the *transitive reductions* of the partial orders — of two of its (maximal) co-lex orders. Characters are sorted according to the standard alphabetical order.

A deterministic **finite automaton** (a **DFA**), is an NFA $\mathcal{D}$ where $|\delta(u, a)| \leq 1$, for any $u \in Q$ and $a \in \Sigma$. If the automaton is deterministic we write $\delta(u, \alpha) = v$ for the unique $v$ such that $\delta(u, \alpha) = \{v\}$ (if defined: we are not assuming a DFA to be complete).

Let $\mathcal{L} \subseteq \Sigma^*$ be a language. An equivalence relation $\sim$ on $\mathrm{Pref}(\mathcal{L})$ is *right-invariant* if for every $\alpha, \beta \in \mathrm{Pref}(\mathcal{L})$ such that $\alpha \sim \beta$ and for every $a \in \Sigma$ it holds $\alpha a \in \mathrm{Pref}(\mathcal{L})$ iff $\beta a \in \mathrm{Pref}(\mathcal{L})$ and, if so, $\alpha a \sim \beta a$. We will extensively use the *Myhill-Nerode equivalence* induced by $\mathcal{L}$, namely, the right-invariant equivalence relation $\equiv_{\mathcal{L}}$ on $\mathrm{Pref}(\mathcal{L})$ such that for every $\alpha, \beta \in \mathrm{Pref}(\mathcal{L})$ it holds:

$$\alpha \equiv_{\mathcal{L}} \beta \iff \{\gamma \in \Sigma^* \mid \alpha\gamma \in \mathcal{L}\} = \{\gamma \in \Sigma^* \mid \beta\gamma \in \mathcal{L}\}.$$

We denote by $\mathcal{D}_{\mathcal{L}}$ the minimum (with respect to state-cardinality) deterministic automaton recognizing a regular language $\mathcal{L}$.

If $u \in Q$, then $\lambda(u)$ denotes the set of labels of edges entering $u$, except when $u = s$ when we also add $\# \notin \Sigma$ to $\lambda(s)$, with $\# \prec e$ for all $e \in \Sigma$ (see e.g., Figure 1 where $s = 0$, $\lambda(s) = \{\#, a\}$, $\lambda(1) = \{a\}$, and $\lambda(5) = \{b, c\}$). If $u \in Q$, by $\min_{\lambda(u)}$, $\max_{\lambda(u)}$ we denote the minimum and the maximum, with respect to the order $\preceq$, among the elements in $\lambda(u)$.

*Orders.* A *(non-strict) partial order* is a pair $(Z, \leq)$, where $\leq$ is a reflexive ($u \leq u$ for all $u \in Z$), transitive ($u \leq v$ and $v \leq w$ implies $u \leq w$, for all $u, v, w \in Z$), and antisymmetric ($u \leq v$ and $v \leq u$ implies $u = v$) binary relation on $Z$.

Two elements $u, v \in Z$ are said to be $\leq$-*comparable* if either $u \leq v$ or $v \leq u$. A partial order can also be described using the relation $u < v$ which holds when $u \leq v$ and $u \neq v$. We write $u \parallel v$ if $u$ and $v$ are not $\leq$-comparable.

If $(Z, \leq)$ is a partial order and $Z' \subseteq Z$ we denote by $(Z', \leq_{Z'})$ the restriction of the partial order $\leq$ to the set $Z'$. To simplify notation, we will also use $(Z', \leq)$ when clear from the context.

A partial order $(Z, \leq)$ is a *total order* if for all $y, z \in Z$, $y$ and $z$ are $\leq$ -comparable.

A subset $Z' \subseteq Z$ is a $\leq$-*chain* if $(Z', \leq)$ is a total order. A family $\{Z_i \mid 1 \leq i \leq p\}$ is a $\leq$-*chain partition* of $Z$ if $\{Z_i \mid 1 \leq i \leq p\}$ is a partition of $Z$ and each $Z_i$ is a $\leq$-chain.

The *width* of a partial order $(Z, \leq)$, denoted by $\mathrm{width}(\leq)$, is the smallest cardinality of a $\leq$-chain partition. A subset $U \subseteq Z$ is a $\leq$-*antichain* if any two distinct elements in $U$ are not $\leq$-comparable. Dilworth's Theorem [37] states that the width of $(Z, \leq)$ is equal to the cardinality of a largest $\leq$-antichain.

If $A, B$ are *disjoint* subsets of a partial order $(Z, \leq)$, then $A < B$ denotes:

$$(\forall a \in A)(\forall b \in B)(a < b).$$

A *monotone sequence* in (a partial order) $(Z, \leq)$ is a sequence $(v_n)_{n\in\mathbb{N}}$ with $v_n \in Z$ and either $v_i \leq v_{i+1}$, for all $i \in \mathbb{N}$, or $v_i \geq v_{i+1}$, for all $i \in \mathbb{N}$.

A subset $C$ of a partial order $(V, \leq)$ is $\leq$-*convex* if for every $u, v, z \in V$ with $u, z \in C$ and $u < v < z$ we have $v \in C$. We drop $\leq$ when clear from the context.

If $\alpha \preceq \alpha' \in \Sigma^*$, we define $[\alpha, \alpha'] = \{\beta : \alpha \preceq \beta \preceq \alpha'\}$; if the relative order between $\alpha, \alpha'$ is not known, we set $[\alpha, \alpha']^{\pm} = [\alpha, \alpha']$, if $\alpha \preceq \alpha'$, while $[\alpha, \alpha']^{\pm} = [\alpha', \alpha]$, if $\alpha' \preceq \alpha$.

*Other assumptions.* Throughout the article, $[a, b]$ with $a, b \in \mathbb{N}$ and $a \leq b$ denotes the integer set $\{a, a + 1, \ldots, b\}$. If $b < a$, then $[a, b] = \emptyset$. All logarithms used in the article are in base 2.

### 2.2 The Co-lex Width of an Automaton

The notion of ordering stands at the core of the most successful compression and pre-processing algorithmic techniques: integer sorting and suffix sorting are two illuminating examples. This concept is well-understood in the case of strings (where the co-lexicographic order of the string's prefixes or the lexicographic order of the string's suffixes are typically used) and has been generalized to a special class of subregular languages (the Wheeler languages) in [47]. The goal of this section is to provide a generalization of the co-lexicographic (*co-lex* for brevity) order among prefixes of a word to an order among the states of an NFA recognizing *any* regular language, by imposing axioms on the accepting NFA. This will allow us to generalize powerful compression and indexing algorithmic techniques from strings to arbitrary regular languages and NFAs, as well as proving relations between the sizes of NFAs and DFAs recognizing a given language.

We capture co-lex orders on an NFA by means of two axioms which ensure a *local* comparability between pairs of states. Given two states $u$ and $v$ such that $u < v$, Axiom 1 of Definition 2.1 imposes that all words in $I_u$ end with letters being smaller than or equal to letters ending words in $I_v$; Axiom 2, instead, requires that the order among states $u$ and $v$ propagates backward when following pairs of equally-labeled transitions. These two axioms generalize to NFAs the familiar notion of prefix sorting: if the NFA is a simple path — i.e., a string —, this order reduces to the well-known co-lexicographic order among the string's prefixes.

*Definition 2.1.* Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA. A *co-lex order* on $\mathcal{N}$ 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 $\max_{\lambda(u)} \preceq \min_{\lambda(v)}$;
(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 $u' \leq v'$.

*Remark 2.2.*

(1) Since $\# \in \lambda(s)$ and $\# \notin \lambda(u)$ for $u \neq s$, then from Axiom 1 it follows that for every $u \in Q$ it holds $u \not< s$.
(2) If $\mathcal{D}$ is a DFA, then we can restate Axiom 2 as follows: 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).

When $\leq$ is a total order, we say that the co-lex order $\leq$ is a *Wheeler order*. Wheeler orders were first introduced in [47] in a slightly less general setting.[3] The class of Wheeler languages — that is, the class of all regular languages recognized by some Wheeler NFA — is rather small: for example, unary languages are Wheeler only if they are finite or co-finite, and all Wheeler languages

[3]More in detail: they required the set $\lambda(u)$ of labels entering in state $u$ to be a singleton for all $u \in Q$ (*input consistency* property). In this article, we drop this requirement and work with arbitrary NFAs.

are star-free (see [3]). Moreover, Wheeler languages are not closed under union, complement, concatenation, and Kleene star [3]. In contrast, as observed in the following remark, any regular automaton admits a co-lex order.

*Remark 2.3.* Every NFA $\mathcal{N}$ admits some co-lex order. For example, the order $\{(u,u) \mid u \in Q\}$ and the order $\{(u,v) \mid \max_{\lambda(u)} \prec \min_{\lambda(v)}\} \cup \{(u,u) \mid u \in Q\}$ are co-lex orders on $\mathcal{N}$.

We note that Axiom 2 implies that the order between two states is not defined whenever their predecessors cannot be unambiguously compared, as observed in the following remark.

*Remark 2.4.* Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA and let $\leq$ be a co-lex order on $\mathcal{N}$. Let $u, v \in Q$ be two distinct states. Then, $u \parallel v$ if at least one of the following holds:

(1) There exist $u', v' \in Q$ and $a \in \Sigma$ such that $u \in \delta(u', a)$, $v \in \delta(v', a)$ and $u' \parallel v'$.
(2) There exist $u', v', u'', v'' \in Q$ and $a, b \in \Sigma$ such that $u \in \delta(u', a) \cap \delta(u'', b)$, $v \in \delta(v', a) \cap \delta(v'', b)$, $u' < v'$ and $v'' < u''$.

Indeed, if e.g., it were $u < v$, then Axiom 2 would imply that in case 1 it should hold $u' \leq v'$ and in case 2 it should hold $u'' \leq v''$ (which is forbidden by the antisymmetry of $\leq$).

More than one non-trivial co-lex order can be given on the same automaton. As an example, consider Figure 1: the automaton on the left admits the two co-lex orders whose Hasse diagrams are depicted on the right. The first, $\leq_1$, is total and states that $2 <_1 3$, while the width of the second one, $\leq_2$, is equal to 2 and $3 <_2 2$ holds. As a matter of fact, in any co-lex order $\leq$ for this automaton in which $3 < 2$ holds, nodes 4 and 5 must be incomparable.

In Section 5 we will prove that a co-lex order over an automaton enables compression and indexing mechanisms whose efficiency is parameterized by the width of the co-lex order (the smaller, the better): this justifies introducing the *co-lex width* of an NFA (Definition 2.5) as a meaningful measure for compression and indexing. In fact, the co-lex width can also be used for further, interesting, language-theoretic consequences — more on this in Sections 3 and 4 for the deterministic case and in [31] for the non-deterministic case.

*Definition 2.5.* The *co-lex width* of an NFA $\mathcal{N}$ is the minimum width of a co-lex order on $\mathcal{N}$:

$$\text{width}(\mathcal{N}) = \mathit{min}\{\text{width}(\leq) \mid \leq \text{ is a co-lex order on } \mathcal{N}\}$$

In Example 2.12 below we shall see that the value $\text{width}(\mathcal{N})$ may depend on the choice of the total order $\preceq$ on $\Sigma$.

As a matter of fact, string sorting stands at the core of the most popular string compression and indexing paradigms, which for this reason also suffer from a sharp dependence on the total alphabet order. For example, the number $r$ of equal-letter runs of the **Burrows-Wheeler transform** (**BWT**) of a string [23] is an important string compressibility parameter (see [48]) and its value depends on the choice of the total order on the alphabet; deciding whether there exists an ordering of the alphabet of a string such that $r$ is bounded by a given value, however, is an NP-complete problem [11]. Despite this limitation, the BWT and the data structures based on it — such as the FM-index [45] and the r-index [48] — are widely used in applications with a fixed (often sub-optimal) alphabet order.

Similarly, in our scenario, it is natural to wonder whether it is possible to determine an ordering of the alphabet that minimizes the width of an automaton. Unfortunately, also this problem is not tractable: deciding whether there exists a total alphabet order under which a given DFA is Wheeler (that is, it has co-lex width equal to one) is already an NP-complete problem [34]. In such situations, one possible way to tame the problem's complexity is to study a more constrained version of the

problem, with the goal of shedding new light on the more general (unconstrained) scenario. For this reason, in this article, we start by fixing a total order on the alphabet and investigating the implications of this choice. In particular, we will prove that, if we fix an order on the alphabet, then the width of a DFA with respect to that order can be determined in polynomial time (see Corollary 4.5). This finding can already be used, for example, as a black-box to test candidate alphabet orderings in search for the one minimizing the automaton's width (more advanced search strategies will be the subject of forthcoming works, bearing in mind that the optimal solution is NP-hard to find).

In the following, we establish preliminary useful properties of the new measure width($\mathcal{N}$). Our first observation is that this measure is linked with the graph's *sparsity*.

Lemma 2.6. *Let $\mathcal{N}$ be an NFA on an alphabet of cardinality $\sigma$ with $n$ states and $|\delta|$ transitions, and let $p = width(\mathcal{N})$. Then:*

$$|\delta| \leq (2n - p)p\sigma.$$

Proof. Let $\mathcal{N}$ be an NFA on an alphabet of cardinality $\sigma$ with $n$ states and $|\delta|$ transitions, and let $p = \text{width}(\mathcal{N})$. Let $\leq$ be a co-lex order of width $p$, $\{Q_i \mid 1 \leq i \leq p\}$ be a $\leq$-chain partition of the set of states, and for all $1 \leq i \leq p$ let $Q_i = \{v_{i,1}, \ldots, v_{i,n_i}\}$, where $n_i = |Q_i|$ and $v_{i,k} < v_{i,k'}$ whenever $k < k'$. Fix $1 \leq i, j \leq p$ and $a \in \Sigma$, and consider all the transitions $(v_{i,k}, v_{j,l}, a)$ labeled with $a$ that leave $Q_i$ and reach $Q_j$. We denote with $e_{i,j,a}$ the number of such transitions; our goal is to establish an upper bound to this quantity for all $i, j, a$. Sort these $e_{i,j,a}$ edges $(v_{i,k}, v_{j,l}, a)$ by the index $l$ of their destination state, breaking ties by the index $k$ of their source state. Now, let us prove that the value $k + l$ is strictly increasing with respect to this order. In order words, we want to prove that if we pick two edges $(v_{i,k}, v_{j,l}, a)$ and $(v_{i,k'}, v_{j,l'}, a)$ being consecutive with respect to the edge order, then $k + l < k' + l'$. By the definition of the edge order, we have $l \leq l'$. If $l = l'$, again by the definition of the edge order we have $k < k'$ and so $k + l < k' + l'$. If $l < l'$, by Axiom 2 of co-lex orders we have $k \leq k'$, and again we conclude $k + l < k' + l'$. Since we have proved that $k + l$ is strictly increasing with respect to the edge order, then from $2 \leq k + l \leq n_i + n_j$ we obtain $e_{i,j,a} \leq n_i + n_j - 1$. Observing that $\sum_{i=1}^{p} n_i = n$, we conclude:

$$|\delta| = \sum_{a\in\Sigma}\sum_{i=1}^{p}\sum_{j=1}^{p} e_{i,j,a} \leq \sum_{a\in\Sigma}\sum_{i=1}^{p}\sum_{j=1}^{p}(n_i + n_j - 1) = 2\sigma pn - \sigma p^2 = (2n - p)p\sigma.$$

□

The above lemma will be useful later, when measuring the size of our NFA encodings as a function of the number of states. Note that Wheeler automata ($p = 1$) have a number of transitions proportional to $O(\sigma n)$. This relation was already noted in the literature [53, Theorem 4].

Next, we move on to studying some preliminary properties of the smallest-width co-lex order. We say that $\leq^*$ is a *refinement* of $\leq$ if, for all $u, v \in Q$, $u \leq v$ implies $u \leq^* v$. Since there are only finitely many co-lex orders over an automaton, every co-lex order $\leq$ is maximally refined by a co-lex order. Moreover, if $\leq^*$ is a refinement of $\leq$, then it must be that width($\leq^*$) is less than or equal to width($\leq$), since every $\leq$-chain partition is also a $\leq^*$-chain partition. This implies that there is always a *maximal* co-lex order $\leq$ on an NFA $\mathcal{N}$ such that width($\mathcal{N}$) = width($\leq$). In general, an NFA admits several maximal co-lex orders of different widths. For example, the two co-lex orders presented in Figure 1 are both maximal and have different widths. This cannot happen over DFAs: in the following lemma we prove that a DFA always admits a unique maximal co-lex order (the *maximum* co-lex order) so that this order realizes the width of the DFA. In particular, the maximum co-lex order refines every co-lex order on the DFA. This simplifies the search for a co-lex order

realizing the width of the automaton and, indeed, in Lemma 4.3 and Corollary 4.5 we prove that such a co-lex order can be determined in polynomial time.

*Definition 2.7.* Let $\mathcal{D}$ be a DFA. The relation $<_{\mathcal{D}}$ over $Q$ is defined by

$$u <_{\mathcal{D}} v \text{ if and only if } (\forall \alpha \in I_u)(\forall \beta \in I_v)\ (\alpha \prec \beta).$$

One can easily prove that $\leq_{\mathcal{D}}$ (that is, $<_{\mathcal{D}} \cup \{(u,u) \mid u \in Q\}$) is a partial order over $Q$. Moreover:

Lemma 2.8. *If $\mathcal{D}$ is a DFA then $(Q, \leq_{\mathcal{D}})$ is the maximum co-lex order on $\mathcal{D}$.*

Proof. First, let us prove that $\leq_{\mathcal{D}}$ is a co-lex order on $\mathcal{D}$.

To see that Axiom 1 holds assume that $u <_{\mathcal{D}} v$: we must prove that $e = \max_{\lambda(u)} \leq e' = \min_{\lambda(v)}$. Notice that it must be $v \neq s$ because the empty string $\varepsilon$ is in $I_s$ and $\varepsilon$ is co-lexicographically smaller than any other string. Hence, $e' > \#$ and if $e = \#$ we are done. Otherwise, there are $\alpha e \in I_u$ and $\alpha' e' \in I_v$, so that $u <_{\mathcal{D}} v$ implies $\alpha e \prec \alpha' e'$ and therefore $e \preceq e'$. As for Axiom 2, assume that $u \in \delta(u', a)$, $v \in \delta(v', a)$, and $u <_{\mathcal{D}} v$. We must prove that $u' <_{\mathcal{D}} v'$. Fixing $\alpha \in I_{u'}$ and $\beta \in I_{v'}$, we must prove that $\alpha \prec \beta$. We have $\alpha a \in I_u$ and $\beta a \in I_v$, hence from $u <_{\mathcal{D}} v$ it follows $\alpha a \prec \beta a$, and therefore $\alpha \prec \beta$.

Let us now prove that $\leq_{\mathcal{D}}$ is the maximum co-lex order.

Suppose, reasoning for contradiction, that $\leq$ is a co-lex order on $\mathcal{D}$ and for some distinct $u, v \in Q$, $u < v$, and $u \not<_{\mathcal{D}} v$. Then, there exist $\alpha \in I_u, \beta \in I_v$ with $\beta \prec \alpha$. Let us fix $u$, $v$, $\alpha$ and $\beta$ with the above properties such that $\beta$ has the minimum possible length. Notice that $\beta$ cannot be the empty word, otherwise $v$ would be the initial state $s$, while $z \not< s$ for all $z \in Q$ (see Remark 2.2). Hence, $\beta = \beta' e$ for some $e \in \Sigma$ and $\beta \prec \alpha$ implies $\alpha = \alpha' f$ for some $f \in \Sigma$. We then have $e \in \lambda(v)$, $f \in \lambda(u)$, and by Axiom 1 of co-lex orders we get $f \leq e$. From $\beta = \beta' e \prec \alpha = \alpha' f$ we conclude $f = e$ and $\beta' \prec \alpha'$. If $u', v'$ are such that $\delta(u', e) = u, \delta(v', e) = v$ and $\alpha' \in I_{u'}, \beta' \in I_{v'}$, then by Axiom 2 of co-lex orders and Remark 2.2 we get $u' < v'$; however, the pair $\alpha', \beta'$ witnesses $u' \not<_{\mathcal{D}} v'$, contradicting the minimality of $\beta$. □

Having proved that $\leq_{\mathcal{D}}$ is the maximum co-lex order over $\mathcal{D}$, we immediately deduce that its characterizing property is satisfied by *any* co-lex order.

Corollary 2.9. *If $\leq$ is a co-lex order over a DFA and $u < v$ then $(\forall \alpha \in I_u)(\forall \beta \in I_v)\ (\alpha \prec \beta)$.*

*Remark 2.10.* The previous corollary shows that Axiom 1 of co-lex orders *propagates* from a *local* level (i.e., letters in $\lambda(u), \lambda(v)$, for which it holds $(\forall e \in \lambda(u))(\forall f \in \lambda(v))(e \leq f)$) to a *global* one (i.e., words in $I_u, I_v$, for which it holds $(\forall \alpha \in I_u)(\forall \beta \in I_v)(\alpha \preceq \beta)$). This works for DFAs because different states are reached by disjoint sets of words: if $u \neq v$ then $I_u \cap I_v = \emptyset$. On NFAs things become more complicated and the existence of a maximum co-lex order is no longer guaranteed.

Lemma 2.8 established that a DFA $\mathcal{D}$ always has a maximum co-lex order $\leq_{\mathcal{D}}$. This lemma will be used in Section 3 to prove that both the cardinality and the width of the automaton resulting from the powerset construction applied to any NFA $\mathcal{N}$ are fixed-parameter linear in $|\mathcal{N}|$ with parameter $\mathrm{width}(\mathcal{N})$. Since $\leq_{\mathcal{D}}$ extends any possible co-lex order on $Q$, it realizes the width of the automaton $\mathcal{D}$, as stated in the following lemma.

Lemma 2.11. *If $\mathcal{D}$ is a DFA then $\mathit{width}(\leq_{\mathcal{D}}) = \mathit{width}(\mathcal{D})$.*

With the next example, we show that the co-lex width of an automaton may depend on the total order on the alphabet.

*Example 2.12.* Let $\mathcal{D}$ be a DFA. Let us show that, in general, the value $\mathrm{width}(\mathcal{D}) = \mathrm{width}(\leq_{\mathcal{D}})$ (Lemma 2.11) may depend on the total order $\preceq$ on the alphabet. Let $\mathcal{D}$ be the DFA in Figure 2.

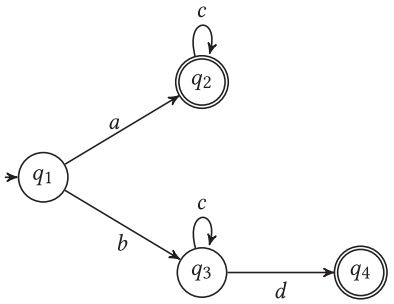



Fig. 2. A DFA $\mathcal{D}$ where the value width($\mathcal{D}$) depends on the total order $\preceq$ on the alphabet.

First, assume that $\preceq$ is the standard alphabetical order such that $a \prec b \prec c \prec d$. We have $q_1 <_{\mathcal{D}} q_2 <_{\mathcal{D}} q_4$ and $q_1 <_{\mathcal{D}} q_3 <_{\mathcal{D}} q_4$, so width($\leq_{\mathcal{D}}$) is at most two. Notice that $q_2$ and $q_3$ are not $\leq_{\mathcal{D}}$-comparable because $acc, ac \in I_{q_2}$, $bc \in I_{q_3}$ and $ac \prec bc \prec acc$, so width($\leq_{\mathcal{D}}$) is equal to two.

Next, assume that $\preceq$ is the total order such that $a \prec c \prec b \prec d$. Then, $q_1 <_{\mathcal{D}} q_2 <_{\mathcal{D}} q_3 <_{\mathcal{D}} q_4$, hence width($\leq_{\mathcal{D}}$) is equal to one.

We now generalize Corollary 2.9 to NFAs, coping with the fact that, on NFAs, sets $I_u, I_v$ may intersect for $u \neq v$ (we shall use this result in Sections 3 and 5).

Lemma 2.13. *Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA and let $\leq$ be a co-lex order on $\mathcal{N}$. If $u < v$, then $(\forall \alpha \in I_u)(\forall \beta \in I_v)(\{\alpha, \beta\} \nsubseteq I_u \cap I_v \Rightarrow \alpha \prec \beta)$.*

Proof. Let $\alpha \in I_u$ and $\beta \in I_v$ such that $\{\alpha, \beta\} \nsubseteq I_u \cap I_v$. We must prove that $\alpha \prec \beta$. Let $\gamma \in \Sigma^*$ be the longest string such that $\alpha = \alpha'\gamma$ and $\beta = \beta'\gamma$, for some $\alpha', \beta' \in \text{Pref}(\mathcal{L}(\mathcal{N}))$. If $\alpha' = \varepsilon$ the claim follows, therefore we can assume $|\alpha'| \geq 1$.

Let $\gamma = c_p \dots c_1$, with $c_i \in \Sigma$ for $i \in \{1, \dots, p\}$ $(p \geq 0)$, $\alpha' = a_q \dots a_1$, with $a_i \in \Sigma$ for $i \in \{1, \dots, q\}$ $(q \geq 1)$, and $\beta' = b_r \dots b_1$, with $b_i \in \Sigma$ for $i \in \{1, \dots, r\}$ $(r \geq 0)$.

Assume $|\gamma| > 0$. Since $\alpha \in I_u$ and $\beta \in I_v$, then there exist $u_1, v_1 \in Q$ such that $\alpha' c_p \dots c_2 \in I_{u_1}$, $\beta' c_p \dots c_2 \in I_{v_1}$, $u \in \delta(u_1, c_1)$ and $v \in \delta(v_1, c_1)$. By Axiom 2, we obtain $u_1 \leq v_1$. However, it cannot be $u_1 = v_1$ because this would imply $\{\alpha, \beta\} \subseteq I_u \cap I_v$, so it must be $u_1 < v_1$. By iterating this argument, we conclude that there exist $u', v' \in Q$ such that $\alpha' \in I_{u'}$, $\beta' \in I_{v'}$ and $u' < v'$, and the same conclusion holds if $\gamma = \varepsilon$ as well.

Now, it cannot be $r = 0$ because this would imply $v' = s$, and $u' < s$ contradicts Remark 2.2. Hence, it must be $|\beta| \geq 1$. By Axiom 1, it must be $a_1 \preceq b_1$. At the same time, the definition of $\gamma$ implies that it cannot be $a_1 = b_1$, so we obtain $a_1 \prec b_1$ and we can conclude $\alpha \prec \beta$. □

Lemma 2.13 has an important implication (which we will use later in Theorem 3.2 and will stand at the core of the encoding and indexing results in Section 5): given a co-lex order on an NFA, then the sets $I_\alpha$'s are convex w.r.t this order.

Corollary 2.14. *Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA and let $\leq$ be a co-lex order on $\mathcal{N}$. If $\alpha \in$ Pref($\mathcal{L}(\mathcal{N})$) then $I_\alpha$ is convex in $(Q, \leq)$.*

Proof. Suppose $u, z \in I_\alpha$ and let $v \in Q$ be such that $u < v < z$. We have to prove that $v \in I_\alpha$. If this were not true, we would have $\alpha \in (I_u \cap I_z) \setminus I_v$. Consider any $\beta \in I_v$. By Lemma 2.13 we would have $\alpha \prec \beta \prec \alpha$, a contradiction. □

The notion of co-lex width of an automaton naturally calls into play the problem of determining the complexity of computing this measure. More precisely, we define the *NFA-width problem* as follows:

*Problem 1* (NFA Width Problem). *Given an NFA $\mathcal{N}$ and an integer $p$, determine whether:*

$$width(\mathcal{N}) \leq p.$$

Problem 1 will be tackled for the particular case of DFAs in Section 4.

A natural question is whether there is a connection between the notion of width and the complexity of regular expressions. The case $width(\mathcal{N}) = 1$ corresponds to the class of Wheeler automata [47]. In [3], it was shown that *Wheeler languages*, that is, regular languages recognized by Wheeler automata, are closed essentially only under intersection and under concatenation with a finite language. On the other hand, with the next two remarks we point out that our notion of width can easily be used to capture complementation, intersection and union.

*Remark 2.15.* Let $\mathcal{D}$ be a DFA. Then, there exists a DFA $\mathcal{D}'$ such that $\mathcal{L}(\mathcal{D}') = \Sigma^* \setminus \mathcal{L}(\mathcal{D})$ and $\text{width}(\mathcal{D}') \leq \text{width}(\mathcal{D}) + 1$. The DFA $\mathcal{D}'$ is obtained by first transforming $\mathcal{D}$ into a complete DFA by adding a non-final sink state that is reached by all transitions not defined in $\mathcal{D}$ (including the ones leaving the sink itself), then switching final and non-final states, and finally removing all states that neither are final, nor allow to reach a final state. In the worst case, the new sink state is not $\leq_{\mathcal{D}'}$-comparable with any other state and therefore the width cannot increase by more than one.

*Remark 2.16.* Let $\mathcal{D}_1 = (Q_1, s_1, \delta_1, F_1), \mathcal{D}_2 = (Q_2, s_2, \delta_2, F_2)$ be DFAs, with $\text{width}(\mathcal{D}_1) = p_1$ and $\text{width}(\mathcal{D}_2) = p_2$. Let us prove that there exists a DFA $\mathcal{D}$ such that $\mathcal{L}(\mathcal{D}) = \mathcal{L}(\mathcal{D}_1) \cap \mathcal{L}(\mathcal{D}_2)$ and $\text{width}(\mathcal{D}) \leq p_1 \cdot p_2$ and a DFA $\mathcal{D}'$ such that $\mathcal{L}(\mathcal{D}') = \mathcal{L}(\mathcal{D}_1) \cup \mathcal{L}(\mathcal{D}_2)$ and $\text{width}(\mathcal{D}') \leq p_1 \cdot p_2 + p_1 + p_2$. In the following, we will implicitly use Lemma 2.11. Let $\{Q_1, \ldots, Q_{p_1}\}$ and $\{Q_1^*, \ldots, Q_{p_2}^*\}$ be a $\leq_{\mathcal{D}_1}$-chain partition of $Q_1$ and a $\leq_{\mathcal{D}_2}$-chain partition of $Q_2$, respectively. In order to build both $\mathcal{D}$ and $\mathcal{D}'$, we first turn $\mathcal{D}_1$ and $\mathcal{D}_2$ into complete DFAs by adding non-final sinks $\star_1, \star_2$ to $Q_1$ and $Q_2$, respectively (like in Remark 2.15), then we build the standard product automaton: the set of states is $(Q_1 \cup \{\star_1\}) \times (Q_2 \cup \{\star_2\})$, the initial state is $(s_1, s_2)$, and the transition function is defined by $\delta((u, v), a) = (\delta_1(u, a), \delta_2(v, a))$, for every state $(u, v)$ and for every $a \in \Sigma$. The difference between $\mathcal{D}$ and $\mathcal{D}'$ lies in how the set of final states is defined.

We define $\mathcal{D}$ by letting $F = F_1 \times F_2$ being the set of all final states, and then removing (i) all states that are not reachable from the initial state and (ii) all states that neither are final, nor allow to reach a final state. Let $\mathcal{D} = (Q, s, \delta, F)$ the resulting DFA at the end of the construction. Notice that $Q \subseteq Q_1 \times Q_2$ (that is, the sinks play no role) because in $\mathcal{D}_1$ and $\mathcal{D}_2$ the sinks are not final and do not allow to reach a final state. Moreover, we have $I_{(u,v)} = I_u \cap I_v$ for every $(u, v) \in Q$, because for every $\alpha \in \Sigma^*$ there exists a path labeled $\alpha$ from $(s_1, s_2)$ to $(u, v)$ on $\mathcal{D}$ if and only there exist a path labeled $\alpha$ from $s_1$ to $u$ on $\mathcal{D}_1$ and a path labeled $\alpha$ from $s_2$ to $v$ on $\mathcal{D}_2$. As a consequence, $\mathcal{L}(\mathcal{D}) = \bigcup_{(u,v)\in F} I_{(u,v)} = \bigcup_{u\in F_1, v\in F_2} I_u \cap I_v = \mathcal{L}(\mathcal{D}_1) \cap \mathcal{L}(\mathcal{D}_2)$. Now, let us prove that $\text{width}(\mathcal{D}) \leq p_1 \cdot p_2$. For every $i = 1, \ldots, p_1$ and for every $j = 1, \ldots, p_2$, define:

$$Q_{i,j} = \{(u, v) \in Q \mid u \in Q_i, v \in Q_j^*\}.$$

Since $\{Q_{i,j} \mid 1 \leq i \leq p_1, 1 \leq j \leq p_2\}$ is a partition of $Q$, we only have to show that every $Q_{i,j}$ is a $\leq_{\mathcal{D}}$-chain. Let $(u_1, v_1), (u_2, v_2)$ be distinct elements in $Q_{i,j}$. Hence, at least one between $u_1 \neq u_2$ and $v_1 \neq v_2$ holds true. Assume that $u_1 \neq u_2$ (the other case is analogous). Since $u_1, u_2 \in Q_i$, then $u_1 <_{\mathcal{D}_1} u_2$ or $u_2 <_{\mathcal{D}_1} u_1$. Assuming without loss of generality that $u_1 <_{\mathcal{D}_1} u_2$, then from $I_{(u_1,v_1)} \subseteq I_{u_1}$ and $I_{(u_2,v_2)} \subseteq I_{u_2}$ we conclude $(u_1, v_1) <_{\mathcal{D}} (u_2, v_2)$.

Next, we define $\mathcal{D}'$ by letting $F' = (F_1 \times Q_2) \cup (Q_1 \times F_2)$ be the set of all final states, and then removing (i) all states that are not reachable from the initial state and removing (ii) all states that neither are final, nor allow to reach a final state. Let $\mathcal{D}' = (Q', (s_1, s_2), \delta, F')$ the resulting DFA at the end of the construction. Again, we have $I_{(u,v)} = I_u \cap I_v$ for every $(u, v) \in Q'$ such that $u \in Q_1$

and $v \in Q_2$, so $\mathcal{L}(\mathcal{D}) = \bigcup_{(u,v)\in F} I_{(u,v)} = \bigcup_{u\in F_1 \vee v\in F_2} I_u \cap I_v = \mathcal{L}(\mathcal{D}_1) \cup \mathcal{L}(\mathcal{D}_2)$. Let us prove that $\text{width}(\mathcal{D}') \leq p_1 \cdot p_2 + p_1 + p_2$. Notice that this time if $(u, v) \in Q'$, then it may happen that $u = \star_1$ or $v = \star_2$ (but not both). Moreover, if $(u, \star_2) \in Q'$, then $I_{(u,\star_2)} = I_u \setminus \text{Pref}(\mathcal{L}_2)$, and if $(\star_1, v) \in Q'$, then $I_{(\star_1,v)} = I_v \setminus \text{Pref}(\mathcal{L}_1)$. For every $i = 1, \ldots, p_1$ and for every $j = 1, \ldots, p_2$, define:

$$Q'_{i,j} = \{(u, v) \in Q' \mid u \in Q_i, v \in Q_j^*\}$$
$$Q'_{i,\star_2} = \{(u, \star_2) \in Q' \mid u \in Q_i\}$$
$$Q'_{\star_1,j} = \{(\star_1, v) \in Q' \mid v \in Q_j^*\}.$$

These sets identify a partition of $Q'$, and like before one can show that each set is a $\leq_{\mathcal{D}'}$-chain. We conclude that $\text{width}(\mathcal{D}') \leq p_1 \cdot p_2 + p_1 + p_2$.

The above two remarks suggest that our notion of width is related with the structural complexity of the regular expressions accepting a given regular language. A more precise and complete analysis (improving these bounds, describing the behavior on NFAs and including the other regular operators, namely, concatenation and Kleene star) will be the subject of forthcoming works (see also Remark 2.18 for the consequences of the above two remarks on the smallest-width DFA recognizing a regular language).

### 2.3 The Co-lex Width of a Regular Language

On the grounds of the definition of automata's width, we also study some implications on the theory of regular languages. We start by defining the width of a regular language based on the co-lex orders of the automata recognizing it.

*Definition 2.17.* Let $\mathcal{L}$ be a regular language.

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

In Example 2.12 we showed that the width of an automaton may depend on the total order $\preceq$ on the alphabet. In Example 4.24 below, we will show that the deterministic and nondeterministic widths of a language may also depend on the order $\preceq$ on the alphabet.

On the grounds of Remarks 2.15 and 2.16, we observe the following relations, which allow us to conclude that already constant-width regular languages form an interesting class:

*Remark 2.18.* Let $\mathcal{L}$, $\mathcal{L}_1$, and $\mathcal{L}_2$ be any regular languages. Then:

(1) $\text{width}^D(\Sigma^* \setminus \mathcal{L}) \leq \text{width}^D(\mathcal{L}) + 1$
(2) $\text{width}^D(\mathcal{L}_1 \cap \mathcal{L}_2) \leq \text{width}^D(\mathcal{L}_1) \cdot \text{width}^D(\mathcal{L}_2)$
(3) $\text{width}^D(\mathcal{L}_1 \cup \mathcal{L}_2) \leq \text{width}^D(\mathcal{L}_1) \cdot \text{width}^D(\mathcal{L}_2) + \text{width}^D(\mathcal{L}_1) + \text{width}^D(\mathcal{L}_2)$.

These inequalities are a direct consequence of Remarks 2.15 and 2.16 by starting from smallest-width DFAs recognizing $\mathcal{L}$, $\mathcal{L}_1$, and $\mathcal{L}_2$.

By Remark 2.18, if $\mathcal{L}$ can be written as the boolean combination of a constant number of Wheeler languages (for example, of a constant number of finite languages), then $\text{width}^N(\mathcal{L}) \leq \text{width}^D(\mathcal{L}) \in O(1)$. Furthermore, this bound holds for any total order $\preceq$ on the alphabet if the starting languages are finite (because finite languages are Wheeler independent of the alphabet order).

Definition 2.17 introduces two hierarchies of regular languages. As shown in Section 5, languages in the first levels of these hierarchies are much easier to index and compress (see Section 2.4 for a definition of the indexing and compression problems for finite-state automata). Hence, it is interesting to determine the correct position of a given language in the above hierarchies, that is, to solve the *language width* problem.

*Problem 2* (Language Width Problem). *Given a regular language $\mathcal{L}$ (by means of a DFA or an NFA recognizing it) and an integer $p$, determine whether:*

$-\ width^D(\mathcal{L}) \leq p.$
$-\ width^N(\mathcal{L}) \leq p.$

The deterministic case will be tackled in Section 4. In the companion article [31] we also study the relationships between the deterministic and the non-deterministic hierarchies, proving that every level of both hierarchies is non-empty and that, apart from level 1, the levels of the two hierarchies do not coincide.

The notion of width of a language naturally calls into play the problem of giving an automaton realizing the width of the language.

*Problem 3* (Minimum-width DFA). *Given a regular language $\mathcal{L}$, can we define a canonical DFA $\mathcal{D}$ such that $\mathcal{L}(\mathcal{D}) = \mathcal{L}$ and $width(\mathcal{D}) = width^D(\mathcal{L})$?*

A solution to the above problem is provided in Section 4.2, Theorem 4.21. We shall prove that the minimum-width DFA problem is *orthogonal* with respect to the well-studied minimum-size DFA problem: it could be necessary to split states in order to minimize the width of a DFA. Hence, reducing the size of a given DFA does not necessarily bring us to the minimum-width DFA for the same language.

This side of the topic is complemented considering the following problem.

*Problem 4* (Automata-free Characterization). *Is there an automata-free characterization of languages with deterministic width less than or equal to $p$?*

Two solutions to Problem 4 will be given: on the one hand, we will prove that we can characterize width-$p$ languages by their co-lex monotone sequences of prefixes. On the other hand, a further solution will be given by proving a generalization of the Myhill-Nerode Theorem for the class of DFAs of width $p$ (for any fixed $p$).

### 2.4 Compression and Indexing

Another problem that we will consider is that of improving the space usage of the existing encoding for automata (such as the one of Chakraborty et al. [25] discussed in Section 1.1), in the case that the automaton's width is small:

*Problem 5* (Compressing Automata). *Can we represent a finite state automaton using $\log \sigma + o(\log n)$ bits per edge, provided its co-lex width $p$ satisfies $\log p \in o(\log n)$?*

The motivation behind Problem 5 is that Wheeler automata, that is, automata of co-lex width equal to 1, can be represented using $\log \sigma + O(1)$ bits per edge [47]. Indeed, in Sections 5.2 and 5.4 we will show that our notion of co-lex order will enable us to solve Problem 5 through a generalization of the powerful ***Burrows-Wheeler transform*** (**BWT**) [23]. The BWT of a text is a permutation that re-arranges the text's characters according to the co-lex order of the prefixes that

precede them.[4] A well-known fact is that the BWT boosts compression and enables efficient indexing (read Problem 6 below) in compressed space [45]. Previous works generalized this transform to trees [44], string sets [64], de Bruijn graphs [17, 63] and Wheeler graphs [47]. In Section 5.2 we complete the picture by generalizing those BWT-based indexes to NFAs (equivalently, to arbitrary labeled graphs) and languages and showing that this index indeed solves Problem 6:

*Problem 6* (Indexing Automata). *Let $\mathcal{N} = (Q, s, \delta, F)$ be a finite-state automaton. Let moreover $T(\alpha)$ be the set of states of $\mathcal{N}$ which are reached by a path labeled with a given word $\alpha$, i.e., $T(\alpha) = \{u \in Q \mid (\exists \beta \in I_u)(\alpha \dashv \beta)\}$. Pre-process $\mathcal{N}$ into a small data structure supporting efficient* subpath queries*, i.e., given a query word $\alpha$ solve:*

- *(Existential queries) Determine whether $T(\alpha) \neq \emptyset$, i.e., whether $\alpha$ matches a substring of some string in the language of $\mathcal{N}$.*
- *(Count queries) Compute the cardinality of $T(\alpha)$.*
- *(Locate queries) Return a representation for all the states in $T(\alpha)$.*

Recent works (see Section 1.1) show that, as opposed to the membership problem on DFAs, all the three above-listed subpath queries are hard even on acyclic DFAs: unless the Strong Exponential Time Hypothesis fails, such queries require quadratic time to be solved off-line [40] and even on-line using any index constructible in polynomial time [41]. In Section 5.3 we provide a linear-space index solving subpath (and thus membership) queries on NFAs in time proportional to $p^2$ per character in the input query, $p$ being the automaton's co-lex width. The index can be built in polynomial time on DFAs and exponential time on NFAs (the latter complexity is due to the hardness of computing the co-lex width of NFAs). For small $p$, our index for DFAs breaks the indexing lower bound of Equi et al. [41].

## 3 CO-LEX WIDTH AND NFA DETERMINIZATION

In this section we show that the notion of width can be used to prove some crucial relationships between an NFA $\mathcal{N}$ and the powerset automaton $\mathrm{Pow}(\mathcal{N})$ obtained from $\mathcal{N}$. First, we bound the width of $\mathrm{Pow}(\mathcal{N})$ in terms of the width of $\mathcal{N}$ and prove that the number of $\mathrm{Pow}(\mathcal{N})$'s states is exponential in $\mathrm{width}(\mathcal{N})$ rather than in the number of $\mathcal{N}$'s states. This implies that several problems easy on DFAs but difficult on NFAs are in fact fixed-parameter tractable with respect to the width.

Recall that, given an NFA $\mathcal{N} = (Q, s, \delta, F)$, the powerset construction algorithm builds an equivalent DFA $\mathrm{Pow}(\mathcal{N}) = (Q^*, s^*, \delta^*, F^*)$ defined as:

- $Q^* = \{I_\alpha \mid \alpha \in \mathrm{Pref}(\mathcal{L}(\mathcal{N}))\}$;
- $s^* = \{s\}$;
- $\delta^*(I_\alpha, a) = I_{\alpha a}$ for all $\alpha \in \Sigma^*$ and $a \in \Sigma$ such that $\alpha a \in \mathrm{Pref}(\mathcal{L}(\mathcal{N}))$;
- $F^* = \{I_\alpha \mid \alpha \in \mathcal{L}(\mathcal{N})\}$.

For $\alpha, \alpha' \in \mathrm{Pref}(\mathcal{L}(\mathcal{N}))$ we have:

$$\delta^*(s^*, \alpha') = I_\alpha \iff I_{\alpha'} = I_\alpha$$

and defining as usual $I_{u^*} = \{\alpha \in \mathrm{Pref}(\mathcal{L}(\mathrm{Pow}(\mathcal{N}))) \mid u^* \in \delta^*(s^*, \alpha)\}$ for $u^* \in Q^*$, we have that for $\alpha \in \mathrm{Pref}(\mathcal{L}(\mathcal{N}))$:

$$I_{I_\alpha} = \{\alpha' \in \mathrm{Pref}(\mathcal{L}(\mathcal{N})) \mid I_{\alpha'} = I_\alpha\}. \tag{1}$$

[4] The original definition considers the lexicographic order of the text's suffixes. We choose a symmetric definition because it can be generalized to NFAs via co-lex orders.

We start with a characterization of the maximum co-lex order on $\mathrm{Pow}(\mathcal{N})$ (which exists by Lemma 2.8).

Lemma 3.1. *Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA and let $\mathrm{Pow}(\mathcal{N}) = (Q^*, s^*, \delta^*, F^*)$ be the powerset automaton obtained from $\mathcal{N}$. Let $\leq_{\mathrm{Pow}(\mathcal{N})}$ be the maximum co-lex order on $\mathrm{Pow}(\mathcal{N})$. Then, for $I_\alpha \neq I_\beta$:*

$$(I_\alpha <_{\mathrm{Pow}(\mathcal{N})} I_\beta) \iff (\forall \alpha', \beta' \in \mathrm{Pref}(\mathcal{L}(\mathcal{N})))((I_{\alpha'} = I_\alpha) \wedge (I_{\beta'} = I_\beta) \to \alpha' \prec \beta')$$

*Moreover, let $\leq$ be a co-lex order on $\mathcal{N}$, and fix $\alpha, \beta \in \mathrm{Pref}(\mathcal{L}(\mathcal{N}))$. Then:*

$$(\exists u \in I_\alpha)(\exists v \in I_\beta)(\{u, v\} \not\subseteq I_\alpha \cap I_\beta \wedge u < v) \Rightarrow (I_\alpha <_{\mathrm{Pow}(\mathcal{N})} I_\beta).$$

Proof. The first part follows immediately from the characterization of the maximum co-lex order over a DFA (Lemma 2.8) and Equation (1). Let us prove the second part. Consider $u \in I_\alpha$ and $v \in I_\beta$ such that $\{u, v\} \not\subseteq I_\alpha \cap I_\beta$ and $u < v$. We prove that $I_\alpha <_{\mathrm{Pow}(\mathcal{N})} I_\beta$ using the characterization of $<_{\mathrm{Pow}(\mathcal{N})}$ given in the first part of the proof. Fix $\alpha', \beta' \in \mathrm{Pref}(\mathcal{L}(\mathcal{N}))$ such that $I_{\alpha'} = I_\alpha$ and $I_{\beta'} = I_\beta$. We must prove that $\alpha' \prec \beta'$. From the hypothesis it follows $u \in I_{\alpha'}, v \in I_{\beta'}$, and $\{u, v\} \not\subseteq I_{\alpha'} \cap I_{\beta'}$ so that $\alpha' \in I_u, \beta' \in I_v$, and $\{\alpha', \beta'\} \not\subseteq I_u \cap I_v$ hold. Hence, $\alpha' \prec \beta'$ follows from $u < v$ and Lemma 2.13. □

We can now prove the main result of this section.

Theorem 3.2. *Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA and let $\mathrm{Pow}(\mathcal{N}) = (Q^*, s^*, \delta^*, F^*)$ be the powerset automaton obtained from $\mathcal{N}$. Let $n = |Q|$ and $p = \mathit{width}(\mathcal{N})$. Then:*

*(1) $\mathit{width}(\mathrm{Pow}(\mathcal{N})) \leq 2^p - 1$;*
*(2) $|Q^*| \leq 2^p(n - p + 1) - 1$.*

Proof. Let $\leq$ be a co-lex order on $\mathcal{N}$ such that $\mathit{width}(\leq) = p$, and let $\{Q_i \mid 1 \leq i \leq p\}$ be a $\leq$-chain partition. Let $\leq_{\mathrm{Pow}(\mathcal{N})}$ be the maximum co-lex order on $\mathrm{Pow}(\mathcal{N})$. For every nonempty $K \subseteq \{1, \ldots, p\}$, define:

$$\mathcal{I}_K = \{I_\alpha \mid (\forall i \in \{1, \ldots, p\})(I_\alpha \cap Q_i \neq \emptyset \iff i \in K)\}.$$

Notice that $Q^*$ is the disjoint union of all $\mathcal{I}_K$. More precisely:

$$Q^* = \bigsqcup_{\emptyset \neq K \subseteq \{1, \ldots, p\}} \mathcal{I}_K. \tag{2}$$

Let us prove that each $\mathcal{I}_K$ is a $\leq_{\mathrm{Pow}(\mathcal{N})}$-chain. Fix $I_\alpha, I_\beta \in \mathcal{I}_K$, with $I_\alpha \neq I_\beta$. We must prove that $I_\alpha$ and $I_\beta$ are $\leq_{\mathrm{Pow}(\mathcal{N})}$-comparable. Since $I_\alpha \neq I_\beta$, there exists either $u \in I_\alpha \setminus I_\beta$ or $v \in I_\beta \setminus I_\alpha$. Assume that there exists $u \in I_\alpha \setminus I_\beta$ (the other case is analogous). In particular, let $i \in \{1, \ldots, p\}$ be the unique integer such that $u \in Q_i$. Since $I_\alpha, I_\beta \in \mathcal{I}_K$, from the definition of $\mathcal{I}_K$ it follows that there exists $v \in I_\beta \cap Q_i$. Notice that $\{u, v\} \not\subseteq I_\alpha \cap I_\beta$ (so in particular $u \neq v$), and since $u, v \in Q_i$ we conclude that $u$ and $v$ are $\leq$-comparable. By Lemma 3.1 we conclude that $I_\alpha$ and $I_\beta$ are $\leq_{\mathrm{Pow}(\mathcal{N})}$-comparable.

(1) The first part of the theorem follows from Equation (2), because each $\mathcal{I}_K$ is a $\leq_{\mathrm{Pow}(\mathcal{N})}$-chain and there are $2^p - 1$ choices for $K$.
(2) Let us prove the second part of the theorem. Fix $\emptyset \neq K \subseteq \{1, \ldots, p\}$. For every $I_\alpha \in \mathcal{I}_K$ and for every $i \in K$, let $m_\alpha^i$ be the smallest element of $I_\alpha \cap Q_i$ (this makes sense because $(Q_i, \leq)$ is totally ordered), and let $M_\alpha^i$ be the largest element of $I_\alpha \cap Q_i$. Fix $I_\alpha, I_\beta \in \mathcal{I}_K$, and note the following:

(a) Assume that for some $i \in K$ it holds $m^i_\alpha < m^i_\beta \vee M^i_\alpha < M^i_\beta$. Then, it must be $I_\alpha <_{\mathrm{Pow}(\mathcal{N})} I_\beta$. Indeed, assume that $m^i_\alpha < m^i_\beta$ (the other case is analogous). We have $m^i_\alpha \in I_\alpha$, $m^i_\beta \in I_\beta$, $\{m^i_\alpha, m^i_\beta\} \not\subseteq I_\alpha \cap I_\beta$ and $m^i_\alpha < m^i_\beta$, so the conclusion follows from Lemma 3.1. Equivalently, we can state that if $I_\alpha <_{\mathrm{Pow}(\mathcal{N})} I_\beta$ then $(\forall i \in K)(m^i_\alpha \leq m^i_\beta \wedge M^i_\alpha \leq M^i_\beta)$.

(b) Assume that for some $i \in K$ it holds $m^i_\alpha = m^i_\beta \wedge M^i_\alpha = M^i_\beta$. By Corollary 2.14, the sets $I_\alpha$ and $I_\beta$ are convex in $(Q, \leq)$. This implies that $I_\alpha \cap Q_i$ and $I_\beta \cap Q_i$ are $\leq_{Q_i}$-convex, and having the same minimum and maximum they must be equal, that is, $I_\alpha \cap Q_i = I_\beta \cap Q_i$.

(c) Assume that $(\forall i \in K)(m^i_\alpha = m^i_\beta \wedge M^i_\alpha = M^i_\beta)$. Then, it must be $I_\alpha = I_\beta$. Indeed, from point (b) we obtain $(\forall i \in K)(I_\alpha \cap Q_i = I_\beta \cap Q_i)$, so $I_\alpha = \bigcup_{i\in K}(I_\alpha \cap Q_i) = \bigcup_{i\in K}(I_\beta \cap Q_i) = I_\beta$. Notice that we can equivalently state that if $I_\alpha \neq I_\beta$, then $(\exists i \in K)(m^i_\alpha \neq m^i_\beta \vee M^i_\alpha \neq M^i_\beta)$.

Fix $I_\alpha, I_\beta \in \mathcal{I}_K$. Now it is easy to show that:

$$I_\alpha <_{\mathrm{Pow}(\mathcal{N})} I_\beta \iff (\forall i \in K)\left(m^i_\alpha \leq m^i_\beta \wedge M^i_\alpha \leq M^i_\beta\right) \wedge \\ \wedge (\exists i \in K)\left(m^i_\alpha < m^i_\beta \vee M^i_\alpha < M^i_\beta\right). \tag{3}$$

Indeed, ($\Leftarrow$) follows from point (a). As for ($\Rightarrow$), notice that $(\forall i \in K)(m^i_\alpha \leq m^i_\beta \wedge M^i_\alpha \leq M^i_\beta)$ again follows from point (a), whereas $(\exists i \in K)(m^i_\alpha < m^i_\beta \vee M^i_\alpha < M^i_\beta)$ follows from point (c).

Let $|m^i_\alpha|$ and $|M^i_\alpha|$ be the positions of $m^i_\alpha$ and $M^i_\alpha$ in the total order $(Q_i, \leq)$ (so $|m^i_\alpha|, |M^i_\alpha| \in \{1, \ldots, |Q_i|\}$). For every $I_\alpha \in \mathcal{I}_K$, define:

$$T(I_\alpha) = \sum_{i\in K}\left(\left|m^i_\alpha\right| + \left|M^i_\alpha\right|\right).$$

By Equation (3), we have that $I_\alpha <_{\mathrm{Pow}(\mathcal{N})} I_\beta$ implies $T(I_\alpha) < T(I_\beta)$, so since $\mathcal{I}_K$ is a $<_{\mathrm{Pow}(\mathcal{N})}$-chain we have that $|\mathcal{I}_K|$ is bounded by the values that $T(I_\alpha)$ can take. For every $I_\alpha \in \mathcal{I}_K$ we have $2|K| \leq T(I_\alpha) \leq 2\sum_{i\in K}|Q_i|$ (because $|m^i_\alpha|, |M^i_\alpha| \in \{1, \ldots, |Q_i|\}$), so:

$$|\mathcal{I}_K| \leq 2\sum_{i\in K}|Q_i| - 2|K| + 1. \tag{4}$$

From Equations (2) and (4), we obtain:

$$|Q^*| = \sum_{\emptyset\subsetneq K\subseteq\{1,\ldots,p\}} |\mathcal{I}_K| \leq \sum_{\emptyset\subsetneq K\subseteq\{1,\ldots,p\}} \left(2\sum_{i\in K}|Q_i| - 2|K| + 1\right) \\ = 2\sum_{\emptyset\subsetneq K\subseteq\{1,\ldots,p\}}\sum_{i\in K}|Q_i| - 2\sum_{\emptyset\subsetneq K\subseteq\{1,\ldots,p\}}|K| + \sum_{\emptyset\subsetneq K\subseteq\{1,\ldots,p\}} 1.$$

Notice that $\sum_{\emptyset\subsetneq K\subseteq\{1,\ldots,p\}}\sum_{i\in K}|Q_i| = 2^{p-1}\sum_{i\in\{1,\ldots,p\}}|Q_i| = 2^{p-1}n$ because every $i \in \{1, \ldots, p\}$ occurs in exactly $2^{p-1}$ subsets of $\{1, \ldots, p\}$. Similarly, we obtain $\sum_{\emptyset\subsetneq K\subseteq\{1,\ldots,p\}}|K| = 2^{p-1}p$ and $\sum_{\emptyset\subsetneq K\subseteq\{1,\ldots,p\}} 1 = 2^p - 1$, We conclude:

$$|Q^*| \leq 2^p n - 2^p p + 2^p - 1 = 2^p(n - p + 1) - 1.$$

□

As a first consequence of Theorem 3.2 we start comparing the non-deterministic and deterministic width hierarchies of regular languages. Clearly, for every regular language $\mathcal{L}$ we have

$\text{width}^N(\mathcal{L}) \leq \text{width}^D(\mathcal{L})$ since DFAs are particular cases of NFAs. Moreover, for languages with $\text{width}^N(\mathcal{L}) = 1$, the so-called Wheeler languages, it is known that the non-deterministic and deterministic widths coincide [2]. Nonetheless, in the companion article [31] we will prove that this property is truly peculiar of Wheeler languages, because the gap between the deterministic and non-deterministic hierarchies is, in general, exponential. Here we prove that Theorem 3.2 provides an *upper* bound for the deterministic width in terms of the non-deterministic width.

Corollary 3.3. *Let $\mathcal{L}$ be a regular language. Then, $\text{width}^D(\mathcal{L}) \leq 2^{\text{width}^N(\mathcal{L})} - 1$.*

Proof. Let $\mathcal{N}$ be an NFA such that $\mathcal{L}(\mathcal{N}) = \mathcal{L}$ and $\text{width}(\mathcal{N}) = \text{width}^N(\mathcal{L})$. By Theorem 3.2, we have $\text{width}^D(\mathcal{L}) \leq \text{width}(Pow(\mathcal{N})) \leq 2^{\text{width}(\mathcal{N})} - 1 = 2^{\text{width}^N(\mathcal{L})} - 1$. □

In a forthcoming work we will prove that the above bound is tight. Notice that the above corollary shows once again that $\text{width}^N(\mathcal{L}) = 1$ implies $\text{width}^D(\mathcal{L}) = 1$, that is, the non-deterministic and deterministic widths are equal for Wheeler languages.

Theorem 3.2 has another intriguing consequence: the PSPACE-complete NFA equivalence problem [79] is fixed-parameter tractable with respect to the widths of the automata. In order to prove this result, we first update the analysis of Hopcroft et al. [55] of the powerset construction algorithm.

Lemma 3.4 (Adapted from [55]). *Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA and let $Pow(\mathcal{N}) = (Q^*, s^*, \delta^*, F^*)$ be the powerset automaton obtained from $\mathcal{N}$. Let $n = |Q|$ and $p = width(\mathcal{N})$. Then, the powerset construction algorithm runs in $O(2^p(n - p + 1)n^2\sigma)$ time.*

Proof. By Theorem 3.2, we know that $N = 2^p(n - p + 1)$ is an upper bound to the number of states of the equivalent DFA. Each state in $Q^*$ consists of $k \leq n$ states $u_1, \ldots, u_k$ of $Q$. For each character $a \in \Sigma$, we need to follow all edges labeled $a$ leaving $u_1, \ldots, u_k$. In the worst case (a complete transition function), this leads to traversing $O(k \cdot n) \subseteq O(n^2)$ edges of the NFA. The final complexity is thus $O(N \cdot n^2 \cdot \sigma)$. □

Corollary 3.5. *We can check the equivalence between two NFAs over an alphabet of size $\sigma$, both with number of states at most $n$ and width at most $p$, in $O(2^p(n - p + 1)n^2\sigma)$ time.*

Proof. First, build the powerset automata, both having at most $N = 2^p(n - p + 1)$ states by Theorem 3.2. This takes $O(Nn^2\sigma)$ time by Lemma 3.4. Finally, DFA equivalence can be tested in $O(N\sigma \log N)$ time by DFA minimization using Hopcroft's algorithm. □

Similarly, the powerset construction can be used to test membership of a word of length $m$ in a regular language expressed as an NFA. When $m$ is much larger than $n$ and $2^p$, this simple analysis of a classical method yields a faster algorithm than the state-of-the-art solution by Thorup and Bille, running in time $O(m \cdot e \cdot \log\log m/(\log m)^{3/2} + m + e)$ [14] where $e$ is the NFA's (equivalently, the regular expression's) size:

Corollary 3.6. *We can test membership of a word of length $m$ in the language recognized by an NFA with $n$ states and co-lex width $p$ on alphabet of size $\sigma$ in $O(2^p(n - p + 1)n^2\sigma + m)$ time.*

## 4 THE DETERMINISTIC WIDTH OF A REGULAR LANGUAGE

In this section, we consider problems related to the notion of width in the deterministic case. In Corollary 4.5 we shall prove that (the NFA width) Problem 1 is polynomial for the case of DFAs. This motivates the search for an efficient algorithm for (the language width) Problem 2. Surprisingly, we show that also this problem admits an efficient (polynomial) solution for any constant width. Given a DFA, we then consider (the Minimum-width DFA) Problem 3. In particular, we prove

that the requests of minimizing the width and the number of states are conflicting: the minimum DFA recognizing a regular language will not, in general, have a minimum width, and among the automata of minimum width recognizing a language there can be non-isomorphic automata with minimum number of states. We then propose two different solutions to Problem 4 (Automata-free characterization for languages of deterministic width $p$): in Corollary 4.22 we consider the behavior of monotone sequences in $(\mathrm{Pref}(\mathcal{L}), \preceq)$, while in Theorem 4.37 we present a Myhill-Nerode result for languages of fixed deterministic width.

The notion of width of an automaton is simpler if we restrict our attention to the class of DFAs. We already proved in Section 2.2 that any DFA admits a (unique) maximum co-lex order $\leq_{\mathcal{D}}$ realizing the width of the automaton $\mathcal{D}$. We shall now prove that the order $\leq_{\mathcal{D}}$ is polynomially computable and so is its width. We start with a characterization of $\leq_{\mathcal{D}}$ in terms of graph reachability.

*Definition 4.1.* We say that a pair $(u', v') \in Q \times Q$ precedes a pair $(u, v) \in Q \times Q$ if $u' \neq v', u \neq v$ and there exists $\alpha \in \Sigma^*$ such that $\delta(u', \alpha) = u, \delta(v', \alpha) = v$.

Lemma 4.2. *Let $\mathcal{D}$ be a DFA and let $u, v \in Q$, with $u \neq v$. Then:*

$$u <_{\mathcal{D}} v \Leftrightarrow \textit{for all pairs } (u', v') \textit{ preceding } (u, v) \textit{ it holds } max_{\lambda(u')} \leq min_{\lambda(v')}.$$

Proof. ($\Rightarrow$) Suppose $u <_{\mathcal{D}} v$. Let $(u', v')$ be a pair preceding $(u, v)$ and let $\gamma \in \Sigma^*$ be such that $\delta(u', \gamma) = u$ and $\delta(v', \gamma) = v$. We must prove that $\max_{\lambda(u')} \leq \min_{\lambda(v')}$. First, notice that it cannot be $v' = s$, otherwise, given $\alpha \in I_{u'}$, we would have $\alpha\gamma \in I_u$ and $\gamma \in I_v$, which contradicts $u <_{\mathcal{D}} v$. So we are only left with proving that if $u' = \delta(u'', e)$ and $v' = \delta(v'', e')$, with $e, e' \in \Sigma$, then $e \leq e'$. Let $\alpha'' \in I_{u''}$ and $\beta'' \in I_{v''}$. Then $\alpha''e\gamma \in I_u, \beta''e'\gamma \in I_v$ and from $u <_{\mathcal{D}} v$ it follows that $\alpha''e\gamma \prec \beta''e'\gamma$, which implies $e \leq e'$.

($\Leftarrow$) Suppose that for all pairs $(u', v')$ preceding $(u, v)$ it holds $\max_{\lambda(u')} \leq \min_{\lambda(v')}$. Let $\alpha \in I_u$ and $\beta \in I_v$. We must prove that $\alpha \prec \beta$. Since $u \neq v$, then $\alpha \neq \beta$. Write $\alpha = \alpha'\gamma$ and $\beta = \beta'\gamma$, where $\alpha'$ and $\beta'$ end with a distinct letter (or, possibly, exactly one of them is equal to the empty string). Let $u', v' \in Q$ be such that $\alpha' \in I_{u'}$ and $\beta' \in I_{v'}$. Then, $(u', v')$ precedes $(u, v)$, so it must be $\max_{\lambda(u')} \leq \min_{\lambda(v')}$. This implies that $v' \neq s$, so $\beta$ is not a suffix of $\alpha$. If $\alpha$ is a suffix of $\beta$, we are done. Otherwise, it must be $\alpha' = \alpha''a$ and $\beta' = \beta''b$, with $a, b \in \Sigma$, $a \neq b$; from $\max_{\lambda(u')} \leq \min_{\lambda(v')}$ it then follows $a \prec b$, which implies $\alpha \prec \beta$. □

Using the previous lemma we are now able to describe a polynomial time algorithm for computing $<_{\mathcal{D}}$. Let $G = (V, F)$ where $V = \{(u, v) \in Q \times Q \mid u \neq v\}$ and $F = \{((u', v'), (u, v)) \in V \times V \mid (\exists e \in \Sigma)(\delta(u', e) = u \wedge \delta(v', e) = v)\}$, where $|F| \leq |\delta|^2$. Intuitively, we will use $G$ to propagate the complement $\not<_{\mathcal{D}}$ of $<_{\mathcal{D}}$. First, mark all nodes $(u, v)$ of $G$ for which $\max_{\lambda(u)} \leq \min_{\lambda(v)}$ does not hold. This process takes $O(|Q|^2)$ time: for any state $u$ we find the minimum and the maximum of $\lambda(u)$ by scanning the transitions of the automaton (total time $O(|\delta|)$); then we decide in constant time when $\max_{\lambda(u)} \leq \min_{\lambda(v)}$ does not hold. Then, mark all nodes reachable on $G$ from marked nodes. This can be done with a simple DFS visit of $G$, initiating the stack with all marked nodes. This process takes $O(|\delta|^2)$ time. By Lemma 4.2, the set of unmarked pairs is $<_{\mathcal{D}}$. Hence, we proved:

Lemma 4.3. *Let $\mathcal{D} = (Q, s, \delta, F)$ be a DFA. We can find the order $\leq_{\mathcal{D}}$ in $O(|\delta|^2)$ time.*

It follows that also the width of a DFA is computable in polynomial time: by the following lemma from [60], the width of a partial order $\leq$ (and an associated $\leq$-chain partition) can always be found in polynomial time from $\leq$. In the following results, *with high probability* means with success probability at least $1 - N^{-c}$, where $N$ is the size of the input and $c$ is a user-defined constant.

Lemma 4.4 ([60]). *Let $(V, \leq)$ be a partial order. A smallest $\leq$-chain partition of $(V, \leq)$ and its width can be found in $\tilde{O}(|V|^2)$ time, with high probability.*

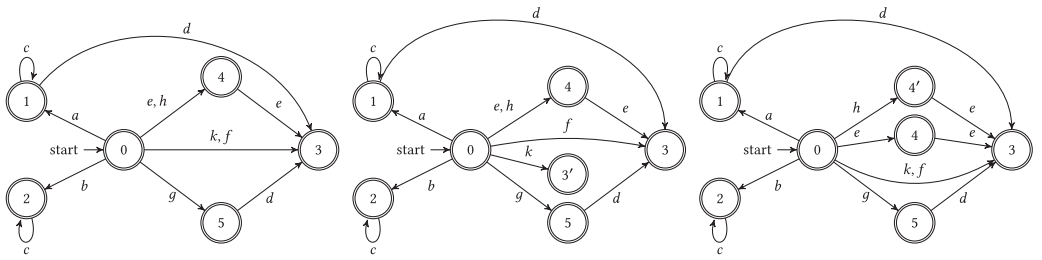



Fig. 3. Three DFAs recognizing the same language.

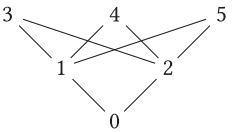



Fig. 4. The Hasse diagram of $\leq_{\mathcal{D}_1}$.

Proof. In [60, Theorem 1.6], it is shown how to compute a minimum chain partition of a DAG with $n$ vertices and $m$ edges in $\tilde{O}(m + n^{3/2})$ time with high probability. The claim follows running this algorithm on the DAG corresponding to $(V, \leq)$, having $|V|$ nodes and $O(|V|^2)$ edges. □

Corollary 4.5. *Let $\mathcal{D} = (Q, s, \delta, F)$ be a DFA. A co-lex order $\leq$ of width equal to width$(\mathcal{D})$ and a corresponding $\leq$-chain partition of cardinality equal to width$(\mathcal{D})$ can be computed in $\tilde{O}(|\delta|^2)$ time with high probability.*

The previous corollary solves Problem 1 for DFAs.[5] As for the width-complexity over NFAs, it is known that the problem is NP-hard, since already deciding whether the width of an NFA is equal to 1 (i.e., deciding whether the NFA is Wheeler) is an NP-complete problem (see [53]).

We have proved that computing the width of a DFA is an "easy" problem. We now move to the natural problem of computing the deterministic width of a regular language, presented by means of a DFA recognizing it.

It would have been nice to have the deterministic width of a language equal to the width of its minimum DFA: if this were true, we could use Corollary 4.5 to determine $\text{width}^D(\mathcal{L})$ in polynomial time with high probability by inspecting its minimum-size accepting automaton. As we show in Example 4.6, unfortunately this is not the case. Moreover, there is, in general, no unique (up to isomorphism) minimum DFA of minimum width recognizing a given regular language.

*Example 4.6.* In Figure 3, three DFAs recognizing the same language $\mathcal{L}$ are shown. We prove that $\text{width}^D(\mathcal{L}) = 2$, the width of the minimum DFA for $\mathcal{L}$ is 3, and there is not a unique minimum automaton among all DFAs recognizing $\mathcal{L}$ of width 2. Consider the DFA $\mathcal{D}_1$ on the left of Figure 3 and let $\mathcal{L} = \mathcal{L}(\mathcal{D}_1)$. The automaton $\mathcal{D}_1$ is a minimum DFA for $\mathcal{L}$. The states $0, \ldots, 5$ are such that: $I_0 = \{\varepsilon\}$, $I_1 = ac^*$, $I_2 = bc^*$, $I_3 = ac^*d \cup \{gd, ee, he, f, k\}$, $I_4 = \{e, h\}$, $I_5 = \{g\}$. States 1 and 2 are $\leq_{\mathcal{D}_1}$-incomparable because $a \in I_1, b \in I_2, ac \in I_1$ and $a \prec b \prec ac$. Similarly, one checks that states 3,4,5 are pairwise $\leq_{\mathcal{D}_1}$-incomparable. On the other hand, 0 is the minimum and states 1,2 precede states 3, 4, 5 in the order $\leq_{\mathcal{D}_1}$. We conclude that the Hasse diagram of the partial order $\leq_{\mathcal{D}_1}$ is the one depicted in Figure 4.

[5]We mention that the bound of Corollary 4.5 has very recently been improved to $O(\min(|Q|^2, |\delta| \log |Q|))$ in two subsequent works [9, 30].

The width of the DFA $\mathcal{D}_1$ is 3 because $\{3, 4, 5\}$ is a largest $\leq_{\mathcal{D}_1}$-antichain. A $\leq_{\mathcal{D}_1}$-chain partition of cardinality 3 is, for example, $\{\{0, 1, 3\}, \{2, 4\}, \{5\}\}$.

Let us prove that $\text{width}^D(\mathcal{L}) \geq 2$. Suppose by contradiction that there exists a DFA $\mathcal{D}$ of width 1 recognizing $\mathcal{L}$. Then, the order $\leq_{\mathcal{D}}$ is total. Moreover, there exists a state $u$ such that two words of the infinite set $ac^* \in \text{Pref}(\mathcal{L})$, say $ac^i, ac^j$ with $i < j$, belong to $I_u$. Consider the word $bc^i \in \text{Pref}(\mathcal{L})$. Since $bc^i \not\equiv_{\mathcal{L}} ac^i$, it follows that $bc^i \notin I_u$. If $u'$ is such that $bc^i \in I_{q'}$, from $ac^i \prec bc^i \prec ac^j$ we have that $u$ and $u'$ are $\leq_{\mathcal{D}}$-incomparable, a contradiction.

Finally, let $\mathcal{D}_2$ be the DFA in the center of Figure 3 and let $\mathcal{D}_3$ be the DFA on the right of Figure 3. Notice that $\mathcal{L}(\mathcal{D}_2) = \mathcal{L}(\mathcal{D}_3) = \mathcal{L}$, and $\mathcal{D}_2$ and $\mathcal{D}_3$ have just one more state than $\mathcal{D}_1$ and are non-isomorphic. We know that $\mathcal{D}_2$ and $\mathcal{D}_3$ cannot have width equal to 1. On the other hand, they both have width 2, as witnessed by the chains $\{\{0, 1, 4\}, \{2, 3, 5, 3'\}\}$ (for $\mathcal{D}_2$) and $\{\{0, 1, 3\}, \{2, 4, 5, 4'\}\}$ (for $\mathcal{D}_3$).

Motivated by the fact that computing the deterministic width of a regular language is not a trivial problem, in the next subsections we develop a set of tools that will ultimately allow us to derive an algorithm solving the problem.

Example 4.6 triggers a further natural and important observation. It is known that languages with deterministic width equal to 1 (that is, Wheeler languages) admit a (unique) minimum-size Wheeler DFA [2]. Note that Example 4.6 implies that no such minimality result holds true for higher levels of the deterministic width hierarchy. In Section 4.5 we will explain why Example 4.6 is not the end of the story and we will derive an adequate notion of minimality.

### 4.1 The Entanglement of a Regular Language

We now exhibit a measure that on the minimum DFA will capture exactly the width of the accepted language: the *entanglement number* of a DFA. We shall use the following terminology: if $\mathcal{D}$ is a DFA and $V \subseteq \text{Pref}(\mathcal{L}(\mathcal{D}))$, then a state $u$ *occurs* in $V$ if there exists $\alpha \in V$ such that $u = \delta(s, \alpha)$.

*Definition 4.7.* Let $\mathcal{D}$ be a DFA with set of states $Q$.

(1) 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) $Q'$.
(2) A set $V \subseteq \text{Pref}(\mathcal{L}(\mathcal{D}))$ is *entangled in* $\mathcal{D}$ if there exists a monotone sequence $(\alpha_i)_{i\in\mathbb{N}}$, with $\alpha_i \in V$ for every $i$, witnessing that the set $\{\delta(s, \alpha) \mid \alpha \in V\}$, consisting of all states occurring in $V$, is entangled.

Moreover, define:

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

Notice that any singleton $\{u\} \subseteq Q$ is entangled, as witnessed by the trivially monotone sequence $(\alpha_i)_{i\in\mathbb{N}}$ where all the $\alpha_i$'s are equal and $\delta(s, \alpha_i) = u$.

As an example consider the entanglement of all DFAs in Figure 3. For any of them the entanglement is two, because the only entangled subset of states is $\{1, 2\}$, as witnessed by the sequence $a \prec b \prec ac \prec bc \prec acc \prec bcc \prec \cdots$.

When two states $u \neq u'$ of a DFA $\mathcal{D}$ belong to an entangled set, there are words $\alpha \prec \beta \prec \alpha'$ such that $\alpha, \alpha' \in I_u, \beta \in I_{u'}$, so that neither $u <_{\mathcal{D}} u'$ nor $u' <_{\mathcal{D}} u$ can hold. In other words, two distinct states $u, u'$ belonging to an entangled set are always $\leq_{\mathcal{D}}$-incomparable. Since by Lemma 2.11, we have $\text{width}(\leq_{\mathcal{D}}) = \text{width}(\mathcal{D})$, it easily follows that the entanglement of a DFA is always smaller than or equal to its width.

Lemma 4.8. *Let $\mathcal{D}$ be a DFA. Then $ent(\mathcal{D}) \leq width(\mathcal{D})$.*

The converse of the above inequality is not always true: for the (minimum) DFA $\mathcal{D}_1$ on the left of Figure 3 we have $\mathrm{ent}(\mathcal{D}_1) = 2$ and $\mathrm{width}(\mathcal{D}_1) = 3$.

Contrary to what happens with the width, we now prove that the entanglement of a regular language is realized by the minimum-size automaton accepting it.

Lemma 4.9. *If $\mathcal{D}_{\mathcal{L}}$ is the minimum-size DFA recognizing $\mathcal{L}$, then $ent(\mathcal{D}_{\mathcal{L}}) = ent(\mathcal{L})$.*

Proof. It is enough to prove that $\mathrm{ent}(\mathcal{D}_{\mathcal{L}}) \leq \mathrm{ent}(\mathcal{D})$, for any DFA $\mathcal{D}$ such that $\mathcal{L}(\mathcal{D}_{\mathcal{L}}) = \mathcal{L}(\mathcal{D})$. Suppose $u_1, \ldots, u_k$ are pairwise distinct states which are entangled in $\mathcal{D}_{\mathcal{L}}$, witnessed by the monotone sequence $(\alpha_i)_{i\in\mathbb{N}}$. Since $\mathcal{D}_{\mathcal{L}}$ is minimum, each $I_{u_j}$ is a union of a finite number of $I_v$, with $v \in Q_{\mathcal{D}}$. The monotone sequence $(\alpha_i)_{i\in\mathbb{N}}$ goes through $u_j$ infinitely often, so there must be a state $v_j \in Q_{\mathcal{D}}$ such that $I_{v_j} \subseteq I_{u_j}$ and $(\alpha_i)_{i\in\mathbb{N}}$ goes through $v_j$ infinitely often. Then $(\alpha_i)_{i\in\mathbb{N}}$ goes through the pairwise distinct states $v_1, \ldots, v_k$ infinitely often and $v_1, \ldots, v_k$ are entangled in $\mathcal{D}$. □

### 4.2 The Hasse Automaton of a Regular Language

Our aim is at proving that the entanglement measure over the minimum DFA $\mathcal{D}_{\mathcal{L}}$ captures the deterministic width of a language $\mathcal{L}$:

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

In order to prove the previous equality we shall describe an automaton, the *Hasse automaton* of $\mathcal{L}$, realizing the width and the entanglement of the language as its width (Theorem 4.21). As a first step, given a DFA $\mathcal{D}$ we prove that there exists an equivalent DFA $\mathcal{D}'$ that realizes the entanglement of $\mathcal{D}$ as its width: $\mathrm{ent}(\mathcal{D}) = \mathrm{width}(\mathcal{D}')$ (Theorem 4.19).

To give an intuition on the construction of the automaton $\mathcal{D}'$ we use the *trace* of the DFA $\mathcal{D}$, that is, the (in general) transfinite sequence: $(\delta(s, \alpha))_{\alpha\in\mathrm{Pref}(\mathcal{L})}$, indexed over the totally ordered set $(\mathrm{Pref}(\mathcal{L}), \preceq)$, where $\mathcal{L} = \mathcal{L}(\mathcal{D})$. We depict below a hypothetical $(\mathrm{Pref}(\mathcal{L}), \preceq)$, together with the trace left by a DFA $\mathcal{D}$ with set of states $\{u_1, u_2, u_3\}$ and $\delta(s, \alpha_i) = \delta(s, \alpha') = u_1$, $\delta(s, \beta_i) = \delta(s, \beta'_i) = u_2$, and $\delta(s, \gamma_i) = u_3$:

$$\begin{array}{ccccccccccccccccccccccccc}
\alpha_1 & \prec & \beta_1 & \prec & \alpha_2 & \prec & \beta_2 & \prec & \cdots & \prec & \alpha_i & \prec & \beta_i & \prec & \cdots & \prec & \beta'_1 & \prec & \gamma_1 & \prec & \beta'_2 & \prec & \gamma_2 & \prec & \cdots & \prec & \beta'_i & \prec & \gamma_i & \prec & \cdots & \prec & \alpha' \\
u_1 & & u_2 & & u_1 & & u_2 & & \ldots & & u_1 & & u_2 & & \ldots & & u_2 & & u_3 & & u_2 & & u_3 & & \ldots & & u_2 & & u_3 & & \ldots & & u_1
\end{array}$$

Consider the entanglement and width of $\mathcal{D}$. Notice that the sets $\{u_1, u_2\}$ and $\{u_2, u_3\}$ are entangled. The set $\{u_1, u_3\}$ is not entangled and therefore the set $\{u_1, u_2, u_3\}$ is not entangled. However, $\{u_1, u_2, u_3\}$ contains pairwise incomparable states. Hence the whole triplet $\{u_1, u_2, u_3\}$ does not contribute to the entanglement but does contribute to the width so that $\mathrm{ent}(\mathcal{D}) = 2 < \mathrm{width}(\mathcal{D}) = 3$.

In general, an automaton where incomparability and entanglement coincide would have $\mathrm{ent}(\mathcal{D}) = \mathrm{width}(\mathcal{D})$. Hence, we would like to force *all* sets of incomparable states in the new automaton $\mathcal{D}'$ to be entangled. To this end, we will first prove that there always exists a finite, ordered partition $\mathcal{V} = \{V_1, \ldots, V_r\}$ of $\mathrm{Pref}(\mathcal{L})$ composed of convex sets which are entangled in $\mathcal{D}$. In the example above we can write $\mathrm{Pref}(\mathcal{L}) = V_1 \cup V_2 \cup V_3$, where:

$$V_1 = \{\alpha_1, \beta_1, \ldots, \alpha_i, \beta_i, \ldots\}, V_2 = \{\beta'_1, \gamma_1, \ldots, \beta'_i, \gamma_i \ldots\}, V_3 = \{\alpha'\},$$

and the states occurring (and entangled) in $V_1, V_2, V_3$ are, respectively: $\{u_1, u_2\}$, $\{u_2, u_3\}$, and $\{u_1\}$. In order to construct an equivalent automaton $\mathcal{D}'$ in which the pairwise incomparability of the three states $u_1, u_2, u_3$ is eliminated and $\mathrm{width}(\mathcal{D}') = 2$, we could try to *duplicate* some of the original states, as it would be the case if the states occurring in $V_1, V_2, V_3$ where, respectively, $\{u_1, u_2\}$, $\{u_2, u_3\}$, and $\{u'_1\}$. To this end, we will consider a refinement $\sim$ of the Myhill-Nerode equivalence on $\mathrm{Pref}(\mathcal{L})$ stating that two strings are equivalent if and only if they are in the same $I_u$ and all $V \in \mathcal{V}$

laying between the two strings intersect $I_u$. In the above example we have $\beta_i \sim \beta'_j$ for all integers $i, j$, because no $V \in \mathcal{V}$ is contained in $[\beta_i, \beta'_j]$, while $\alpha_1 \nsim \alpha'$, since $V_2 \subseteq [\alpha_1, \alpha']$ but $V_2 \cap I_{u_1} = \emptyset$.

We will prove that the equivalence $\sim$ decomposes the set of words reaching $u$ into a *finite* number of $\sim$-classes and induces a well-defined quotient automaton $\mathcal{D}'$ equivalent to $\mathcal{D}$.

By construction, in the automaton $\mathcal{D}'$ any set of $\prec_{\mathcal{D}'}$-incomparable states $\{u_1, \ldots, u_k\}$ will occur in at least an element $V \in \mathcal{V}$, so that, $V$ being entangled, they will contribute to the entanglement number of $\mathcal{D}$ and width$(\mathcal{D}')$ = ent$(\mathcal{D})$ will follow.

In our example, the new automaton $\mathcal{D}'$ will leave the following trace:

$$\begin{array}{ccccccccccccccccccccccccc}
\alpha_1 & \prec & \beta_1 & \prec & \alpha_2 & \prec & \beta_2 & \prec & \cdots & \prec & \alpha_i & \prec & \beta_i & \prec & \cdots & \prec & \beta'_1 & \prec & \gamma_1 & \prec & \beta'_2 & \prec & \gamma_2 & \prec & \cdots & \prec & \beta'_i & \prec & \gamma_i & \prec & \cdots & \prec & \alpha' \\
u_1 & & u_2 & & u_1 & & u_2 & & \ldots & & u_1 & & u_2 & & \ldots & & u_2 & & u_3 & & u_2 & & u_3 & & \ldots & & u_2 & & u_3 & & \ldots & & u'_1
\end{array}$$

and ent$(\mathcal{D})$ = width$(\mathcal{D}')$ = 2.

In order to give a formal definition of $\mathcal{D}'$ we need some properties of entangled sets. Since these properties hold true with respect to a partition of a generic total order, we state and prove them in a more general setting in Appendix A, while we state them without proof in this section. In particular, most of the properties of $\mathcal{D}'$ will be proved using a finite partition of $(\text{Pref}(\mathcal{L}(\mathcal{D})), \preceq)$, composed by entangled, convex sets.

*Definition 4.10.* If $\mathcal{D}$ is a DFA, we say that a partition $\mathcal{V}$ of Pref$(\mathcal{L}(\mathcal{D}))$ is an *entangled, convex decomposition of* $\mathcal{D}$ (*e.c. decomposition*, for short) if all the elements of $\mathcal{V}$ are convex in $(\text{Pref}(\mathcal{L}(\mathcal{D})), \preceq)$ and entangled in $\mathcal{D}$.

Theorem 4.11. *If $\mathcal{D}$ is a DFA, then there exists a finite partition $\mathcal{V}$ of Pref$(\mathcal{L}(\mathcal{D}))$ which is an e.c. decomposition of $\mathcal{D}$.*

Proof. The existence of such a decomposition is guaranteed by Theorem A.5 of Appendix A, applied to $(Z, \leq) = (\text{Pref}(\mathcal{L}(\mathcal{D})), \preceq)$ and $\mathcal{P} = \{I_u \mid u \in Q\}$. □

Using an e.c. decomposition of an automaton $\mathcal{D}$ we can express a condition implying the equality width$(\mathcal{D})$ = ent$(\mathcal{D})$.

Lemma 4.12. *Let $\mathcal{D}$ be a DFA. Suppose $V_1 \prec V_2 \prec \cdots \prec V_m$ is an e.c. decomposition of $\mathcal{D}$ such that for every $u \in Q$ there are $1 \leq i \leq j \leq m$ with:*

$$I_u \subseteq V_i \cup V_{i+1} \cup \cdots \cup V_j, \quad \text{and} \quad V_h \cap I_u \neq \emptyset, \ \text{for all} \ \ i \leq h \leq j.$$

*Then, width$(\mathcal{D})$ = ent$(\mathcal{D})$.*

Proof. We already proved that ent$(\mathcal{D}) \leq$ width$(\mathcal{D})$ in Lemma 4.8. In order to prove the reverse inequality, let width$(\mathcal{D}) = p$ and let $u_1, \ldots, u_p$ be $p$ $\preceq_{\mathcal{D}}$-incomparable states. If we prove that $u_1, \ldots, u_p$ are entangled, we are done. Fix $i \in \{1, \ldots, p\}$ and denote by $W_i$ the convex set $V_h \cup V_{h+1} \cup \cdots \cup V_k$ where $I_{u_i} \subseteq V_h \cup V_{h+1} \cup \cdots \cup V_k$ and $V_j \cap I_{u_i} \neq \emptyset$, for all $h \leq j \leq k$. From the incomparability of the $u_i$'s it follows that $W_i \cap W_j \neq \emptyset$, for all pairs $i, j$, so that $\bigcap_i W_i \neq \emptyset$ by Lemma A.13 in Appendix A. Since all $W_i$'s are unions of consecutive elements in the partition $\mathcal{V}$, there must be an element $V \in \mathcal{V}$ with $V \subseteq \bigcap_i W_i$. Such a $V$ must contain an occurrence of every $u_i$, and since $V$ is an entangled set, we conclude that $u_1, \ldots, u_p$ are entangled. □

The previous lemma suggests that in order to construct an automaton $\mathcal{D}'$ recognizing the same language as $\mathcal{D}$ and satisfying width$(\mathcal{D}')$ = ent$(\mathcal{D}')$ = ent$(\mathcal{D})$, we might *duplicate* some states in $\mathcal{D}$ in order to ensure that the new automaton $\mathcal{D}'$ satisfies the condition of the previous Lemma. Consider two words $\alpha \prec \alpha'$ reaching the same state $u$ of $\mathcal{D}$: if the convex $[\alpha, \alpha']$ is not contained in a union of consecutive elements of the partition $\mathcal{V}$, all having an occurrence of $u$, then in $\mathcal{D}'$ we duplicate the state $u$ into $u$ and $u'$, with $\alpha$ reaching $u$ and $\alpha'$ reaching $u'$ (see Figure 5).

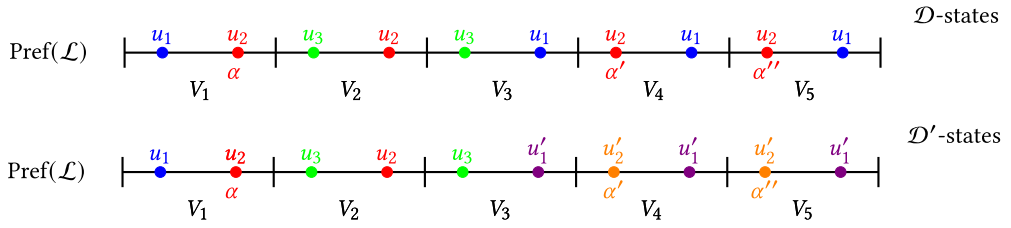



Fig. 5. The upper line represents the words in Pref($\mathcal{L}$), partitioned by the e.c. decomposition $\mathcal{V}$, where some words are highlighted and labeled by the state they reach in $\mathcal{D}$. The lower line still represents Pref($\mathcal{L}$), with the same e.c. decomposition, but now the highlighted words are labeled by states of $\mathcal{D}'$. Note that the strings $\alpha, \alpha'$ reach the same state in $\mathcal{D}$, but in different states in $\mathcal{D}'$, because $V_3 \subseteq [\alpha, \alpha']$, and $V_3 \cap I_{u_2} = \emptyset$. However, the words $\alpha', \alpha''$ reach the same state in both automata.

As usual, an equivalence relation $\sim_{\mathcal{D}}$ over Pref($\mathcal{L}$) is used to introduce the new states of the automaton $\mathcal{D}'$. In order to maintain the definition of $\mathcal{D}'$ independent from any particular e.c. decomposition $\mathcal{V}$, in the definition below we use generic entangled convex sets instead of elements of an e.c. decomposition. In Lemma 4.17 we prove that this is equivalent to using elements of an e.c. decomposition of minimum cardinality.

*Definition 4.13.* Let $\mathcal{D}$ be a DFA and let $\sim_{\mathcal{D}}$ be the equivalence relation on Pref($\mathcal{L}(D)$) defined as follows: $\alpha \sim_{\mathcal{D}} \alpha'$ if and only if:

- $\delta(s, \alpha) = \delta(s, \alpha')$ and
- there are entangled convex sets $C_1, \dots, C_n \subseteq \text{Pref}(\mathcal{L}(\mathcal{D}))$ such that:
  - $[\alpha, \alpha']^{\pm} \subseteq \bigcup_{i=1}^{n} C_i$;
  - $C_i \cap I_{\delta(s,\alpha)} \neq \emptyset$, for all $i \in \{1, \dots, n\}$.

Note that $\sim_{\mathcal{D}}$ is indeed an equivalence relation (in particular, it is transitive). When the DFA $\mathcal{D}$ is clear from the context, we shall drop the subscript $\mathcal{D}$ in $\sim_{\mathcal{D}}$.

In the following lemma we prove that the equivalence $\sim$ has finite index and $\mathcal{L}(\mathcal{D})$ is equal to the union of some of its classes.

Lemma 4.14. *Let $\mathcal{D}$ be a DFA. Then, $\sim$ has a finite number of classes on Pref($\mathcal{L}(\mathcal{D})$) and $\mathcal{L}(\mathcal{D})$ is equal to the union of some $\sim$-classes.*

Proof. Consider an e.c. decomposition $\mathcal{V} = \{V_1, \dots, V_m\}$ of $\mathcal{D}$, whose existence is guaranteed by Theorem 4.11. Since all $V_i$'s are entangled convex sets in Pref($\mathcal{L}(\mathcal{D})$), two words belonging to the same $V_i$ and ending in the same state belong to the same $\sim$-class. Hence, the number of $\sim$-classes is at most $m \times |Q|$. Moreover, $\alpha \sim \beta$ implies $\delta(s, \alpha) = \delta(s, \beta)$. Hence, $\alpha \in \mathcal{L}(\mathcal{D})$ and $\alpha \sim \beta$ imply $\beta \in \mathcal{L}(\mathcal{D})$, proving that $\mathcal{L}(\mathcal{D})$ is equal to the union of some $\sim$-classes. □

Lemma 4.15. *Let $\mathcal{D}$ be a DFA. Then, the equivalence relation $\sim$ is right-invariant.*

Proof. Let $\mathcal{L} = \mathcal{L}(\mathcal{D})$, assume $\alpha \sim \alpha'$ and let $a \in \Sigma$ be such that $\alpha a \in \text{Pref}(\mathcal{L})$. We must prove that $\alpha' a \in \text{Pref}(\mathcal{L})$ and $\alpha a \sim \alpha' a$. Since $\alpha \sim \alpha'$ we know that $\delta(s, \alpha) = \delta(s, \alpha')$ and there exist entangled convex sets $C_1, \dots, C_n$ such that $[\alpha, \alpha']^{\pm} \subseteq C_1 \cup \dots \cup C_n$ and $C_i \cap I_{\delta(s,\alpha)} \neq \emptyset$ for all $i = 1, \dots, n$. We must prove that $\alpha' a \in \text{Pref}(\mathcal{L})$, $\delta(s, \alpha a) = \delta(s, \alpha' a)$, and there exist entangled convex sets $C'_1, \dots, C'_{n'}$ such that $[\alpha a, \alpha' a]^{\pm} \subseteq C'_1 \cup \dots \cup C'_{n'}$ and $C'_i \cap I_{\delta(s,\alpha a)} \neq \emptyset$ for all $i = 1, \dots, n'$.

From $\delta(s, \alpha) = \delta(s, \alpha')$ and $\alpha a \in \text{Pref}(\mathcal{L})$ we immediately obtain $\alpha' a \in \text{Pref}(\mathcal{L})$ and $\delta(s, \alpha a) = \delta(s, \alpha' a)$. Moreover, from $[\alpha, \alpha']^{\pm} \subseteq C_1 \cup \dots \cup C_n$ we obtain $[\alpha a, \alpha' a]^{\pm} = [\alpha, \alpha']^{\pm} a \subseteq C_1 a \cup \dots \cup C_n a$, and from $C_i \cap I_{\delta(s,\alpha)} \neq \emptyset$ we obtain $C_i a \cap I_{\delta(s,\alpha a)} \neq \emptyset$. We are only left with showing that every $C_i a = \{\gamma a \mid \gamma \in C_i\}$ is an entangled convex set. The fact that they are convex follows directly

from the definition of co-lex ordering. Let us prove that the $C_i a$'s are entangled. Fix $i$ and consider a monotone sequence $(\alpha_j)_{j\in\mathbb{N}}$ witnessing that $C_i$ is entangled. Then $(\alpha_j a)_{j\in\mathbb{N}}$ is a monotone sequence witnessing that $C_i a$ is entangled. □

We are now ready to complete the construction of the automaton $\mathcal{D}'$ using $\sim$.

*Definition 4.16.* Let $\mathcal{D}$ be a DFA and let $\mathcal{L} = \mathcal{L}(\mathcal{D})$. Define $\mathcal{D}' = (Q', s', \delta', F')$ by:

$-Q' = \{[\alpha]_\sim : \alpha \in \mathrm{Pref}(\mathcal{L})\}$;
$-\delta'([\alpha]_\sim, a) = [\alpha a]_\sim$ for every $\alpha \in \mathrm{Pref}(\mathcal{L})$ and for every $a \in \Sigma$ such that $\alpha a \in \mathrm{Pref}(\mathcal{L})$;
$-s' = [\varepsilon]_\sim$;
$-F' = \{[\alpha]_\sim : \alpha \in \mathcal{L}\}$.

The equivalence relation $\sim$ is right-invariant (Lemma 4.15), has finite index, and $\mathcal{L}$ is the union of some $\sim$-classes (Lemma 4.14). Hence, $\mathcal{D}'$ is a well-defined DFA, and $\alpha \in [\beta]_\sim \iff \delta'(s', \alpha) = [\beta]_\sim$, which implies that for every $\alpha \in \mathrm{Pref}(\mathcal{L})$ it holds:

$$I_{[\alpha]_\sim} = [\alpha]_\sim \tag{5}$$

and so $\mathcal{L}(\mathcal{D}') = \mathcal{L}$.

In the following lemma we prove that it is safe to replace $C_1, \ldots, C_n$ in Definition 4.13 by the elements of a *minimum-size* e.c. decomposition, that is, an e.c. decomposition with minimum cardinality.

Lemma 4.17. *Let $\mathcal{D}$ be a DFA and let $\mathcal{V} = \{V_1, \ldots, V_r\}$, with $V_1 \prec \cdots \prec V_r$, be a minimum-size e.c. decomposition of $\mathcal{D}$. Then, $\alpha \sim \alpha'$ holds if and only if:*

- *$\delta(s, \alpha) = \delta(s, \alpha')$ and*
- *there exist integers $i \le j$ such that:*
  *$-[\alpha, \alpha']^{\pm} \subseteq \bigcup_{h=i}^{j} V_h$;*
  *$-V_h \cap I_{\delta(s,\alpha)} \neq \emptyset$, for all $h \in \{i, \ldots, j\}$.*

Proof. To prove that $\alpha \sim \alpha'$ holds under the above hypotheses, it is sufficient to recall that $V_i, \ldots, V_j$ are entangled convex sets and apply Definition 4.13.

Let us prove the reverse implication. Pick $\alpha \sim \alpha' \in \mathrm{Pref}(\mathcal{L}(\mathcal{D}))$ and let $C_1, \ldots, C_n$ be entangled convex sets such that $[\alpha, \alpha']^{\pm} \subseteq \bigcup_{i=1}^{n} C_i$ and $C_i \cap I_u \neq \emptyset$, for every $i = 1, \ldots, n$, where $u = \delta(s, \alpha) = \delta(s, \alpha')$. By Lemma A.12 of Appendix A we can assume that $C_1 \prec C_2 \prec \cdots \prec C_n$. Let $i \le j$ be such that $[\alpha, \alpha']^{\pm} \subseteq \bigcup_{h=i}^{j} V_h$ and $V_h \cap [\alpha, \alpha']^{\pm} \neq \emptyset$ for all $h \in \{i, \ldots, j\}$. We just have to prove that $V_h \cap I_u \neq \emptyset$, for all $h \in \{i, \ldots, j\}$. From $V_h \cap [\alpha, \alpha']^{\pm} \neq \emptyset$ it follows that either $V_h$ contains $\alpha$ or $\alpha'$, or $V_h \subseteq [\alpha, \alpha']^{\pm} \subseteq \bigcup_{i=1}^{n} C_i$. In the first case we have $V_h \cap I_u \neq \emptyset$ because $\alpha, \alpha' \in I_u$, while in the second case $V_h \cap I_u \neq \emptyset$ follows from Lemma A.9 of Appendix A. □

Let $\mathcal{D}'$ be the automaton of Definition 4.16. The following corollary allows us to consider an e.c. decomposition of $\mathcal{D}$ of minimum size as an e.c. decomposition of $\mathcal{D}'$.

Corollary 4.18. *Any e.c. decomposition of minimum size $V_1 \prec \cdots \prec V_r$ of $\mathcal{D}$ is also an e.c. decomposition of $\mathcal{D}'$. Moreover, for all $u' \in Q'$, there exist $i \le j$ such that $I_{u'} \subseteq \bigcup_{h=i}^{j} V_h$ and $V_h \cap I_{u'} \neq \emptyset$, for $h = i, \ldots, j$.*

Proof. In order to prove that an e.c. decomposition of minimum size $V_1 \prec \cdots \prec V_r$ of $\mathcal{D}$ is also an e.c. decomposition of $\mathcal{D}'$ we just have to check that $V_h$ is entangled in $\mathcal{D}'$, for all $h = 1, \ldots, r$. Let $u'_1, \ldots, u'_k$ be the pairwise distinct $\mathcal{D}'$-states occurring in $V_h$. Notice that by the definition of $\delta'$ we have $\{u'_1, \ldots, u'_k\} = \{\delta'(s', \alpha) \mid \alpha \in V_h\} = \{[\alpha]_\sim \mid \alpha \in V_h\}$. Hence, for every $j = 1, \ldots, k$

there exists $\alpha_j \in V_h$ such that $u'_j = [\alpha_j]_\sim$. Then the $\mathcal{D}$-states $u_j = \delta(s, \alpha_j)$, for $j = 1, \dots, k$, occur in $V_h$. Notice that $u_1, \dots, u_k$ are pairwise distinct as well: if $u_i$ were equal to $u_j$ for $i \neq j$, then from Lemma 4.17 we would have $\alpha_i \sim \alpha_j$ and $u'_i = [\alpha_i]_\sim = [\alpha_j]_\sim = u'_j$ would follow. Since $V_h$ is entangled in $\mathcal{D}$, there exists a monotone sequence $(\beta_i)_{i\in\mathbb{N}}$ in $V_h$ reaching each $u_j$ infinitely many times. Fix $j \in \{1, \dots, k\}$. If $\delta(s, \beta_i) = u_j$ then from $\delta(s, \alpha_j) = u_j$ and $\beta_i, \alpha_j \in V_h$ it follows $\alpha_j \sim \beta_i$ again by Lemma 4.17, so that $\delta'(s', \beta_i) = [\beta_i]_\sim = [\alpha_j]_\sim = u'_j$ in $\mathcal{D}'$. It follows that the sequence $(\beta_i)_{i\in\mathbb{N}}$ reaches $u'_j$ infinitely many times. Hence, $V_h$ is entangled in $\mathcal{D}'$.

As for the second part of the Corollary, if $u' = [\alpha]_\sim \in Q'$ then $I_{u'} = [\alpha]_\sim$ (see Equation (5) above). Let $i$ ($j$, respectively) be the minimum (maximum) index $h$ with $[\alpha]_\sim \cap V_h \neq \emptyset$; then $[\alpha]_\sim \subseteq V_i \cup \dots \cup V_j$. Fix $h \in \{i, \dots, j\}$ and consider $u = \delta(s, \alpha)$. Since there exist $\alpha', \alpha'' \in \mathrm{Pref}(\mathcal{L}(\mathcal{D}))$ such that $\alpha' \sim \alpha \sim \alpha''$, $\alpha' \in V_i$ and $\alpha'' \in V_j$, then, Lemma 4.17 implies $\delta(s, \alpha') = \delta(s, \alpha'') = \delta(s, \alpha) = u$ and $V_h \cap I_u \neq \emptyset$. Pick $\beta \in V_h \cap I_u$. Then, the same lemma implies $\beta \sim \alpha$ so that $\beta \in I_{u'}$ and $V_h \cap I_{u'} \neq \emptyset$ as well. □

We can now prove that $\mathcal{D}'$ has width equal (to its entanglement and) to the entanglement of $\mathcal{D}$.

Theorem 4.19. *If $\mathcal{D}$ is a DFA, then $ent(\mathcal{D}) = ent(\mathcal{D}') = width(\mathcal{D}')$.*

Proof. Let us prove that $\mathrm{ent}(\mathcal{D}') \leq \mathrm{ent}(\mathcal{D})$. Consider an entangled collection $\{[\alpha_1]_\sim, \dots, [\alpha_h]_\sim\}$ of $h$ states in $\mathcal{D}'$. Then, there is a monotone sequence $(\gamma_i)_{i\in\mathbb{N}}$ such that, for each $j \in \{1, \dots, h\}$ we have $\delta'(s', \gamma_i) = [\alpha_j]_\sim$ for infinitely many $i$'s. Let $\mathcal{V}$ be a minimum-size finite e.c. decomposition of $\mathcal{D}$. Since $\mathcal{V}$ is a finite partition and all the elements of $\mathcal{V}$ are convex, there exists $V \in \mathcal{V}$ and $n_0$ such that $\gamma_i \in V$ for all $i \geq n_0$. In particular, there are words $\beta_1, \dots, \beta_h$ in $V$ such that $\delta'(s', \beta_k) = [\alpha_k]_\sim$, for every $k = 1, \dots, h$. Define $u_k = \delta(s, \beta_k)$ and notice that the states $u_1, \dots, u_h$ are pairwise distinct. In fact, if $u_r = u_s$ for $r \neq s$, then by Lemma 4.17 we would have $\beta_r \sim \beta_s$ and $[\alpha_r]_\sim = \delta'(s', \beta_r) = [\beta_r]_\sim = [\beta_s]_\sim = \delta'(s', \beta_s) = [\alpha_s]_\sim$, a contradiction. Moreover, $\{u_1, \dots, u_h\}$ is an entangled set in $\mathcal{D}$, because all these states occur in $V$ (as witnessed by $\beta_1, \dots, \beta_h$) and $V$ is an element of an e.c. decomposition. Since this holds for any collection of entangled states in $\mathcal{D}'$, it follows that $\mathrm{ent}(\mathcal{D}') \leq \mathrm{ent}(\mathcal{D})$.

Let us now prove that $\mathrm{ent}(\mathcal{D}) \leq \mathrm{ent}(\mathcal{D}')$. Let $\{u_1, \dots, u_h\}$ be an entangled set of $h$ states in $\mathcal{D}$, witnessed by some monotone sequence $(\alpha_i)_{i\in\mathbb{N}}$. Every $I_{u_k}$ is equal to a finite union of some $I_{u'}$'s, with $u' \in Q'$, and $(\alpha_i)_{i\in\mathbb{N}}$ goes through any $u_k$ infinitely many times. Therefore, for all $k = 1, \dots h$ there exist $u'_k \in Q'$ such that $I_{u'_k} \subseteq I_{u_k}$ and $(\alpha_i)_{i\in\mathbb{N}}$ goes through $u'_k$ infinitely many times. We conclude that $u'_1, \dots, u'_h$ are pairwise distinct and $\{u'_1, \dots, u'_h\}$ is an entangled set of states in $\mathcal{D}'$, which implies $\mathrm{ent}(\mathcal{D}) \leq \mathrm{ent}(\mathcal{D}')$.

Finally, we prove that $\mathrm{width}(\mathcal{D}') = \mathrm{ent}(\mathcal{D}')$. If $\mathcal{V}$ is an e.c. decomposition of $\mathcal{D}$ of minimum size, then Corollary 4.18 implies that $\mathcal{V}$ is an e.c. decomposition of $\mathcal{D}'$ satisfying the hypothesis of Lemma 4.12 so that $\mathrm{width}(\mathcal{D}') = \mathrm{ent}(\mathcal{D}')$ follows from this lemma. □

If we start from the minimum DFA $\mathcal{D}_\mathcal{L}$ of a regular language $\mathcal{L}$, then, as we shall see in Theorem 4.21, the automaton $\mathcal{D}'_\mathcal{L}$ acquires a special role because it realizes the deterministic width of the language $\mathcal{L}$.

*Definition 4.20.* If $\mathcal{D}_\mathcal{L}$ is the minimum DFA of a regular language $\mathcal{L}$, the DFA $\mathcal{D}'_\mathcal{L}$ is called the *Hasse automaton* for $\mathcal{L}$ and it is denoted by $\mathcal{H}_\mathcal{L}$.

The above definition is motivated by the fact that the width of the language can be "visualized" by the Hasse diagram of the partial order $\leq_{\mathcal{H}_\mathcal{L}}$.

Theorem 4.21. *If $\mathcal{D}_\mathcal{L}$ is the minimum DFA of the regular language $\mathcal{L}$, then:*

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

Proof. By Lemma 4.9 we have $\text{ent}(\mathcal{D}_{\mathcal{L}}) = \text{ent}(\mathcal{L})$. Since $\text{ent}(\mathcal{D}) \leq \text{width}(\mathcal{D})$ for all DFAs (Lemma 4.8), we obtain $\text{ent}(\mathcal{L}) \leq \text{width}^D(\mathcal{L})$, while from Theorem 4.19 we know that $\text{width}(\mathcal{H}_{\mathcal{L}}) = \text{ent}(\mathcal{D}_{\mathcal{L}})$. Hence, we have:

$$\text{width}(\mathcal{H}_{\mathcal{L}}) = \text{ent}(\mathcal{D}_{\mathcal{L}}) = \text{ent}(\mathcal{L}) \leq \text{width}^D(\mathcal{L}) \leq \text{width}(\mathcal{H}_{\mathcal{L}})$$

and the conclusion follows. □

The previous theorem allow us to give a first answer to Problem 4, that is, we provide an automata-free characterization of the deterministic width of a regular language. Recall that a property is *eventually* true for a sequence if it holds true for all but finitely many elements of the sequence.

Corollary 4.22. *Let $\mathcal{L}$ be a regular language. Then $\text{width}^D(\mathcal{L}) \leq p$ iff every (co-lexicographically) monotone sequence in Pref($\mathcal{L}$) is eventually included in at most $p$ classes of the Myhill-Nerode equivalence $\equiv_{\mathcal{L}}$.*

Proof. Let $\mathcal{D}_{\mathcal{L}}$ be the minimum DFA for $\mathcal{L}$. By definition, $k$ states $u_1, \ldots, u_k$ are entangled in $\mathcal{D}_{\mathcal{L}}$ iff there exists a monotone sequence $(\alpha_j)_{j \in \mathbb{N}}$ such that, for each $i = 1, \ldots, k$, we have $\delta(s, \alpha_j) = u_i$ for infinitely many $j$'s. Moreover, since $\mathcal{D}_{\mathcal{L}}$ is minimum, if $i \neq i'$ a word arriving in $u_i$ and a word arriving in $u_{i'}$ belong to different $\equiv_{\mathcal{L}}$-classes. Hence, $\text{ent}(\mathcal{D}_{\mathcal{L}}) > p$ iff there exists a monotone sequence in Pref($\mathcal{L}$) which eventually reaches more than $p$ classes of the Myhill-Nerode equivalence $\equiv_{\mathcal{L}}$ infinitely often, and the corollary follows from the previous theorem. □

Summarizing, the Hasse automaton $\mathcal{H}_{\mathcal{L}}$ captures the deterministic width of a language. An interesting open question is whether it is possible to devise an effective procedure to build the Hasse automaton. More generally, also in view of the indexing and compression applications in Section 5, we have two conflicting objectives: minimizing the width and minimizing the number of states. We will explore the latter objective in Section 4.5.

### 4.3 Computing the Deterministic Width of a Regular Language

In this section we shall use Theorem 4.21 — stating that the deterministic width of $\mathcal{L}$ is equal to the entanglement of the minimum DFA for $\mathcal{L}$ — to study the complexity of Problem 2 — the problem of finding the deterministic width of a language recognized by a given automaton. We show that if we are given a regular language $\mathcal{L}$ by means of a DFA $\mathcal{D}$ accepting $\mathcal{L}$ and a positive integer $p$, then the problem

$$\text{width}^D(\mathcal{L}) \stackrel{?}{\leq} p$$

is solvable in polynomial time for constant values of $p$. More precisely, we show that the problem of computing $\text{width}^D(\mathcal{L})$ is in the class XP with parameter $p$. This result is achieved by exhibiting a dynamic programming algorithm that extends the ideas introduced in [3] when solving the corresponding problem for Wheeler languages.

Theorem 4.21 suggests that the minimum DFA contains all "topological" information required to compute the width of a language. In the next theorem we clarify this intuition by providing a graph-theoretical characterization of the deterministic width of a language based on the minimum DFA recognizing the language.

Theorem 4.23. *Let $\mathcal{L}$ be a regular language and let $\mathcal{D}_{\mathcal{L}}$ be the minimum DFA of $\mathcal{L}$, with set of states $Q$. Let $k \geq 2$ be an integer. Then, $\text{width}^D(\mathcal{L}) \geq k$ if and only if there exist strings $\mu_1, \ldots, \mu_k, \gamma$ and pairwise distinct states $u_1, \ldots, u_k \in Q$, such that for every $j = 1, \ldots, k$:*

*(1) $\mu_j$ labels a path from the initial state $s$ to $u_j$;*
*(2) $\gamma$ labels a cycle starting (and ending) at $u_j$;*

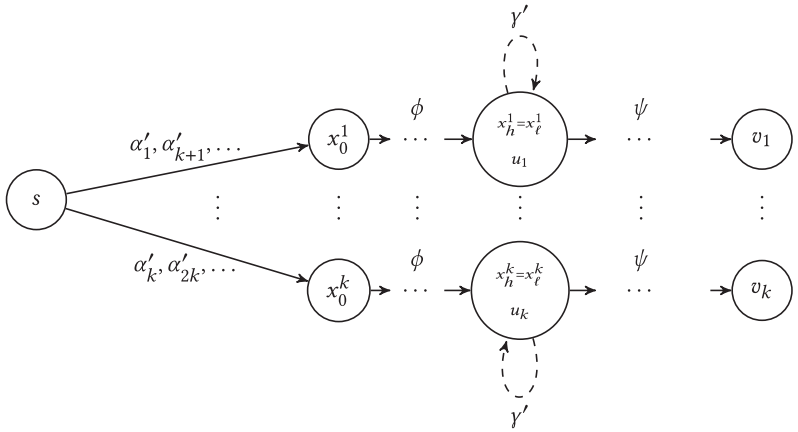



Fig. 6. A glimpse into the minimum DFA used in the proof of Theorem 4.23.

*(3) either* $\mu_1, \dots, \mu_k \prec \gamma$ *or* $\gamma \prec \mu_1, \dots, \mu_k$*;*
*(4)* $\gamma$ *is not a suffix of* $\mu_j$*.*

Proof. By Theorem 4.21 we have $\text{width}^D(\mathcal{L}) = \text{ent}(\mathcal{D}_\mathcal{L})$. We begin by proving that, if the stated conditions hold true, then $\text{ent}(\mathcal{D}_\mathcal{L}) \geq k$. Notice that for every integer $i$ we have $\mu_j \gamma^i \in I_{u_j}$. Moreover, the $\mu_j$'s are pairwise distinct because the $u_j$'s are pairwise distinct, so without loss of generality we can assume $\mu_1 \prec \cdots \prec \mu_k$.

(1) If $\mu_1 \prec \cdots \prec \mu_k \prec \gamma$, consider the increasing sequence:

$$\mu_1 \prec \cdots \prec \mu_k \prec \mu_1\gamma \prec \cdots \prec \mu_k\gamma \prec \mu_1\gamma^2 \prec \cdots \prec \mu_k\gamma^2 \prec \mu_1\gamma^3 \prec \cdots \prec \mu_k\gamma^3 \prec \dots$$

(2) If $\gamma \prec \mu_1 \prec \cdots \prec \mu_k$, consider the decreasing sequence:

$$\mu_k \succ \cdots \succ \mu_1 \succ \mu_k\gamma \succ \cdots \succ \mu_1\gamma \succ \mu_k\gamma^2 \succ \cdots \succ \mu_1\gamma^2 \succ \mu_k\gamma^3 \succ \cdots \succ \mu_1\gamma^3 \succ \dots$$

where $\mu_1\gamma^i \succ \mu_k\gamma^{i+1}$ holds because $\mu_1 \succ \gamma$ and $\gamma$ is not a suffix of $\mu_1$.

In both cases the sequence witnesses that $\{u_1, \dots, u_k\}$ is an entangled set of distinct states, so $\text{ent}(\mathcal{D}_\mathcal{L}) \geq k$.

Conversely, assume that $\text{ent}(\mathcal{D}_\mathcal{L}) \geq k$. This means that there exist distinct states $v_1, \dots, v_k$ and a monotone sequence $(\alpha_i)_{i\in\mathbb{N}}$ that reaches each of the $v_j$'s infinitely many times. Let us show that, up to taking subsequences, we can assume not only that $(\alpha_i)_{i\in\mathbb{N}}$ reaches each of the $v_j$'s infinitely many times, but it also satisfies additional properties.

— Since $|\Sigma|$ and $|Q|$ are finite and $(\alpha_i)_{i\in\mathbb{N}}$ is monotone, then up to removing a finite number of initial elements we can assume that all $\alpha_i$'s end with the same $m = |Q|^k$ characters, and we can write $\alpha_i = \alpha'_i\theta$, for some $\theta \in \Sigma^m$. Notice that such a new monotone sequence $(\alpha_i)_{i\in\mathbb{N}}$ still reaches each of the $v_j$'s infinitely many times.
— Up to taking a subsequence of the new $(\alpha_i)_{i\in\mathbb{N}}$, we can assume that $\alpha_i$ reaches $v_j$ if and only if $i - j$ is a multiple of $k$, that is, $\alpha_j, \alpha_{k+j}, \alpha_{2k+j}, \dots \in I_{v_j}$. Notice that such such a new monotone sequence $(\alpha_i)_{i\in\mathbb{N}}$ still satifies $\alpha_i = \alpha'_i\theta$ for every $i$.
— Since $|Q|$ is finite, up to taking a subsequence of the new $(\alpha_i)_{i\in\mathbb{N}}$ we can assume that all $\alpha_i$'s reaching the same $v_j$ spell the suffix $\theta$ visiting the same $m + 1$ states $x_0^j, x_1^j, \dots, x_m^j = v_j$.

Consider the $k$-tuples $(x_s^1, \dots, x_s^k)$, for $s \in \{0, \dots, m\}$ (corresponding to the states in column in Figure 6). There are $m + 1 = |Q|^k + 1$ such $k$-tuples and therefore two of them must be equal.

That is, there exist $h, \ell$, with $0 \le h < \ell \le m$, such that $(x_h^1, \ldots, x_h^k) = (x_\ell^1, \ldots, x_\ell^k)$. Hence, for all $j \in \{1, \ldots, k\}$ there is a cycle $x_h^j, x_{h+1}^j, \ldots, x_\ell^j$, all these cycles are labeled by the same string $\gamma'$, and we can write $\theta = \phi\gamma'\psi$ for some $\phi$ and $\psi$.

Let $u_1, u_2, \ldots, u_k$ be the pairwise distinct states $x_h^1, x_h^2, \ldots, x_h^k$ (they are distinct, because if $u_i = u_j$ for some $i \neq j$ we would have $v_i = v_j$). Hence, we have $k$ pairwise distinct states and $k$ equally labeled cycles. In order to fulfill the remaining conditions of the theorem, we proceed as follows. Considering the monotone sequence $(\alpha_i'\phi)_{i\in\mathbb{N}}$, reaching each of the $u_i$'s infinitely many times, we may suppose without loss of generality (possibly eliminating a finite number of initial elements) that all $\alpha_i'\phi$'s are co-lexicographically larger than $\gamma'$ or they are all co-lexicographically smaller than $\gamma'$.

If $\gamma'$ is not a suffix of any $\alpha_i'\phi$ we can choose $\gamma = \gamma'$ and, considering $k$ words $\mu_1, \ldots, \mu_k$ of the sequence $(\alpha_i'\phi)_{i\in\mathbb{N}}$ arriving in $u_1, \ldots, u_k$, respectively, we are done.

Otherwise, if $\gamma'$ is a suffix of some $\alpha_i'\phi$, pick $2k-1$ strings $\delta_1, \ldots, \delta_{2k-1}$ in the sequence $(\alpha_i'\phi)_{i\in\mathbb{N}}$ such that $\delta_k$ ends in $u_k$, while $\delta_i$ and $\delta_{k+i}$ end in $u_i$ for $i = 1, \ldots, k-1$, and

$$\delta_1 \prec \cdots \prec \delta_k \prec \cdots \prec \delta_{2k-1}.$$

Let $r$ be an integer such that $|(\gamma')^r| > |\delta_i|$ for every $i = 1, \ldots, 2k-1$. Then $\gamma = (\gamma')^r$ is the label of a cycle from $u_i$, for every $i = 1, \ldots, k$, and $\gamma$ is not a suffix of $\delta_i$, for every $i = 1, \ldots, 2k-1$. We distinguish two cases:

(1) $\delta_k \prec \gamma$. In this case, let $\mu_1, \ldots, \mu_k$ be equal to $\delta_1, \ldots, \delta_k$, respectively.
(2) $\gamma \prec \delta_k$. In this case, let $\mu_1, \ldots, \mu_k$ be equal to $\delta_k, \ldots, \delta_{2k-1}$, respectively.

In both cases, we have either $\mu_1, \ldots, \mu_k \prec \gamma$ or $\gamma \prec \mu_1, \ldots, \mu_k$ and the conclusion follows. □

*Example 4.24.* Let $\mathcal{L}$ be a regular language. Let us prove that, in general, $\text{width}^D(\mathcal{L})$ and $\text{width}^N(\mathcal{L})$ may depend on the total order $\preceq$ on the alphabet. Let $\mathcal{D}$ be the DFA in Figure 2, and let $\mathcal{L}$ be the language recognized by $\mathcal{D}$. Notice that $\mathcal{D}$ is the minimum DFA recognizing $\mathcal{L}$.

First, assume that $\preceq$ is the standard alphabetical order such that $a \prec b \prec c \prec d$. Let us prove that $\text{width}^D(\mathcal{L}) = 2$. From Example 2.12, we obtain $\text{width}^D(\mathcal{L}) \le 2$, and from Theorem 4.23 we obtain $\text{width}^D(\mathcal{L}) \ge 2$ by choosing $u_1 = q_2, u_2 = q_3, \mu_1 = a, \mu_2 = b, \gamma = c$. Notice that Corollary 3.3 implies that $\text{width}^N(\mathcal{L}) = \text{width}^D(\mathcal{L}) = 2$.

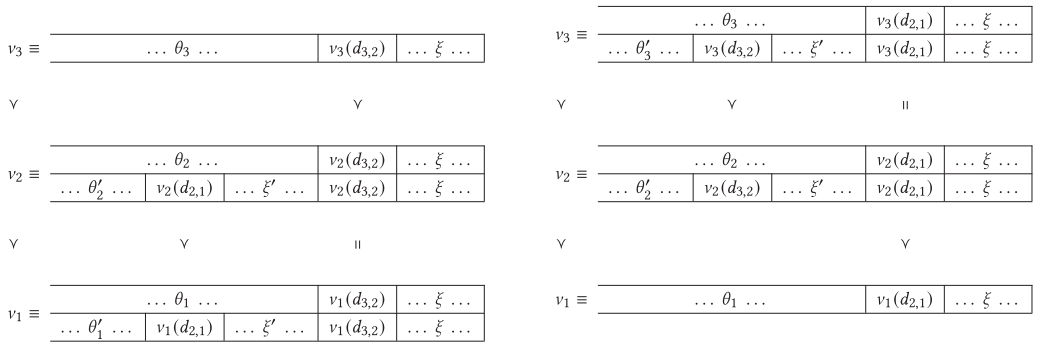
Next, let $\preceq$ be the total order such that $a \prec c \prec b \prec d$. From Example 2.12 we immediately obtain $\text{width}^N(\mathcal{L}) = \text{width}^D(\mathcal{L}) = 1$.

The strings $\mu_1, \ldots, \mu_k$, and $\gamma$ of Theorem 4.23 can be determined by a dynamic programming algorithm whose running time can be computed using the following lemma.

Lemma 4.25. *Let $\mathcal{D}$ be a DFA with set of states $Q$, and let $s_1, q_1, \ldots, s_h, q_h \in Q$. Suppose there are strings $v_1 \preceq \cdots \preceq v_h$ such that $\delta(s_i, v_i) = q_i$, for every $i = 1, \ldots, h$. Then, there exist strings $v_1' \preceq \cdots \preceq v_h'$ such that, for every $i, j \in \{1, \ldots, h\}$, it holds:*

$-\delta(s_i, v_i') = q_i$;
$-v_i = v_j$ *iff* $v_i' = v_j'$;
$-v_i \dashv v_j$ *iff* $v_i' \dashv v_j'$;
$-|v_i'| \le h - 2 + \sum_{t=1}^{h} |Q|^t$.

Proof. We will prove the lemma for $h = 3$ (the extension to the general case is straightforward). Given $\varphi \in \Sigma^*$, we denote by $\varphi(k)$ the $k$-th letter of $\varphi$ from the right (if $|\varphi| < k$ we write $\varphi(k) = \varepsilon$, where $\varepsilon$ is the empty string); therefore, if $\varphi \neq \varepsilon$, then $\varphi(1)$ is the last letter of $\varphi$.




(a) Case $\nu_1 \prec \nu_2 \prec \nu_3$ and $d_{3,2} < d_{2,1}$.

(b) Case $\nu_1 \prec \nu_2 \prec \nu_3$ and $d_{2,1} < d_{3,2}$.

Fig. 7. Proof of Lemma 4.25.

Let $\nu_1 \preceq \nu_2 \preceq \nu_3$ be strings with $\delta(s_i, \nu_i) = q_i$, for $i = 1, 2, 3$. Let $d_{3,2}$ be the first position from the right where $\nu_3$ and $\nu_2$ differ (if $\nu_3 = \nu_2$, let $d_{3,2} = |\nu_3|$). Since $\nu_2 \preceq \nu_3$, we have $d_{3,2} \leq |\nu_3|$. Defining $d_{2,1}$ similarly, we have $d_{2,1} \leq |\nu_2|$.

We distinguish three cases.

(1) $d_{3,2} = d_{2,1}$.
(2) $d_{3,2} < d_{2,1}$ (see Figure 7(a), assuming that $\nu_1 \prec \nu_2 \prec \nu_3$).
(3) $d_{2,1} < d_{3,2}$ (see Figure 7(b), assuming that $\nu_1 \prec \nu_2 \prec \nu_3$).

Case 3 is analogous to case 2 and will not be considered. In cases 1 and 2, $|\nu_3| \geq d_{3,2}$ and $|\nu_2| \geq d_{2,1} \geq d_{3,2}$, so that $\nu_3, \nu_2$, and $\nu_1$ end with the same (possibly empty) word $\xi$ with $|\xi| = d_{3,2} - 1$. Summing up:

$$\nu_1 = \theta_1\nu_1(d_{3,2})\xi \preceq \nu_2 = \theta_2\nu_2(d_{3,2})\xi \preceq \nu_3 = \theta_3\nu_3(d_{3,2})\xi$$

for some (possibly empty) strings $\theta_1, \theta_2, \theta_3$.

Without loss of generality, we may assume that $|\xi| \leq |Q|^3$. Indeed, if $|\xi| > |Q|^3$ then when we consider the triples of states visited while reading the last $|\xi|$ letters in computations from $s_i$ to $q_i$ following $\nu_i$, for $i = 1, 2, 3$, we should have met a repetition. If this were the case, we could erase a common factor from $\xi$, obtaining a shorter word $\xi_1$ such that $\theta_1\nu_1(d_{3,2})\xi_1 \preceq \theta_2\nu_2(d_{3,2})\xi_1 \preceq \theta_3\nu_3(d_{3,2})\xi_1$, with the three new strings starting in $s_1, s_2, s_3$ and ending in $q_1, q_2, q_3$, respectively, respecting equalities and suffixes. Then we can repeat the argument until we reach a word not longer than $|Q|^3$.

If $d_{3,2} = d_{2,1}$, the order between the $\nu_i's$ is settled in position $d_{3,2}$. Let $r_1, r_2, r_3$ be the states reached from $s_1, s_2, s_3$ by reading $\theta_1, \theta_2, \theta_3$, respectively. For $i = 1, 2, 3$, let $\bar{\theta}_i$ be the label of a simple path from $s_i$ to $r_i$ and let $\nu_i' = \bar{\theta}_i\nu_i(d_{3,2})\xi$. Then $\delta(s_i, \nu_i') = q_i$, $\nu_1' \preceq \nu_2' \preceq \nu_3'$, $|\nu_1'|, |\nu_2'|, |\nu_3'| \leq |Q| + |Q|^3$.

If $d_{3,2} < d_{2,1}$ we have $\nu_1(d_{3,2}) = \nu_2(d_{3,2})$. Moreover, $\theta_1$ and $\theta_2$ end with the same word $\xi'$ with $|\xi'| = d_{2,1} - d_{3,2} - 1$ (see Figure 7(a)), and we can write

$$\theta_1 = \theta_1'\nu_1(d_{2,1})\xi' \preceq \theta_2 = \theta_2'\nu_2(d_{2,1})\xi'.$$

Arguing as before we can assume, without loss of generality, that $|\xi'| \leq |Q|^2$. Moreover, we have $\nu_1(d_{2,1}) \preceq \nu_2(d_{2,1})$ and, as before, we can assume that $\theta_1', \theta_2'$ and $\theta_3$ label simple paths. Therefore, $|\theta_1'|, |\theta_2'|, |\theta_3| \leq |Q| - 1$. Hence, in this case we can find $\nu_1', \nu_2', \nu_3'$ such that $\delta(s_i, \nu_i') = q_i$, $\nu_1' \preceq \nu_2' \preceq \nu_3'$, and $|\nu_1'|, |\nu_2'|, |\nu_3'| \leq 1 + |Q| + |Q|^2 + |Q|^3$.

Finally, the construction implies that $\nu_i = \nu_j$ iff $\nu_i' = \nu_j'$, and $\nu_i \dashv \nu_j$ iff $\nu_i' \dashv \nu_j'$. □

We are now ready for a computational variant of Theorem 4.23.

Corollary 4.26. *Let $\mathcal{L}$ be a regular language and let $\mathcal{D}_{\mathcal{L}}$ be the minimum DFA of $\mathcal{L}$, with set of states $Q$. Let $k \geq 2$ be an integer. Then, $\text{width}^D(\mathcal{L}) \geq k$ if and only if there exist strings $\mu_1, \ldots, \mu_k$, and $\gamma$ and there exist pairwise distinct $u_1, \ldots, u_k \in Q$ such that, for every $j = 1, \ldots, k$:*

*(1) $\mu_j$ labels a path from the initial state $s$ to $u_j$;*
*(2) $\gamma$ labels a cycle starting (and ending) at $u_j$;*
*(3) either $\mu_1, \ldots, \mu_k \prec \gamma$ or $\gamma \prec \mu_1, \ldots, \mu_k$;*
*(4) $|\mu_1|, \ldots, |\mu_k| < |\gamma| \leq 2(2k - 2 + \sum_{t=1}^{2k} |Q|^t)$.*

Proof. ($\Leftarrow$) Since condition 4. implies that $\gamma$ is not a suffix of any of the $\mu_j$, $\text{width}^D(\mathcal{L}) \geq k$ follows from Theorem 4.23.
($\Rightarrow$) If $\text{width}^D(\mathcal{L}) \geq k$, we use Theorem 4.23 and find words $\mu'_1, \ldots, \mu'_k, \gamma'$ and pairwise distinct states $u_1, \ldots, u_k \in Q$ such that for every $j = 1, \ldots, k$:

(1) $\mu'_j$ labels a path from the initial state $s$ to $u_j$;
(2) $\gamma'$ labels a cycle starting (and ending) at $u_j$;
(3) either $\mu'_1, \ldots, \mu'_k \prec \gamma'$ or $\gamma' \prec \mu'_1, \ldots, \mu'_k$;
(4) $\gamma'$ is not a suffix of $\mu'_j$.

We only consider the case $\gamma' \prec \mu'_1, \ldots, \mu'_k$, since in the case $\mu'_1, \ldots, \mu'_k \prec \gamma'$ the proof is similar. Up to an index permutation, we may suppose without loss of generality that $\gamma' \prec \mu'_1 \prec \cdots \prec \mu'_k$. Consider the $2k$-words $v_i$ and states $s_i, q_i$ defined, for $i = 1, \ldots 2k$, as follows:

$-\, v_1 = \cdots = v_k = \gamma'$, $s_1 = q_1 = u_1, \ldots, s_k = q_k = u_k$;
$-\, v_{k+i} = \mu'_i$, $s_{k+i} = s$, $q_{k+i} = u_i$ for $i = 1, \ldots, k$, where $s$ is the initial state of $\mathcal{D}_{\mathcal{L}}$.

If we apply Lemma 4.25 to these $2k$-words, we obtain words $v'_1 = \cdots = v'_k \prec v'_{k+1} \prec \cdots \prec v'_{2k}$ such that, for all $i = 1, \ldots, k$:

(1) $\delta(u_i, v'_1) = u_i$, that is, $v'_1$ labels a cycle from every $u_i$;
(2) $\delta(s, v'_{k+i}) = u_i$;
(3) $v'_1$ is not a suffix of $v'_{k+i}$;
(4) $|v'_1|, |v'_{k+i}| \leq 2k - 2 + \sum_{t=1}^{2k} |Q|^t$.

Let $r$ be the smallest integer such that $|(v'_1)^r| > max\{|v'_{k+i}| \mid i \in \{1, \ldots, k\}\}$. Let $\mu_1 = v'_{k+1}, \ldots, \mu_k = v'_{2k}, \gamma = (v'_1)^r$. Since $v'_1$ is not a suffix of $\mu_i$, for all $i = 1, \ldots, k$, from $v'_1 \prec \mu_1 \prec \cdots \prec \mu_k$ it follows $\gamma = (v'_1)^r \prec \mu_1 \prec \cdots \prec \mu_k$, Moreover:

$$|\mu_1|, \ldots, |\mu_k| < |\gamma| \leq max\{|v'_{k+i}| \mid i \in \{1, \ldots, k\}\} + |v'_1| \leq 2\left(2k - 2 + \sum_{t=1}^{2k} |Q|^t\right)$$

and the conclusion follows. □

We can finally provide an algorithmic solution to Problem 2 in the deterministic case.

Theorem 4.27. *Let $\mathcal{L}$ be a regular language, given as input by means of any DFA $\mathcal{D} = (Q, s, \delta, F)$ recognizing $\mathcal{L}$. Then, for any integer $p \geq 1$ we can decide whether $\text{width}^D(\mathcal{L}) \leq p$ in time $|\delta|^{O(p)}$.*

Proof. We exhibit a dynamic programming algorithm based on Corollary 4.26, plugging in the value $k = p + 1$ and returning true if and only if $\text{width}^D(\mathcal{L}) \geq k$ is false.

First, note that the alphabet's size is never larger than the number of transitions: $\sigma \leq |\delta|$, and that $|Q| \leq |\delta| + 1$ since we assume that each state can be reached from $s$. Up to minimizing $\mathcal{D}$ (with Hopcroft's algorithm, running in time $O(|Q|\sigma \log |Q|) \leq |\delta|^{O(1)}$ we can assume that

$\mathcal{D} = \mathcal{D}_{\mathcal{L}}$ is the minimum DFA recognizing $\mathcal{L}$. Let $N' = 2(2k - 2 + \sum_{t=1}^{2k} |Q|^t)$ be the upper bound to the lengths of the strings $\mu_i$ $(1 \le i \le k)$ and $\gamma$ that need to be considered, and let $N = N' + 1$ be the number of states in a path labeled by a string of length $N'$. Asymptotically, note that $N \le |Q|^{O(k)} \le |\delta|^{O(k)}$. The high-level idea of the algorithm is as follows. First, in condition (3) of Corollary 4.26, we focus on finding paths $\mu_j$'s smaller than $\gamma$, as the other case (all $\mu_j$'s larger than $\gamma$) can be solved with a symmetric strategy. Then:

(1) For each state $u$ and each length $2 \le \ell \le N$, we compute the co-lexicographically smallest path of length (number of states) $\ell$ connecting $s$ with $u$.
(2) For each $k$-tuple $u_1, \ldots, u_k$ and each length $\ell \le N$, we compute the co-lexicographically largest string $\gamma$ labeling $k$ cycles of length (number of states) $\ell$ originating (respectively, ending) from (respectively, in) all the states $u_1, \ldots, u_k$.

Steps (1) and (2) could be naively solved by enumerating the strings $\mu_1, \ldots, \mu_k$, and $\gamma$ and trying all possible combinations of states $u_1, \ldots, u_k$. Because of the string enumeration step, however, this strategy would be exponential in $N$, i.e., doubly-exponential in $k$. We show that a dynamic programming strategy is exponentially faster.

`Step (1)`. This construction is identical to the one used in [3] for the Wheeler case ($p = 1$). For completeness, we report it here. Let $\pi_{u,\ell}$, with $u \in Q$ and $2 \le \ell \le N$, denote the predecessor of $u$ such that the co-lexicographically smallest path of length (number of states) $\ell$ connecting the source $s$ to $u$ passes through $\pi_{u,\ell}$ as follows: $s \rightsquigarrow \pi_{u,\ell} \to u$. The node $\pi_{u,\ell}$ coincides with $s$ if $\ell = 2$ and $u$ is a successor of $s$; in this case, the path is simply $s \to u$. If there is no path of length $\ell$ connecting $s$ with $u$, then we write $\pi_{u,\ell} = \perp$. We show that the set $\{\pi_{u,\ell} : 2 \le \ell \le N,\ u \in Q\}$ stores in just polynomial space all co-lexicographically smallest paths of any fixed length $2 \le \ell \le N$ from the source to any node $u$. We denote such a path — to be intended as a sequence $u_1 \to \cdots \to u_\ell$ of states — with $\alpha_\ell(u)$. The node sequence $\alpha_\ell(u)$ can be obtained recursively (in $O(\ell)$ steps) as $\alpha_\ell(u) = \alpha_{\ell-1}(\pi_{u,\ell}) \to u$, where $\alpha_1(s) = s$ by convention. Note also that $\alpha_\ell(u)$ does not fully specify the sequence of edges (and thus labels) connecting those $\ell$ states, since two states may be connected by multiple (differently labeled) edges. However, the corresponding co-lexicographically smallest sequence $\lambda^-(\alpha_\ell(u))$ of $\ell - 1$ labels is uniquely defined as follows:

$$\begin{cases} \lambda^-(\alpha_\ell(u)) = \min\{a \in \Sigma \mid \delta(s,a) = u\} & \text{if } \ell = 2, \\ \lambda^-(\alpha_\ell(u)) = \lambda^-(\alpha_{\ell-1}(\pi_{u,\ell}) \to u) = \lambda^-(\alpha_{\ell-1}(\pi_{u,\ell})) \cdot \min\{a \in \Sigma \mid \delta(\pi_{u,\ell}, a) = u\} & \text{if } \ell > 2. \end{cases}$$

It is not hard to see that each $\pi_{u,\ell}$ can be computed in $|\delta|^{O(k)}$ time using dynamic programming. First, we set $\pi_{u,2} = s$ for all successors $u$ of $s$. Then, for $\ell = 3, \ldots, N$:

$$\pi_{u,\ell} = \operatorname*{argmin}_{v \in \mathrm{Pred}(u)} \Big( \lambda^-(\alpha_{\ell-1}(v)) \cdot \min\{a \in \Sigma \mid \delta(v,a) = u\} \Big),$$

where $\mathrm{Pred}(u)$ is the set of all predecessors of $u$ and the argmin operator compares strings in co-lex order. In the equation above, if none of the $\alpha_{\ell-1}(v)$ are well-defined (because there is no path of length $\ell - 1$ from $s$ to $v$), then $\pi_{u,\ell} = \perp$. Note that computing any particular $\pi_{u,\ell}$ requires comparing co-lexicographically $|\mathrm{Pred(u)}| \le |Q|$ strings of length at most $\ell \le N \le |\delta|^{O(k)}$, which overall amounts to $|\delta|^{O(k)}$ time. Since there are $|Q| \times N = |\delta|^{O(k)}$ variables $\pi_{u,\ell}$ and each can be computed in time $|\delta|^{O(k)}$, overall Step (1) takes time $|\delta|^{O(k)}$. This completes the description of Step (1).

`Step (2)`. Fix a $k$-tuple $u_1, \ldots, u_k$ and a length $2 \le \ell \le N$. Our goal is now to show how to compute the co-lexicographically largest string $\gamma$ of length $\ell - 1$ labeling $k$ cycles of length

(number of states) $\ell$ originating (respectively, ending) from (respectively, in) all the states $u_1, \ldots, u_k$. Our final strategy will iterate over all such $k$-tuples of states (in time exponential in $k$) in order to find one satisfying the conditions of Corollary 4.26.

Our goal can again be solved by dynamic programming. Let $u_1, \ldots, u_k$ and $u'_1, \ldots, u'_k$ be two $k$-tuples of states, and let $2 \leq \ell \leq N$. Let moreover $\pi_{u_1,\ldots,u_k,u'_1,\ldots,u'_k,\ell}$ be the $k$-tuple $\langle u''_1, \ldots, u''_k \rangle$ of states (if it exists) such that there exists a string $\gamma$ of length $\ell - 1$ with the following properties:

- For each $1 \leq i \leq k$, there is a path $u_i \rightsquigarrow u''_i \to u'_i$ of length (number of nodes) $\ell$ labeled with $\gamma$, and
- $\gamma$ is the co-lexicographically largest string satisfying the above property.

If such a string $\gamma$ does not exist, then we set $\pi_{u_1,\ldots,u_k,u'_1,\ldots,u'_k,\ell} = \perp$.

Remember that we fix $u_1, \ldots, u_k$. For $\ell = 2$ and each $k$-tuple $u'_1, \ldots, u'_k$, it is easy to compute $\pi_{u_1,\ldots,u_k,u'_1,\ldots,u'_k,\ell}$: this $k$-tuple is $\langle u_1, \ldots, u_k \rangle$ (all paths have length 2) if and only if there exists $c \in \Sigma$ such that $u'_i = \delta(u_i, c)$ for all $1 \leq i \leq k$ (otherwise it does not exist). Then, $\gamma$ is formed by one character: the largest such $c$.

For $\ell > 2$, the $k$-tuple $\pi_{u_1,\ldots,u_k,u'_1,\ldots,u'_k,\ell}$ can be computed as follows. Assume we have computed those variables for all lengths $\ell' < \ell$. Note that for each such $\ell' < \ell$ and $k$-tuple $u''_1, \ldots, u''_k$, the variables $\pi_{u_1,\ldots,u_k,u''_1,\ldots,u''_k,\ell'}$ identify $k$ paths $u_i \rightsquigarrow u''_i$ of length (number of nodes) $\ell'$. Let us denote with $\alpha_{\ell'}(u''_i)$ such paths, for $1 \leq i \leq k$.

Then, $\pi_{u_1,\ldots,u_k,u'_1,\ldots,u'_k,\ell}$ is equal to $\langle u''_1, \ldots, u''_k \rangle$ maximizing co-lexicographically the string $\gamma' \cdot c$ defined as follows:

(1) $u'_i = \delta(u''_i, c)$ for all $1 \leq i \leq k$,
(2) $\pi_{u_1,\ldots,u_k,u''_1,\ldots,u''_k,\ell-1} \neq \perp$, and
(3) $\gamma'$ is the co-lexicographically largest string labeling all the paths $\alpha_{\ell-1}(u''_i)$. Note that this string exists by condition (2), and it can be easily built by following those paths in parallel (choosing, at each step, the largest character labeling all the $k$ considered edges of the $k$ paths).

If no $c \in \Sigma$ satisfies condition (1), or condition (2) cannot be met, then $\pi_{u_1,\ldots,u_k,u'_1,\ldots,u'_k,\ell} = \perp$.

Note that $\pi_{u_1,\ldots,u_k,u_1,\ldots,u_k,\ell}$ allows us to identify (if it exists) the largest string $\gamma$ of length $\ell - 1$ labeling $k$ cycles originating and ending in each $u_i$, for $1 \leq i \leq k$.

Each tuple $\pi_{u_1,\ldots,u_k,u'_1,\ldots,u'_k,\ell}$ can be computed in $|\delta|^{O(k)}$ time by dynamic programming (in order of increasing $\ell$), and there are $|\delta|^{O(k)}$ such tuples to be computed (there are $|Q|^{O(k)} \leq |\delta|^{O(k)}$ ways of choosing $u_1, \ldots, u_k, u'_1, \ldots, u'_k$, and $N \leq |\delta|^{O(k)}$). Overall, also Step (2) can therefore be solved in $|\delta|^{O(k)}$ time.

To sum up, we can check if the conditions of Corollary 4.26 hold as follows:

(1) We compute $\pi_{u,\ell}$ for each $u \in Q$ and $\ell \leq N$. This identifies a string $\mu_u^\ell$ for each such pair $u \in Q$ and $\ell \leq N$: the co-lexicographically smallest one, of length $\ell - 1$, labeling a path connecting $s$ with $u$.
(2) For each $k$-tuple $u_1, \ldots, u_k$ and each $\ell \leq N$, we compute $\pi_{u_1,\ldots,u_k,u_1,\ldots,u_k,\ell}$. This identifies a string $\gamma^\ell_{u_1,\ldots,u_k}$ for each such tuple $u_1, \ldots, u_k$ and $\ell \leq N$: the co-lexicographically largest one, of length $\ell - 1$, labeling $k$ cycles originating and ending in each $u_i$, for $1 \leq i \leq k$.
(3) We identify the $k$-tuple $u_1, \ldots, u_k$ and the lengths $\ell_i < \ell \leq N$ (if they exist) such that $\mu^{\ell_i}_{u_i} \prec \gamma^\ell_{u_1,\ldots,u_k}$ for all $1 \leq i \leq k$.

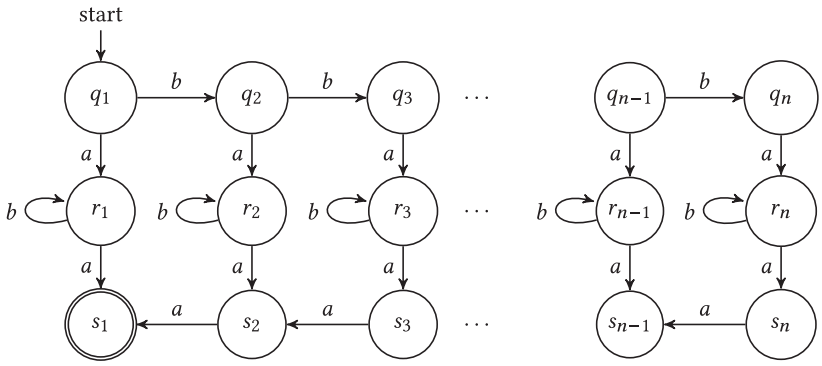



Fig. 8. A minimum DFA $\mathcal{D}_n$ recognizing a star-free language $\mathcal{L}_n$ with $\text{width}^D(\mathcal{L}_n) = n$ for the two possible orders on the alphabet $\{a, b\}$.

The conditions of Corollary 4.26 hold if and only if step 3 above succeeds for at least one $k$-tuple $u_1, \ldots, u_k$ and lengths $\ell_i < \ell \leq N$, for $1 \leq i \leq k$. Overall, the algorithm terminates in $|\delta|^{O(k)} = |\delta|^{O(p)}$ time. □

### 4.4 Relation with Star-Free Languages

Theorem 4.23 allows us to describe the levels of the width hierarchy looking to cycles in the minimum automata for the languages. This results resemble another very well known result on a class of subregular languages, the star-free ones, which can also be described by inspecting the cycles in the minimum DFA for the language.

*Definition 4.28.* A regular language is said to be star-free if it can be described by a regular expression constructed from the letters of the alphabet, the empty set symbol, all boolean operators (including complementation), and concatenation (but no Kleene star).

A well-known automata characterization of star-free languages is given by using counters. A counter in a DFA is a sequence of pairwise-distinct states $u_0, \ldots, u_n$ (with $n \geq 1$) such that there exists a non-empty string $\alpha$ with $\delta(u_0, \alpha) = u_1, \ldots, \delta(u_{n-1}, \alpha) = u_n, \delta(u_n, \alpha) = u_0$. A language is star-free if and only if its minimum DFA has no counters [67, 76].

We can easily prove that a Wheeler language, i.e., a language $\mathcal{L}$ with $\text{width}^N(\mathcal{L}) = \text{width}^D(\mathcal{L}) = 1$ for a fixed order of the alphabet, is always star-free. Indeed, if the minimum DFA for a language has a counter $u_0, \ldots, u_n$ with string $\alpha$, and $\gamma \in I_{u_0}$, then $(\gamma\alpha^n)_{n\in\mathbb{N}}$ is a monotone sequence (increasing or decreasing depending on which string between $\gamma$ and $\gamma\alpha$ is smaller) which is not ultimately included in one class of the Myhill-Nerode equivalence $\equiv_{\mathcal{L}}$ (because in a minimum DFA the $I_u$'s are exactly equal to Nerode classes). Hence, the language is not Wheeler by Corollary 4.22.

This implies that the first level of the deterministic width hierarchy is included in the class of star-free languages. On the other hand, in the next example we prove that there is an infinite sequence of star-free languages $(\mathcal{L}_n)_{n\in\mathbb{N}}$ over the two letter alphabet $\{a, b\}$ such that $\text{width}^D(\mathcal{L}_n) = n$, for both total orders $\preceq$ on $\{a, b\}$.

*Example 4.29.* In Figure 8 we depicted a DFA $\mathcal{D}_n$ with $3n$ states accepting the language $\mathcal{L}_n = \bigcup_{j=0}^{n-1} b^j a b^* a^{j+1}$. Notice that:

(1) for every state $u$ and for every $1 \leq j \leq n$, we have that $\delta(u, aba^j)$ is defined and final if and only if $u = q_j$;

(2) for every state $u$ in the second or third row and for every $1 \leq j \leq n$, we have that $\delta(u, ba^j)$ is defined and final if and only if $u = r_j$;
(3) for every state $u$ in the third row and for every $1 \leq j \leq n$, we have that $\delta(u, a^{j-1})$ is defined and final if and only if $u = s_j$.

We conclude that $\mathcal{D}_n$ is the minimum DFA of $\mathcal{L}_n$.

Since $\mathcal{D}_n$ has no counters (because every cycle is a self-loop), the above mentioned characterization of star-free languages tells us that $\mathcal{L}_n$ is star-free. Let us prove that $\text{ent}(\mathcal{D}_n) = n$, so that $\text{width}^D(\mathcal{L}_n) = n$ follows from Theorem 4.21. Notice that (1) states in the first row are reached by only one string, (2) states in the second row are reached infinitely many times only by string ending with $b$, and (3) states in the third row are reached only by strings ending with $a$. This implies $\text{ent}(\mathcal{D}_n) \leq n$, because the words belonging to a monotone sequence witnessing an entanglement between states will definitely end by the same letter, so only states belonging to the same row may belong to an entangled set. Finally, the $n$ states in the second level are entangled, as it witnessed by the monotone sequence:

$$a \prec ba \prec bba \prec \cdots \prec b^{n-1}a \prec ab \prec bab \prec bbab \prec \cdots \prec b^{n-1}ab \prec \cdots \prec ab^k \prec bab^k \prec bbab^k \prec \dots$$

if $a \prec b$, and by the monotone sequence:

$$b^{n-1}a \succ b^{n-2}a \succ \cdots \succ a \succ b^{n-1}ab \succ b^{n-2}ab \succ \cdots \succ ab \succ \cdots \succ b^{n-1}ab^k \succ b^{n-2}ab^k \succ \cdots \succ ab^k \succ \dots$$

if $b \prec a$. Hence, in both cases we have $\text{ent}(\mathcal{D}_n) = n$.

### 4.5 The Convex Myhill-Nerode Theorem

In the previous sections we described a hierarchy of regular languages by means of their deterministic widths. A natural question is whether a corresponding Myhill-Nerode theorem can be provided for every level of the hierarchy: given a regular language $\mathcal{L}$, if we consider all DFAs recognizing $\mathcal{L}$ and having width equal to $\text{width}^D(\mathcal{L})$, is there a unique such DFA having the minimum number of states? In general, the answer is "no", as showed in Example 4.6.

The non-uniqueness can be explained as follows. If a DFA of width $p$ recognizes $\mathcal{L}$, then $\text{Pref}(\mathcal{L})$ can be partitioned into $p$ sets, each of which consists of the (disjoint) union of some pairwise comparable $I_q$'s. However, in general the partition into $p$ sets is not unique, so it may happen that two distinct partitions lead to two non-isomorphic minimal DFAs with the same number of states. For example, in Figure 3, we see two non-isomorphic DFAs (center and right) realizing the width of the language and with the minimum number of states among all DFAs recognizing the same language and realizing the width of the language: the chain partition $\{\{0, 1, 4\}, \{2, 3, 5, 3'\}\}$ of the DFA in the center induces the partition $\{ac^* \cup \{\varepsilon, e, h\}, bc^* \cup ac^*d \cup \{gd, ee, he, f, k, g\}\}$ of $\text{Pref}(\mathcal{L})$, whereas the chain partition $\{\{0, 1, 3\}, \{2, 4, 5, 4'\}\}$ of the DFA on the right induces the partition $\{ac^* \cup ac^*d \cup \{\varepsilon, gd, ee, he, f, k\}, bc^* \cup \{e, h, g\}\}$ of $\text{Pref}(\mathcal{L})$.

This example shows that no uniqueness results can be ensured as long as partitions are not fixed. But what happens if we fix a partition? As we will prove in this section, once a partition is fixed, it is possible to prove a full Myhill-Nerode theorem, thereby providing a DFA-free characterization of languages of width equal to $p$ and a minimum DFA for these languages.

More formally, let $\mathcal{D} = (Q, s, \delta, F)$ be a DFA, and let $\{Q_i \mid 1 \leq i \leq p\}$ be a $\leq_{\mathcal{D}}$-chain partition of $Q$. For every $i \in \{1, \dots, p\}$, define:

$$\text{Pref}(\mathcal{L}(\mathcal{D}))^i = \{\alpha \in \text{Pref}(\mathcal{L}(\mathcal{D})) \mid \delta(s, \alpha) \in Q_i\}.$$

Then $\{\text{Pref}(\mathcal{L}(\mathcal{D})^i \mid 1 \leq i \leq p\}$ is a partition of $\text{Pref}(\mathcal{L}(\mathcal{D}))$, and from now on we will think of such a partition as fixed. We now consider the class of all DFAs accepting $\mathcal{L}$ and inducing the considered partition.

*Definition 4.30.* Let $\mathcal{D} = (Q, s, \delta, F)$ be a DFA, and let $\mathcal{P} = \{U_1, \ldots, U_p\}$ be a partition of $\mathrm{Pref}(\mathcal{L}(\mathcal{D}))$. We say that $\mathcal{D}$ is $\mathcal{P}$-sortable if there exists a $\leq_{\mathcal{D}}$-chain partition $\{Q_i \mid 1 \leq i \leq p\}$ such that for every $i \in \{1, \ldots, p\}$:

$$\mathrm{Pref}(\mathcal{L}(\mathcal{D}))^i = U_i.$$

We wish to give a DFA-free characterization of languages $\mathcal{L}$ and partitions $\mathcal{P}$ of $\mathrm{Pref}(\mathcal{L})$ for which there exists a $\mathcal{P}$-sortable DFA. As in the Myhill-Nerode theorem, we aim at determining which properties an equivalence relation $\sim$ should satisfy to ensure that a canonical construction provides a $\mathcal{P}$-sortable DFA. First, $\mathcal{L}$ must be regular, so $\sim$ is expected to be right-invariant. In order to develop some intuition on the required properties, let us consider an equivalence relation which plays a key role in the classical Myhill-Nerode theorem. Let $\mathcal{D} = (Q, s, \delta, F)$ be a $\mathcal{P}$-sortable DFA, and let $\equiv_{\mathcal{D}}$ be the equivalence relation on $\mathrm{Pref}(\mathcal{L}(\mathcal{D}))$ defined by

$$\alpha \equiv_{\mathcal{D}} \beta \Leftrightarrow \delta(s, \alpha) = \delta(s, \beta).$$

Notice that equivalent strings end up in the same element of $\mathcal{P}$ (*$\mathcal{P}$-consistency*), and since all states in each $\leq_{\mathcal{D}}$-chain $Q_i$ are comparable, then each $I_q$ must be convex in the corresponding element of $\mathcal{P}$ (*$\mathcal{P}$-convexity*). More formally we consider the following definition, where, for every $\alpha \in \mathrm{Pref}(\mathcal{L})$, we denote by $U_\alpha$ the unique element $U_i$ of $\mathcal{P}$ such that $\alpha \in U_i$.

*Definition 4.31.* Let $\mathcal{L} \subseteq \Sigma^*$ be a language, and let $\sim$ be an equivalence relation on $\mathrm{Pref}(\mathcal{L})$. Let $\mathcal{P} = \{U_1, \ldots, U_p\}$ be a partition of $\mathrm{Pref}(\mathcal{L})$.

(1) We say that $\sim$ is *$\mathcal{P}$-consistent* if for every $\alpha, \beta \in \mathrm{Pref}(\mathcal{L})$, if $\alpha \sim \beta$, then $U_\alpha = U_\beta$.
(2) Assume that $\sim$ is $\mathcal{P}$-consistent. We say that $\sim$ is *$\mathcal{P}$-convex* if for every $\alpha \in \mathrm{Pref}(\mathcal{L})$ we have that $[\alpha]_\sim$ is a convex in $(U_\alpha, \leq)$.

As we now prove, these are exactly the required properties for a DFA-free characterization.

Let $\mathcal{L} \subseteq \Sigma^*$ be a language, and let $\sim$ be an equivalence relation on $\mathrm{Pref}(\mathcal{L})$. We say that $\sim$ *respects* $\mathrm{Pref}(\mathcal{L})$ if:

$$(\forall \alpha, \beta \in \mathrm{Pref}(\mathcal{L}))(\forall \phi \in \Sigma^*)(\alpha \sim \beta \wedge \alpha\phi \in \mathrm{Pref}(\mathcal{L}) \rightarrow \beta\phi \in \mathrm{Pref}(\mathcal{L})).$$

Now, let us define the right-invariant, $\mathcal{P}$-consistent and $\mathcal{P}$-convex refinements of an equivalence relation $\sim$.

(1) Assume that $\sim$ respects $\mathrm{Pref}(\mathcal{L})$. For every $\alpha, \beta \in \mathrm{Pref}(\mathcal{L})$, define:

$$\alpha \sim^r \beta \iff (\forall \phi \in \Sigma^*)(\alpha\phi \in \mathrm{Pref}(\mathcal{L}) \rightarrow \alpha\phi \sim \beta\phi).$$

We say that $\sim^r$ is the *right-invariant refinement* of $\sim$.
(2) Let $\mathcal{P} = \{U_1, \ldots, U_p\}$ be a partition of $\mathrm{Pref}(\mathcal{L})$. For every $\alpha, \beta \in \mathrm{Pref}(\mathcal{L})$, define:

$$\alpha \sim^{cs} \beta \iff (\alpha \sim \beta) \wedge (U_\alpha = U_\beta)$$

We say that $\sim^{cs}$ is the *$\mathcal{P}$-consistent refinement* of $\sim$.
(3) Let $\mathcal{P} = \{U_1, \ldots, U_p\}$ be a partition of $\mathrm{Pref}(\mathcal{L})$. Assume that $\sim$ is $\mathcal{P}$-consistent. For every $\alpha, \gamma \in \mathrm{Pref}(\mathcal{L})$, define:

$$\begin{aligned}\alpha \sim^{cv} \gamma \iff\ & (\alpha \sim \gamma) \wedge \\ & \wedge (\forall \beta \in \mathrm{Pref}(\mathcal{L}))(((U_\alpha = U_\beta) \wedge (\min\{\alpha, \gamma\} \prec \beta \prec \max\{\alpha, \gamma\}) \rightarrow \alpha \sim \beta).\end{aligned}$$

We say that $\sim^{cv}$ is the *$\mathcal{P}$-convex refinement* of $\sim$.

It is easy to check that $\sim^r$ is the coarsest right-invariant equivalence relation refining $\sim$, $\sim^{cs}$ is the coarsest $\mathcal{P}$-consistent equivalence relation refining $\sim$ and $\sim^{cv}$ is the coarsest $\mathcal{P}$-convex equivalence relation refining $\sim$.

We wish to prove that any equivalence relation that respects Pref($\mathcal{L}$) admits a coarsest refinement being $\mathcal{P}$-consistent, $\mathcal{P}$-convex and right-invariant at once, because then we will be able to define an equivalence relation inducing the minimum ($\mathcal{P}$-sortable) DFA. We first prove that if we use the operators $cv$ and $r$, in this order, over a $\mathcal{P}$-consistent and right-invariant equivalence relation we do not lose $\mathcal{P}$-consistency, nor right-invariance, and we gain $\mathcal{P}$-convexity.

Lemma 4.32. *Let $\mathcal{L} \subseteq \Sigma^*$ be a language, and let $\mathcal{P}$ be a partition of Pref($\mathcal{L}$). If $\sim$ is a $\mathcal{P}$-consistent and right-invariant equivalence relation on Pref($\mathcal{L}$), then the relation $(\sim^{cv})^r$ is $\mathcal{P}$-consistent, $\mathcal{P}$-convex and right-invariant.*

Proof. By definition $(\sim^{cv})^r$ is a right-invariant refinement. Moreover, $\sim^{cv}$ and $(\sim^{cv})^r$ are $\mathcal{P}$-consistent because they are refinements of the $\mathcal{P}$-consistent equivalence relation $\sim$. Let us prove that $(\sim^{cv})^r$ is $\mathcal{P}$-convex. Assume that $\alpha, \beta, \gamma \in \text{Pref}(\mathcal{L})$ are such that $\alpha(\sim^{cv})^r\gamma$, $\alpha \prec \beta \prec \gamma$ and $U_\alpha = U_\beta$. Being $(\sim^{cv})^r$ a $\mathcal{P}$-consistent relation, we have $U_\alpha = U_\beta = U_\gamma$. We must prove that $\alpha(\sim^{cv})^r\beta$. Fix $\phi \in \Sigma^*$ such that $\alpha\phi \in \text{Pref}(\mathcal{L})$. We must prove that $\alpha\phi \sim^{cv} \beta\phi$. Now, $\alpha(\sim^{cv})^r\gamma$ implies $\alpha \sim^{cv} \gamma$. Since $\alpha \prec \beta \prec \gamma$ and $U_\alpha = U_\beta = U_\gamma$, then the $\mathcal{P}$-convexity of $\sim^{cv}$ implies $\alpha \sim^{cv} \beta$. In particular, $\alpha \sim \beta$. Since $\sim$ is right-invariant we have $\alpha\phi \sim \beta\phi$, and from the $\mathcal{P}$-consistency of $\sim$ we obtain $U_{\alpha\phi} = U_{\beta\phi}$. Moreover, $\alpha(\sim^{cv})^r\gamma$ implies $\alpha\phi(\sim^{cv})^r\gamma\phi$ by right-invariance, so $\alpha\phi \sim^{cv} \gamma\phi$. By $\mathcal{P}$-convexity, from $\alpha\phi \sim^{cv} \gamma\phi$, $U_{\alpha\phi} = U_{\beta\phi}$ and $\alpha\phi \prec \beta\phi \prec \gamma\phi$ (since $\alpha \prec \beta \prec \gamma$) we conclude $\alpha\phi \sim^{cv} \beta\phi$. □

Corollary 4.33. *Let $\mathcal{L} \subseteq \Sigma^*$ be a nonempty language, and let $\mathcal{P}$ be a partition of Pref($\mathcal{L}$). Let $\sim$ be an equivalence relation that respects Pref($\mathcal{L}$). Then, there exists a (unique) coarsest $\mathcal{P}$-consistent, $\mathcal{P}$-convex and right-invariant equivalence relation refining $\sim$.*

Proof. The equivalence relation $(\sim^{cs})^r$ is $\mathcal{P}$-consistent (because it is a refinement of the $\mathcal{P}$-consistent equivalence relation $\sim^{cs}$) and right-invariant (by definition it is a right-invariant refinement), so by Lemma 4.32 the equivalence relation $(((\sim^{cs})^r)^{cv})^r$ is $\mathcal{P}$-consistent, $\mathcal{P}$-convex and right-invariant. Moreover, every $\mathcal{P}$-consistent, $\mathcal{P}$-convex and right-invariant equivalence relation refining $\sim$ must also refine $(((\sim^{cs})^r)^{cv})^r$, so $(((\sim^{cs})^r)^{cv})^r$ is the coarsest $\mathcal{P}$-consistent, $\mathcal{P}$-convex and right-invariant equivalence relation refining $\sim$. □

Corollary 4.33 allows us to give the following definition.

*Definition 4.34.* Let $\mathcal{L} \subseteq \Sigma^*$ be a language, and let $\mathcal{P} = \{U_1, \ldots, U_p\}$ be a partition of Pref($\mathcal{L}$). Denote by $\equiv^{\mathcal{P}}_{\mathcal{L}}$ the coarsest $\mathcal{P}$-consistent, $\mathcal{P}$-convex and right-invariant equivalence relation refining the Myhill-Nerode equivalence $\equiv_{\mathcal{L}}$.

In particular, since $\mathcal{L}$ is the union of some $\equiv_{\mathcal{L}}$-classes, we also have that $\mathcal{L}$ is the union of some $\equiv^{\mathcal{P}}_{\mathcal{L}}$-classes.

Recall that, given a DFA $\mathcal{D} = (Q, s, \delta, F)$, the equivalence relation $\equiv_{\mathcal{D}}$ on Pref($\mathcal{L}(\mathcal{D})$) is the one such that:

$$\alpha \equiv_{\mathcal{D}} \beta \iff \delta(s, \alpha) = \delta(s, \beta).$$

Here are the key properties of $\equiv_{\mathcal{D}}$, when $\mathcal{D}$ is a $\mathcal{P}$-sortable DFA.

Lemma 4.35. *Let $\mathcal{D} = (Q, s, \delta, F)$ be a $\mathcal{P}$-sortable DFA, where $\mathcal{P} = \{U_1, \ldots, U_p\}$ is a partition of Pref($\mathcal{L}$) for $\mathcal{L} = \mathcal{L}(\mathcal{D})$. Then, $\equiv_{\mathcal{D}}$ has finite index, it respects Pref($\mathcal{L}$), it is right-invariant, $\mathcal{P}$-consistent, $\mathcal{P}$-convex, it refines $\equiv^{\mathcal{P}}_{\mathcal{L}}$, and $\mathcal{L}$ is the union of some $\equiv_{\mathcal{D}}$-classes. In particular, $\equiv^{\mathcal{P}}_{\mathcal{L}}$ has finite index.*

Proof. The relation $\equiv_{\mathcal{D}}$ has index equal to $|Q|$. It respects $\mathrm{Pref}(\mathcal{L})$ because if $\alpha \equiv_{\mathcal{D}} \beta$ and $\phi \in \Sigma^*$ satisfies $\alpha\phi \in \mathrm{Pref}(\mathcal{L})$, then there exists $\gamma$ with $\alpha\phi\gamma \in \mathcal{L}$ so $\delta(s, \alpha\phi\gamma) \in F$. Since $\delta(s, \alpha) = \delta(s, \beta)$ we obtain $\delta(s, \alpha\phi\gamma) = \delta(s, \beta\phi\gamma)$ and so $\beta\phi\gamma \in \mathcal{L}$ and $\beta\phi \in \mathrm{Pref}(\mathcal{L})$ follows. Moreover, it is right-invariant because if $\alpha \equiv_{\mathcal{D}} \beta$ and $\phi \in \Sigma^*$ is such that $\alpha\phi\gamma \in \mathcal{L}$, then $\beta\phi \in \mathrm{Pref}(\mathcal{L})$ and from $\delta(s, \alpha) = \delta(s, \beta)$ we obtain $\delta(s, \alpha\phi) = \delta(s, \beta\phi)$.

For every $\alpha \in \mathrm{Pref}(\mathcal{L})$ we have $[\alpha]_{\equiv_{\mathcal{D}}} = I_{\delta(s,\alpha)}$, which implies that $\equiv_{\mathcal{D}}$ is $\mathcal{P}$-consistent. Moreover, $\equiv_{\mathcal{D}}$ is $\mathcal{P}$-convex, that is, for every $\alpha \in \mathrm{Pref}(\mathcal{L})$ we have that $[\alpha]_{\equiv_{\mathcal{D}}} = I_{\delta(s,\alpha)}$ is convex in $U_\alpha$, because if $u_1, \ldots, u_k \in Q$ are such that $U_\alpha = \bigcup_{i=1}^{k} I_{u_i}$, then the $u_i$'s must be pairwise $\leq_{\mathcal{D}}$-comparable, being in the same $\leq_{\mathcal{D}}$-chain. Moreover, $\equiv_{\mathcal{D}}$ refines $\equiv_{\mathcal{L}}$, because $\alpha \equiv_{\mathcal{D}} \beta$ implies that for every $\phi \in \Sigma^*$ we have $\delta(s, \alpha\phi) = \delta(s, \beta\phi)$ and so $\alpha\phi \in \mathcal{L}$ iff $\beta\phi \in \mathcal{L}$. Since $\equiv_{\mathcal{L}}^{\mathcal{P}}$ is the coarsest $\mathcal{P}$-consistent, $\mathcal{P}$-convex and right-invariant equivalence relation refining $\equiv_{\mathcal{L}}$, and $\equiv_{\mathcal{D}}$ is a $\mathcal{P}$-consistent, $\mathcal{P}$-convex and right-invariant equivalence relation refining $\equiv_{\mathcal{L}}$, we conclude that $\equiv_{\mathcal{D}}$ also refines $\equiv_{\mathcal{L}}^{\mathcal{P}}$, which in particular implies that $\mathcal{L}$ is the union of some $\equiv_{\mathcal{D}}$-classes. We know that $\equiv_{\mathcal{D}}$ has finite index, so $\equiv_{\mathcal{L}}^{\mathcal{P}}$ has finite index. □

We can now explain how to canonically build a $\mathcal{P}$-sortable DFA starting from an equivalence relation.

Lemma 4.36. *Let $\mathcal{L} \subseteq \Sigma^*$ be a language, and let $\mathcal{P} = \{U_1, \ldots, U_p\}$ be a partition of $\mathrm{Pref}(\mathcal{L})$. Assume that $\mathcal{L}$ is the union of some classes of a $\mathcal{P}$-consistent, $\mathcal{P}$-convex, right-invariant equivalence relation $\sim$ on $\mathrm{Pref}(\mathcal{L})$ of finite index. Then, $\mathcal{L}$ is recognized by a $\mathcal{P}$-sortable DFA $\mathcal{D}_\sim = (Q_\sim, s_\sim, \delta_\sim, F_\sim)$ such that:*

*(1) $|Q_\sim|$ is equal to the index of $\sim$;*
*(2) $\equiv_{D_\sim}$ and $\sim$ are the same equivalence relation (in particular, $|Q_\sim|$ is equal to the index of $\equiv_{D_\sim}$).*

*Moreover, if $\mathcal{B}$ is a $\mathcal{P}$-sortable DFA that recognizes $\mathcal{L}$, then $\mathcal{D}_{\equiv_{\mathcal{B}}}$ is isomorphic to $\mathcal{B}$.*

Proof. Define the DFA $\mathcal{D}_\sim = (Q_\sim, s_\sim, \delta_\sim, F_\sim)$ as follows.

$-Q_\sim = \{[\alpha]_\sim \mid \alpha \in \mathrm{Pref}(\mathcal{L})\}$;
$-s_\sim = [\varepsilon]_\sim$, where $\varepsilon$ is the empty string;
$-\delta_\sim([\alpha]_\sim, a) = [\alpha a]_\sim$, for every $\alpha \in \Sigma^*$ and $a \in \Sigma$ such that $\alpha a \in \mathrm{Pref}(\mathcal{L})$.
$-F_\sim = \{[\alpha]_\sim \mid \alpha \in \mathcal{L}\}$.

Since $\sim$ is right-invariant, it has finite index and $\mathcal{L}$ is the union of some $\sim$-classes, then $\mathcal{D}_\sim$ is a well-defined DFA and:

$$\alpha \in [\beta]_\sim \iff \delta_\sim(s_\sim, \alpha) = [\beta]_\sim. \tag{6}$$

which implies that for every $\alpha \in \mathrm{Pref}(\mathcal{L})$ it holds $I_{[\alpha]_\sim} = [\alpha]_\sim$, and so $\mathcal{L}(\mathcal{D}_\sim) = \mathcal{L}$.

For every $i \in \{1, \ldots, p\}$, define:

$$Q_i = \{[\alpha]_\sim \mid U_\alpha = U_i\}.$$

Notice that each $Q_i$ is well-defined because $\sim$ is $\mathcal{P}$-consistent, and each $Q_i$ is a $\leq_{\mathcal{D}_\sim}$-chain because $\sim$ is $\mathcal{P}$-convex. It follows that $\{Q_i \mid 1 \leq i \leq p\}$ is a $\leq_{\mathcal{D}_\sim}$-chain partition of $Q_\sim$.

From Equation (6) we obtain:

$$\begin{aligned}\mathrm{Pref}(\mathcal{L}(\mathcal{D}_\sim))^i &= \{\alpha \in \mathrm{Pref}(\mathcal{L}(\mathcal{D}_\sim)) \mid \delta_\sim(s_\sim, \alpha) \in Q_i\} \\ &= \{\alpha \in \mathrm{Pref}(\mathcal{L}(\mathcal{D}_\sim)) \mid (\exists [\beta]_\sim \in Q_i\ \alpha \in [\beta]_\sim)\} \\ &= \{\alpha \in \mathrm{Pref}(\mathcal{L}(\mathcal{D}_\sim)) \mid U_\alpha = U_i\} = U_i.\end{aligned}$$

In other words, $\mathcal{D}_\sim$ witnesses that $\mathcal{L}$ is recognized by a $\mathcal{P}$-sortable DFA. Moreover:

(1) The number of states of $\mathcal{D}_\sim$ is clearly equal to the index of $\sim$.
(2) By Equation (6):

$$\alpha \equiv_{D_\sim} \beta \iff \delta_\sim(s_\sim, \alpha) = \delta_\sim(s_\sim, \beta) \iff [\alpha]_\sim = [\beta]_\sim \iff \alpha \sim \beta$$

so $\equiv_{D_\sim}$ and $\sim$ are the same equivalence relation.

Finally, suppose $\mathcal{B}$ is a $\mathcal{P}$-sortable DFA that recognizes $\mathcal{L}$. Notice that by Lemma 4.35 we have that $\equiv_{\mathcal{B}}$ is a $\mathcal{P}$-consistent, $\mathcal{P}$-convex, right-invariant equivalence relation on Pref($\mathcal{L}$) of finite index such that $\mathcal{L}$ is the union of some $\equiv_{\mathcal{B}}$-classes, so $\mathcal{D}_{\equiv_{\mathcal{B}}}$ is well-defined. Call $Q_{\mathcal{B}}$ the set of states of $\mathcal{B}$, and let $\phi : Q_{\equiv_{\mathcal{B}}} \to Q_{\mathcal{B}}$ be the function sending $[\alpha]_{\equiv_{\mathcal{B}}}$ into the state in $Q_{\mathcal{B}}$ reached by reading $\alpha$. Notice that $\phi$ is well-defined because by the definition of $\equiv_{\mathcal{B}}$ we obtain that all strings in $[\alpha]_{\equiv_{\mathcal{B}}}$ reach the same state of $\mathcal{B}$. It is easy to check that $\phi$ determines an isomorphism between $\mathcal{D}_{\equiv_{\mathcal{B}}}$ and $\mathcal{B}$. □

We now have all the required definitions to state our Myhill-Nerode theorem, which generalizes the one for Wheeler languages [2].

Theorem 4.37 (Convex Myhill-Nerode Theorem). *Let $\mathcal{L}$ be a language. Let $\mathcal{P}$ be a partition of Pref($\mathcal{L}$). The following are equivalent:*

*(1) $\mathcal{L}$ is recognized by a $\mathcal{P}$-sortable DFA.*
*(2) $\equiv^{\mathcal{P}}_{\mathcal{L}}$ has finite index.*
*(3) $\mathcal{L}$ is the union of some classes of a $\mathcal{P}$-consistent, $\mathcal{P}$-convex, right-invariant equivalence relation on Pref($\mathcal{L}$) of finite index.*

*Moreover, if one of the above statements is true (and so all the above statements are true), then there exists a unique minimum $\mathcal{P}$-sortable DFA recognizing $\mathcal{L}$ (that is, two $\mathcal{P}$-sortable DFAs recognizing $\mathcal{L}$ having the minimum number of states must be isomorphic).*

Proof. (1) → (2) It follows from Lemma 4.35.
(2) → (3) The desired equivalence relation is simply $\equiv^{\mathcal{P}}_{\mathcal{L}}$.
(3) → (1) It follows from Lemma 4.36.
Now, let us prove that the minimum DFA is $\mathcal{D}_{\equiv^{\mathcal{P}}_{\mathcal{L}}}$ as defined in Lemma 4.36. First, $\mathcal{D}_{\equiv^{\mathcal{P}}_{\mathcal{L}}}$ is well-defined because $\equiv^{\mathcal{P}}_{\mathcal{L}}$ is $\mathcal{P}$-consistent, $\mathcal{P}$-convex and right-invariant by definition; moreover, it has finite index and $\mathcal{L}$ is the union of some $\equiv^{\mathcal{P}}_{\mathcal{L}}$-equivalence classes by Lemma 4.35. Now, the number of states of $\mathcal{D}_{\equiv^{\mathcal{P}}_{\mathcal{L}}}$ is equal to the index of $\equiv^{\mathcal{P}}_{\mathcal{L}}$, or equivalently, of $\equiv_{\mathcal{D}_{\equiv^{\mathcal{P}}_{\mathcal{L}}}}$. On the other hand, let $\mathcal{B}$ be any $\mathcal{P}$-sortable DFA recognizing $\mathcal{L}$ non-isomorphic to $\mathcal{D}_{\equiv^{\mathcal{P}}_{\mathcal{L}}}$. Then $\equiv_{\mathcal{B}}$ is a refinement of $\equiv^{\mathcal{P}}_{\mathcal{L}}$ by Lemma 4.35, and it must be a strict refinement of $\equiv^{\mathcal{P}}_{\mathcal{L}}$, otherwise $\mathcal{D}_{\equiv^{\mathcal{P}}_{\mathcal{L}}}$ would be equal to $\mathcal{D}_{\equiv_{\mathcal{B}}}$, which by Lemma 4.36 is isomorphic to $\mathcal{B}$, a contradiction. We conclude that the index of $\equiv^{\mathcal{P}}_{\mathcal{L}}$ is smaller than the index of $\equiv_{\mathcal{B}}$, so again by Lemma 4.36 the number of states of $\mathcal{D}_{\equiv^{\mathcal{P}}_{\mathcal{L}}}$ is smaller than the number of states of $\mathcal{D}_{\equiv_{\mathcal{B}}}$ and so of $\mathcal{B}$. □

Notice that for a language $\mathcal{L}$ Definition 4.30 implies that $\text{width}^D(\mathcal{L}) = p$ if and only if (i) there exists a partition $\mathcal{P}$ of size $p$ such that $\mathcal{L}$ is recognized by a $\mathcal{P}$-sortable DFA and (ii) for every partition $\mathcal{P}'$ of size less than $p$ it holds that $\mathcal{L}$ is not recognized by a $\mathcal{P}'$-sortable DFA. As a consequence, $\text{width}^D(\mathcal{L}) = p$ if and only if the minimum cardinality of a partition $\mathcal{P}$ of Pref($\mathcal{L}$) that satisfies any of the statements in Theorem 4.37 is equal to $p$. Given a $\mathcal{P}$-sortable DFA recognizing $\mathcal{L}$, it can be shown that the minimum $\mathcal{P}$-sortable DFA recognizing $\mathcal{L}$ can be built in polynomial time by generalizing the algorithm in [1] (we do not provide the algorithmic details here because they would take us away from the main ideas that we want to convey).

## 5 WIDTH-AWARE ENCODINGS AND INDEXES FOR REGULAR LANGUAGES

In this section, we present compressed data structures for automata solving the *compression* and the *indexing* problems, that is Problems 5 and 6 of Section 2.4.

When presenting our data structures in detail, we will assume to be working with *integer* alphabets of the form $\Sigma = [0, \sigma - 1]$, that is, alphabets formed by all integers $\{0, 1, \ldots, \sigma - 1\}$. Our data structure results hold in the word RAM model with words of size $w \in \Theta(\log u)$ bits, where $u$ is the size of the input under consideration (for example, $u$ may be the size of an automaton or the length of a string, depending on the input of the algorithm under consideration). When not specified otherwise, the space of our data structures is measured in words.

Given an array $S = S[1]S[2] \ldots S[|S|]$, let $S[l, r] = S[l]S[l+1] \ldots S[r-1]S[r]$, if $1 \leq l \leq r \leq |S|$, and $S[l, r] = \emptyset$ if $l > r$.

Recall that the zero-order entropy of a sequence $S \in \Sigma^n$ of length $n$ over alphabet $\Sigma$ is $H_0(S) = \sum_{c \in \Sigma} \frac{|S|_c}{n} \log_2 \frac{n}{|S|_c}$, where $|S|_c$ denotes the number of occurrences of character $c$ in $S$. We will use some well-known properties of $H_0(S)$: the quantity $nH_0(S)$ is a lower bound to the length of any encoding of $S$ that encodes each character independently from the others via a prefix code of the alphabet $\Sigma$, and in particular $H_0(S) \leq \log_2 \sigma$.

### 5.1 Path Coherence and Lower Bounds

The reason why Wheeler automata admit an efficient indexing mechanism lies in two key observations: (i) on finite total orders a convex set can be expressed with $O(1)$ words by specifying its endpoints, and (ii) the set of states reached by a path labeled with a given string $\alpha$ forms a convex set (*path-coherence*). We now show that the convex property holds true also for co-lex orders by generalizing the result in [47].

Lemma 5.1 (Path-coherence). *Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA, $\leq$ be a co-lex order on $\mathcal{N}$, $\alpha \in \Sigma^*$, and $U$ be a $\leq$-convex set of states. Then, the set $U'$ of all states in $Q$ that can be reached from $U$ by following edges whose labels, when concatenated, yield $\alpha$, is still a (possibly empty) $\leq$-convex set.*

Proof. We proceed by induction on $|\alpha|$. If $|\alpha| = 0$, then $\alpha = \varepsilon$, and we are done. Now assume $|\alpha| \geq 1$. We can write $\alpha = \alpha' a$, with $\alpha' \in \Sigma^*$, $a \in \Sigma$. Let $u, v, z \in Q$ such that $u < v < z$ and $u, z \in U'$. We must prove that $v \in U'$. By the inductive hypothesis, the set $U''$ of all states in $Q$ that can be reached from some state in $U$ by following edges whose labels, when concatenated, yield $\alpha'$, is a $\leq$-convex set. In particular, there exist $u', z' \in U''$ such that $u \in \delta(u', a)$ and $z \in \delta(z', a)$. Since $a \in \lambda(u) \cap \lambda(z)$ and $u < v < z$, then $\lambda(v) = \{a\}$ (otherwise by Axiom 1 we would obtain a contradiction), so there exists $v' \in Q$ such that $v \in \delta(v', a)$. From $u < v < z$ and Axiom 2 we obtain $u' \leq v' \leq z'$. Since $u', z' \in U''$ and $U''$ is a $\leq$-convex set, then $v' \in U''$, and so $v \in U'$. □

Note that Corollary 2.14 in Section 2.2 also follows from Lemma 5.1 by picking $U = \{s\}$, because then $U' = I_\alpha$.

As we will see, the above result implies that indexing mechanism can be extended to arbitrary finite automata by updating *one* $\leq$-convex set for each character of the query pattern. This, however, does not mean that, in general, indexing can be performed as efficiently as on Wheeler automata: as we show next, in general, it is not possible to represent a $\leq$-convex set in a partial order using constant space.

Lemma 5.2. *The following hold:*

1. *Any partial order $(V, \leq)$ of width $p$ has at least $2^p$ distinct $\leq$-convex subsets.*
2. *For any $n$ and $p$ such that $1 \leq p \leq n$, there exists a partial order $(V, \leq)$ of width $p$ and $|V| = n$ with at least $(n/p)^p$ distinct $\leq$-convex subsets.*

Proof. (1) Since $V$ has width $p$, there exists an antichain $A$ of cardinality $p$. It is easy to see that any subset $I \subseteq A$ is a distinct $\leq$-convex set. The bound $2^p$ follows. (2) Consider a partial order formed by $p$ mutually-incomparable total orders $V_i$, all having $n/p$ elements. Since any total order of cardinality $n/p$ has $(n/p+1)(n/p)/2+1$ distinct convex sets and any combination of $\leq_{V_i}$-convex sets forms a distinct $\leq$-convex set, we obtain at least

$$\prod_{i=1}^{p}((n/p+1)(n/p)/2+1) \geq \prod_{i=1}^{p} n/p = (n/p)^p$$

distinct $\leq$-convex sets. □

*Remark 5.3.* Given an NFA $\mathcal{N}$ with $n$ states and a co-lex order $\leq$ of width $p$ on $\mathcal{N}$, Lemma 5.2 implies an information-theoretic lower bound of $p$ bits for expressing a $\leq$-convex set, which increases to $\Omega(p\log(n/p))$ bits in the worst case. This means that, up to (possibly) a logarithmic factor, in the word RAM model $\Omega(p)$ time is needed to manipulate one $\leq$-convex set.

*Remark 5.4.* If $(V,\leq)$ is a partial order, $V' \subseteq V$ and $U$ is a convex subset of $(V,\leq)$, then $U \cap V'$ is a convex set over the restricted partial order $(V',\leq_{V'})$. In particular, if $\{V_i \mid 1 \leq i \leq p\}$ is a partition of $V$ then any $\leq$-convex set $U$ is the disjoint union of $p$ (possibly empty) sets $U_1,\ldots,U_p$, where $U_i = U \cap V_i$ is a convex set over the restriction $(V_i,\leq_{V_i})$.

The above remarks motivate the following strategy. Letting $p$ be the width of a partial order $\leq$, by Dilworth's theorem [37] there exists a $\leq$-chain partition $\{Q_i \mid 1 \leq i \leq p\}$ of $Q$ into $p$ chains. Then, Remark 5.4 implies that a $\leq$-convex set can be encoded by at most $p$ convex sets, each contained in a distinct chain, using $O(p)$ words. This encoding is essentially optimal by Remark 5.3.

Using the above mentioned strategy we can now refine Lemma 5.1 (path-coherence) and its corollary (Corollary 2.14).

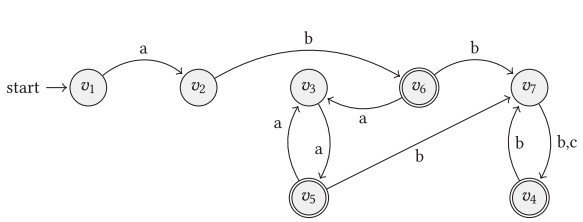
Lemma 5.5. *Let $\mathcal{N} = (Q,s,\delta,F)$ be an NFA, $\leq$ be a co-lex order on $\mathcal{N}$, $\{Q_i\}_{i=1}^{p}$ be a $\leq$-chain partition of $\mathcal{N}$, $\alpha \in \Sigma^*$, and $U$ be a $\leq$-convex set of states. Then, the set $U'$ of all states in $Q$ that can be reached from $U$ by following edges whose labels, when concatenated, yield $\alpha$, is the disjoint union of $p$ (possibly empty) sets $U'_1,\ldots,U'_p$, where $U'_i = U' \cap Q_i$ is $\leq_{Q_i}$-convex, for $i = 1,\ldots,p$.*

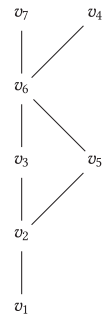
*In particular, if $\alpha \in Pref(\mathcal{L}(\mathcal{A}))$ then, $I_\alpha$ is the disjoint union of $p$ (possibly empty) sets $I^1_\alpha,\ldots,I^p_\alpha$, where $I^i_\alpha = I_\alpha \cap Q_i$ is $\leq_{Q_i}$-convex, for $i = 1,\ldots,p$.*

### 5.2 Encoding DFAs and Languages: The Automaton BWT (aBWT)

Let us define a representation of an automaton that is a generalization of the well-known BWT of a string [23]. We call this generalization the ***automaton Burrows-Wheeler transform*** (**aBWT**) and, just like the BWT of a string is an encoding of the string (that is, distinct strings have distinct BWTs), we will show that the aBWT of a DFA is an encoding of the DFA. We will also see that, on NFAs, the aBWT allow us to reconstruct the accepted language and to efficiently solve the string matching problem (Problem 6), but in general it is not an encoding since it is not sufficient to reconstruct the NFA's topology. A variant, using slightly more space and encoding NFAs, will be presented in Section 5.4.

The aBWT is given for an automaton $\mathcal{N} = (Q,s,\delta,F)$ and it depends on a co-lex order $\leq$ endowed with a fixed $\leq$-chain partition $\{Q_i \mid 1 \leq i \leq p\}$ of $Q$ (we assume $s \in Q_1$, so $s$ is the first element of $Q_1$). An intuition behind the aBWT is provided in Figure 9: after sorting the states in a total order which agrees with the co-lex order $\leq$ on pairs whose elements belong to the same class of the partition $\{Q_i \mid 1 \leq i \leq p\}$ and drawing the transition function's adjacency matrix in this order, we build five sequences collecting the chain borders (`CHAIN`), a boolean flag per state




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | CHAIN | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| | | FINAL | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| | | IN_DEG | 1 | 01 | 001 | 001 | 01 | 01 | 0001 |
| OUT_DEG | OUT | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 01 | (1,a) | 1 | | (1,a) | | | | | |
| 01 | (2,b) | 2 | | | | | | (2,b) | |
| 01 | (2,a) | 3 | | | | | (2,a) | | |
| 01 | (2,b) | 4 | | | | | | | (2,b) |
| 001 | (1,a),(2,b) | 5 | | | (1,a) | | | | (2,b) |
| 001 | (1,a),(2,b) | 6 | | | (1,a) | | | | (2,b) |
| 001 | (1,b),(1,c) | 7 | | | | (1,b)<br>(1,c) | | | |

Fig. 9. A DFA $\mathcal{D}$ accepting $\mathcal{L} = ab(aa)^*(b(b+c))^*$, together with the Hasse diagram of its maximum co-lex order $\leq$ and the adjacency matrix of $\mathcal{D}$. In the following examples we consider the $\leq$-chain partition given by $Q_1 = \{v_1, v_2, v_3, v_4\}$, $Q_2 = \{v_5, v_6, v_7\}$. The adjacency matrix is sorted according to the total order $Q = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$. The two different shades of gray divide the edges by destination chain (either 1 or 2). Each edge is represented in this matrix as the pair $(i, c)$, where $i$ is the destination chain and $c \in \Sigma$ is the edge's label. This way of visualizing the adjacency matrix can be viewed as a two-dimensional representation of the automaton Burrows-Wheeler transform (aBWT, Definition 5.6). The aBWT can be linearized in five sequences, as shown here and in Example 5.7.

marking final states (FINAL), the states' in-degrees (IN_DEG), the states' out-degrees (OUT_DEG), and the states' labels and destination chains (OUT).

Definition 5.6 *(aBWT of an Automaton).* Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA and let $e = |\delta|$ be the number of $\mathcal{N}$-transitions. Let $\leq$ be a co-lex order on $\mathcal{N}$, and let $\{Q_i \mid 1 \leq i \leq p\}$ be a $\leq$-chain partition of $Q$, where w.l.o.g. $s \in Q_1$. Let $\pi(v)$ be the unique map such that $v \in Q_{\pi(v)}$ and consider the total state order $Q = \{v_1, \ldots, v_n\}$ such that, for every $1 \leq i < j \leq n$, it holds[6] $\pi(v_i) < \pi(v_j) \vee (\pi(v_i) = \pi(v_j) \wedge v_i < v_j)$. The *automaton Burrows-Wheeler transform* $\mathtt{aBWT}(\mathcal{N}, \leq, \{Q_i \mid 1 \leq i \leq p\})$ of $(\mathcal{N}, \leq, \{Q_i \mid 1 \leq i \leq p\})$ consists of the following sequences.

- CHAIN $\in \{0,1\}^n$ is such that the $i$th bit is equal to 1 if and only if $v_i$ is the first state of some chain $Q_j$.
- FINAL $\in \{0,1\}^n$ is such that the $i$th bit is equal to 1 if and only if $v_i \in F$.
- IN_DEG $\in \{0,1\}^{e+n}$ stores the nodes' in-degrees in unary. More precisely, (1) IN_DEG contains exactly $n$ characters equal to 1, (2) IN_DEG contains exactly $e$ characters equal to 0, and (3) the number of zeros between the $(i-1)$-th character equal to one (or the beginning of the sequence if $i = 1$) and the $i$th character equal to 1 yields the in-degree of $v_i$.

[6]Notice the overload on symbol $\leq$, also used to indicate the co-lex order among states.

— OUT_DEG $\in \{0,1\}^{e+n}$ stores the nodes' out-degrees in unary. More precisely, (1) OUT_DEG contains exactly $n$ characters equal to 1, (2) OUT_DEG contains exactly $e$ characters equal to 0, and (3) the number of zeros between the $(i-1)$-th character equal to one (or the beginning of the sequence if $i = 1$) and the $i$th character equal to 1 yields the out-degree of $v_i$.
— OUT stores the edges' labels and destination chains, as follows. Sort all edges $(v_j, v_i, c)$ by their starting state $v_j$ according to their index $j$. Edges originating from the same state are further sorted by their label $c$. Edges sharing the starting state and label are further sorted by destination node $v_i$. Then, OUT is obtained by concatenating the pairs $(\pi(v_i), c)$ for all edges $(v_j, v_i, c)$ sorted in this order.

*Example 5.7.* The aBWT of $(\mathcal{D}, \leq, \{Q_i \mid 1 \leq i \leq 2\})$ in Figure 9 consists of the following sequences:

— CHAIN = 1000100.
— FINAL = 0001110.
— IN_DEG = 10100100101010001.
— OUT_DEG = 01010101001001001.
— OUT = $(1,a)(2,b)(2,a)(2,b)(1,a)(2,b)(1,a)(2,b)(1,b)(1,c)$.

It is not hard to show that the aBWT generalizes all existing approaches [17, 23, 44, 47, 63, 64], for which $p = 1$ always holds (and so sequences CHAIN and the first components of the pairs in OUT are uninformative). For example, on strings also OUT_DEG and IN_DEG are uninformative (FINAL does not apply); the only sequence left is the concatenation of the second components of the pairs in OUT, that is, the classic Burrows-Wheeler transform (to be precise, its co-lexicographic variant).

In this section, we will prove that if we only know the aBWT of an automaton we can reconstruct all the sets $I^i_\alpha$ of Lemma 5.5 (we recall that $I^i_\alpha$ is the set of all states in the $i$th chain being connected with the source by a path labeled $\alpha$), and in particular we can retrieve the language of the automaton. To this end, we first define some auxiliary sets of states of an NFA — $S(\alpha)$ and $L(\alpha)$ — and we prove that, for any $1 \leq i \leq p$, on the $i$th chain the convex set corresponding to $I^i_\alpha$ lays between (the convex sets) $S(\alpha) \cap Q_i$ and $L(\alpha) \cap Q_i$. Intuitively, $S(\alpha)$ (respectively, $L(\alpha)$) is the set of all states $u$ whose associated regular language $I_u$ contains only strings co-lexicographically strictly smaller (respectively, larger) than $\alpha$.

*Definition 5.8.* Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA, $\leq$ be a co-lex order on $\mathcal{N}$, and $\{Q_i \mid 1 \leq i \leq p\}$ be a $\leq$-chain partition of $Q$. Let $\alpha \in \Sigma^*$. Define:

$$S(\alpha) = \{u \in Q \mid (\forall \beta \in I_u)(\beta \prec \alpha)\}$$
$$L(\alpha) = \{u \in Q \mid (\forall \beta \in I_u)(\alpha \prec \beta)\}.$$

Moreover, for every $i = 1, \ldots, p$ define $S_i(\alpha) = S(\alpha) \cap Q_i$ and $L_i(\alpha) = L(\alpha) \cap Q_i$.

In the following, we see a $\leq$-chain $Q_i$ as an array of sorted elements, so $Q_i[j]$ and $Q_i[1,k]$ denote the $j$th smallest state in $Q_i$ and the $k$ smallest states in $Q_i$, respectively.

In Lemma 5.9 we show that in order to compute $I_\alpha$ it will be sufficient to compute $S(\alpha)$ and $L(\alpha)$.

Lemma 5.9. *Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA, $\leq$ be a co-lex order on $\mathcal{N}$, $\{Q_i \mid 1 \leq i \leq p\}$ be a $\leq$-chain partition of $Q$, and $\alpha \in \Sigma^*$.*

*(1) If $u, v \in Q$ are such that $u \leq v$ and $v \in S(\alpha)$, then $u \in S(\alpha)$. In particular, for every $i = 1, \ldots, p$ there exists $0 \leq l_i \leq |Q_i|$ such that $S_i(\alpha) = Q_i[1, l_i]$ (namely, $l_i = |S_i(\alpha)|$).*

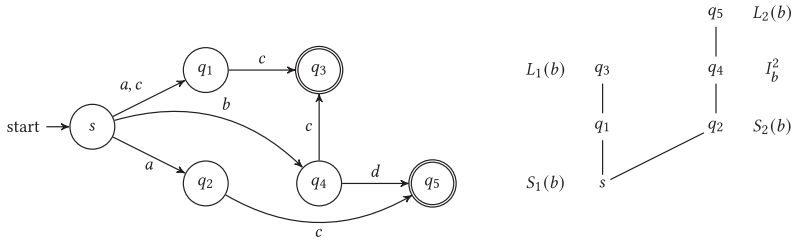



Fig. 10. Consider the NFA in the figure and the Hasse diagram of a co-lex order $\leq$ with chain partition $Q_1 = \{s, q_1, q_3\}$ and $Q_2 = \{q_2, q_4, q_5\}$. If we consider the word $b$, then, on the one hand, $I^1_b = \emptyset$ and $\{S_1(b) = \{s\}, L_1(b) = \{q_3\}\}$ is not a partition of $Q_1$. On the other hand, since $I^2_b = \{q_4\} \neq \emptyset$, then $\{S_2(b) = \{q_2\}, I^2_b, L_1(b) = \{q_5\}\}$ is an ordered $Q_2$-partition.

(2) *If $u, v \in Q$ are such that $u \leq v$ and $u \in L(\alpha)$, then $v \in L(\alpha)$. In particular, for every $i = 1, \ldots, p$ there exists $1 \leq r_i \leq |Q_i| + 1$ such that $L_i(\alpha) = Q_i[r_i, |Q_i|]$ (namely, $r_i = |Q_i| - |L_i(\alpha)| + 1$).*
(3) *$I_\alpha$, $S(\alpha)$, and $L(\alpha)$ are pairwise disjoint. In particular, it always holds that $l_i < r_i$.*
(4) *Let $1 \leq i \leq p$. If $I^i_\alpha \neq \emptyset$, then $I^i_\alpha = Q_i[l_i + 1, r_i - 1]$, that is, $\{S_i(\alpha), I^i_\alpha, L_i(\alpha)\}$ is an ordered partition of $Q_i$.*

Proof.

(1) Let $\beta \in I_u$. We must prove that $\beta \prec \alpha$. Now, if $\beta \in I_v$, from $v \in S(\alpha)$ we obtain $\beta \prec \alpha$. If $\beta \notin I_v$, then for any $\gamma \in I_v$ we have $\beta \prec \gamma$ by Lemma 2.13. Again, we have $\gamma \prec \alpha$, so we conclude $\beta \prec \alpha$.
(2) Analogous to the previous point.
(3) We have $I_\alpha \cap S(\alpha) = \emptyset$ because if $u \in I_\alpha$, then $\alpha \in I_u$, so $u \notin S(\alpha)$. Similarly, $I_\alpha \cap L(\alpha) = \emptyset$. Finally, we have $S(\alpha) \cap L(\alpha) = \emptyset$ because if there existed $u \in S(\alpha) \cap L(\alpha)$, then for any $\beta \in I_u$ (there exists at least one such $\beta$ since $I_u \neq \emptyset$) we would obtain $\beta \prec \alpha \prec \beta$, a contradiction.
(4) To begin with, let us prove that, for every $v \in I_\alpha$, (1) if $u < v$, then either $u \in I_\alpha$ or $u \in S(\alpha)$, and (2) if $v < z$, then either $z \in I_\alpha$ or $z \in L(\alpha)$. We only prove (1), the proof of (2) being analogous. Assume that $u \notin I_\alpha$, and let $\beta \in I_u$. We must prove that $\beta \prec \alpha$ and, since $\alpha \in I_v \setminus I_u$, this follows from Lemma 2.13.
Now, let $1 \leq i \leq p$ be such that $I^i_\alpha \neq \emptyset$, and let us prove that $\{S_i(\alpha), I^i_\alpha, L_i(\alpha)\}$ is an ordered partition of $Q_i$. Consider $u \in I^i_\alpha$. Then, if $v \in Q_i \setminus I^i_\alpha$ we have either $v < u$ or $u < v$, hence what we have proved above implies that either $v \in S_i(\alpha)$ or $v \in L_i(\alpha)$. Therefore, if $I^i_\alpha \neq \emptyset$ then $\{S_i(\alpha), I^i_\alpha, L_i(\alpha)\}$ is an ordered partition of $Q_i$ and point 4 follows.

□

*Remark 5.10.* Notice that if $I^i_\alpha = \emptyset$ then $\{S_i(\alpha), L_i(\alpha)\}$ is not, in general, an ordered partition of $Q_i$, as shown in Figure 10.

Our next step is to show how to recursively compute the sets $S(\alpha)$ and $L(\alpha)$ defined above. We begin with the following two lemmas.

Lemma 5.11. *Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA, $\leq$ be a co-lex order on $\mathcal{N}$, $\{Q_i \mid 1 \leq i \leq p\}$ be a $\leq$-chain partition of $Q$, $\alpha' \in \Sigma^*$, $a \in \Sigma$, and $u \in Q$.*

*(1) $u \in S(\alpha' a)$ if and only if (1) $max_{\lambda(u)} \leq a$ and (2) if $u' \in Q$ is such that $u \in \delta(u', a)$, then $u' \in S(\alpha')$.*

*(2)* $u \in L(\alpha' a)$ *if and only if (1)* $a \preceq min_{\lambda(u)}$ *and (2) if* $u' \in Q$ *is such that* $u \in \delta(u', a)$*, then* $u' \in L(\alpha')$*.*

Proof. Let us prove the first statement.

($\Rightarrow$) Let $c \in \Sigma$ such that $c \in \lambda(u) \setminus \{\#\}$. Let $u' \in Q$ such that $u \in \delta(u', c)$, and let $\beta' \in I_{u'}$. Then $\beta' c \in I_u$, so from $u \in S(\alpha' a)$ we obtain $\beta' c \prec \alpha' a$, which implies $c \preceq a$. Now assume that $c = a$. Suppose for sake of contradiction that $u' \notin S(\alpha')$. This means that there exists $\gamma' \in I_{u'}$ such that $\alpha' \preceq \gamma'$. This implies $\alpha' a \preceq \gamma' a$ and, since $\gamma' a \in I_u$, we obtain $u \notin S(\alpha' a)$, a contradiction.

($\Leftarrow$) Let $\beta \in I_u$. We must prove that $\beta \prec \alpha' a$. If $\beta = \varepsilon$ we are done, because $\varepsilon \prec \alpha' a$. Now assume that $\beta = \beta' b$. This means that there exists $u' \in Q$ such that $u \in \delta(u', b)$ and $\beta' \in I_{u'}$. We know that $b \preceq a$. If $b \prec a$, then $\beta \prec \alpha' a$ and we are done. If $b = a$, then $u' \in S(\alpha')$, so $\beta' \prec \alpha'$, which implies $\beta \prec \alpha' a$.

The proof of the second statement is analogous (the only difference being that in ($\Leftarrow$) it must necessarily be $\beta \neq \varepsilon$, because $a \preceq \min_{\lambda(u)}$). □

Lemma 5.12. *Let* $\mathcal{N} = (Q, s, \delta, F)$ *be an NFA,* $\preceq$ *be a co-lex order on* $\mathcal{N}$*,* $\{Q_i \mid 1 \leq i \leq p\}$ *be a* $\preceq$*-chain partition of* $Q$*,* $\alpha' \in \Sigma^*$*, and* $a \in \Sigma$*. Fix* $1 \leq i \leq p$*, and let* $S_i(\alpha' a) = Q_i[1, l_i]$ *and* $L_i(\alpha' a) = Q_i[r_i, |Q_i|]$*.*

*(1) If* $u' \in S(\alpha')$ *and* $u \in Q_i$ *are such that* $u \in \delta(u', a)$*, then* $u \in Q_i[1, \min\{l_i + 1, |Q_i|\}]$*.*
*(2) If* $u' \in L(\alpha')$ *and* $u \in Q_i$ *are such that* $u \in \delta(u', a)$*, then* $u \in Q_i[\max\{r_i - 1, 1\}, |Q_i|]$*.*

Proof. We only prove the first statement, the proof of the second statement being entirely analogous. We can assume $l_i < |Q_i| - 1$, otherwise the conclusion is trivial. If $a \prec \max(\lambda(Q_i[l_i + 1]))$ the conclusion is immediate by Axiom 1, so we can assume $\max(\lambda(Q_i[l_i + 1])) \preceq a$. We know that $Q_i[l_i + 1] \notin S(\alpha' a)$, so by Lemma 5.11 there exists $v' \in Q$ such that $Q_i[l_i + 1] \in \delta(v', a)$ and $v' \notin S(\alpha')$. Suppose for sake of contradiction that $Q_i[l_i + 1] \prec u$. By Axiom 2 we obtain $v' \preceq u'$. From $u' \in S(\alpha')$ and Lemma 5.9 we conclude $v' \in S(\alpha')$, a contradiction. □

The following definition is instrumental in giving an operative variant of Lemma 5.11 (i.e., Lemma 5.14) to be used in our algorithms.

*Definition 5.13.* Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA, $\preceq$ be a co-lex order on $\mathcal{N}$, and $\{Q_i \mid 1 \leq i \leq p\}$ be a $\preceq$-chain partition of $Q$. Let $U \subseteq Q$. We denote by $\mathtt{in}(U, a)$ the number of edges labeled with character $a$ that enter states in $U$:

$$\mathtt{in}(U, a) = |\{(u', u) \mid u' \in Q, u \in U, u \in \delta(u', a)\}|.$$

We denote by $\mathtt{out}(U, i, a)$ the number of edges labeled with character $a$ that leave states in $U$ and enter the $i$th chain:

$$\mathtt{out}(U, i, a) = |\{(u', u) \mid u' \in U, u \in Q_i, u \in \delta(u', a)\}|.$$

In the following lemma, we show how to compute the convex sets corresponding to $S_i(\alpha' a)$ and $L_i(\alpha' a)$, for every $i = 1, \ldots, p$, using the above definitions.

Lemma 5.14. *Let* $\mathcal{N} = (Q, s, \delta, F)$ *be an NFA,* $\preceq$ *be a co-lex order on* $\mathcal{N}$*,* $\{Q_i \mid 1 \leq i \leq p\}$ *be a* $\preceq$*-chain partition of* $Q$*,* $\alpha' \in \Sigma^*$*,* $a \in \Sigma$*, and* $\alpha = \alpha' a$*. For every* $j = 1, \ldots, p$*, let* $S_j(\alpha') = Q_j[1, l'_j]$ *and* $L_j(\alpha') = Q_j[r'_j, |Q_j|]$*. Fix* $1 \leq i \leq p$*, and let* $S_i(\alpha) = Q_i[1, l_i]$ *and* $L_i(\alpha) = Q_i[r_i, |Q_i|]$*.*

*(1) Let* $x = \mathtt{out}(S(\alpha), i, a) = \sum_{j=1}^{p} \mathtt{out}(Q_j[1, l'_j], i, a)$*. Then,* $l_i$ *is the largest integer* $0 \leq k \leq |Q_i|$ *such that (i)* $\mathtt{in}(Q_i[1, k], a) \leq x$*, and (ii) if* $k \geq 1$*, then* $\max(\lambda(Q_i[k])) \preceq a$*.*

(2) *Let* $y = \mathtt{out}(L(\alpha), i, a) = \sum_{j=1}^{p} \mathtt{out}(Q_j[r'_j, |Q_j|], i, a)$. *Then,* $r_i$ *is the smallest integer* $1 \le k \le |Q_i| + 1$ *such that (i)* $\mathtt{in}(Q_i[k, |Q_i|], a) \le y$, *and (ii) if* $k \le |Q_i|$, *then* $a \preceq \min(\lambda(Q_i[k]))$.

Proof. Again, we just prove the first statement since the proof of the second one is analogous.

Let $z_i$ be the largest integer $0 \le k \le |Q_i|$ such that (i) $\mathtt{in}(Q_i[1, k], a) \le x$, and (ii) if $k \ge 1$, then $\max(\lambda(Q_i[k])) \preceq a$. We want to prove that $l_i = z_i$.

($\le$) The conclusion is immediate if $l_i = 0$, so we can assume $l_i \ge 1$. It will suffice to prove that $\mathtt{in}(Q_i[1, l_i], a) \le x$ and $\max(\lambda(Q_i[l_i])) \preceq a$. This follows from Lemma 5.11 and the definition of $x$.

($\ge$) The conclusion is immediate if $l_i = |Q_i|$, so we can assume $l_i < |Q_i|$. We only have to prove that if $l_i + 1 \le k \le |Q_i|$, then either $\mathtt{in}(Q_i[1, k], a) > x$ or $\max(\lambda(Q_i[k])) \succ a$. By Axiom 1, it will suffice to prove that we have $\mathtt{in}(Q_i[1, l_i + 1], a) > x$ or $\max(\lambda(Q_i[l_i + 1])) \succ a$. Assume that $\max(\lambda(Q_i[l_i + 1])) \preceq a$. Since $Q_i[l_i + 1] \notin S(\alpha)$, by Lemma 5.11 there exists $v' \in Q$ such that $Q_i[l_i + 1] \in \delta(v', a)$ and $v' \notin S(\alpha')$. We will conclude that $\mathtt{in}(Q_i[1, l_i + 1], a) > x$ if we show that for every $j = 1, \dots, p$, if $u' \in Q_j$ and $u \in Q_i$ are such that $u' \in Q_j[1, l'_j]$ (and so $u' \in S(\alpha')$) and $u \in \delta(u', a)$, then it must be $u \in Q_i[1, l_i + 1]$. This follows from Lemma 5.12. □

We now use Lemma 5.14 to retrieve the language of the automaton starting from the aBWT.

Lemma 5.15. *Let* $\mathcal{N} = (Q, s, \delta, F)$ *be an NFA,* $\le$ *be a co-lex order on* $\mathcal{N}$, *and* $\{Q_i \mid 1 \le i \le p\}$ *be a* $\le$*-chain partition of* $Q$, *with* $s \in Q_1$. *Let* $v_1, \dots, v_n$ *be the ordering of* $Q$ *defined in Definition 5.6. Assume that we do not know* $\mathcal{N}$, *but we only know* $\mathtt{aBWT}(\mathcal{N}, \le, \{Q_i \mid 1 \le i \le p\})$. *Then, for every* $\alpha \in \Sigma^*$ *we can retrieve the set* $\{i \in \{1, \dots, n\} \mid \alpha \in I_{v_i}\}$, *which yields* $\delta(s, \alpha)$.

Proof. First, let us prove that for every $k = 1, \dots, n$, we can retrieve the labels — with multiplicities — of all edges entering $v_k$. By scanning CHAIN we can retrieve the integers $k_1$ and $k_2$ such that the states in chain $Q_i$ are $v_{k_1}, v_{k_1+1}, \dots, v_{k_2-1}, v_{k_2}$. By scanning OUT, which stores the label and the destinational chain of each edge, we can retrieve how many edges enter chain $Q_i$, and we can retrieve the labels - with multiplicities - of all edges entering chain $Q_i$. By considering the substring of IN_DEG between the $(k_1 - 1)$-th one and the $k_2$-th one we can retrieve the in-degrees of all states in chain $Q_i$. Since we know the labels - with multiplicities - of all edges entering chain $Q_i$ and the in-degrees of all states in chain $Q_i$, by Axiom 1 we can retrieve the labels - with multiplicities - of all edges entering each node in chain $Q_i$: order the multiset of incoming edge labels, scan the nodes in $Q_i$ in order, and assign the labels to each node in $Q_i$ in agreement with their in-degrees.

Let us prove that for every $i = 1, \dots, p$ we can retrieve the integers $l_i$ and $r_i$ such that $S_i(\alpha) = Q_i[1, l_i]$ and $L_i(\alpha) = Q_i[r_i, |Q_i|]$. We proceed by induction on $|\alpha|$. If $|\alpha| = 0$, then $\alpha = \varepsilon$, so for every $i = 1, \dots, p$ we have $l_i = 0$, for every $i = 2, \dots, p$ we have $r_i = 1$, and $r_1 = 2$. Now, assume $|\alpha| > 0$. We can write $\alpha = \alpha' a$, with $\alpha' \in \Sigma^*$ and $a \in \Sigma$. By the inductive hypothesis, for $j = 1, \dots, p$ we know the integers $l'_j$ and $r'_j$ such that $S_j(\alpha') = Q_j[1, l'_j]$ and $L_j(\alpha') = Q_j[r'_j, |Q_j|]$. Notice that by using OUT_DEG and OUT we can compute $\mathtt{out}(Q_j[1, l'_j], i, a)$ for every $j = 1, \dots, p$ (see Definition 5.13). Since we know the labels - with multiplicities - of all edges entering each state, we can also compute $\mathtt{in}(Q_i[c, d], a)$ and $\lambda(Q_i[k]))$ for every $i = 1, \dots, p$, $1 \le c \le d \le |Q_i|$, and $1 \le k \le |Q_i|$. By Lemma 5.14 we conclude that we can compute $l_i$ and $r_i$ for every $i = 1, \dots, p$, and we are done.

Now, let us prove that for every $\alpha \in \Sigma^*$ we can retrieve the set $\{i \in \{1, \dots, n\} \mid \alpha \in I_{v_i}\}$. We proceed by induction on $|\alpha|$. If $|\alpha| = 0$, then $\alpha = \varepsilon$ and $\{i \in \{1, \dots, n\} \mid \varepsilon \in I_{v_i}\} = \{1\}$. Now, assume $|\alpha| > 0$. We can write $\alpha = \alpha' a$, with $\alpha' \in \Sigma^*$ and $a \in \Sigma$. By the inductive hypothesis, we know $\{i \in \{1, \dots, n\} \mid \alpha' \in I_{v_i}\}$. For every $i = 1, \dots, p$ we decide whether $I^i_\alpha \neq \emptyset$ by using $\{i \in \{1, \dots, n\} \mid \alpha' \in I_{v_i}\}$, OUT_DEG and OUT. If $I^i_\alpha \neq \emptyset$, then by Lemma 5.9 we know that

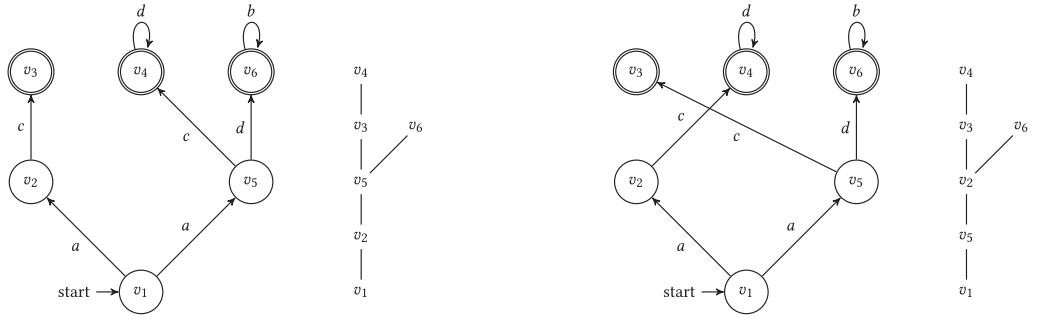



Fig. 11. Consider the two non-isomorphic NFAs $\mathcal{N}_1$ and $\mathcal{N}_2$ in the figure, both with set of states $Q = \{v_1, v_2, v_3, v_4, v_5, v_6\}$. Let $\leq_1$ and $\leq_2$ be the maximal co-lex orders given by the Hasse diagrams shown in the figure, and notice that in both cases if we consider $Q_1 = \{v_1, v_2, v_3, v_4\}$ and $Q_2 = \{v_5, v_6\}$ we obtain a minimum-size chain partition $\mathcal{Q} = \{Q_1, Q_2\}$. It is easy to check that $\mathtt{aBWT}(\mathcal{N}_1, \leq_1, \mathcal{Q}) = \mathtt{aBWT}(\mathcal{N}_2, \leq_2, \mathcal{Q})$ because in both cases we have $\mathtt{CHAIN} = 100010$, $\mathtt{FINAL} = 001101$, $\mathtt{OUT\_DEG} = 0010110100101$, $\mathtt{OUT} = (1, a)(2, a)(1, c)(1, d)(1, c)(2, d)(2, b)$, and $\mathtt{IN\_DEG} = 1010100101001$. Consistently with Theorem 5.19, we have that $\mathcal{N}_1$ and $\mathcal{N}_2$ are not distinguished by their paths.

$I^i_\alpha = Q_i[l_i + 1, r_i - 1]$, and we know how to determine $l_i$ and $r_i$. Hence, we can easily compute $\{i \in \{1, \ldots, n\} \mid \alpha \in I_{v_i}\}$. □

Corollary 5.16. *If* $\mathtt{aBWT}(\mathcal{N}, \leq, \{Q_i \mid 1 \leq i \leq p\}) = \mathtt{aBWT}(\mathcal{N}', \leq', \{Q'_i \mid 1 \leq i \leq p'\})$*, then:*

*(1)* $p = p'$*;*
*(2) for every* $1 \leq i \leq p$ *we have* $|Q_i| = |Q'_i|$*;*
*(3)* $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{N}')$*.*

Proof. Since $\mathcal{N}$ and $\mathcal{N}'$ share the sequence $\mathtt{CHAIN}$, it must be $p = p'$ and $|Q_i| = |Q'_i|$ for every $i$. Fix a string $\alpha \in \Sigma^*$. Then, by Lemma 5.15 we conclude that the set $\{i \in \{1, \ldots, n\} \mid \alpha \in I_{v_i}\}$ is the same for both $\mathcal{N}$ and $\mathcal{N}'$. Since $\mathcal{N}$ and $\mathcal{N}'$ share also the sequence $\mathtt{FINAL}$, we conclude that $\alpha$ is accepted by $\mathcal{N}$ if and only if it is accepted by $\mathcal{N}'$. □

Corollary 5.16 ensures that $\mathtt{aBWT}(\mathcal{N}, \leq, \{Q_i \mid 1 \leq i \leq p\})$ is enough to reconstruct the language $\mathcal{L}(\mathcal{N})$ of an NFA. Similarly to the string case, however (where the BWT is augmented with light data structures in order to achieve efficient indexing with the *FM-index* [45]), we will need additional data structures built on top of the aBWT in order to solve efficiently string matching queries. In Section 5.3 we will show how to extend the FM-index to automata by augmenting the aBWT with light data structures.

While Corollary 5.16 establishes that the aBWT preserves the automaton's language, it does not state anything about whether it preserves the automaton's topology. In fact, we now show that this is not, in general, the case.

*Definition 5.17.* Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA. We say that $\mathcal{N}$ is *distinguished by its paths* if for every $v \in Q$ there exists $\alpha \in \Sigma^*$ such that $I_\alpha = \{v\}$.

*Remark 5.18.* If an NFA is not distinguished by its paths, then, in general, we cannot retrieve its topology from its aBWT, because there exist two non-isomorphic NFAs having the same aBWT: see Figure 11 for an example.

Let us prove that $\mathtt{aBWT}(\mathcal{N}, \leq, \{Q_i \mid 1 \leq i \leq p\})$ is a one-to-one encoding for the class of automata which are distinguished by paths.

Theorem 5.19. *Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA, $\leq$ be a co-lex order on $\mathcal{N}$, and $\{Q_i \mid 1 \leq i \leq p\}$ be a $\leq$-chain partition of $Q$, with $s \in Q_1$. Assume that we do not know $\mathcal{N}$, but we only know* $\mathtt{aBWT}(\mathcal{N}, \leq, \{Q_i \mid 1 \leq i \leq p\})$. *Then, we can decide whether $\mathcal{N}$ is distinguished by its paths and, if so, we can retrieve $\mathcal{N}$.*

Proof. Let $v_1, \ldots, v_n$ be the ordering of $Q$ in Definition 5.6. We know that $v_1$ is the initial state and for every $j = 1, \ldots, n$ we can decide whether $v_j$ is final by using $\mathtt{FINAL}$. Now, for every $\alpha \in \Sigma^*$, let $C_\alpha = \{i \in \{1, \ldots, n\} \mid \alpha \in I_{v_i}\}$. Notice that we can compute $C_\alpha$ for every $\alpha \in \Sigma^*$ by Lemma 5.15. Consider a list that contains pairs of the form $(\alpha, C_\alpha)$. Initially, the list contains only $(\varepsilon, \{1\})$. Remove recursively an element $(\alpha, C_\alpha)$ and for every $a \in \Sigma$ add $(\alpha a, C_{\alpha a})$ to the list if and only if $C_{\alpha a}$ is nonempty and it is not the second element of a pair which is or has already been in the list. This implies that after at most $|\Sigma| \cdot 2^n$ steps the list is empty, and any non-empty $C_\alpha$ has been the second element of some pair in the list. Then, we conclude that $\mathcal{N}$ is distinguished by its paths if and only for every $k = 1, \ldots, n$ the set $\{v_k\}$ has been the second element of some pair in the list. In particular, if $\mathcal{N}$ is distinguished by its paths, then for every $k = 1, \ldots, n$ we know a string $\alpha' \in \Sigma^*$ such that $C_{\alpha'} = \{k\}$. We are only left with showing that we can use $\alpha'$ to retrieve all edges leaving $v_k$. Fix a character $a \in \Sigma$. Then, compute $C_{\alpha' a}$ using again Lemma 5.15. Then, $v_k$ has $|C_{\alpha' a}|$ outgoing edges labeled $a$, whose indexes are given by $C_{\alpha' a}$. □

Since any DFA is distinguished by its paths, we obtain the following corollary:

Corollary 5.20. *The aBWT is a one-to-one encoding over DFAs.*

By counting the number of bits required by the aBWT, we can determine the size of our encoding for NFAs that are distinguished by their paths:

Corollary 5.21. *Let $\mathcal{N}$ be an NFA that is distinguished by its paths (for example, a DFA), and let $p = width(\mathcal{N})$. Then, we can store $\mathcal{N}$ using $\log(p\sigma) + O(1)$ bits per transition. If $\mathcal{N}$ is a DFA, this space can also be expressed as (at most) $\sigma \log(p\sigma) + O(\sigma)$ bits per state. If $\mathcal{N}$ is an NFA, this space can also be expressed as (at most) $2p\sigma \log(p\sigma) + O(p\sigma)$ bits per state.*

Proof. The bound of $\log(p\sigma) + O(1)$ bits per transition follows directly from Definition 5.6 and Theorem 5.19. Letting $|\delta|$ denote the number of transitions and $n$ denote the number of states, on DFAs the naive bound $|\delta| \leq n\sigma$ holds; this allows us to derive the bound of $\sigma \log(p\sigma) + O(\sigma)$ bits per state on DFAs. On arbitrary NFAs, we can use the bound $|\delta| \leq 2p\sigma n$ implied by Lemma 2.6, yielding the bound of $2p\sigma \log(p\sigma) + O(p\sigma)$ bits per state on NFAs. □

We stress that, while not being an encoding of the NFA, the aBWT still allows to reconstruct the language of the automaton and — as we will show in the next subsection — to solve subpath queries (Problem 6) by returning the convex set of all states reached by a path labeled with a given input query string. In Section 5.4 we will augment the aBWT and obtain an injective encoding of arbitrary NFAs.

### 5.3 An Index for NFAs and Languages

We now show how to support subpath queries by augmenting the aBWT with light data structures and turning it into an index. In fact, our structure is a generalization of the FM-index [45] to arbitrary automata. This contribution will solve Problem 6. Our index can be built in polynomial time for DFAs and exponential time for NFAs. In our companion article [31] we will present an index for NFAs that can be built in polynomial time by circumventing the NP-hardness of computing a co-lex order of minimum width: the solution will be to switch to co-lex *relations* (see also [29]).

Solving subpath queries on an NFA requires finding the subset $T(\alpha)$ of its states reached by some path labeled by the query string $\alpha$. In turn, note that there is a path labeled $\alpha$ ending in state $u$ if and only if $I_u$ contains a string suffixed by $\alpha$. This motivates the following definition.

*Definition 5.22.* Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA, $\leq$ be a co-lex order on $\mathcal{N}$, $\{Q_i \mid 1 \leq i \leq p\}$ be a $\leq$-chain partition of $Q$, and $\alpha \in \Sigma^*$. Define:

$$T(\alpha) = \{u \in Q \mid (\exists \beta \in I_u)(\alpha \dashv \beta)\},$$
$$R(\alpha) = S(\alpha) \cup T(\alpha) = \{u \in Q \mid (\forall \beta \in I_u)(\beta \prec \alpha) \vee (\exists \beta \in I_u)(\alpha \dashv \beta)\}.$$

Moreover, for every $i = 1, \ldots, p$ define $T_i(\alpha) = T(\alpha) \cap Q_i$ and $R_i(\alpha) = R(\alpha) \cap Q_i$.

Intuitively, $T(\alpha)$ contains all states reached by a path labeled with $\alpha$, while $R(\alpha)$ contains all the states that are either reached by a string suffixed by $\alpha$, or only reached by strings co-lexicographically smaller than $\alpha$. Note that the goal of an index solving subpath queries (Problem 6) is to compute the (cardinality of the) set $T(\alpha)$. The aim of the next lemma is to show that, once a co-lex order is fixed, $T(\alpha)$ always forms a range (a convex set). Indeed, we now prove a counterpart of Lemma 5.9, where for any $\alpha \in \Sigma^*$ we showed that $S_i(\alpha) = Q_i[1, l_i]$, for some $0 \leq l_i \leq |Q_i|$.

Lemma 5.23. *Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA, $\leq$ be a co-lex order on $\mathcal{N}$, and $\{Q_i \mid 1 \leq i \leq p\}$ be a $\leq$-chain partition of $Q$. Let $\alpha \in \Sigma^*$. Then:*

*(1) $S(\alpha) \cap T(\alpha) = \emptyset$.*
*(2) $T(\alpha)$ is $\leq$-convex.*
*(3) If $u, v \in Q$ are such that $u \leq v$ and $v \in R(\alpha)$, then $u \in R(\alpha)$. In particular, for every $i = 1, \ldots, p$ there exists $0 \leq t_i \leq |Q_i|$ such that $T_i(\alpha) = Q_i[|S_i(\alpha)| + 1, t_i]$ (namely, $t_i = |R_i(\alpha)|$).*

Proof.

(1) If $u \in T(\alpha)$, then there exists $\beta \in I_u$ such that $\alpha \dashv \beta$. In particular, $\alpha \preceq \beta$, so $u \notin S(\alpha) = \{v \in Q \mid (\forall \beta \in I_v)(\beta \prec \alpha)\}$.
(2) It follows from Lemma 5.1 by picking $U = Q$.
(3) If $v \in S(\alpha)$, then $u \in S(\alpha)$ by Lemma 5.9 and so $u \in R(\alpha)$. Now, assume that $v \in T(\alpha)$. If $u \in T(\alpha)$ we are done. If $u \notin T(\alpha)$ (and therefore $u \neq v$), we want to prove that $u \in S(\alpha)$, which implies $u \in R(\alpha)$. Fix $\beta \in I_u$; we must prove that $\beta \prec \alpha$. Since $v \in T(\alpha)$, then there exists $\gamma \in \Sigma^*$ such that $\gamma\alpha \in I_v$. Moreover, $\gamma\alpha \notin I_u$ because $u \notin T(\alpha)$. Since $u \prec v$, by Lemma 2.13 we conclude $\beta \prec \gamma\alpha$. Since $u \notin T(\alpha)$ implies $\alpha \not\dashv \beta$, from $\beta \prec \gamma\alpha$ we conclude $\beta \prec \alpha$.

□

We now show how to recursively compute the range on each chain $Q_i$ corresponding to $R_i(\alpha)$. Note that, by Lemma 5.14, we can assume to be able to recursively compute the range on each chain $Q_i$ corresponding to $S_i(\alpha)$. A computational variant of Lemma 5.23 will allow us to compute $T_i(\alpha)$ on each chain $1 \leq i \leq p$. Each recursive step of this procedure — dubbed here *forward search* — will stand at the core of our index.

Lemma 5.24 (Forward Search). *Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA, $\leq$ be a co-lex order on $\mathcal{N}$, and $\{Q_i \mid 1 \leq i \leq p\}$ be a $\leq$-chain partition of $Q$. Let $\alpha' \in \Sigma^*$, $a \in \Sigma$ and $\alpha = \alpha' a$. For every $1 \leq i, j \leq p$, let:*

*— $S_j(\alpha') = Q_j[1, l'_j]$;*
*— $R_j(\alpha') = Q_j[1, t'_j]$;*
*— $S_i(\alpha) = Q_i[1, l_i]$;*
*— $R_i(\alpha) = Q_i[1, t_i]$.*

*Fix $1 \leq i \leq p$, and define $c = \sum_{j=1}^{p} \mathsf{out}(Q_j[1, l'_j], i, a)$ and $d = \sum_{j=1}^{p} \mathsf{out}(Q_j[1, t'_j], i, a)$. Then $d \geq c$, and:*

*(1) If $d = c$, then $T_i(\alpha) = \emptyset$ and so $t_i = l_i$.*
*(2) If $d > c$, then $T_i(\alpha) \neq \emptyset$ and $t_i$, with $1 \leq t_i \leq |Q_i|$, is the smallest integer such that $\mathsf{in}(Q_i[1, t_i], a) \geq d$.*

*In particular, $l_i$ can be computed by means of Lemma 5.14, and:*

$$T_i(\alpha) = Q_i[l_i + 1, t_i].$$

Proof. Since $S(\alpha) \subseteq R(\alpha)$, we have $d \geq c$. Now, notice that $T_i(\alpha) \neq \emptyset$ if and only there exists an edge labeled $a$ leaving a state in $T(\alpha')$ and reaching chain $Q_i$, if and only if $d > c$. Hence, in the following we can assume $T_i(\alpha) \neq \emptyset$. In particular, this implies $l_i < |Q_i|$ and $t_i \geq l_i + 1$. By Lemma 5.12 all edges labeled $a$, leaving a state in $S(\alpha')$, and reaching chain $Q_i$ must end in $Q_i[1, l_i + 1]$. At the same time, since $T_i(\alpha) \neq \emptyset$, the definition of $t_i$ implies that there exists $v' \in T(\alpha')$ (and so $v' \in R(\alpha')$) such that $Q_i[t_i] \in \delta(v', a)$. Hence, the conclusion will follow if we prove that if $u', u \in Q$ are such that $u \in Q_i[1, t_i - 1]$ and $u \in \delta(u', a)$, then $u' \in R(\alpha')$. Since $u < Q_i[t_i]$, from Axiom 2 we obtain $u' \leq v'$ and since $v' \in R(\alpha')$, from Lemma 5.23 we conclude $u' \in R(\alpha')$. □

The next step is to show how to implement the forward search procedure of Lemma 5.24 using fast and small data structures, thereby obtaining an index. Before presenting the main result of this section (the aBWT-index of an automaton, Theorem 5.29), we report a few results on data structures that will be the building blocks of our index. In the following lemma, $H_0(S)$ is the zero-order entropy of $S \in \Sigma^*$.

Lemma 5.25 (Succinct String [10], Theorem 5.2 and [68], Section 6.3). *Let $S \in \Sigma^n$ be a string over an integer alphabet $\Sigma = [0, \sigma - 1]$ of size $\sigma \leq n$. Then, there exists a data structure of $nH_0(S)(1 + o(1)) + O(n)$ bits supporting the following operations in time $O(\log\log\sigma)$:*

—Access: *$S[i]$, for any $1 \leq i \leq n$.*
—Rank: *$S.rank(i, c) = |\{j \in \{1, \ldots, i\} \mid S[j] = c\}|$, for any $1 \leq i \leq n$ and $c \in \Sigma$.*
—Select: *$S.select(i, c)$ equals the integer $j$ such that $S[j] = c$ and $S.rank(j, c) = i$, for any $1 \leq i \leq S.rank(n, c)$ and $c \in \Sigma$.*

*Given $S$, the data structure can be built in $O(n \log\log\sigma)$ worst-case time.*

In other words, the operation $S[i]$ simply returns the $i$th character appearing in $S$, the operation $S.rank(i, c)$ returns the number of occurrences of character $c$ among the first $i$ characters of $S$, and the operation $S.select(i, c)$ returns the position of the $i$th occurrences of character $c$ in $S$ (if it exists). In the following, it will be expedient to assume $S.rank(0, c) = S.select(0, c) = 0$, for $c \in \Sigma$, and $S.select(i, c) = n + 1$, for $c \in \Sigma$ and $i > S.rank(n, c)$.

Note that Lemma 5.25 requires the cardinality $\sigma$ of the alphabet to be no larger than the length of the string. However, this will turn out to be too restrictive, for two reasons: (1) we would like to be able to handle also automata labeled with larger alphabets and, most importantly, (2) in our data structures (see the proof of Theorem 5.29) we will also need to manage *rank* and *select* queries over strings defined not on $\Sigma$, but $[1, p] \times \Sigma$ (where $p \leq n$ is an integer specifying the width of the underlying automaton), so even if $\sigma \leq n$ it may still be $p \cdot \sigma > n$. With the following lemma we cover this more general case. The requirement $|\Sigma| \leq n^{O(1)}$ ensures that characters fit in a constant number of computer memory words and thus they can be manipulated in constant time (as it is customary in the data compression field, we recall that in this article we assume a computer memory word to be formed by $\Theta(\log n)$ bits — see Section 2.1). Note that we lose fast access functionality (which however will not be required in our application of this data structure).

Lemma 5.26 (Succinct String Over Large Alphabet). *Let $S \in \Sigma^n$ be a string over an integer alphabet $\Sigma = [0, \sigma - 1]$ of size $\sigma = |\Sigma| \leq n^{O(1)}$. Then, there exists a data structure of $nH_0(S)(1 + o(1)) + O(n)$ bits, where $H_0(S)$ is the zero-order entropy of $S$, supporting the following operations in time $O(\log\log\sigma)$:*

- —Rank: $S.rank(i, c) = |\{j \in \{1, \ldots, i\} \mid S[j] = c\}|$, *for* $1 \leq i \leq n$ *and* $c \in \Sigma$ *that occurs in* $S$.
- —Select: $S.select(i, c)$ *equals the integer* $j$ *such that* $S[j] = c$ *and* $S.rank(j, c) = i$, *for any* $1 \leq i \leq S.rank(n, c)$ *and for any character* $c \in \Sigma$ *that occurs in* $S$.

*Given $S$, the data structure can be built in expected $O(n \log\log\sigma)$ time.*

Proof. If $\sigma \leq n$, then we simply use the structure of Lemma 5.25. Otherwise ($\sigma > n$), let $\Sigma' = \{S[i] \mid 1 \leq i \leq n\}$ be the *effective alphabet* of $S$. We build a minimal perfect hash function $h : \Sigma \to [0, |\Sigma'| - 1]$ mapping (injectively) $\Sigma'$ to the numbers in the range $[0, |\Sigma'| - 1]$ and mapping arbitrarily $\Sigma \setminus \Sigma'$ to the range $[0, |\Sigma'| - 1]$. We store $h$ using the structure described in [54]. This structure can be built in $O(n)$ expected time, uses $O(n)$ bits of space, and answers queries of the form $h(x)$ in $O(1)$ worst-case time. Note that $|\Sigma'| \leq n$, so we can build the structure of Lemma 5.25 starting from the string $S' \in [0, |\Sigma'| - 1]^n$ defined as $S'[i] = h(S[i])$. Then, *rank* and *select* operations on $S$ can be answered as $S.rank(i, c) = S'.rank(i, h(c))$ and $S.select(i, c) = S'.select(i, h(c))$, provided that $c \in \Sigma'$. Notice that the zero-order entropies of $S$ and $S'$ coincide, since the character's frequencies remain the same after applying $h$ to the characters of $S$. We conclude that the overall space of the data structure is at most $nH_0(S)(1 + o(1)) + O(n)$ bits. □

Finally, we need a fully-indexable dictionary data structure. Such a data structure encodes a set of integers and supports efficiently a variant of *rank* and *select* queries as defined below:

Lemma 5.27 (Fully-indexable Dictionary [43], Theorem 4.1). *A set $A = \{x_1, \ldots, x_n\} \subseteq [1, u]$ of cardinality $n$ can be represented with a data structure of $n \log(u/n) + O(n)$ bits so that the following operations can be implemented in $O(\log\log(u/n))$ time:*

- —Rank: $A.rank(x) = |\{y \in A \mid y \leq x\}|$, *for any* $1 \leq x \leq u$.
- —Select: $A.select(i) = x$ *such that* $x \in A$ *and* $A.rank(x) = i$, *for any* $1 \leq i \leq |A|$.

*Given $A$ as input, the data structure can be built in $O(n)$ worst-case time.*

*Remark 5.28.* The queries of Lemma 5.27 can be used to solve in $O(\log\log(u/n))$ time also:

- —*Predecessor:* the largest element of $A$ smaller than or equal to $x$, if it exists. For any $1 \leq x \leq u$, $A.pred(x) = A.select(A.rank(x))$ if $A.rank(x) > 0$, and $A.pred(x) = \perp$ otherwise.
- —*Strict-Successor:* the smallest element of $A$ strictly greater than $x$, if it exists. For any $1 \leq x \leq u$, $A.succ(x) = A.select(A.rank(x) + 1)$ if $A.rank(x) < |A|$, and $A.succ(x) = \perp$ otherwise.
- —*Membership:* For any $1 \leq x \leq u$, $x \in A$ if and only if $x = A.pred(x)$.

We are ready to present the main result of this section (Theorem 5.29): a linear-space index supporting subpath queries on any automaton (Problem 6) in time proportional to $p^2 \cdot \log\log(p\sigma)$ per query character ($p$ being the automaton's width).

Theorem 5.29 (aBWT-index of a Finite-state Automaton). *Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA on alphabet $\Sigma$ of size $\sigma = |\Sigma| \leq e^{O(1)}$, where $e = |\delta|$ is the number of $\mathcal{N}$-transitions. Assume that we are given a $\leq$-chain partition $\{Q_i \mid 1 \leq i \leq p\}$, for some co-lex order $\leq$ of width $p$. Then, in expected time $O(e \log\log\sigma)$, we can build a data structure using $e \log(p\sigma)(1 + o(1)) + O(e)$ bits that, given a query string $\alpha \in \Sigma^m$, answers the following queries in $O(m \cdot p^2 \cdot \log\log(p\sigma))$ time:*

(1) *compute the set $T(\alpha)$ of all states reached by a path on $\mathcal{N}$ labeled $\alpha$, represented by means of $p$ ranges on the chains in $\{Q_i \mid 1 \leq i \leq p\}$;*
(2) *compute the set $I_\alpha$ of all states reached by a path labeled with $\alpha$ originating in the source, represented by means of $p$ ranges on the chains in $\{Q_i \mid 1 \leq i \leq p\}$ and, in particular, decide whether $\alpha \in \mathcal{L}(\mathcal{N})$.*

Proof. Let $n = |Q|$. In this proof, we assume that the states of $Q$ have been sorted like in Definition 5.6: if $\pi(v)$, for $v \in Q$, is the unique integer such that $v \in Q_{\pi(v)}$, then we consider the ordering $v_1, \ldots, v_n$ of $Q$ such that for every $1 \leq i < j \leq n$ it holds $\pi(v_i) < \pi(v_j) \vee (\pi(v_i) = \pi(v_j) \wedge v_i < v_j)$. Moreover, we assume that $s \in Q_1$ (again like in Definition 5.6), so $s = v_1$. For every $i = 1, \ldots, p$, let $e_i = |\{(u, v, a) \mid \delta(u, a) = v, u \in Q, v \in Q_i, a \in \Sigma\}|$ be the number of edges entering the $i$th chain, let $\Sigma_i = (\bigcup_{u \in Q_i} \lambda(u)) \setminus \{\#\}$ be the set of characters labeling edges entering the $i$th chain, and let $\sigma_i = |\Sigma_i|$.

We store the following data structures:

— One fully-indexable succinct dictionary (Lemma 5.27) on each $\Sigma_i$ to map $\Sigma_i \subseteq [0, \sigma - 1]$ to $[0, \sigma_i - 1]$. The total number of required bits is $\sum_{i=1}^{p}(\sigma_i \log(\sigma/\sigma_i) + O(\sigma_i)) \leq \sum_{i=1}^{p}(e_i \log(\sigma/\sigma_i) + O(e_i)) = e \log \sigma - \sum_{i=1}^{p}(e_i \log \sigma_i) + O(e)$. As a consequence, we can solve rank, select, predecessor, strict-successor and membership queries on each dictionary in $O(\log\log(\sigma/\sigma_i)) \subseteq O(\log\log\sigma)$ time.
— The bitvector $\texttt{CHAIN} \in \{0,1\}^n$ of Definition 5.6 represented by the data structure of Lemma 5.25. The number of required bits is $nH_0(\texttt{CHAIN})(1 + o(1)) + O(n) = O(n) \subseteq O(e)$. As a consequence, we can solve rank and select queries on CHAIN in $O(1)$ time. In particular, in $O(1)$ time we can compute $|Q_i|$, for $i = 1, \ldots, p$, because $|Q_i| = \texttt{CHAIN}.select(i + 1, 1) - \texttt{CHAIN}.select(i, 1)$.
— The bitvector $\texttt{FINAL} \in \{0,1\}^n$ of Definition 5.6 represented by the data structure of Lemma 5.25. The number of required bits is again $O(n) \subseteq O(e)$.
— The bitvector $\texttt{OUT\_DEG} \in \{0,1\}^{e+n}$ of Definition 5.6 represented by the data structure of Lemma 5.25. The number of required bits is $(n + e)H_0(\texttt{OUT\_DEG})(1 + o(1)) + O(n + e) \subseteq O(e)$. As a consequence, we can solve rank and select queries on OUT_DEG in $O(1)$ time.
— The string $\texttt{OUT} \in ([1, p] \times \Sigma)^e$ of Definition 5.6 represented by the data structure of Lemma 5.26 (the assumption on the size on the alphabet in Lemma 5.26 is satisfied because $|[1, p] \times \sigma| = p \cdot \sigma \leq n \cdot \sigma \leq (e + 1) \cdot \sigma = e^{O(1)}$). The number of required bits is $eH_0(\texttt{OUT})(1 + o(1)) + O(e)$. We will bound the quantity $eH_0(\texttt{OUT})$ by exhibiting a prefix-free encoding of OUT. The key idea is that if $(i, c) \in [1, p] \times \Sigma$ occurs in OUT, then it must be $c \in \Sigma_i$, so we can encode $(i, c)$ by using $\lceil \log(p + 1) \rceil \leq \log p + 1$ bits encoding $i$, followed by $\lceil \log(\sigma_i + 1) \rceil \leq \log \sigma_i + 1$ bits encoding $c$ (note that this part depends on $i$). We clearly obtain a prefix code, so we conclude $eH_0(\texttt{OUT}) \leq \sum_{i=1}^{p} e_i(\log p + \log \sigma_i + O(1)) = e \log p + \sum_{i=1}^{p}(e_i \log \sigma_i) + O(e)$ bits. Observing that $\sum_{i=1}^{p}(e_i \log \sigma_i) \leq e \log \sigma$, we conclude that the number of required bits for OUT is bounded by $eH_0(\texttt{OUT})(1 + o(1)) + O(e) = (e \log p + \sum_{i=1}^{p}(e_i \log \sigma_i) + O(e))(1 + o(1)) + O(e) \leq (1 + o(1))e \log p + \sum_{i=1}^{p}(e_i \log \sigma_i) + o(e \log \sigma) + O(e)$ bits. Notice that in $O(\log\log(p\sigma))$ time we can solve rank and select queries on OUT (that is, queries $\texttt{OUT}.rank(j, (i, c))$ and $\texttt{OUT}.select(j, (i, c))$) for all $1 \leq i \leq p$ and for all $c \in \Sigma$. Indeed, given $i$ and $c$, we first check whether $c \in \Sigma_i$ by solving a membership query on the dictionary for $\Sigma_i$ in $O(\log\log\sigma)$ time. If $c \notin \Sigma_i$, then we immediately conclude that $\texttt{OUT}.rank(j, (i, c)) = 0$ and $\texttt{OUT}.select(j, (i, c))$ is undefined. If $c \in \Sigma_i$, then $(i, c)$ appears in OUT, so the conclusion follows from Lemma 5.26.

— The bitvector $\texttt{IN\_DEG} \in \{0,1\}^{e+n}$ of Definition 5.6 represented by the data structure of Lemma 5.25. The number of required bits is again $O(e)$. As a consequence, we can solve rank and select queries on IN_DEG in $O(1)$ time.
— A bitvector IN′, represented by the data structure of Lemma 5.25, built as follows. We sort all edges $(v_j, v_i, c)$ by end state $v_i$ and, if the end state is the same, by label $c$. Then, we build a string $\texttt{IN} \in \Sigma^e$ by concatenating all labels of the sorted edges. Finally, $\texttt{IN}' \in \{0,1\}^e$ is the bitvector such that $\texttt{IN}'[k] = 1$ if and only if $k = 1$ or $\texttt{IN}[k] \neq \texttt{IN}[k-1]$ or the $k$-th edge and the $(k-1)$-th edge reach distinct chains. The number of required bits is $O(e)$.

For an example, consider the automaton of Figure 9. All sequences except for bitvector IN′ are reported in Example 5.7. To build bitvector IN′, we first build the string IN of all incoming labels of the sorted edges: IN = *aaabcabbbb*. Then, bitvector IN′ marks with a bit "1" (i) the first character of each maximal unary substring in IN, and (ii) the characters of IN labeling the first edge in each chain: IN′ = 1001111000.

By adding up the space of all components (note that the terms $-\sum_{i=1}^{p}(e_i \log \sigma_i)$ in the dictionaries $\Sigma_i$ and $\sum_{i=1}^{p}(e_i \log \sigma_i)$ in sequence OUT cancel out), we conclude that our data structures take at most $e \log(p\sigma)(1 + o(1)) + O(e)$ bits.

We proceed by showing how to solve queries 1 (string matching) and 2 (membership).

(1) Let us prove that, given a query string $\alpha \in \Sigma^m$, we can use our data structure to compute, in time $O(m \cdot p^2 \cdot \log\log(p\sigma))$, the set $T(\alpha)$ of all states reached by a $\alpha$-path on $\mathcal{N}$, presented by $p$ convex sets on the chains $\{Q_i \mid 1 \leq i \leq p\}$. By Lemma 5.24, it will suffice to show how to compute $R(\alpha)$ and $S(\alpha)$. We can recursively compute each $R(\alpha)$ and $S(\alpha)$ in time proportional to $m$ by computing $R(\alpha')$ and $S(\alpha')$ for all prefixes $\alpha'$ of $\alpha$. Hence, we only have to show that we can update $R(\alpha')$ and $S(\alpha')$ with a new character, in $O(p^2 \cdot \log\log(p\sigma))$ time. We start with the empty prefix $\varepsilon$, whose corresponding sets are $R(\varepsilon) = Q$ and $S(\varepsilon) = \emptyset$. For the update we apply Lemmas 5.14 and 5.24. An inspection of the two lemmas reveals the we can update $R(\alpha')$ and $S(\alpha')$ with a new character by means of $O(p^2)$ calls to the following queries.

— (op1) for any $1 \leq i, j \leq p$, $1 \leq k \leq |Q_j|$, and $a \in \Sigma$, compute $\texttt{out}(Q_j[1,k], i, a)$;
— (op2) for any $1 \leq i \leq p$, $a \in \Sigma$, and $h \geq 0$, find the largest integer $0 \leq k \leq |Q_i|$ such that $\texttt{in}(Q_i[1,k], a) \leq h$;
— (op3) for any $1 \leq i \leq p$, $a \in \Sigma$, and $z \geq 1$, find the smallest integer $1 \leq t \leq |Q_i|$ such that $\texttt{in}(Q_i[1,t], a) \geq z$, if it exists, otherwise report that it does not exist.
— (op4) for any $1 \leq i \leq p$ and $a \in \Sigma$, find the largest integer $0 \leq h \leq |Q_i|$ such that, if $h \geq 1$, then $\max(\lambda(Q_i[h])) \leq a$.

As a consequence, we are left to show that we can solve each query in $O(\log\log(p\sigma))$ time.

(op1). The states in $Q_j[1,k]$ correspond to the convex set of all states (in the total order $v_1, \ldots, v_n$ of Definition 5.6) whose endpoints are $v_l$ and $v_r$, where $l = \texttt{CHAIN}.select(j,1)$ and $r = l + k - 1$. Considering the order of edges used to define OUT, the set of all edges leaving a state in $Q_j[1,k]$ forms a convex set in OUT and in OUT_DEG. Define:

— $x = \texttt{OUT\_DEG}.rank(\texttt{OUT\_DEG}.select(l-1,1), 0)$;
— $y = \texttt{OUT\_DEG}.rank(\texttt{OUT\_DEG}.select(r,1), 0)$.

Notice that $x$ is equal to the number of edges leaving all states before $v_l$, while $y$ is the number of edges leaving all states up to $v_r$ included. As a consequence, we have $x \leq y$. If $x = y$, then there are no edges leaving states in $Q_j[1,k]$ and we can immediately conclude $\texttt{out}(Q_j[1,k], i, a) = 0$. Assuming $x < y$, $x + 1$ and $y$ are the endpoints of the convex set of all edges leaving states in $Q_j[1,k]$. Hence, we are left with counting the number of such edges labeled $a$ and reaching

chain $Q_i$. Notice that $\texttt{OUT}.rank(x, (i, a))$ is the number of all edges labeled $a$ and reaching chain $i$ whose start state comes before $v_l$, whereas $\texttt{OUT}.rank(y, (i, a))$ is the number of all edges labeled $a$ and reaching $i$ whose start state comes before or is equal to $v_r$. We can then conclude that $\texttt{out}(Q_j[1, k], i, a) = \texttt{OUT}.rank(y, (i, a)) - \texttt{OUT}.rank(x, (i, a))$.

(op2). First, we check whether $a \in \Sigma_i$ by a membership query on the dictionary for $\Sigma_i$. If $a \notin \Sigma_i$, we immediately conclude that the largest $k$ with the desired properties is $k = |Q_i|$. Assume $a \in \Sigma_i$ and notice that the states in $Q_i$ correspond to the convex set of all states whose endpoints are $v_l$ and $v_r$, where $l = \texttt{CHAIN}.select(i, 1)$ and $r = \texttt{CHAIN}.select(i + 1, 1) - 1$. Considering the order of edges used to define IN, the set of all edges entering a state in $Q_j[1, k]$ forms a convex set in IN and in IN_DEG. Define

$-x = \texttt{IN\_DEG}.rank(\texttt{IN\_DEG}.select(l - 1, 1), 0)$;
$-y = \texttt{IN\_DEG}.rank(\texttt{IN\_DEG}.select(r, 1), 0)$.

Notice that $x$ is equal to the number of edges reaching all states before $v_l$, while $y$ is the number of edges reaching all states coming before or equal to $v_r$. Since $a \in \Sigma_i$, we have $x < y$, and $x + 1$ and $y$ are the endpoints of the convex set of all edges reaching a state in $Q_i$. The next step is to determine the smallest edge labeled $a$ reaching a state in $Q_i$. First, notice that the number of characters smaller than or equal to $a$ in $\Sigma_i$ can be retrieved, by Lemma 5.27, as $\Sigma_i.rank(a)$. Notice that $f = \texttt{IN}'.rank(x, 1)$ yields the number of 0-runs in $\texttt{IN}'$ pertaining to chains before chain $Q_i$ so that, since we know that $a \in \Sigma_i$, then $g = \texttt{IN}'.select(f + \Sigma_i.rank(a), 1)$ yields the smallest edge labeled $a$ in the convex set of all edges reaching a state in $Q_i$. We distinguish two cases for the parameter $h$ of op2:

- $h = 0$. In this case, the largest $k$ with the desired properties is equal to the position on chain $Q_i$ of the state reached by the $g$-th edge minus one. The index of the state reached by the $g$-th edge is given by $p = \texttt{IN\_DEG}.rank(\texttt{IN\_DEG}.select(g, 0), 1) + 1$, so the largest $k$ with the desired properties is $k = p - l$.
- $h > 0$. The quantity $h' = \texttt{IN}'.select(f + \Sigma_i.rank(a) + 1, 1) - g = \texttt{IN}'.select(f + \Sigma_i.rank(a) + 1, 1) - \texttt{IN}'.select(f + \Sigma_i.rank(a), 1)$ yields the number of edges labeled $a$ in the convex set of all edges reaching a state in $Q_i$. If $h' \leq h$, then we conclude that the largest $k$ is $|Q_i|$. Hence, assume that $h' > h$. We immediately obtain that the $(h + 1)$-th smallest edge labeled $a$ reaching a state in $Q_i$ is the $(g + h)$-th edge, and the largest $k$ with the desired properties is equal to the position on chain $Q_i$ of the state reached by the $(g + h)$-th edge minus one. Analogously to case 1, the index of the state reached by this edge is given by $p = \texttt{IN\_DEG}.rank(\texttt{IN\_DEG}.select(g + h, 0), 1) + 1$, so the largest $k$ with the desired properties is $k = p - l$.

(op3). We simply use operation (op2) to compute the largest integer $0 \leq k \leq |Q_i|$ such that $\texttt{in}(Q_i[1, k], a) \leq z - 1$. If $k = |Q_i|$, then the desired integer does not exist, otherwise it is equal to $k + 1$.

(op4). We first decide whether $\Sigma_i.succ(a)$ is defined. If it is not defined, then the largest integer with the desired property is $|Q_i|$. Now assume that $\Sigma_i.succ(a)$ is defined. Then the largest integer with the desired property is simply the largest integer $0 \leq k \leq |Q_i| - 1$ such that $\texttt{in}(Q_i[1, k], \Sigma_i.succ(a)) \leq 0$, which can be computed using (op2).

(2) Let us prove that, given a query string $\alpha \in \Sigma^m$, we can use our data structure to compute $I_\alpha$ in $O(m \cdot p^2 \cdot \log\log(p\sigma))$ time, represented as $p$ ranges on the chains in $\{Q_i \mid 1 \leq i \leq p\}$. We claim that it will suffice to run the same algorithm used in the previous point, starting with $R = \{v_1\}$ and $S = \emptyset$. Indeed, consider the automaton $\mathcal{N}'$ obtained from $\mathcal{N}$ by adding a new initial state $v_0$ and adding exactly one edge from $v_0$ to $v_1$ (the old initial state) labeled with #, a character smaller than

every character in the alphabet $\Sigma$. Let $\leq'$ the co-lex order on $\mathcal{N}'$ obtained from $\leq$ by adding the pair $\{(v_0, v_1)\}$, and consider the $\leq'$-chain partition obtained from $\{Q_i \mid 1 \leq i \leq p\}$ by adding $v_0$ to $Q_1$. It is immediate to notice that for every $k = 1, \ldots, n$ and for every string $\alpha \in \Sigma^*$ we have that $v_k \in I_\alpha$ on $\mathcal{N}$ if and only if $v_k \in T(\#\alpha)$ on $\mathcal{N}'$. Since $v_0$ has no incoming edges and on $\mathcal{N}'$ we have $R(\#) = \{v_0, v_1\}$ and $S(\#) = \{v_0\}$, the conclusion follows. Now, given $I_\alpha$, we can easily check whether $\alpha \in \mathcal{L}(\mathcal{N})$. Indeed, for every $i = 1, \ldots, p$ we know the integers $l_i$ and $t_i$ such that $I^i_\alpha = Q_i[l_i + 1, t_i]$, and we decide whether some of these states are final by computing $f = \texttt{CHAIN}.select(i, 1)$, and then checking whether $\texttt{FINAL}.rank(f + l_i - 1, 1) - \texttt{FINAL}.rank(f + t_i - 1, 1)$ is larger than zero. □

Notice that, in order to apply Theorem 5.29, we need a chain partition of a co-lex order over the automaton. In the case of a DFA, Corollary 4.5 allows us to compute in polynomial time (with high probability) a $\leq$-chain partition of optimal width, so from Theorem 5.29 we obtain the following result.

Corollary 5.30. *Let $\mathcal{D} = (Q, s, \delta, F)$ be a DFA. Then, the data structure of Theorem 5.29 can be built on $\mathcal{D}$ with parameter $p = width(\mathcal{D})$ in expected $\tilde{O}(|\delta|^2)$ time.*

Note that the data structure of Corollary 5.30 implicitly supports the navigation of the labeled graph underlying $\mathcal{D}$ starting from the initial state: a node is represented as the $p$ ranges (on the $p$ chains) obtained when computing $I_\alpha$. This enables deciding whether $\alpha \in \mathcal{L}(\mathcal{D})$ in $O(p^2 \log\log(p\sigma))$ time per character of $\alpha$. Even if testing membership with our structure is inefficient on DFAs for large $p$ (on a classic DFA representation, this operation can be easily implemented in $O(|\alpha|)$ time), for small values of $p$ on NFAs our membership procedure can be much faster than the classical ones (determinization or dynamic programming), provided our index has been built on the NFA.

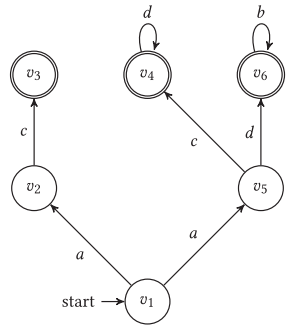
### 5.4 Encoding NFAs

We now show a simple extension of the aBWT of Definition 5.6, yielding an injective encoding of NFAs. It turns out that in order to achieve these goals it is sufficient to collect the components of the aBWT and, in addition, the origin chain of every edge (see Figure 12 for an example):

*Definition 5.31.* Let $\mathcal{N} = (Q, s, \delta, F)$ be an NFA and let $e = |\delta|$ be the number of $\mathcal{N}$-transitions. Let $\leq$ be a co-lex order on $\mathcal{N}$, and let $\{Q_i \mid 1 \leq i \leq p\}$ be a $\leq$-chain partition of $Q$, where w.l.o.g. $s \in Q_1$. Let $\pi(v)$ and $Q = \{v_1, \ldots, v_n\}$ be the map and the total state order defined in Definition 5.6. Define a new sequence $\texttt{IN\_CHAIN} \in [1, p]^e$, storing the edges' origin chains, as follows. Sort all edges $(v_j, v_i, c)$ by increasing destination index $i$, breaking ties by label $c$ and then by origin index $j$. Then, IN_CHAIN is obtained by concatenating the elements $\pi(v_j)$ for all edges $(v_j, v_i, c)$ sorted in this order.

The following lemma presents a function that will ultimately allow us to show that our augmented aBWT is indeed an injective encoding on NFAs.

Lemma 5.32. *Let $1 \leq i, j \leq p$ and $a \in \Sigma$. Let $w$ be the number of edges labeled with character $a$ leaving any state in $Q_j$ and entering any state in $Q_i$. Let $B_{j,i,a} = (f_1, f_2, \ldots, f_w)$ be state indices such that (i) $1 \leq f_1 \leq f_2 \leq \cdots \leq f_w \leq n$, (ii) if $1 \leq k \leq n$ occurs in $B_{j,i,a}$, then $\pi(v_k) = j$ and (iii) if $1 \leq k \leq n$ occurs $t \geq 1$ times in $B_{j,i,a}$, then there exist exactly $t$ edges labeled with character $a$ leaving $v_k$ and entering a state in $Q_i$. Let $C_{j,i,a} = (g_1, g_2, \ldots, g_w)$ be state indices such that (i) $1 \leq g_1 \leq g_2 \leq \cdots \leq g_w \leq n$, (ii) if $1 \leq h \leq n$ occurs in $C_{j,i,a}$, then $\pi(v_h) = i$ and (iii) if $1 \leq h \leq n$ occurs $t \geq 1$ times in $C_{j,i,a}$, then there exist exactly $t$ edges labeled with character $a$ entering $v_h$ and leaving a state in $Q_j$. Then, $\{(v_{f_\ell}, v_{g_\ell}, a) \mid 1 \leq \ell \leq w\}$ is the set of all edges labeled with character $a$, leaving a state in $Q_j$ and entering a state in $Q_i$.*




| | | CHAIN | 1 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | FINAL | 0 | 0 | 1 | 1 | 0 | 1 |
| | | IN_DEG | 1 | 01 | 01 | 001 | 01 | 001 |
| | | IN_CHAIN | | 1 | 1 | 21 | 1 | 22 |
| OUT_DEG | OUT | | 1 | 2 | 3 | 4 | 5 | 6 |
| 001 | (1,a),(2,a) | 1 | | (1,1,a) | | | (1,2,a) | |
| 01 | (1,c) | 2 | | | (1,1,c) | | | |
| 1 | | 3 | | | | | | |
| 01 | (1,d) | 4 | | | | (1,1,d) | | |
| 001 | (1,c),(2,d) | 5 | | | | (2,1,c) | | (2,2,d) |
| 01 | (2,b) | 6 | | | | | | (2,2,b) |

Fig. 12. Augmented aBWT of an NFA (the one in Figure 11 on the left), using the chain partition $\{\{v_1, v_2, v_3, v_4\}, \{v_5, v_6\}\}$. In addition to the aBWT of Definition 5.6, we add a sequence IN_CHAIN collecting the origin chain of every edge. For each labeled edge $(u, v, a)$, in the adjacency matrix we show the triple $(\pi(u), \pi(v), a)$, that is, the origin chain, destination chain, and label of the edge (the matrix is visually divided in 4 sectors, corresponding to all combinations of origin and destination chains). Vector IN_CHAIN collects vertically the incoming chains, that is, the first component of each triple in the corresponding column of the adjacency matrix.

Proof. Consider the set of all edges labeled with character $a$, leaving a state in $Q_j$ and entering a state in $Q_i$. Then, $B_{j,i,a}$ is obtained by picking and sorting all start states of these edges, and $C_{j,i,a}$ is obtained by picking and sorting all end states of these edges. The conclusion follows from Axiom 2. □

Lemma 5.32 allows reconstructing the topology of a NFA starting from our augmented aBWT, as we show in the next lemma.

Lemma 5.33. *The aBWT of Definition 5.6, in addition to sequence* IN_CHAIN *of Definition 5.31, is a one-to-one encoding over the NFAs.*

Proof. From CHAIN and FINAL, we can retrieve the chain of each state, and we can decide which states are final. We only have to show how to retrieve the set $\{(v_f, v_g, a) \mid v_g \in \delta(v_f, a),\ v_f, v_g \in Q, a \in \Sigma\}$ of all NFA's transitions. By Lemma 5.32, we only have to prove that for every $1 \leq i, j \leq p$, and for every $a \in \Sigma$ we can retrieve $B_{j,i,a}$ and $C_{j,i,a}$. We will use ideas similar to those employed in the proof of Lemma 5.15.

Let us show how to retrieve $B_{j,i,a}$ for every $1 \leq i, j \leq p$, and for every $a \in \Sigma$. Fix $i$, $j$ and $a$. From OUT_DEG we can retrieve the number of edges leaving each state in the $j$th chain. Then, the definition of OUT implies that, for every state in the $j$th chain, we can retrieve the label and the

destination chain of each edge leaving the state, so we can decide how many times a state in the $j$th chain occurs in $B_{j,i,a}$.

Let us show how to retrieve $C_{j,i,a}$ for every $1 \leq i, j \leq p$, and for every $a \in \Sigma$. Fix $i$, $j$ and $a$. From IN_DEG we can retrieve the number of edges entering each state in the $i$th chain. From OUT we can retrieve all characters (with multiplicities) labeling some edge entering the $i$th-chain. As a consequence, Axiom 1 implies that, for every state in the $i$th chain, we can retrieve the label of each edge entering the state. Moreover, the definition of IN_CHAIN implies that, for every state in the $i$th cain, we can retrieve the start chain of each edge entering the state, so we can decide how many times a state in the $i$th chain occurs in $C_{j,i,a}$. □

By analyzing the space required by our extension of the aBWT and applying Lemma 5.33, we obtain:

Corollary 5.34. *Let $\mathcal{N}$ be an NFA, and let $p = width(\mathcal{N})$. Then, we can store $\mathcal{N}$ using $\log(p^2\sigma) + O(1)$ bits per transition. This space can also be expressed as (at most) $2p\sigma \log(p^2\sigma) + O(p\sigma)$ bits per state.*

Proof. The bound of $\log(p^2\sigma) + O(1)$ bits per transition follows easily from the definitions of aBWT (Definition 5.6) and IN_CHAIN (Definition 5.31). In order to bound this space as a function of the number of states, we use the bound $|\delta| \leq 2p\sigma n$ implied by Lemma 2.6, where $|\delta|$ is the number of transitions. □

In Theorem 5.29 we provided an aBWT-index supporting pattern matching queries on any NFA. Being a superset of the aBWT, our (indexed) augmented aBWT can clearly support the same operations (in the same running times) of Theorem 5.29, albeit using additional $\log p$ bits per edge. Actually, these operations turn out to be much simpler on the augmented aBWT than on the aBWT (thanks to the new sequence IN_CHAIN). This is possible because by means of our augmented aBWT, given *any* convex set $U$ of states (represented by means of $p$ ranges on the chains) and a character $c$, one can compute the convex set of all states that can be reached from $U$ by following edges labeled $c$ (see Lemma 5.1). However, the queries' running times remain the same as in Theorem 5.29 so we will not describe them here.

## 6 CONCLUSIONS

In this article, we considered the theoretical and practical implications of studying a specific *partial* order on the states of a finite automaton. The considered partial order is a particularly natural one: the one obtained lifting to sets the *co-lex* order of strings reaching every given state of the underlying automaton.

In this work, we extensively argued that our proposed point of view allows us to sensibly classify regular languages and their complexities, from a both *practically* and *theoretically* interesting perspective. The central measure for the classification we put forward is the (minimum) *width* of the above mentioned partial order. In this article and in [31] we show that such measure induces a proper hierarchy of regular languages and make a number of observations on the levels of this hierarchy. An interesting feature of the classification obtained is that languages at higher levels of the hierarchy have a larger *entangled* collection of states that makes them less prone to index-ability — and, consequently, to any subsequent algorithmic analysis.

From a theoretical perspective, we showed that a canonical automaton minimizing the width to its entanglement size (the *Hasse automaton* of a language), can be built at any given level.

From a practical perspective, in the last part of the article we introduced a technique to build very efficient indexes, exploiting the partial order considered. As it turned out, the indexes proposed

can be used in both the deterministic and non-deterministic case, but can be built in polynomial time only in the former case. In the companion article [31] we show how to overcome this issue.

As a matter of fact, our full proposal can be divided in two macro steps: in the first one, the one presented in this article, we illustrated and proved results that mainly apply to the deterministic-width hierarchy and to indexing and encoding regular languages (a task that we have shown to be computationally easy on DFAs). In a subsequent article, we will illustrate the extensions of our classification results to the non-deterministic-width hierarchy, as well as polynomial-time algorithms for sorting non-deterministic automata by means of *co-lex relations* (a more general concept than co-lex orders, enabling indexing and avoiding NP-hardness).

### A Glimpse into the Non-Deterministic Case

What happens if we consider the notion of width applied to non-deterministic automata instead of deterministic ones? In this paragraph, we sketch the main results that will be presented in the companion article [31].

(1) There exist NFAs without a maximum co-lex order, suggesting that Problem 1 (Automata-width problem) could become more complex when applied to NFAs. Indeed, in [53] it is proved that even deciding whether $\text{width}(\mathcal{N}) = 1$ (i.e., deciding whether $\mathcal{N}$ is Wheeler) is an NP-complete problem. For this reason, a smallest-width NFA index cannot be built in polynomial time by means of co-lex orders (assuming $P \neq NP$). We will show that the solution relies on switching to *co-lex relations* (see [29]): the smallest-width relation can be computed in polynomial time, it enables indexing, and its width is never larger than that of a smallest-width co-lex order.

(2) Another way to bypass the previous obstacle to indexing NFAs is to consider the class consisting of all NFAs admitting a maximum co-lex order. We will prove that this class — dubbed the $\mathcal{MAX}$ class — is a polynomial time decidable class of automata, strictly between the DFA and NFA classes, for which the maximum co-lex order is always computable in polynomial time. This implies that the NP-completeness of determining the width of an NFA is due to automata outside the $\mathcal{MAX}$ class.

(3) The language-width problem becomes more complex: starting from an NFA $\mathcal{N}$, already deciding whether $\text{width}^D(\mathcal{L}(\mathcal{N})) = 1$ (equivalently, $\text{width}^N(\mathcal{L}(\mathcal{N})) = 1$) is a PSPACE complete problem (see [35]).

(4) The language hierarchies based on the two (deterministic/non-deterministic) notions of width do not coincide, except for level 1 (Wheeler languages). More precisely, we will prove that there exist regular languages $\mathcal{L}$, whose deterministic width $p > 1$ can be chosen arbitrarily large, such that $p = \text{width}^D(\mathcal{L}) = 2^{\text{width}^N(\mathcal{L})} - 1$, matching the upper-bound given in Corollary 3.3.

## INDEX



## APPENDIX

## A PARTITIONS AND ORDERS

This section is devoted to the proof of Theorem 4.11 and to other useful properties of entangled convex sets. All these results follow from general results valid for arbitrary total orders and partitions. From now on, we fix a total order $(Z, \leq)$ and a finite partition $\mathcal{P} = \{P_1, \ldots, P_m\}$ of $Z$. We first give a notion of entanglement, with respect to $\mathcal{P}$, for subsets $X \subseteq Z$. The main result of this section, Theorem A.5, states that there always exists a finite, *ordered* partition $\mathcal{V}$ of $Z$ composed of entangled convex sets.

It is convenient to think of the elements of $\mathcal{P}$ as letters of an alphabet, forming finite or infinite strings while labeling element of $Z$. A finite string $P_1 \ldots P_k \in \mathcal{P}^*$ is said to be *generated* by $X \subseteq Z$, if there exists a sequence $x_1 \leq \cdots \leq x_k$ of elements in $X$ such that $x_j \in P_j$, for all $j = 1, \ldots, k$. We also say that $P_1 \ldots P_k$ *occurs in* $X$ *at* $x_1, \ldots, x_k$. Similarly, an infinite string $P_1 \ldots P_k \cdots \in \mathcal{P}^\omega$ is generated by $X \subseteq Z$ if there exists a monotone sequence $(x_i)_{i \in \mathbb{N}}$ of elements in $X$ such that $x_j \in P_j$, for all $j \in \mathbb{N}$. Notice that if $X$ is a finite set, then there exists an index $i_0$ such that for every

$i \geq i_0$ it holds $P_i = P_{i_0}$. We can now re-state the notion of entanglement in this, more general, context.

*Definition A.1.* Let $(Z, \leq)$ be a total order, let $\mathcal{P}$ be a partition of $Z$, and let $X \subseteq Z$.

(1) We define $\mathcal{P}_X = \{P \in \mathcal{P} : P \cap X \neq \emptyset\}$.
(2) If $\mathcal{P}' = \{P_1, \ldots, P_m\} \subseteq \mathcal{P}$, we say that $\mathcal{P}'$ is *entangled* in $X$ if the infinite string $(P_1 \ldots P_m)^\omega$ is generated by $X$.
(3) We say that $X$ is *entangled* if $\mathcal{P}_X$ is entangled in $X$.

The property of $X$ being entangled is captured by the occurrence of an infinite string $(P_1 \ldots P_m)^\omega$. In fact, as proved in the following lemma, finding $(P_1 \ldots P_m)^k$ for arbitrarily big $k$ is sufficient to guarantee the existence of $(P_1 \ldots P_m)^\omega$.

Lemma A.2. *Let $(Z, \leq)$ be a total order, let $\mathcal{P}$ be a partition of $Z$, let $X \subseteq Z$, and let $\mathcal{P}' = \{P_1, \ldots, P_m\} \subseteq \mathcal{P}$. The following are equivalent:*

*(1) For every $k \in \mathbb{N}$, the string $(P_1 \ldots P_m)^k$ is generated by $X$.*
*(2) $(P_1 \ldots P_m)^\omega$ is generated by $X$.*

Proof. The nontrivial implication is (1) $\Rightarrow$ (2). If $m = 1$, then by choosing $k = 1$ we obtain that there exists $x \in X$ such that $x \in P_1$, so $(P_1)^\omega$ occurs in $X$, as witnessed by the monotone sequence $(x_i)_{i \in \mathbb{N}}$ such that $x_i = x$ for every $i \in \mathbb{N}$. Thus, in the following, we can assume $m \geq 2$. This implies that for every $k$, if $(P_1 \ldots P_m)^k$ occurs in $X$ at $x_1, x_2, \ldots, x_{m_k}$, then $x_1 < x_2 < \ldots < x_{m_k}$, that is, the inequalities are strict. If (1) holds, we prove that we can find an infinite family $(Y_i)_{i \geq 1}$ of pairwise disjoint subsets of $X$, each containing an occurrence of $(P_1 \ldots P_m)$, such that for every pair of distinct integers $i, j$ it holds either $Y_i < Y_j$ (that is, each element in $Y_i$ is smaller than each element in $Y_j$) or $Y_j < Y_i$. This will imply (2), because if the set $\{i \geq 1 | (\forall j > i)(Y_i < Y_j)\}$ is infinite, then (2) is witnessed by an increasing sequence, and if $\{i \geq 1 | (\forall j > i)(Y_i < Y_j)\}$ is finite, then (2) is witnessed by a decreasing sequence.

Let us show a recursive construction of $(Y_i)_{i \geq 1}$. We say that $\langle X_1, X_2 \rangle$ is a *split* of $X$ if $\{X_1, X_2\}$ is a partition of $X$ and $X_1 < X_2$. Given a split $\langle X_1, X_2 \rangle$ of $X$, we claim that (1) must hold for either $X_1$ or $X_2$ (or both). In fact, reasoning by contradiction, assume there exists $\bar{k}$ such that the string $(P_1 \ldots P_m)^{\bar{k}}$ is neither generated by $X_1$ nor by $X_2$. This promptly leads to a contradiction, since $(P_1 \ldots P_m)^{2\bar{k}}$ is generated by $X$ and hence, if $(P_1 \ldots P_m)^{\bar{k}}$ is not generated by $X_1$, then it must be generated by $X_2$.

Now consider an occurrence of $(P_1 \ldots P_m)^2$ generated by $X$ and a split $\langle X_1, X_2 \rangle$ such that $(P_1 \ldots P_m)$ is generated by $X_1$ and $(P_1 \ldots P_m)$ is generated by $X_2$. Now, if (1) holds for $X_1$, then define $Y_1 = X_2$ and repeat the construction using $X_1$ instead of $X$. If (1) holds for $X_2$, then define $Y_1 = X_1$ and repeat the construction using $X_2$ instead of $X$. We can then recursively define a family $(Y_i)_{i \geq 1}$ with the desired properties. □

We now introduce the notion of an *entangled convex decomposition*, whose aim is at identifying entangled regions of $(Z, \leq)$ with respect to a partition $\mathcal{P}$.

*Definition A.3.* Let $(Z, \leq)$ be a total order and let $\mathcal{P}$ be a partition of $Z$. We say that a partition $\mathcal{V}$ of $Z$ is an *entangled, convex decomposition of $\mathcal{P}$ in $(Z, \leq)$* (*e.c. decomposition*, for short) if all the elements of $\mathcal{V}$ are entangled (w.r.t. the partition $\mathcal{P}$) convex sets in $(Z, \leq)$.

*Example A.4.* Consider the total order $(\mathbb{Z}, \leq)$, where $\mathbb{Z}$ is the set of all integers and $\leq$ is the usual order on $\mathbb{Z}$. Let $\mathcal{P} = \{P_1, P_2, P_3\}$ be the partition of $\mathbb{Z}$ defined as follows:

$$P_1 = \{n \leq 0 : n \text{ is odd}\} \cup \{n > 0 : n \equiv 1 \bmod 3\}$$
$$P_2 = \{n \leq 0 : n \text{ is even}\} \cup \{n > 0 : n \equiv 2 \bmod 3\},$$
$$P_3 = \{n > 0 : n \equiv 0 \bmod 3\}$$

The partition $\mathcal{P}$ generates the following *trace* over $\mathbb{Z}$:

$$\ldots P_1P_2P_1P_2 \ldots P_1P_2P_1P_2P_3P_1P_2P_3 \ldots$$

Now define $\mathcal{V} = \{V_1, V_2\}$, where $V_1 = \{n \in \mathbb{Z} : n \leq 0\}$, $V_2 = \{n \in \mathbb{Z} : n > 0\}$. It is immediate to check that $\mathcal{V}$ is an e.c. decomposition of $\mathcal{P}$ in $(\mathbb{Z}, \leq)$. More trivially, even $\mathcal{V}' = \{\mathbb{Z}\}$ is an e.c. decomposition of $\mathcal{P}$ in $(\mathbb{Z}, \leq)$.

Below we prove that if $\mathcal{P}$ is a finite partition, then there always exists a *finite* e.c. decomposition of $\mathcal{P}$.

Theorem A.5. *Let $(Z, \leq)$ be a total order, and let $\mathcal{P} = \{P_1, \ldots, P_m\}$ be a finite partition of $Z$. Then, $\mathcal{P}$ admits a finite e.c. decomposition in $(Z, \leq)$.*

Proof. We proceed by induction on $m = |\mathcal{P}|$. If $m = 1$, then $\mathcal{P} = \{Z\}$, so $\{Z\}$ is an e.c. decomposition of $\mathcal{P}$ in $(Z, \leq)$. Assume $m \geq 2$ and notice that we may also assume that the sequence $(P_1 \ldots P_m)^\omega$ is *not* generated by $Z$, otherwise the partition $\mathcal{V} = \{Z\}$ is a finite e.c. decomposition of $\mathcal{P}$ in $(Z, \leq)$ and we are done. Since $(P_1 \ldots P_m)^\omega$ is not generated by $Z$, for any permutation $\pi$ of the set $\{1, \ldots, m\}$ the sequence $(P_{\pi(1)}, \ldots, P_{\pi(m)})^\omega$ is not generated by $Z$ and therefore, by Lemma A.2, for any $\pi$ there exists an integer $s_\pi$ such that $(P_{\pi(1)}, \ldots, P_{\pi(m)})^{s_\pi}$ is not generated by $Z$. Using this property we prove that there exists a finite partition $\mathcal{V}$ of $Z$ into convex sets such that for every $V \in \mathcal{V}$ and every $\pi$, the string $(P_{\pi(1)}, \ldots, P_{\pi(m)})^2$ does not occur in $V$.

Consider the following procedure:

1: $\mathcal{V} \leftarrow \{Z\}$;{initialise the partition}
2: **for** $\pi$ permutation of $\{1, \ldots, m\}$ **do**
3: **while** exists an element in $\mathcal{V}$ generating $(P_{\pi(1)} \ldots P_{\pi(m)})^2$ **do**
4: let $V \in \mathcal{V}$ generating $(P_{\pi(1)} \ldots P_{\pi(m)})^2$;
5: $\mathcal{V} \leftarrow \mathcal{V} \setminus V$;
6: let $\alpha_1 < \cdots < \alpha_m < \alpha'_1 < \cdots < \alpha'_m$ in $V$ be such that $\alpha_j, \alpha'_j \in P_{\pi(j)}$, for $j = 1, \ldots, m$;
7: $\mathcal{V} \leftarrow \mathcal{V} \cup \{\{\alpha \in V \mid \alpha \leq \alpha_m\}, \{\alpha \in V \mid \alpha > \alpha_m\}\}$;
8: **end while**
9: **end for**
10: **return** $\mathcal{V}$

The procedure starts with $\mathcal{V} = \{Z\}$ and recursively partitions a $V$ in $\mathcal{V}$ into two nonempty convex sets as long as $(P_{\pi(1)}, \ldots, P_{\pi(m)})^2$ occurs in $V$. Notice that the procedure ends after at most $\sum_\pi s_\pi$ iterations, returning a finite partition $\mathcal{V}$ of $Z$ into convex sets.

Now fix $V \in \mathcal{V}$ and consider the partition $\mathcal{P}_{|V} = \{P \cap V \mid P \in \mathcal{P} \wedge P \cap V \neq \emptyset\} = \{P \cap V \mid P \in \mathcal{P}_V\}$ of $V$, where $\mathcal{P}_V$ is as in Definition A.1 and $|\mathcal{P}_{|_V}| \leq m$. To complete the proof it will be enough to prove that $\mathcal{P}_{|V}$ admits a finite e.c. decomposition in $(V, \leq)$ because then a finite e.c. decomposition of $\mathcal{P}$ in $(Z, \leq)$ can be obtained by merging all the decompositions of $\mathcal{P}_{|V}$, for $V \in \mathcal{V}$.

If $|\mathcal{P}_{|V}| < m$, then $\mathcal{P}_{|V}$ admits an e.c. decomposition by inductive hypothesis. Otherwise, $|\mathcal{P}_{|V}| = m$, say $\mathcal{P}_{|V} = \{P'_1, \dots, P'_m\}$. By construction, we know that for any permutation $\pi$ of $\{1, \dots, m\}$ the string $(P'_{\pi(1)}, \dots, P'_{\pi(m)})^2$ does not occur in $V$. Example A.6 below may help with an intuition for the rest of the argument. Let $k \geq 1$ be the number of distinct permutations $\pi$ of $\{1, \dots, m\}$ such that $(P'_{\pi(1)}, \dots, P'_{\pi(m)})$ occurs in $V$. We proceed by induction on $k$. If $k = 1$, then each set $P'_j$ is convex and trivially entangled, so $\{P'_j \mid 1 \leq j \leq m\}$ is an e.c. decomposition of $\mathcal{P}_{|V}$ in $(V, \leq)$. Now assume $k \geq 2$ and fix a permutation $\pi$ such that $(P'_{\pi(1)}, \dots, P'_{\pi(m)})$ occurs in $V$. Define:

$$V_1 = \{\alpha \in V \mid \exists\, \alpha_1, \dots, \alpha_m \text{ with } \alpha \leq \alpha_1 < \cdots < \alpha_m \text{ and } \alpha_i \in P'_{\pi(i)}\}$$

and $V_2 = V \setminus V_1$. Let us prove that $V_1$ and $V_2$ are nonempty. Just observe that if $\alpha_1, \dots, \alpha_m$ is a witness for $(P'_{\pi(1)}, \dots, P'_{\pi(m)})$ in $V$, then $\alpha_1 \in V_1$. Moreover, $\alpha_m \in V \setminus V_1 = V_2$, otherwise, since $m > 1$ and $P'_{\pi(1)} \neq P'_{\pi(m)}$, the sequence $(P'_{\pi(1)}, \dots, P'_{\pi(m)})^2$ would occur in $V$. The previous observation implies also that $(P'_{\pi(1)}, \dots, P'_{\pi(m)})$ does not occur in $V_1$, nor in $V_2$. Moreover $V_1$ and $V_2$ are clearly convex. To conclude it will suffice to prove that the $V_i$-partition $\mathcal{P}_{|V_i} = \{P \cap V_i \mid P \in \mathcal{P}_{|V}, P \cap V_i \neq \emptyset\}$ admits a finite e.c. decomposition in $(V_i, \leq)$, for $i = 1, 2$. For any given $i$, if $|\mathcal{P}_{|V_i}| < m$, we conclude by the inductive hypothesis on $m$. If, instead, $|\mathcal{P}_{|V_i}| = m$, say $\mathcal{P}_{|V_i} = \{P''_1, \dots, P''_m\}$, we use the inductive hypothesis on $k$: the number of distinct permutations $\pi'$ of $\{1, \dots, m\}$ such that $(P''_{\pi'(1)}, \dots, P''_{\pi'(m)})$ occurs in $V_i$ is less than $k$, since $(P'_{\pi(1)}, \dots, P'_{\pi(m)})$ occurs in $V$ while $(P''_{\pi(1)}, \dots, P''_{\pi(m)})$ does not occur in $V_i$. □

*Example A.6.* We give an example of the final part of the construction described in Theorem A.5. Suppose $\mathcal{P}_{|V} = \{P'_1, P'_2, P'_3\}$ leaves the following trace over $(V, \leq)$:

$$(P'_1P'_2)^\omega (P'_3P'_2)^\omega (P'_1)^\omega (P'_2)^\omega.$$

Notice that, as assumed in the last part of the above proof, for any permutation $\pi$ of $\{1, 2, 3\}$ the sequence $(P'_{\pi(1)}P'_{\pi(2)}P'_{\pi(3)})^2$ does not appear in $V$; however, the sequence $(P'_{\pi(1)}P'_{\pi(2)}P'_{\pi(3)})$ appears in $V$ for $\pi = id$. If we fix $\pi = id$ and consider the sets $V_1, V_2$ as in the above proof, then the partition $\mathcal{P}_{|V}$ leaves the following traces on the sets $V_1, V_2$:

$$(P'_1P'_2)^\omega \qquad \text{and} \qquad (P'_3P'_2)^\omega (P'_1)^\omega (P'_2)^\omega,$$

respectively. Notice that $P'_3$ does not appear in $V_1$, while the sequence $(P'_1P'_2P'_3)$ does not appear in $V_2$, so that the inductive hypothesis can be applied.

We say that an e.c. decomposition of $\mathcal{P}$ in $(Z, \leq)$ is a *minimum-size* e.c. decomposition if it has minimum cardinality. As shown in the following remark, minimum-size e.c. decompositions ensure additional interesting properties.

*Remark A.7.* Let $(Z, \leq)$ be a total order, let $\mathcal{P}$ be a finite partition of $Z$, and let $\mathcal{V} = \{V_1, \dots, V_r\}$ be a minimum-size e.c. decomposition of $\mathcal{P}$ in $(Z, \leq)$, where $V_1 < \cdots < V_r$. Then, for every $1 \leq i < r$, we have $\mathcal{P}_{V_i} \nsubseteq \mathcal{P}_{V_{i+1}}$, where $\mathcal{P}_{V_i} = \{P \in \mathcal{P} \mid P \cap V_i \neq \emptyset\}$ (see Definition A.1). In fact, if this were not the case, $\mathcal{V}' = \{V_1, \dots, V_{i-1}, V_i \cup V_{i+1}, \dots V_r\}$ would be a smaller size e.c. decomposition of $\mathcal{P}$ in $(Z, \leq)$. Similarly, for every $1 < i \leq r$, it must be $\mathcal{P}_{V_i} \nsubseteq \mathcal{P}_{V_{i-1}}$. In conclusion, for every $i = 1, \dots, r$, there exist $R_i \in \mathcal{P}_{V_i} \setminus \mathcal{P}_{V_{i+1}}$ and $L_i \in \mathcal{P}_{V_i} \setminus \mathcal{P}_{V_{i-1}}$, where we assume $V_0 = V_{r+1} = \emptyset$.

In general, a minimum-size e.c. decomposition is not unique.

*Example A.8.* Let us show that even in the special case when $(Z, \leq) = (\mathrm{Pref}(\mathcal{L}(\mathcal{D})), \leq)$ and $\mathcal{P} = \{I_u \mid u \in Q\}$ we can have more than one minimum-size e.c. decomposition.

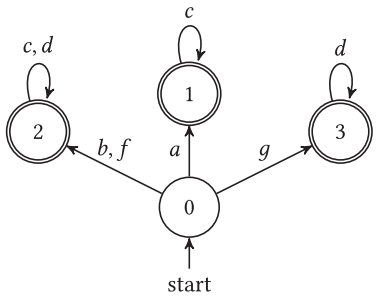



Fig. 13. An automaton $\mathcal{D}$ admitting two distinct minimum-size e.c. decompositions.

Consider the DFA $\mathcal{D}$ in Figure 13. Notice that in every e.c. decomposition of $\mathcal{D}$ one element is $\{\varepsilon\}$, because $I_0 = \{\varepsilon\}$. Moreover, every e.c. decomposition of $\mathcal{D}$ must have cardinality at least three, because, since $1 <_{\mathcal{D}} 3$, states 1 and 3 are not entangled. It is easy to check that:

$$\mathcal{V} = \{\{\varepsilon\}, \{ac^* \cup bc^*\}, \{[b(c+d)^* \setminus bc^*] \cup f(c+d)^* \cup gd^*\}\}$$

and:

$$\mathcal{V}' = \{\{\varepsilon\}, \{ac^* \cup b(c+d)^* \cup [f(c+d)^* \setminus fd^*]\}, \{fd^* \cup gd^*\}\},$$

are two distinct minimum-size e.c. decompositions of $\mathcal{D}$.

We need the following lemmas on entangled convex sets in Section 4.2. Assume that $(Z, \leq)$ is a total order and $\mathcal{P}$ is a partition of $Z$.

Lemma A.9. *Let $\mathcal{V}$ be a minimum-size e.c. decomposition of $\mathcal{P}$. Assume that $V \in \mathcal{V}$ is such that there exist $C_1 < \cdots < C_n$ entangled convex sets with $V \subseteq \bigcup_{i=1}^{n} C_i$. Then, for all $P \in \mathcal{P}$ it holds:*

$$\forall i \in \{1, \ldots, n\}(C_i \cap P \neq \emptyset) \rightarrow V \cap P \neq \emptyset.$$

Proof. If $C_j \subseteq V$ for some $j$, then $V \cap P \neq \emptyset$ since $V \cap P \supseteq C_j \cap P \neq \emptyset$. Otherwise, consider the smallest $i$ such that $C_i \cap V \neq \emptyset$. Since $C_1 < \cdots < C_n$, $V$ is convex and $V$ does not contain any $C_j$, it must be $V \subseteq C_i \cup C_{i+1}$, (where we assume $C_{i+1} = \emptyset$ if $i = n$). Let $\mathcal{V} = \{V_1, \ldots, V_r\}$ with $V_1 < \cdots < V_r$. For all $j = 1, \ldots, r$, consider the elements $R_j \in \mathcal{P}_{V_j} \setminus \mathcal{P}_{V_{j+1}}$ and $L_j \in \mathcal{P}_{V_j} \setminus \mathcal{P}_{V_{j-1}}$ (where we assume $V_0 = V_{r+1} = \emptyset$), as in Remark A.7. Let $s$ be such that $V = V_s$. We distinguish three cases.

(1) $V_s \cap C_{i+1} = \emptyset$. In this case, it must be $V_s \subseteq C_i$. Let $V_{s-h}, V_{s-h+1}, \ldots, V_s, \ldots, V_{s+k-1}, V_{s+k}$ $(h, k \geq 0)$ be all the elements of $\mathcal{V}$ contained in $C_i$. Since $V_1 < \cdots < V_r$ , we conclude:

$$V_{s-h} \cup \cdots \cup V_s \cup \cdots \cup V_{s+k} \subseteq C_i \subseteq V_{s-h-1} \cup V_{s-h} \cup \cdots \cup V_s \cup \cdots \cup V_{s+k} \cup V_{s+k+1}.$$

We know that $L_{s-h}, \ldots, L_s, R_s, \ldots R_{s+k}$ occur in $V_{s-h} \cup \cdots \cup V_{s+k}$, so they also occur in $C_i$. Moreover, we also know that $P$ occurs in $C_i$. Since $C_i$ is entangled, there exists a sequence $\alpha_{s-h} \leq \cdots \leq \alpha_s \leq \beta \leq \gamma_s \leq \cdots \leq \gamma_{s+k}$ of elements in $C_i$ witnessing that the ordered sequence $L_{s-h}, \ldots, L_s, P, R_s, \ldots R_{s+k}$ occurs in $C_i$; it follows that $L_{s-h}, \ldots, L_s, P, R_s, \ldots R_{s+k}$ occurs in $V_{s-h-1} \cup V_{s-h} \cup \cdots \cup V_s \cup \cdots \cup V_{s+k} \cup V_{s+k+1}$ as well. Since $L_{s-h}$ does not occur in $V_{s-h-1}$, then the ordered sequence $L_{s-h+1}, \ldots, L_s, P, R_s, \ldots R_{s+k}$ occurs in $V_{s-h} \cup \cdots \cup V_s \cup \cdots \cup V_{s+k} \cup V_{s+k+1}$. Now, $L_{s-h+1}$ does not occur in $V_{s-h}$, so the ordered sequence $L_{s-h+2}, \ldots, L_s, P, R_s, \ldots R_{s+k}$ occurs in $V_{s-h+1} \cup \cdots \cup V_s \cup \cdots \cup V_{s+k} \cup V_{s+k+1}$. Proceeding in this way, we obtain that the sequence $P, R_s, \ldots R_{s+k}$ occurs in this order in $V_s \cup \cdots \cup V_{s+k} \cup V_{s+k+1}$. Now suppose for sake of a contradiction that $P$ does not occur in $V_s$. As before we obtain that $R_s, \ldots R_{s+k}$ occurs in this order in $V_{s+1} \cup \cdots \cup V_{s+k} \cup V_{s+k+1}$, $R_{s+1}, \ldots R_{s+k}$ occurs in this order in $V_{s+2} \cup \cdots \cup V_{s+k} \cup V_{s+k+1}$, and so on. We finally conclude that $R_{s+k}$ occurs in $V_{s+k+1}$, a contradiction.

(2) $V_s \cap C_i = \emptyset$. In this case, it must be $V_s \subseteq C_{i+1}$ and one concludes as in the previous case.
(3) $V_s \cap C_i \neq \emptyset$ and $V_s \cap C_{i+1} \neq \emptyset$. In this case, let $V_{s-h}, \dots, V_{s-1}$ ($h \geq 0$) be all elements of $\mathcal{V}$ contained in $C_i$, and let $V_{s+1}, \dots, V_{s+k}$ ($k \geq 0$) be all elements of $\mathcal{V}$ contained in $C_{i+1}$. As before:

$$V_{s-h} \cup \dots \cup V_{s-1} \subseteq C_i \subseteq V_{s-h-1} \cup V_{s-h} \cup \dots \cup V_{s-1} \cup V_s$$

and:

$$V_{s+1} \cup \dots \cup V_{s+k} \subseteq C_{i+1} \subseteq V_s \cup V_{s+1} \cup \dots \cup V_{s+k} \cup V_{s+k+1}.$$

Now, assume by contradiction that $P$ does not occur in $V_s$. First, let us prove that $L_s$ does not occur in $C_i$. Suppose by contradiction that $L_s$ occurs in $C_i$. We know that $L_{s-h}, \dots, L_{s-1}$ occurs in $C_i$, and we also know that $P$ occurs in $C_i$. Since $C_i$ is entangled, then $L_{s-h}, \dots, L_{s-1}, L_s, P$ should occur in this order in $C_i$ and so also in $V_{s-h-1} \cup V_{s-h} \cup \dots \cup V_{s-1} \cup V_s$; however, reasoning as in case 1, this would imply that $P$ occurs in $V_s$, a contradiction. Analogously, one shows that $R_s$ does not occur in $C_{i+1}$.
Since $R_s$ and $L_s$ occur in $V_s$, then there exists a monotone sequence in $V_s$ whose trace consists of alternating values of $R_s$ and $L_s$. But $V_s \subseteq C_i \cup C_{i+1}$ and $C_i \prec C_{i+1}$, so the monotone sequence is definitely contained in $C_i$ or $C_{i+1}$. In the first case we would obtain that $L_s$ occurs in $C_i$, and in the second case we would obtain that $R_s$ occurs in $C_{i+1}$, so in both cases we reach a contradiction.

□

Lemma A.10. *Let $C \subseteq Z$ be an entangled convex set and consider any pair of convex sets $C_1, C_2$ such that $C = C_1 \cup C_2$. Then, there exists $i \in \{1, 2\}$ such that $C_i$ is entangled and $\mathcal{P}_{C_i} = \mathcal{P}_C$.*

Proof. Let $(z_i)_{i \geq 1}$ be a monotone sequence witnessing the entanglement of $C$. Then, infinitely many $z_j$'s appear in either $C_1$ or $C_2$ (or both). In the former case $C_1$ is entangled: since $C_1$ is convex, if $j_0 \geq 1$ is such that $z_{j_0} \in C_1$, then the subsequence $(z_j)_{j \geq j_0}$ is in $C_1$ and, clearly, we have $\mathcal{P}_{C_1} = \mathcal{P}_C$. In the latter case, analogously, $C_2$ is entangled and $\mathcal{P}_{C_2} = \mathcal{P}_C$. □

Lemma A.11. *Let $C_1, C_2 \subseteq Z$ be entangled convex sets. Then, at least one the following holds true:*

*(1) $C_1 \setminus C_2$ is entangled and convex and $\mathcal{P}_{C_1 \setminus C_2} = \mathcal{P}_{C_1}$;*
*(2) $C_2 \setminus C_1$ is entangled and convex and $\mathcal{P}_{C_2 \setminus C_1} = \mathcal{P}_{C_2}$;*
*(3) $C_1 \cup C_2$ is entangled and convex and $\mathcal{P}_{C_1 \cup C_2} = \mathcal{P}_{C_1}$ or $\mathcal{P}_{C_1 \cup C_2} = \mathcal{P}_{C_2}$.*

Proof. If $C_1 \cap C_2 = \emptyset$, (1) and (2) hold. If $C_2 \subseteq C_1$ or $C_1 \subseteq C_2$, (3) holds. In the remaining cases observe that $C_1 \setminus C_2$, $C_2 \setminus C_1$, $C_1 \cap C_2$ and $C_1 \cup C_2$ are convex. Since $C_1 = (C_1 \setminus C_2) \cup (C_1 \cap C_2)$ and $C_2 = (C_2 \setminus C_1) \cup (C_1 \cap C_2)$, by Lemma A.10 we conclude that at least one the following holds true:

(1) $C_1 \setminus C_2$ is entangled and convex and $\mathcal{P}_{C_1 \setminus C_2} = \mathcal{P}_{C_1}$, or $C_2 \setminus C_1$ is entangled and convex and $\mathcal{P}_{C_2 \setminus C_1} = \mathcal{P}_{C_2}$;
(2) the intersection $C_1 \cap C_2$ is entangled and convex and $\mathcal{P}_{C_1 \cap C_2} = \mathcal{P}_{C_1} = \mathcal{P}_{C_2}$.

In the first case we are done, while in the second case we have $\mathcal{P}_{C_1 \cap C_2} = \mathcal{P}_{C_1} = \mathcal{P}_{C_2} = \mathcal{P}_{C_1} \cup \mathcal{P}_{C_2} = \mathcal{P}_{C_1 \cup C_2}$. Since $C_1 \cap C_2 \subseteq C_1 \cup C_2$ and $C_1 \cap C_2$ is entangled, we conclude that $C_1 \cup C_2$ is entangled. □

Lemma A.12. *Let $C_1, \dots, C_n \subseteq Z$ be entangled convex sets. Then, there exist $m \leq n$ pairwise disjoint, entangled convex sets $C'_1, \dots, C'_m \subseteq Z$, such that:*

*— $C'_1 < \dots < C'_m$ and $\bigcup_{i=1}^n C_i = \bigcup_{i=1}^m C'_i$;*
*— if $P \in \mathcal{P}$ occurs in all $C_i$'s, then it occurs in all $C'_i$'s as well.*

Proof. We can suppose without loss of generality that the $C_i$'s are non-empty and we proceed by induction on the number of intersections $r = |\{(i,j) | i < j \wedge C_i \cap C_j \neq \emptyset\}|$.

If $r = 0$, then the $C_i$'s are pairwise disjoint and, since they are convex, they are comparable. Hence, it is sufficient to take $C'_1, \ldots, C'_n$ as the permutation of the $C_i$'s such that $C'_1 < \cdots < C'_n$.

Now assume $r \geq 1$ and let, without loss of generality, $C_1 \cap C_2 \neq \emptyset$. We now produce a new sequence of at most $n$ entangled convex sets to which we can apply the inductive hypothesis. By Lemma A.11 at least one among $C_1 \setminus C_2$, $C_2 \setminus C_1$ and $C_1 \cup C_2$, is an entangled convex set. If $C_1 \cup C_2$ is an entangled convex set, then let $C_1 \cup C_2, C_3, \ldots, C_n$ be the new sequence. Otherwise, if $C_1 \setminus C_2$ (the case $C_2 \setminus C_1$ analogous) is entangled, let the new sequence be $C_1 \setminus C_2, C_2, C_3, \ldots, C_n$. In both cases the number of intersections decreases: this is clear in the first case, while in the second case $C_1 \setminus C_2 \subseteq C_1$ and $(C_1 \setminus C_2) \cap C_2 = \emptyset$.

In all the above cases Lemma A.11 implies that if a $P \in \mathcal{P}$ occurs in all $C_i$'s, then it occurs in all elements of the new family and we can conclude by the inductive hypothesis. □

Below we prove a fairly intuitive result on the intersection of convex sets.

Lemma A.13. *Let $(Z, \leq)$ be a total order. If $C_1, \ldots, C_n \subseteq Z$ are non-empty, convex sets such that $C_i \cap C_j \neq \emptyset$ for all $i, j \in \{1, \ldots, n\}$, then $\bigcap_{i=1}^{n} C_i \neq \emptyset$.*

Proof. We proceed by induction on $n$. Cases $n = 1, 2$ are trivial, so assume $n \geq 3$. For every $i \in \{1, \ldots, n\}$, the set:

$$\bigcap_{\substack{k \in \{1, \ldots, n\} \\ k \neq i}} C_k,$$

is nonempty by the inductive hypothesis, so we can pick an element $d_i$. If for some distinct $i$ and $j$ we have $d_i = d_j$, then such an element witnesses that $\bigcap_{i=1}^{n} C_i \neq \emptyset$. Otherwise, assume without loss of generality that $d_1 < \cdots < d_n$. Fix any integer $j$ such that $1 < j < n$, and let us prove that $d_j$ witnesses that $\bigcap_{i=1}^{n} C_i \neq \emptyset$. We only have to prove that $d_j \in C_j$. This follows from $d_1, d_n \in C_j$ and the fact that $C_j$ is convex. □

## ACKNOWLEDGMENTS

We wish to thank the anonymous reviewers for improving the quality of the presentation by providing valuable suggestion. We wish to thank Gonzalo Navarro for pointing out the correct references for the succinct string data structures used in Lemma 5.26 and Manuel Cáceres for pointing out the work [60].

## REFERENCES

[1] Jarno Alanko, Nicola Cotumaccio, and Nicola Prezza. 2022. Linear-time minimization of wheeler DFAs. In *Proceedings of the 2022 Data Compression Conference*. IEEE, 53–62. DOI: https://doi.org/10.1109/DCC52660.2022.00013

[2] Jarno Alanko, Giovanna D'Agostino, Alberto Policriti, and Nicola Prezza. 2020. Regular languages meet prefix sorting. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms*. Shuchi Chawla (Ed.), SIAM, 911–930. DOI: https://doi.org/10.1137/1.9781611975994.55

[3] Jarno Alanko, Giovanna D'Agostino, Alberto Policriti, and Nicola Prezza. 2021. Wheeler languages. *Information & Computation* 281 (2021), 104820. DOI: https://doi.org/10.1016/j.ic.2021.104820

[4] Jarno N. Alanko, Travis Gagie, Gonzalo Navarro, and Louisa Seelbach Benkner. 2019. Tunneling on Wheeler graphs. In *Proceedings of the Data Compression Conference*. Ali Bilgin, Michael W. Marcellin, Joan Serra-Sagristà, and James A. Storer (Eds.), IEEE, 122–131. DOI: https://doi.org/10.1109/DCC.2019.00020

[5] Renzo Angles and Claudio Gutierrez. 2008. Survey of graph database models. *ACM Computing Surveys* 40, 1(2008), 39 pages. DOI: https://doi.org/10.1145/1322432.1322433

[6] Arturs Backurs and Piotr Indyk. 2016. Which regular expression patterns are hard to match?. In *Proceedings of the 2016 IEEE 57th Annual Symposium on Foundations of Computer Science*. IEEE, 457–466.

[7] Uwe Baier, Timo Beller, and Enno Ohlebusch. 2015. Graphical pan-genome analysis with compressed suffix trees and the Burrows–Wheeler transform. *Bioinformatics* 32, 4(2015), 497–504. DOI: https://doi.org/10.1093/bioinformatics/btv603

[8] Ruben Becker, Davide Cenzato, Sung-Hwan Kim, Bojana Kodric, Alberto Policriti, and Nicola Prezza. 2023. Optimal Wheeler Language Recognition. arXiv:2306.04737. Retrieved from https://arxiv.org/abs/2306.04737

[9] Ruben Becker, Manuel Cáceres, Davide Cenzato, Sung-Hwan Kim, Bojana Kodric, Francisco Olivares, and Nicola Prezza. 2023. Sorting finite automata via partition refinement. In *Proceedings of the 31st Annual European Symposium on Algorithms*. arxiv:2305.05129 [cs.DS]

[10] D. Belazzougui and G. Navarro. 2015. Optimal lower and upper bounds for representing sequences. *ACM Transactions on Algorithms* 11, 4 (2015) 1–21.

[11] Jason W. Bentley, Daniel Gibney, and Sharma V. Thankachan. 2020. On the complexity of BWT-runs minimization via alphabet reordering. In *Proceedings of the 28th Annual European Symposium on Algorithms*. Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders (Eds.), Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 15:1–15:13. DOI: https://doi.org/10.4230/LIPIcs.ESA.2020.15

[12] Giulia Bernardini, Pawel Gawrychowski, Nadia Pisanti, Solon P. Pissis, Giovanna Rosone. 2019. Even faster elastic-degenerate string matching via fast matrix multiplication. In *46th International Colloquium on Automata, Languages, and Programming (ICALP'19)*, Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi (Eds.). Vol. 132, Schloss Dagstuhl – Leibniz Center for Informatics, Dagstuhl, 21:1–21:15. http://drops.dagstuhl.de/opus/volltexte/2019/10597

[13] Maciej Besta and Torsten Hoefler. 2019. Survey and taxonomy of lossless graph compression and space-efficient graph representations. arXiv:1806.01799. Retrieved from https://arxiv.org/abs/1806.01799

[14] Philip Bille and Mikkel Thorup. 2009. Faster regular expression matching. In *Proceedings of the Automata, Languages and Programming*. Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris Nikoletseas, and Wolfgang Thomas (Eds.), Springer, Berlin, 171–182.

[15] Guy E. Blelloch and Arash Farzan. 2010. Succinct representations of separable graphs. In *Proceedings of the Combinatorial Pattern Matching*. Amihood Amir and Laxmi Parida (Eds.), Springer, Berlin, 138–150.

[16] Henning Bordihn, Markus Holzer, and Martin Kutrib. 2009. Determination of finite automata accepting subregular languages. *Theoretical Computer Science* 410, 35 (2009), 3209–3222. DOI: https://doi.org/10.1016/j.tcs.2009.05.019 Descriptional Complexity of Formal Systems.

[17] Alexander Bowe, Taku Onodera, Kunihiko Sadakane, and Tetsuo Shibuya. 2012. Succinct de Bruijn graphs. In *Proceedings of the Algorithms in Bioinformatics*. Ben Raphael and Jijun Tang (Eds.), Springer, Berlin, 225–235.

[18] Karl Bringmann, Allan Grønlund, and Kasper Green Larsen. 2017. A dichotomy for regular expression membership testing. In *Proceedings of the 2017 IEEE 58th Annual Symposium on Foundations of Computer Science*. IEEE, 307–318.

[19] Karl Bringmann and Marvin Künnemann. 2015. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE, 79–97.

[20] Nieves R. Brisaboa, Susana Ladra, and Gonzalo Navarro. 2009. k2-trees for compact web graph representation. In *Proceedings of the SPIRE* 9 (2009), 18–30.

[21] Janusz A. Brzozowski and Rina Cohen. 1969. On decompositions of regular events. *Journal of the ACM* 16, 1 (1969), 132–144.

[22] Janusz A. Brzozowski and Faith E. Fich. 1980. Languages of R-trivial Monoids. *Journal of Computer and System Sciences* 20, 1 (1980), 32–49. DOI: https://doi.org/10.1016/0022-0000(80)90003-3

[23] Michael Burrows and David J. Wheeler. 1994. *A Block-sorting Lossless Data Compression Algorithm*. Technical Report 124. Digital Equipment Corporation.

[24] Manuel Caceres. 2023. Parameterized algorithms for string matching to DAGs: Funnels and beyond. In *34th Annual Symposium on Combinatorial Pattern Matching (CPM'23)*, Bulteau, Laurent and Lipták, Zsuzsanna (Eds.). Vol. 259, Schloss Dagstuhl – Leibniz Center for Informatics, Dagstuhl, 7:1–7:19. https://drops.dagstuhl.de/opus/volltexte/2023/17961

[25] Sankardeep Chakraborty, Roberto Grossi, Kunihiko Sadakane, and Srinivasa Rao Satti. 2021. Succinct representations for (Non) deterministic finite automata. In *Proceedings of the LATA*. 55–67.

[26] Sankardeep Chakraborty and Seungbum Jo. 2023. Compact representation of interval graphs and circular-arc graphs of bounded degree and chromatic number. *Theoretical Computer Science* 941 (2023), 156–166. DOI: https://doi.org/10.1016/j.tcs.2022.11.010

[27] Marek Chrobak. 1986. Finite automata and unary languages. *Theoretical Computer Science* 47 (1986), 149–158.

[28] Francisco Claude and Gonzalo Navarro. 2007. A fast and compact Web graph representation. In *Proceedings of the International Symposium on String Processing and Information Retrieval*. Springer, 118–129.

[29] Nicola Cotumaccio. 2022. Graphs can be succinctly indexed for pattern matching in $O(|E|^2 + |V|^{5/2})$ time. In *Proceedings of the 2022 Data Compression Conference*. IEEE, 272–281. DOI: https://doi.org/10.1109/DCC52660.2022.00035
[30] Nicola Cotumaccio. 2023. Prefix Sorting DFAs: A Recursive Algorithm. arXiv:2305.02526. Retrieved from https://arxiv.org/abs/2305.02526
[31] Nicola Cotumaccio, Giovanna D'Agostino, Alberto Policriti, and Nicola Prezza. 2021. Co-lexicographically ordering automata and regular languages – part II. arXiv:2102.06798. Retrieved from https://arxiv.org/abs/2102.06798
[32] Nicola Cotumaccio and Nicola Prezza. 2021. On indexing and compressing finite automata. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms*. Dániel Marx (Ed.), SIAM, 2585–2599. DOI: https://doi.org/10.1137/1.9781611976465.153
[33] Isabel F. Cruz, Alberto O. Mendelzon, and Peter T. Wood. 1987. A graphical query language supporting recursion. *ACM SIGMOD Record* 16, 3 (1987), 323–330.
[34] Giovanna D'Agostino, Davide Martincigh, and Alberto Policriti. 2021. Ordering regular languages: A danger zone. In *Proceedings of the 22nd Italian Conference on Theoretical Computer Science*. Claudio Sacerdoti Coen and Ivano Salvo (Eds.), CEUR-WS.org, 46–69. Retrieved from http://ceur-ws.org/Vol-3072/paper5.pdf
[35] Giovanna D'Agostino, Davide Martincigh, and Alberto Policriti. 2023. Ordering regular languages and automata: Complexity. *Theoretical Computer Science* 949 (2023), 113709. DOI: https://doi.org/10.1016/j.tcs.2023.113709
[36] Narsingh Deo and Bruce Litow. 1998. A structural approach to graph compression. In *Proceedngs of the 23th MFCS Workshop on Communications*. 91–101.
[37] R. P. Dilworth. 1950. A decomposition theorem for partially ordered sets. *Annals of Mathematics* 51, 1 (1950), 161–166. Retrieved from http://www.jstor.org/stable/1969503
[38] Lawrence C. Eggan. 1963. Transition graphs and the star-height of regular events. *Michigan Mathematical Journal* 10, 4 (1963), 385–397.
[39] Joost Engelfriet. 1997. *Context-Free Graph Grammars*. Springer, Berlin, 125–213. DOI: https://doi.org/10.1007/978-3-642-59126-6_3
[40] Massimo Equi, Roberto Grossi, Veli Mäkinen, and Alexandru I. Tomescu. 2019. On the complexity of string matching for graphs. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming*. Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi (Eds.), Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 55:1–55:15. DOI: https://doi.org/10.4230/LIPIcs.ICALP.2019.55
[41] Massimo Equi, Veli Mäkinen, and Alexandru I. Tomescu. 2021. Graphs cannot be indexed in polynomial time for sub-quadratic time string matching, unless SETH fails. In *SOFSEM 2021: Theory and Practice of Computer Science - 47th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2021, Bolzano-Bozen, Italy, January 25–29, 2021, Proceedings*. Tomás Bures, Riccardo Dondi, Johann Gamper, Giovanna Guerrini, Tomasz Jurdzinski, Claus Pahl, Florian Sikora, and Prudence W. H. Wong (Eds.), Lecture Notes in Computer Science, Vol. 12607, Springer, 608–622. DOI: https://doi.org/10.1007/978-3-030-67731-2_44
[42] Arash Farzan and Shahin Kamali. 2014. Compact navigation and distance oracles for graphs with small treewidth. *Algorithmica* 69, 1 (2014), 92–116.
[43] Guy Feigenblat, Ely Porat, and Ariel Shiftan. 2016. Linear time succinct indexable dictionary construction with applications. In *Proceedings of the 2016 Data Compression Conference*. IEEE, 13–22. DOI: https://doi.org/10.1109/DCC.2016.70
[44] P. Ferragina, F. Luccio, G. Manzini, and S. Muthukrishnan. 2005. Structuring labeled trees for optimal succinctness, and beyond. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, 184–193. DOI: https://doi.org/10.1109/SFCS.2005.69
[45] P. Ferragina and G. Manzini. 2000. Opportunistic data structures with applications. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 390–398. DOI: https://doi.org/10.1109/SFCS.2000.892127
[46] L. Ferres, J. Fuentes-Sepúlveda, T. Gagie, M. He, and G. Navarro. 2020. Fast and compact planar embeddings. *Computational Geometry Theory and Applications* 89 (2020).
[47] Travis Gagie, Giovanni Manzini, and Jouni Sirén. 2017. Wheeler graphs: A framework for BWT-based data structures. *Theoretical Computer Science* 698 (2017), 67–78. DOI: https://doi.org/10.1016/j.tcs.2017.06.016 Algorithms, Strings and Theoretical Approaches in the Big Data Era (In Honor of the 60th Birthday of Professor Raffaele Giancarlo).
[48] Travis Gagie, Gonzalo Navarro, and Nicola Prezza. 2020. Fully functional suffix trees and optimal text searching in BWT-runs bounded space. *Journal of the ACM* 67, 1(2020), 54 pages. DOI: https://doi.org/10.1145/3375890
[49] Daniel Gibney. 2020. An efficient elastic-degenerate text index? Not likely. In *Proceedings of the International Symposium on String Processing and Information Retrieval*. Springer, 76–88.
[50] Daniel Gibney, Gary Hoppenworth, and Sharma V. Thankachan. 2021. Simple reductions from formula-SAT to pattern matching on labeled graphs and subtree isomorphism. In *Proceedings of the 4th Symposium on Simplicity in Algorithms*. Hung Viet Le and Valerie King (Eds.), SIAM, 232–242. DOI: https://doi.org/10.1137/1.9781611976496.26

[51] Daniel Gibney, Gary Hoppenworth, and Sharma V. Thankachan. 2021. Simple reductions from formula-SAT to pattern matching on labeled graphs and subtree isomorphism. In *Proceedings of the Symposium on Simplicity in Algorithms*. SIAM, 232–242.

[52] Daniel Gibney and Sharma V. Thankachan. 2021. Text indexing for regular expression matching. *Algorithms* 14, 5 (2021), 133.

[53] Daniel Gibney and Sharma V. Thankachan. 2022. On the complexity of recognizing Wheeler graphs. *Algorithmica* 84, 3 (2022), 784–814.

[54] Torben Hagerup and Torsten Tholey. 2001. Efficient minimal perfect hashing in nearly minimal space. In *Proceedings of the Annual Symposium on Theoretical Aspects of Computer Science*. Springer, 317–326.

[55] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2006. *Introduction to Automata Theory, Languages, and Computation* (3rd ed). Addison-Wesley Longman Publishing Co., Inc.

[56] Russell Impagliazzo and Ramamohan Paturi. 2001. On the complexity of k-SAT. *Journal of Computer and System Sciences* 62, 2 (2001), 367–375.

[57] Jesper Jansson, Kunihiko Sadakane, and Wing-Kin Sung. 2012. Ultra-succinct representation of ordered trees with applications. *Journal of Computer and System Sciences* 78, 2 (2012), 619–631.

[58] Shahin Kamali. 2018. Compact representation of graphs of small clique-width. *Algorithmica* 80, 7 (2018), 2106–2131.

[59] Shahin Kamali. 2022. Compact representation of graphs with bounded bandwidth or treedepth. *Information and Computation* 285 (2022), 104867. DOI: https://doi.org/10.1016/j.ic.2022.104867

[60] Shimon Kogan and Merav Parter. 2022. Beating matrix multiplication for $n^{1/3}$-directed shortcuts. In *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming*. Mikołaj Bojańczyk, Emanuela Merelli, and David P. Woodruff (Eds.), Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 82:1–82:20. DOI: https://doi.org/10.4230/LIPIcs.ICALP.2022.82

[61] Leonid Libkin. 2004. *Elements of Finite Model Theory*. Springer. DOI: https://doi.org/10.1007/978-3-662-07003-1

[62] Anirban Majumdar and Denis Kuperberg. 2019. Computing the width of non-deterministic automata. *Logical Methods in Computer Science* 15, 4 (2019), 10:1–10:31.

[63] Veli Mäkinen, Niko Välimäki, and Jouni Sirén. 2014. Indexing graphs for path queries with applications in genome research. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 11, 2 (2014), 375–388.

[64] S. Mantaci, A. Restivo, G. Rosone, and M. Sciortino. 2007. An extension of the Burrows–Wheeler Transform. *Theoretical Computer Science* 387, 3 (2007), 298–312. DOI: https://doi.org/10.1016/j.tcs.2007.07.014 The Burrows-Wheeler Transform.

[65] Giovanni Manzini. 1999. The Burrows-Wheeler transform: Theory and practice. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science*. Springer, 34–47.

[66] Tomás Masopust and Markus Krötzsch. 2021. Partially ordered automata and piecewise testability. *Logical Methods in Computer Science* 17, 2 (2021), 14:1–14:36. Retrieved from https://lmcs.episciences.org/7475

[67] Robert McNaughton and Seymour A Papert. 1971. *Counter-Free Automata (MIT Research Monograph No. 65)*. The MIT Press.

[68] Gonzalo Navarro. 2016. *Compact Data Structures: A Practical Approach*. Cambridge University Press.

[69] Abhinav Nellore, Austin Nguyen, and Reid F. Thompson. 2021. An invertible transform for efficient string matching in labeled digraphs. In *Proceedings of the 32nd Annual Symposium on Combinatorial Pattern Matching*. Paweł Gawrychowski and Tatiana Starikovskaya (Eds.), Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 20:1–20:14. Retrieved from https://drops.dagstuhl.de/opus/volltexte/2021/13971

[70] Aaron Potechin and Jeffrey O. Shallit. 2020. Lengths of words accepted by nondeterministic finite automata. *Information Processing Letters* 162 (2020), 105993. DOI: https://doi.org/10.1016/j.ipl.2020.105993

[71] Nicola Prezza. 2021. On locating paths in compressed tries. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms*. Dániel Marx (Ed.), SIAM, 744–760. DOI: https://doi.org/10.1137/1.9781611976465.47

[72] Nicola Rizzo, Alexandru I. Tomescu, and Alberto Policriti. 2022. Solving string problems on graphs using the labeled direct product. *Algorithmica* 84, 10 (2022), 3008–3033.

[73] Fred S. Roberts. 1969. On the boxicity and cubicity of a graph. *Recent Progress in Combinatorics* 1, 1 (1969), 301–310.

[74] Kai Salomaa and Sheng Yu. 1997. NFA to DFA transformation for finite languages over arbitrary languages. *Journal of Automata, Languages and Combinatorics* 2, 3 (1997), 177–186.

[75] Thomas Schwentick, Denis Thérien, and Heribert Vollmer. 2001. Partially-ordered two-way automata: a new characterization of DA. In *Proceedings of theInternational Conference on Developments in Language Theory*. Werner Kuich, Grzegorz Rozenberg, and Arto Salomaa (Eds.), Lecture Notes in Computer Science, Vol. 2295, Springer, 239–250. DOI: https://doi.org/10.1007/3-540-46011-X_20

[76] M. P. Schützenberger. 1965. On finite monoids having only trivial subgroups. *Information and Control* 8, 2 (1965), 190–194. DOI: https://doi.org/10.1016/S0019-9958(65)90108-7

[77] H.-J. Shyr and G. Thierrin. 1974. Ordered automata and associated languages. *Tamkang J. Math* 5, 1 (1974), 9–20.
[78] Jouni Sirén, Niko Välimäki, and Veli Mäkinen. 2014. Indexing graphs for path queries with applications in genome research. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 11, 2(2014), 375–388. DOI: https://doi.org/10.1109/TCBB.2013.2297101
[79] L. J. Stockmeyer and A. R. Meyer. 1973. Word problems requiring exponential time (preliminary report). In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing*. Association for Computing Machinery, New York, NY, 1–9. DOI: https://doi.org/10.1145/800125.804029
[80] Howard Straubing. 1981. A generalization of the Schützenberger product of finite monoids. *Theoretical Computer Science* 13, 2 (1981), 137–150.
[81] Denis Thérien. 1981. Classification of finite monoids: The language approach. *Theoretical Computer Science* 14, 2 (1981), 195–208.
[82] Chengcheng Xu, Shuhui Chen, Jinshu Su, S. M. Yiu, and Lucas C. K. Hui. 2016. A survey on regular expression matching for deep packet inspection: Applications, algorithms, and hardware platforms. *IEEE Communications Surveys & Tutorials* 18, 4 (2016), 2991–3029. DOI: https://doi.org/10.1109/COMST.2016.2566669
[83] Tatsuya Yanagita, Sankardeep Chakraborty, Kunihiko Sadakane, and Srinivasa Rao Satti. 2022. Space-efficient data structure for posets with applications. In *Proceedings of the 18th Scandinavian Symposium and Workshops on Algorithm Theory*. Artur Czumaj and Qin Xin (Eds.), Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 33:1–33:16. DOI: https://doi.org/10.4230/LIPIcs.SWAT.2022.33

Received 10 August 2022; revised 15 March 2023; accepted 22 June 2023