The complexity of separation for levels in concatenation hierarchies

Thomas Place
LaBRI Bordeaux University and IUF, France

Marc Zeitoun
LaBRI Bordeaux University, France

Abstract

We investigate the complexity of the separation problem associated to classes of regular languages. For a class $\mathcal{C}$, $\mathcal{C}$-separation takes two regular languages as input and asks whether there exists a third language in $\mathcal{C}$ which includes the first and is disjoint from the second. First, in contrast with the situation for the classical membership problem, we prove that for most classes $\mathcal{C}$, the complexity of $\mathcal{C}$-separation does not depend on how the input languages are represented: it is the same for nondeterministic finite automata and monoid morphisms. Then, we investigate specific classes belonging to finitely based concatenation hierarchies. It was recently proved that the problem is always decidable for levels 1/2 and 1 of any such hierarchy (with inefficient algorithms). Here, we build on these results to show that when the alphabet is fixed, there are polynomial time algorithms for both levels. Finally, we investigate levels 3/2 and 2 of the famous Straubing-Thérien hierarchy. We show that separation is PSpace-complete for level 3/2 and between PSpace-hard and EXPTime for level 2.

2012 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases Regular languages, separation, concatenation hierarchies, complexity

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding Both authors acknowledge support from the DeLTA project (ANR-16-CE40-0007).

1 Introduction

For more than 50 years, a significant research effort in theoretical computer science was made to solve the membership problem for regular languages. This problem consists in determining whether a class of regular languages is decidable, that is, whether there is an algorithm inputing a regular language and outputing 'yes' if the language belongs to the investigated class, and 'no' otherwise.

Many results were obtained in a long and fruitful line of research. The most prominent one is certainly Schützenberger's theorem [19], which gives such an algorithm for the class of star-free languages. For most interesting classes also, we know precisely the computational cost of the membership problem. As can be expected, this cost depends on the way the input language is given. Indeed, there are several ways to input a regular language. For instance, it can be given by a nondeterministic finite automaton (NFA), or, alternately, by a morphism into a finite monoid. While obtaining an NFA representation from a morphism into a monoid has only a linear cost, the converse direction is much more expensive: from an NFA with n states, the smallest monoid recognizing the same language may have an exponential number of elements (the standard construction yields 2^{n^2} elements). This explains why the complexity of the membership problem depends on the representation of the input. For

instance, for the class of star-free languages, it is PSpace-complete if one starts from NFAs (and actually, even from DFAs [2]) while it is NL when starting from monoid morphisms.

Recently, another problem, called separation, has replaced membership as the cornerstone in the investigation of regular languages. It takes as input *two* regular langages instead of one, and asks whether there exists a third language from the class under investigation including the first input language and having empty intersection with the second one. This problem has served recently as a major ingredient in the resolution of difficult membership problems, such as the so-called dot-depth two problem [16] which remained open for 40 years (see [13, 18, 6] for recent surveys on the topic). Dot-depth two is a class belonging to a famous *concatenation hierarchy* which stratifies the star-free languages: the dot-depth [1]. A specific concatenation hierarchy is built in a generic way. One starts from a base class (level 0 of the hierarchy) and builds increasingly growing classes (called levels and denoted by 1/2, 1, 3/2, 2, ...) by alternating two standard closure operations: polynomial and Boolean closure. Concatenation hierarchies account for a significant part of the open questions in this research area. The state of the art regarding separation is captured by only three results [17, 9]: in finitely based concatenation hierarchies (i.e. those whose basis is a finite class) levels 1/2, 1 and 3/2 have decidable separation. Moreover, using specific transfer results [15], this can be pushed to the levels 3/2 and 2 for the two most famous finitely based hierarchies: the dot-depth [1] and the Straubing-Thérien hierarchy [21, 22].

Unlike the situation for membership and despite these recent decidability results for separability in concatenation hierarchies, the complexity of the problems and of the corresponding algorithms has not been investigated so far (except for the class of piecewise testable languages [3, 11, 5], which is level 1 in the Straubing-Thérien hierarchy). The aim of this paper is to establish such complexity results. Our contributions are the following:

- We present a **generic** reduction, which shows that for many natural classes, the way the input is given (by NFAs or finite monoids) has **no impact** on the complexity of the separation problem. This is proved using two LogSpace reductions from one problem to the other. This situation is surprising and opposite to that of the membership problem, where an exponential blow-up is unavoidable when going from NFAs to monoids.
- Building on the results of [17], we show that when the alphabet is fixed, there are polynomial time algorithms for levels 1/2 and 1 in any finitely based hierarchy.
- We investigate levels 3/2 and 2 of the famous Straubing-Thérien hierarchy, and we show that separation is PSpace-complete for level 3/2 and between PSpace-hard and EXPTime for level 2. The upper bounds are based on the results of [17] while the lower bounds are based on independent reductions.

Organization. In Section 2, we give preliminary terminology on the objects investigated in the paper. Sections 3, 4 and 5 are then devoted to the three above points. Due to space limitations, many proofs are postponed to the full version of the paper.

2 Preliminaries

In this section, we present the key objects of this paper. We define words and regular languages, classes of languages, the separation problem and finally, concatenation hierarchies.

2.1 Words and regular languages

An alphabet is a *finite* set A of symbols, called *letters*. Given some alphabet A, we denote by A^+ the set of all nonempty finite words and by A^* the set of all finite words over A (*i.e.*,

$A^* = A^+ \cup \{\varepsilon\}$). If $u \in A^*$ and $v \in A^*$ we write $u \cdot v \in A^*$ or $uv \in A^*$ for the concatenation of u and v. A *language* over an alphabet A is a subset of A^*. Abusing terminology, if $u \in A^*$ is some word, we denote by u the singleton language $\{u\}$. It is standard to extend concatenation to languages: given $K, L \subseteq A^*$, we write $KL = \{uv \mid u \in K \text{ and } v \in L\}$. Moreover, we also consider marked concatenation, which is less standard. Given $K, L \subseteq A^*$, *a marked concatenation* of K with L is a language of the form KaL, for some $a \in A$.

We consider *regular languages*, which can be equivalently defined by *regular expressions*, *nondeterministic finite automata* (NFAs), *finite monoids* or *monadic second-order logic* (MSO). In the paper, we investigate the separation problem which takes regular languages as input. Since we are focused on complexity, how we represent these languages in our inputs matters. We shall consider two kinds of representations: NFAs and monoids. Let us briefly recall these objects and fix the terminology (we refer the reader to [7] for details).

NFAs. An NFA is a tuple $\mathcal{A} = (A, Q, \delta, I, F)$ where A is an alphabet, Q a finite set of states, $\delta \subseteq Q \times A \times Q$ a set of transitions, $I \subseteq Q$ a set of initial states and $F \subseteq Q$ a set of final states. The language $L(\mathcal{A}) \subseteq A^*$ consists of all words labeling a run from an initial state to a final state. The regular languages are exactly those which are recognized by an NFA. Finally, we write "DFA" for *deterministic* finite automata, which are defined in the standard way.

Monoids. We turn to the algebraic definition of regular languages. A *monoid* is a set M endowed with an associative multiplication $(s, t) \mapsto s \cdot t$ (also denoted by st) having a neutral element 1_M, *i.e.*, such that $1_M \cdot s = s \cdot 1_M = s$ for every $s \in M$. An *idempotent* of a monoid M is an element $e \in M$ such that $ee = e$.

Observe that A^* is a monoid whose multiplication is concatenation (the neutral element is ε). Thus, we may consider monoid morphisms $\alpha : A^* \to M$ where M is an arbitrary monoid. Given such a morphism, we say that a language $L \subseteq A^*$ is *recognized* by α when there exists a set $F \subseteq M$ such that $L = \alpha^{-1}(F)$. It is well-known that the regular languages are also those which are recognized by a morphism into a *finite* monoid. When representing a regular language L by a morphism into a finite monoid, one needs to give both the morphism $\alpha : A^* \to M$ (*i.e.*, the image of each letter) and the set $F \subseteq M$ such that $L = \alpha^{-1}(F)$.

2.2 Classes of languages and separation

A class of languages $\mathcal{C}$ is a correspondence $A \mapsto \mathcal{C}(A)$ which, to an alphabet A, associates a set of languages $\mathcal{C}(A)$ over A.

▶ Remark. When two alphabets A, B satisfy $A \subseteq B$, the definition of classes does not require $\mathcal{C}(A)$ and $\mathcal{C}(B)$ to be comparable. In fact, it may happen that a particular language $L \subseteq A^* \subseteq B^*$ belongs to $\mathcal{C}(A)$ but not to $\mathcal{C}(B)$ (or the opposite). For example, we may consider the class $\mathcal{C}$ defined by $\mathcal{C}(A) = \{\emptyset, A^*\}$ for every alphabet A. When $A \subsetneq B$, we have $A^* \in \mathcal{C}(A)$ while $A^* \notin \mathcal{C}(B)$.

We say that $\mathcal{C}$ is a *lattice* when for every alphabet A, we have $\emptyset, A^* \in \mathcal{C}(A)$ and $\mathcal{C}(A)$ is closed under finite union and finite intersection: for any $K, L \in \mathcal{C}(A)$, we have $K \cup L \in \mathcal{C}(A)$ and $K \cap L \in \mathcal{C}(A)$. Moreover, a *Boolean algebra* is a lattice $\mathcal{C}$ which is additionally closed under complement: for any $L \in \mathcal{C}(A)$, we have $A^* \setminus L \in \mathcal{C}(A)$. Finally, a class $\mathcal{C}$ is *quotienting* if it is closed under quotients. That is, for every alphabet A, $L \in \mathcal{C}(A)$ and word $u \in A^*$, the following properties hold:

$$u^{-1}L \stackrel{\text{def}}{=} \{w \in A^* \mid uw \in L\} \quad \text{and} \quad Lu^{-1} \stackrel{\text{def}}{=} \{w \in A^* \mid wu \in L\} \quad \text{both belong to } \mathcal{C}(A).$$

All classes that we consider in the paper are (at least) quotienting lattices consisting of *regular languages*. Moreover, some of them satisfy an additional property called *closure under inverse image*.

Recall that A^* is a monoid for any alphabet A. We say that a class $\mathcal{C}$ is *closed under inverse image* if for every two alphabets A, B, every monoid morphism $\alpha : A^* \to B^*$ and every language $L \in \mathcal{C}(B)$, we have $\alpha^{-1}(L) \in \mathcal{C}(A)$. A quotienting lattice (resp. quotienting Boolean algebra) closed under inverse image is called a *positive variety* (resp. *variety*).

Separation. Consider a class of languages $\mathcal{C}$. Given an alphabet A and two languages $L_1, L_2 \subseteq A^*$, we say that L_1 is $\mathcal{C}$-separable from L_2 when there exists a third language $K \in \mathcal{C}(A)$ such that $L_1 \subseteq K$ and $L_2 \cap K = \emptyset$. In particular, K is called a *separator* in $\mathcal{C}$. The $\mathcal{C}$-separation problem is now defined as follows:

Input: An alphabet A and two regular languages $L_1, L_2 \subseteq A^*$.
Output: Is L_1 $\mathcal{C}$-separable from L_2 ?

▶ Remark. Separation generalizes the simpler *membership problem*, which asks whether a single regular language belongs to $\mathcal{C}$. Indeed $L \in \mathcal{C}$ if and only if L is $\mathcal{C}$-separable from $A^* \setminus L$ (which is also regular and computable from L).

Most papers on separation are mainly concerned about decidability. Hence, they do not go beyond the above presentation of the problem (see [3, 16, 12, 17] for example). However, this paper specifically investigates complexity. Consequently, we shall need to be more precise and take additional parameters into account. First, it will be important to specify whether the alphabet over which the input languages is part of the input (as above) or a constant. When considering separation for some fixed alphabet A, we shall speak of "$\mathcal{C}(A)$-separation". When the alphabet is part of the input, we simply speak of "$\mathcal{C}$-separation".

Another important parameter is how the two input languages are represented. We shall consider NFAs and monoids. We speak of *separation for* NFAs *and separation for monoids*. Note that one may efficiently reduce the latter to the former. Indeed, given a language $L \subseteq A^*$ recognized by some morphism $\alpha : A^* \to M$, it is simple to efficiently compute a NFA with $|M|$ states recognizing L (see [7] for example). Hence, we have the following lemma.

▶ **Lemma 1.** *For any class $\mathcal{C}$, there is a* LogSpace *reduction from $\mathcal{C}$-separation for monoids to $\mathcal{C}$-separation for* NFAs.

Getting an efficient reduction for the converse direction is much more difficult since going from NFAs (or even DFAs) to monoids usually involves an exponential blow-up. However, we shall see in Section 3 that for many natural classes $\mathcal{C}$, this is actually possible.

2.3 Concatenation hierarchies

We now briefly recall the definition of concatenation hierarchies. We refer the reader to [18] for a more detailed presentation. A particular concatenation hierarchy is built from a starting class of languages $\mathcal{C}$, which is called its *basis*. In order to get robust properties, we restrict $\mathcal{C}$ to be a quotienting Boolean algebra of regular languages. The basis is the only parameter in the construction. Once fixed, the construction is generic: each new level is built from the previous one by applying generic operators: either Boolean closure, or polynomial closure. Let us first define these two operators.

Definition. Consider a class $\mathcal{C}$. We denote by $\mathit{Bool}(\mathcal{C})$ the *Boolean closure* of $\mathcal{C}$: for every alphabet A, $\mathit{Bool}(\mathcal{C})(A)$ is the least set containing $\mathcal{C}(A)$ and closed under Boolean operations. Moreover, we denote by $\mathit{Pol}(\mathcal{C})$ the *polynomial closure* of $\mathcal{C}$: for every alphabet A,

$Pol(\mathcal{C})(A)$ is the least set containing $\mathcal{C}(A)$ and closed under union and marked concatenation (if $K, L \in Pol(\mathcal{C})(A)$ and $a \in A$, then $K \cup L, KaL \in Pol(\mathcal{C})(A)$).

Consider a quotienting Boolean algebra of regular languages $\mathcal{C}$. The concatenation hierarchy of basis $\mathcal{C}$ is defined as follows. Languages are classified into levels of two kinds: full levels (denoted by 0, 1, 2,...) and half levels (denoted by 1/2, 3/2, 5/2,...). Level 0 is the basis (*i.e.*, $\mathcal{C}$) and for every $n \in \mathbb{N}$,

- The *half level* $n + 1/2$ is the *polynomial closure* of the previous full level, *i.e.*, of level n.
- The *full level* $n + 1$ is the *Boolean closure* of the previous half level, *i.e.*, of level $n + 1/2$.

$$0 \xrightarrow{Pol} 1/2 \xrightarrow[Bool]{} 1 \xrightarrow{Pol} 3/2 \xrightarrow[Bool]{} 2 \xrightarrow{Pol} 5/2 \cdots\cdots$$

We write $\frac{1}{2}\mathbb{N} = \{0, 1/2, 1, 2, 3/2, 3, \dots\}$ for the set of all possible levels in a concatenation hierarchy. Moreover, for any basis $\mathcal{C}$ and $n \in \frac{1}{2}\mathbb{N}$, we write $\mathcal{C}[n]$ for level n in the concatenation hierarchy of basis $\mathcal{C}$. It is known that every half-level is a quotienting lattice and every full level is a quotienting Boolean algebra (see [18] for a recent proof).

We are interested in finitely based concatenation hierarchies: if $\mathcal{C}$ is the basis, then $\mathcal{C}(A)$ is finite for every alphabet A. Indeed, it was shown in [17] that for such hierarchies separation is always decidable for the levels 1/2 and 1 (in fact, while we do not discuss this in the paper, this is also true for level 3/2, see [9] for a preliminary version). In Section 4, we build on the results of [17] and show that when the alphabet is fixed, this can be achieved in polynomial time for both levels 1/2 and 1. Moreover, we shall also investigate the famous *Straubing-Thérien* hierarchy in Section 5. Our motivation for investigating this hierarchy in particular is that the results of [17] can be pushed to levels 3/2 and 2 in this special case.

3 Handling NFAs

In this section, we investigate how the representation of input languages impact the complexity of separation. We prove that for many natural classes $\mathcal{C}$ (including most of those considered in the paper), $\mathcal{C}$-separation has the same complexity for NFAs as for monoids. Because of these results, we shall be able to restrict ourselves to monoids in later sections.

▶ Remark. This result highlights a striking difference between separation and the simpler membership problem. For most classes $\mathcal{C}$, $\mathcal{C}$-membership is strictly harder for NFAs than for monoids. This is because when starting from a NFA, typical membership algorithms require to either determinize $\mathcal{A}$ or compute a monoid morphism recognizing $L(\mathcal{A})$ which involves an exponential blow-up in both cases. Our results show that the situation differs for separation.

We already have a generic efficient reduction from $\mathcal{C}$-separation for monoids to $\mathcal{C}$-separation for NFAs (see Lemma 1). Here, we investigate the opposite direction: given some class $\mathcal{C}$, is it possible to *efficiently* reduce $\mathcal{C}$-separation for NFAs to $\mathcal{C}$-separation for monoids ? As far as we know, there exists no such reduction which is generic to all classes $\mathcal{C}$.

▶ Remark. There exists an *inefficient* generic reduction from separation for NFAs to the separation for monoids. Given as input two NFAs $\mathcal{A}_1, \mathcal{A}_2$, one may compute monoid morphisms recognizing $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$. This approach is not satisfying as it involves an exponential blow-up: we end-up with monoids M_i of size $2^{|Q_i|^2}$ where Q_i is the set of states of $\mathcal{A}_i$.

Here, we present a set of conditions applying to a pair of classes $(\mathcal{C}, \mathcal{D})$. When they are satisfied, there exists an efficient reduction from $\mathcal{C}$-separation for NFAs to $\mathcal{D}$-separation for monoids. By themselves, these conditions are abstract. However, we highlight two concrete applications. First, for every positive variety $\mathcal{C}$, the pair $(\mathcal{C}, \mathcal{C})$ satisfies the conditions. Second,

for every finitely based concatenation hierarchies of basis $\mathcal{C}$, there exists another finite basis $\mathcal{D}$ such that for every $n \in \frac{1}{2}\mathbb{N}$, the pair $(\mathcal{C}[n], \mathcal{D}[n])$ satisfies the conditions

We first introduce the notions we need to present the reduction and the conditions required to apply it. Then, we state the reduction itself and its applications.

3.1 Generic theorem

We fix a special two letter alphabet $\mathbb{E} = \{0, 1\}$. For the sake of improved readability, we abuse terminology and assume that when considering an arbitrary alphabet A, it always has empty intersection with $\mathbb{E}$. This is harmless as we may work up to bijective renaming.

We exhibit conditions applying to a pair of classes $(\mathcal{C}, \mathcal{D})$. Then, we prove that they imply the existence of an efficient reduction from $\mathcal{C}$-separation for NFAs to $\mathcal{D}$-separation for monoids. This reduction is based on a construction which takes as input a NFA $\mathcal{A}$ (over some arbitrary alphabet A) and builds a modified version of the language $L(\mathcal{A})$ (over $A \cup \mathbb{E}$) which is recognized by a "small" monoid. Our conditions involve two kinds of hypotheses:

1. First, we need properties related to inverse image: "$\mathcal{D}$ must be an an extension of $\mathcal{C}$".
2. The construction is parametrized by an object called "tagging". We need an algorithm which builds special taggings (with respect to $\mathcal{D}$) efficiently.

We now make these two notions more precise. Let us start with extension.

Extensions. Consider two classes $\mathcal{C}$ and $\mathcal{D}$. We say that $\mathcal{D}$ is an extension of $\mathcal{C}$ when for every alphabet A, the two following conditions hold:

- If $\gamma : (A \cup \mathbb{E})^* \to A^*$ is the morphism defined by $\gamma(a) = a$ for $a \in A$ and $\gamma(b) = \varepsilon$ for $b \in \mathbb{E}$, then for every $K \in \mathcal{C}(A)$, we have $\gamma^{-1}(K) \in \mathcal{D}(A \cup \mathbb{E})$.
- For every $u \in \mathbb{E}^*$, if $\lambda_u : A^* \to (A \cup \mathbb{E})^*$ is the morphism defined by $\lambda_u(a) = au$ for $a \in A$, then for every $K \in \mathcal{D}(A \cup \mathbb{E})$, we have $\lambda_u^{-1}(K) \in \mathcal{C}(A)$.

Positive varieties give an important example of extension. Since they are closed under inverse image, it is immediate that for every positive variety $\mathcal{C}$, $\mathcal{C}$ is an extension of itself.

Taggings. A *tagging* is a pair $P = (\tau : \mathbb{E}^* \to T, G)$ where τ is a morphism into a finite monoid and $G \subseteq T$. We call $|G|$ the *rank* of P and $|T|$ its size. Moreover, given some NFA $\mathcal{A} = (A, Q, \delta, I, F)$, P is *compatible with* $\mathcal{A}$ when the rank $|G|$ is larger than $|\delta|$.

For our reduction, we shall require special taggings. Consider a class $\mathcal{D}$ and a tagging $P = (\tau : \mathbb{E}^* \to T, G)$. We say that P *fools* $\mathcal{D}$ when, for every alphabet A and every morphism $\alpha : (A \cup \mathbb{E})^* \to M$ into a finite monoid M, if all languages recognized by α belong to $Bool(\mathcal{D})(A \cup \mathbb{E})$, then, there exists $s \in M$, such that for every $t \in G$, we have $w_t \in \mathbb{E}^*$ which satisfies $\alpha(w_t) = s$ and $\tau(w_t) = t$.

Our reduction requires an efficient algorithm for computing taggings which fool the output class $\mathcal{D}$. Specifically, we say that a class $\mathcal{D}$ is *smooth* when, given as input $k \in \mathbb{N}$, one may compute in LogSpace (with respect to k) a tagging of rank at least k which fools $\mathcal{D}$.

Main theorem. We may now state our generic reduction theorem. The statement has two variants depending on whether the alphabet is fixed or not.

▶ **Theorem 2.** *Let $\mathcal{C}, \mathcal{D}$ be quotienting lattices such that $\mathcal{D}$ is smooth and extends $\mathcal{C}$. Then the two following properties hold:*

- *There is a* LogSpace *reduction from $\mathcal{C}$-separation for* NFAs *to $\mathcal{D}$-separation for monoids.*
- *For every fixed alphabet A, there is a* LogSpace *reduction from $\mathcal{C}(A)$-separation for* NFAs *to $\mathcal{D}(A \cup \mathbb{E})$-separation for monoids.*

We have two main applications of Theorem 2 which we present at the end of the section. Let us first describe the reduction. As we explained, we use a construction building a language recognized by a "small" monoid out of an input NFA and a compatible tagging.

Consider a NFA $\mathcal{A} = (A, Q, \delta, I, F)$ and let $P = (\tau : \mathbb{E}^* \to T, G)$ be a compatible tagging (i.e. $|\delta| \leq |G|$). We associate a new language $L[\mathcal{A}, P]$ over the alphabet $A \cup \mathbb{E}$ and show that one may efficiently compute a recognizing monoid whose size is polynomial with respect to $|Q|$ and the rank of P (i.e $|G|$). The construction involves two steps. We first define an intermediary language $K[\mathcal{A}, P]$ over the alphabet $A \times T$ and then define $L[\mathcal{A}, P]$ from it.

We define $K[\mathcal{A}, P] \subseteq (A \times T)^*$ as the language recognized by a new NFA $\mathcal{A}[P]$ which is built by relabeling the transitions of $\mathcal{A}$. Note that the definition of $\mathcal{A}[P]$ depends on arbitrary linear orders on G and δ. We let $\mathcal{A}[P] = (A \times T, Q, \delta[P], I, F)$ where $\delta[P]$ is obtained by relabeling the transitions of $\mathcal{A}$ as follows. Given $i \leq |\delta|$, if $(q_i, a_i, r_i) \in \delta$ is the i-th transition of $\mathcal{A}$, we replace it with the transition $(q_i, (a_i, t_i), r_i) \in \delta[P]$ where $t_i \in G$ is the i-th element of G (recall that $|\delta| \leq |G|$ by hypothesis).

▶ Remark. A key property of $\mathcal{A}[P]$ is that, by definition, all transitions are labeled by distinct letters in $A \times T$. This implies that $K[\mathcal{A}, P] = L(\mathcal{A}[P])$ is recognized by a monoid of size at most $|Q|^2 + 2$.

We may now define the language $L[\mathcal{A}, P] \subseteq (A \cup \mathbb{E})^*$. Observe that we have a natural map $\mu : (A\mathbb{E}^*)^* \to (A \times T)^*$. Indeed, consider $w \in (A\mathbb{E}^*)^*$. Since $A \cap \mathbb{E} = \emptyset$ (recall that this is a global assumption), it is immediate that w admits a *unique* decomposition $w = a_1 w_1 \cdots a_n w_n$ with $a_1, \ldots, a_n \in A$ and $w_1, \ldots, w_n \in \mathbb{E}^*$. Hence, we may define $\mu(w) = (a_1, P(w_1)) \cdots (a_n, P(w_n)) \in (A \times T)^*$. Finally, we define,

$$L[\mathcal{A}, P] = \mathbb{E}^* \cdot \mu^{-1}(K[\mathcal{A}, P]) \subseteq (A \cup \mathbb{E})^*$$

We may now state the two key properties of $L[\mathcal{A}, P]$ upon which Theorem 2 is based. It is recognized by a small monoid and the construction is connected to the separation.

▶ **Proposition 3.** *Given a* NFA $\mathcal{A} = (A, Q, \delta, I, F)$ *and a compatible tagging* P *of rank* n*, one may compute in* LogSpace *a monoid morphism* $\alpha : (A \cup \mathbb{E})^* \to M$ *recognizing* $L[\mathcal{A}, P]$ *and such that* $|M| \leq n + |A| \times n^2 \times (|Q|^2 + 2)$.

▶ **Proposition 4.** *Let* $\mathcal{C}, \mathcal{D}$ *be quotienting lattices such that* $\mathcal{D}$ *extends* $\mathcal{C}$*. Consider two* NFAs $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *over some alphabet* A *and let* P *be a compatible tagging that fools* $\mathcal{D}$*. Then,* $L(\mathcal{A}_1)$ *is* $\mathcal{C}(A)$*-separable from* $L(\mathcal{A}_2)$ *if and only if* $L[\mathcal{A}_1, P]$ *is* $\mathcal{D}(A \cup \mathbb{E})$*-separable from* $L[\mathcal{A}_2, P]$.

Let us explain why these two propositions imply Theorem 2. Let $\mathcal{C}, \mathcal{D}$ be quotienting lattices such that $\mathcal{D}$ is smooth and extends $\mathcal{C}$. We show that the second assertion in the theorem holds (the first one is proved similarly).

Consider two NFAs $\mathcal{A}_i = (A, Q_j, \delta_j, I_j, F_j)$ for $j = 1, 2$. We let $k = max(|\delta_1|, |\delta_2|)$. Since $\mathcal{D}$ is smooth, we may compute (in LogSpace) a tagging $P = (\tau : \mathbb{E}^* \to T, G)$ of rank $|G| \geq k$. Then, we may use Proposition 3 to compute (in LogSpace) monoid morphisms recognizing $L[\mathcal{A}_1, P]$ and $L[\mathcal{A}_2, P]$. Finally, by Proposition 4, $L(\mathcal{A}_1)$ is $\mathcal{C}(A)$-separable from $L(\mathcal{A}_2)$ if and only if $L[\mathcal{A}_1, P]$ is $\mathcal{D}(A \cup \mathbb{E})$-separable from $L[\mathcal{A}_2, P]$. Altogether, this construction is a LogSpace reduction to $\mathcal{D}$-separation for monoids which concludes the proof.

3.2 Applications

We now present the two main applications of Theorem 2. We start with the most simple one positive varieties. Indeed, we have the following lemma.

▶ **Lemma 5.** *Let $\mathcal{C}$ be a positive variety. Then, $\mathcal{C}$ is an extension of itself. Moreover, if $Bool(\mathcal{C}) \neq \mathrm{REG}$, then $\mathcal{C}$ is smooth.*

That a positive variety is an extension of itself is immediate (one uses closure under inverse image). The difficulty is to prove smoothness. We may now combine Theorem 2 with Lemma 5 to get the following corollary.

▶ **Corollary 6.** *Let $\mathcal{C}$ be a positive variety such that $Bool(\mathcal{C}) \neq \mathrm{REG}$. There exists a* LogSpace *reduction from $\mathcal{C}$-separation for* NFAs *to $\mathcal{C}$-separation for monoids.*

Corollary 6 implies that for any positive variety $\mathcal{C}$, the complexity of $\mathcal{C}$-separation is the same for monoids and NFAs. We illustrate this with an example: the *star-free languages*.

▶ **Example 7.** Consider the star-free languages (SF): for every alphabet A, $\mathrm{SF}(A)$ is the least set of languages containing all singletons $\{a\}$ for $a \in A$ and closed under Boolean operations and concatenation. It is folklore and simple to verify that SF is a variety. It is known that SF-membership is in NL for monoids (this is immediate from Schützenberger's theorem [19]). On the other hand, SF-membership is PSpace-complete for NFAs. In fact, it is shown in [2] that PSpace-completeness still holds for *deterministic* finite automata (DFAs).

For SF-separation, we may combine Corollary 6 with existing results to obtain that the problem is in EXPTime and PSpace-hard for both NFAs and monoids. Indeed, the EXPTime upper bounds is proved in [14] for monoids and we may lift it to NFAs with Corollary 6. Finally, the PSpace lower bound follows from [2]: SF-membership is PSpace-hard for DFAs. This yields that SF-separation is PSpace-hard for both DFAs and NFAs (by reduction from membership to separation which is easily achieved in LogSpace when starting from a DFA). Using Corollary 6 again, we get that SF-separation is PSpace-hard for monoids as well. ◀

We turn to our second application: finitely based concatenation hierarchies. Consider a finite quotienting Boolean algebra $\mathcal{C}$. We associate another finite quotienting Boolean algebra $\mathcal{C}_{\mathbb{E}}$ which we only define for alphabets of the form $A \cup \mathbb{E}$ (this is harmless: $\mathcal{C}_{\mathbb{E}}$ is used as the output class of our reduction). Let A be an alphabet and consider the morphism $\gamma : (A \cup \mathbb{E})^* \to A^*$ defined by $\gamma(a) = a$ for $a \in A$ and $\gamma(0) = \gamma(1) = \varepsilon$. We define,

$$\mathcal{C}_{\mathbb{E}}(A \cup \mathbb{E}) = \{\gamma^{-1}(L) \mid L \in \mathcal{C}(A)\}$$

It is straightforward to verify that $\mathcal{C}_{\mathbb{E}}$ remains a finite quotienting Boolean algebra. Moreover, we have the following lemma.

▶ **Lemma 8.** *Let $\mathcal{C}$ be a finite quotienting Boolean algebra. For every $n \in \frac{1}{2}\mathbb{N}$, $\mathcal{C}_{\mathbb{E}}[n]$ is smooth and an extension of $\mathcal{C}[n]$.*

In view of Theorem 2, we get the following corollary which provides a generic reduction for levels within finitely based hierarchies.

▶ **Corollary 9.** *Let $\mathcal{C}$ be a finite basis and $n \in \frac{1}{2}\mathbb{N}$. There exists a* LogSpace *reduction from $\mathcal{C}[n]$-separation for* NFAs *to $\mathcal{C}_{\mathbb{E}}[n]$-separation for monoids.*

4 Generic upper bounds for low levels in finitely based hierarchies

In this section, we present generic complexity results for the fixed alphabet separation problem associated to the lower levels in finitely based concatenation hierarchies. More precisely, we show that for every finite basis $\mathcal{C}$ and every alphabet A, $\mathcal{C}[1/2](A)$- and $\mathcal{C}[1](A)$-separation are respectively in NL and in P. These upper bounds hold for both monoids and NFAs: we prove them for monoids and lift the results to NFAs using the reduction of Corollary 9.

▶ Remark. We do **not** present new proofs for the decidability of $\mathcal{C}[1/2]$- and $\mathcal{C}[1]$-separation when $\mathcal{C}$ is a finite quotienting Boolean algebra. These are difficult results which are proved in [17]. Instead, we recall the (inefficient) procedures which were originally presented in [17] and carefully analyze and optimize them in order to get the above upper bounds.

For the sake of avoiding clutter, we fix an arbitrary finite quotienting Boolean algebra $\mathcal{C}$ and an alphabet A for the section.

4.1 Key sub-procedure

The algorithms $\mathcal{C}[1/2](A)$- and $\mathcal{C}[1](A)$-separation presented in [17] are based on a common sub-procedure. This remains true for the improved algorithms which we present in the paper. In fact, this sub-procedure is exactly what we improve to get the announced upper complexity bounds. We detail this point here. Note that the algorithms require considering special monoid morphisms (called "$\mathcal{C}$-compatible") as input. We first define this notion.

$\mathcal{C}$-compatible morphisms. Since $\mathcal{C}$ is finite, one associates a classical equivalence $\sim_{\mathcal{C}}$ defined on A^*. Given $u, v \in A^*$, we write $u \sim_{\mathcal{C}} v$ if and only if $u \in L \Leftrightarrow v \in L$ for all $L \in \mathcal{C}(A)$. Given $w \in A^*$, we write $[w]_{\mathcal{C}} \subseteq A^*$ for its $\sim_{\mathcal{C}}$-class. Since $\mathcal{C}$ is a finite quotienting Boolean algebra, $\sim_{\mathcal{C}}$ is a congruence of finite index for concatenation (see [18] for a proof). Hence, the quotient $A^*/\sim_{\mathcal{C}}$ is a monoid and the map $w \mapsto [w]_{\mathcal{C}}$ a morphism.

Consider a morphism $\alpha : A^* \to M$ into a finite monoid M. We say that α is $\mathcal{C}$-compatible when there exists a *monoid morphism* $s \mapsto [s]_{\mathcal{C}}$ from M to $A^*/\sim_{\mathcal{C}}$ such that for every $w \in A^*$, we have $[w]_{\mathcal{C}} = [\alpha(w)]_{\mathcal{C}}$. Intuitively, the definition means that α "computes" the $\sim_{\mathcal{C}}$-classes of words in A^*. The following lemma is used to compute $\mathcal{C}$-compatible morphisms (note that the LogSpace bound holds because $\mathcal{C}$ and A is fixed).

▶ **Lemma 10.** *Given two morphisms recognizing regular languages $L_1, L_2 \subseteq A^*$ as input, one may compute in* LogSpace *a $\mathcal{C}$-compatible morphism which recognizes both L_1 and L_2.*

In view of Lemma 10, we shall assume in this section without loss of generality that our input in separation for monoids is a single $\mathcal{C}$-compatible morphism recognizing the two languages that need to be separated.

Sub-procedure. Consider two $\mathcal{C}$-compatible morphisms $\alpha : A^* \to M$ and $\beta : A^* \to N$. We say that a subset of N is *good* (for β) when it contains $\beta(A^*)$ and is closed under multiplication. For every good subset S of N, we associate a subset of $M \times 2^N$. We then consider the problem of deciding whether specific elements belong to it (this is the sub-procedure used in the separation algorithms).

▶ Remark. The set $M \times 2^N$ is clearly a monoid for the componentwise multiplication. Hence we may multiply its elements and speak of idempotents in $M \times 2^N$.

An (α, β, S)-*tree* is an unranked ordered tree. Each node x must carry a label $lab(x) \in M \times 2^N$ and there are three possible kinds of nodes:

- **Leaves**: x has no children and $lab(x) = (\alpha(w), \{\beta(w)\})$ for some $w \in A^*$.
- **Binary**: x has exactly two children x_1 and x_2. Moreover, if $(s_1, T_1) = lab(x_1)$ and $(s_2, T_2) = lab(x_2)$, then $lab(x) = (s_1 s_2, T)$ with $T \subseteq T_1 T_2$.
- **S-Operation**: x has a unique child y. Moreover, the following must be satisfied:
  1. The label $lab(y)$ is an idempotent $(e, E) \in M \times 2^N$.
  2. $lab(x) = (e, T)$ with $T \subseteq E \cdot \{t \in S \mid [e]_{\mathcal{C}} = [t]_{\mathcal{C}} \in S\} \cdot E$.

We are interested in deciding whether elements in $M \times 2^N$ are the root label of some computation tree. Observe that computing all such elements is easily achieved with a least fixpoint procedure: one starts from the set of leaf labels and saturates this set with three operations corresponding to the two kinds of inner nodes. This is the approach used in [17] (actually, the set of all root labels is directly defined as a least fixpoint and (α, β, S)-trees are not considered). However, this is costly since the computed set may have exponential size with respect to $|N|$. Hence, this approach is not suitable for getting efficient algorithms. Fortunately, solving $\mathcal{C}[1/2](A)$- and $\mathcal{C}[1](A)$-separation does not require to have the whole set of possible root labels in hand. Instead, we shall only need to consider the elements $(s, T) \in M \times 2^N$ which are the root label of some tree **and** such that T is a **singleton set**. It turns out that these specific elements can be computed efficiently. We state this in the next theorem which is the key technical result and main contribution of this section.

▶ **Theorem 11.** *Consider two $\mathcal{C}$-compatible morphisms $\alpha : A^* \to M$ and $\beta : A^* \to N$ and a good subset $S \subseteq N$. Given $s \in M$ and $t \in N$, one may test in* NL *with respect to $|M|$ and $|N|$ whether there exists an (α, β, S)-tree with root label $(s, \{t\})$.*

Theorem 11 is proved in the full version of the paper. We only present a brief outline which highlights two propositions about (α, β, S)-trees upon which the theorem is based.

We first define a complexity measure for (α, β, S)-trees. Consider two $\mathcal{C}$-compatible morphisms $\alpha : A^* \to M$ and $\beta : A^* \to N$ as well as a good subset $S \subseteq N$. Given an (α, β, S)-tree $\mathbb{T}$, we define the *operational height of* $\mathbb{T}$ as the greatest number $h \in \mathbb{N}$ such that $\mathbb{T}$ contains a branch with h S-operation nodes.

Our first result is a weaker version of Theorem 11. It considers the special case when we restrict ourselves to (α, β, S)-trees whose operational heights are bounded by a constant.

▶ **Proposition 12.** *Let $h \in \mathbb{N}$ be a constant and consider two $\mathcal{C}$-compatible morphisms $\alpha : A^* \to M$ and $\beta : A^* \to N$ and a good subset $S \subseteq N$. Given $s \in M$ and $t \in N$, one may test in* NL *with respect to $|M|$ and $|N|$ whether there exists an (α, β, S)-tree of operational height at most h and with root label $(s, \{t\})$.*

Our second result complements the first one: in Theorem 11, it suffices to consider (α, β, S)-trees whose operational heights are bounded by a constant (depending only on the class $\mathcal{C}$ and the alphabet A which are fixed here). Let us first define this constant. Given a finite monoid M, we define the $\mathcal{J}$-depth of M as the greatest number $h \in \mathbb{N}$ such that one may find h pairwise distinct elements $s_1, \ldots, s_h \in M$ such that for every $i < h$, $s_{i+1} = x s_i y$ for some $x, y \in M$

▶ Remark. The term "$\mathcal{J}$-depth" comes from the Green's relations which are defined on any monoid [4]. We do not discuss this point here.

Recall that the quotient set $A^*/{\sim_{\mathcal{C}}}$ is a monoid. Consequently, it has a $\mathcal{J}$-depth. Our second result is as follows.

▶ **Proposition 13.** *Let $h \in \mathbb{N}$ be the $\mathcal{J}$-depth of $A^*/{\sim_{\mathcal{C}}}$. Consider two $\mathcal{C}$-compatible morphisms $\alpha : A^* \to M$ and $\beta : A^* \to N$, and a good subset $S \subseteq N$. Then, for every $(s, T) \in M \times 2^N$, the following properties are equivalent:*

1. *(s, T) is the root label of some (α, β, S)-tree.*
2. *(s, T) is the root label of some (α, β, S)-tree whose operational height is at most h.*

In view of Proposition 13, Theorem 11 is an immediate consequence of Proposition 12 applied in the special case when h is the $\mathcal{J}$-depth of $A^*/{\sim_{\mathcal{C}}}$ and $m = 1$.

4.2 Applications

We now combine Theorem 11 with the results of [17] to get the upper complexity bounds for $\mathcal{C}[1/2](A)$- and $\mathcal{C}[1](A)$-separation that we announced at the begging of the section.

Application to $\mathcal{C}[1/2]$. Let us first recall the connection between $\mathcal{C}[1/2]$-separation and (α, β, S)-trees. The result is taken from [17].

▶ **Theorem 14** ([17]). *Let $\alpha : A^* \to M$ be a $\mathcal{C}$-compatible morphism and $F_0, F_1 \subseteq M$. Moreover, let $S = \alpha(A^*) \subseteq M$. The two following properties are equivalent:*

- *$\alpha^{-1}(F_0)$ is $\mathcal{C}[1/2]$-separable from $\alpha^{-1}(F_1)$.*
- *for every $s_0 \in F_0$ and $s_1 \in F_1$, there exists no (α, α, S)-tree with root label $(s_0, \{s_1\})$.*

By Theorem 11 and the Immerman–Szelepcsényi theorem (which states that NL = *co*-NL), it is straightforward to verify that checking whether the second assertion in Theorem 14 holds can be done in NL with respect to $|M|$. Therefore, the theorem implies that $\mathcal{C}[1/2](A)$-separation for monoids is in NL. This is lifted to NFAs using Corollary 9.

▶ **Corollary 15.** *For every finite basis $\mathcal{C}$ and alphabet A, $\mathcal{C}[1/2](A)$-separation is in* NL *for both* NFAs *and monoids.*

Application to $\mathcal{C}[1]$. We start by recalling the $\mathcal{C}[1]$-separation algorithm which is again taken from [17]. In this case, we consider an auxiliary sub-procedure which relies on (α, β, S)-trees.

Consider a $\mathcal{C}$-compatible morphism $\alpha : A^* \to M$. Observe that M^2 is a monoid for the componentwise multiplication. We let $\beta : A^* \to M^2$ as the morphism defined by $\beta(w) = (\alpha(w), \alpha(w))$ for every $w \in A^*$. Clearly, β is $\mathcal{C}$-compatible: given $(s, t) \in M^2$, it suffices to define $[(s, t)]_{\mathcal{C}} = [s]_{\mathcal{C}}$. Using (α, β, S)-trees, we define a procedure $S \mapsto \mathit{Red}(\alpha, S)$ which takes as input a good subset $S \subseteq M^2$ (for β) and outputs a subset $\mathit{Red}(\alpha, S) \subseteq S$.

$$\mathit{Red}(\alpha, S) = \{(s, t) \in S \mid (s, \{(t, s)\}) \in M \times 2^{M^2} \text{ is the root label of an } (\alpha, \beta, S)\text{-tree}\} \subseteq S$$

It is straightforward to verify that $\mathit{Red}(\alpha, S)$ remains a good subset of M^2. We now have the following theorem which is taken from [17].

▶ **Theorem 16** ([17]). *Let $\alpha : A^* \to M$ be a morphism into a finite monoid and $F_0, F_1 \subseteq M$. Moreover, let $S \subseteq M^2$ be the greatest subset of $\alpha(A^*) \times \alpha(A^*)$ such that $\mathit{Red}(\alpha, S) = S$. Then, the two following properties are equivalent:*

- *$\alpha^{-1}(F_0)$ is $\mathit{Bool}(\mathit{Pol}(\mathcal{C}))$-separable from $\alpha^{-1}(F_1)$.*
- *for every $s_0 \in F_0$ and $s_1 \in F_1$, $(s_0, s_1) \notin S$.*

Observe that Theorem 11 implies that given an arbitrary good subset S of $\alpha(A^*) \times \alpha(A^*)$, one may compute $\mathit{Red}(\alpha, S) \subseteq S$ in P with respect to $|M|$. Therefore, the greatest subset S of $\alpha(A^*) \times \alpha(A^*)$ such that $\mathit{Red}(\alpha, S) = S$ can be computed in P using a greatest fixpoint algorithm. Consequently, Theorem 16 yields that $\mathcal{C}[1](A)$-separation for monoids is in P. Again, this is lifted to NFAs using Corollary 9.

▶ **Corollary 17.** *For every finite basis $\mathcal{C}$ and alphabet A, $\mathcal{C}[1](A)$-separation is in* P *for both* NFAs *and monoids.*

5 The Straubing-Thérien hierarchy

In this final section, we consider one of the most famous concatenation hierarchies: the Straubing-Thérien hierarchy [21, 22]. We investigate the complexity of separation for the levels 3/2 and 2.

▶ Remark. Here, the alphabet is part of the input. For fixed alphabets, these levels can be handled with the generic results presented in the previous section (see Theorem 18 below).

The basis of the Straubing-Thérien hierarchy is the trivial variety ST[0] defined by $\mathrm{ST}[0](A) = \{\emptyset, A^*\}$ for every alphabet A. It is known and simple to verify (using induction) that all half levels are positive varieties and all full levels are varieties.

The complexity of separation for the level one (ST[1]) has already been given a lot of attention. Indeed, this level corresponds to a famous class which was introduced independently from concatenation hierarchies: the piecewise testable languages [20]. It was shown independently in [3] and [11] that ST[1]-separation is in P for NFAs (and therefore for DFAs and monoids as well). Moreover, it was also shown in [5] that the problem is actually P-complete for NFAs and DFAs[1]. Additionally, it is shown in [3] that ST[1/2]-separation is in NL.

In the paper, we are mainly interested in the levels ST[3/2] and ST[2]. Indeed, the Straubing-Thérien hierarchy has a unique property: the generic separation results of [17] apply to these two levels as well. Indeed, these are also the levels 1/2 and 1 in another finitely based hierarchy. Consider the class AT of *alphabet testable languages*. For every alphabet A, $\mathrm{AT}(A)$ is the set of all Boolean combinations of languages A^*aA^* for $a \in A$. One may verify that AT is a variety and that $\mathrm{AT}(A)$ is finite for every alphabet A. Moreover, we have the following theorem which is due to Pin and Straubing [8] (see [18] for a modern proof).

▶ **Theorem 18** ([8]). *For every $n \in \frac{1}{2}\mathbb{N}$, we have $\mathrm{AT}[n] = \mathrm{ST}[n+1]$.*

The theorem implies that $\mathrm{ST}[3/2] = \mathrm{AT}[1/2]$ and $\mathrm{ST}[2] = \mathrm{AT}[1]$. Therefore, the results of [17] yield the decidability of separation for both ST[3/2] and ST[2] (the latter is the main result of [17]). As expected, this section investigates complexity for these two problems.

5.1 The level 3/2

We have the following tight complexity bound for ST[3/2]-separation.

▶ **Theorem 19.** ST[3/2]*-separation is* PSpace*-complete for both* NFAs *and monoids.*

The PSpace upper bound is proved by building on the techniques introduced in the previous section for handling the level 1/2 of an arbitrary finitely based hierarchies. Indeed, we have $\mathrm{ST}[3/2] = \mathrm{AT}[1/2]$ by Theorem 18. However, let us point out that obtaining this upper bound requires some additional work: the results of Section 4 apply to the setting in which the alphabet is fixed, this is not the case here. In particular, this is why we end up with a PSpace upper bound instead of the generic NL upper presented in Corollary 15. The detailed proof is postponed to the full version of the paper.

In this abstract, we focus on proving that ST[3/2]-separation is PSpace-hard. The proof is presented for NFAs: the result can then be lifted to monoids with Corollary 6 since ST[3/2] is a positive variety. We use a LogSpace reduction from the quantified Boolean formula problem (QBF) which is among the most famous PSpace-complete problems.

We first describe the reduction. For every quantified Boolean formula Ψ, we explain how to construct two languages L_Ψ and L'_Ψ. It will be immediate from the presentation that given Ψ as input, one may compute NFAs for L_Ψ and L'_Ψ in LogSpace. Then, we show that

[1] Since ST[1] is a variety, P-completeness for ST[1]-separation can also be lifted to monoids using Corollary 6.

this construction is the desired reduction: Ψ is true if and only if L_Ψ is not ST[3/2]-separable from L'_Ψ.

Consider a quantified Boolean formula Ψ and let n be the number of variables it involves. We assume without loss of generality that Ψ is in prenex normal form and that the quantifier-free part of Ψ is in conjunctive normal form (QBF remains PSpace-complete when restricted to such formulas). That is,

$$\Psi = Q_n\ x_n \cdots Q_1\ x_1\ \varphi$$

where $x_1 \dots x_n$ are the variables of Ψ, $Q_1, \dots, Q_n \in \{\exists, \forall\}$ are quantifiers and φ is a quantifier-free Boolean formula involving the variables $x_1 \dots x_n$ which is in conjunctive normal form.

We describe the two regular languages L_Ψ, L'_Ψ by providing regular expressions recognizing them. Let us first specify the alphabet over which these languages are defined. For each variable x_i occurring in Ψ, we create two letters that we write x_i and $\overline{x_i}$. Moreover, we let,

$$X = \{x_1, \dots, x_n\} \quad \text{and} \quad \overline{X} = \{\overline{x_1}, \dots, \overline{x_n}\}$$

Additionally, our alphabet also contains the following letters: $\#_1, \dots, \#_i, \$$. For $0 \leq i \leq n$, we define an alphabet B_i. We have:

$$B_0 = X \cup \overline{X} \quad \text{and} \quad B_i = X \cup \overline{X} \cup \{\#_1, \dots, \#_i, \$\}$$

Our languages are defined over the alphabet B_n: $L_\Psi, L'_\Psi \subseteq B_n^*$. They are built by induction: for $0 \leq i \leq n$ we describe two languages $L_i, L'_i \subseteq B_i^*$ (starting with the case $i = 0$). The languages L_Ψ, L'_Ψ are then defined as L_n, L'_n.

Construction of L_0, L'_0. The language L_0 is defined as $L_0 = (B_0)^*$. The language L'_0 is defined from the quantifier-free Boolean formula φ. Recall that by hypothesis φ is in conjunctive normal form: $\varphi = \bigwedge_{j \leq k} \varphi_j$ were φ_i is a disjunction of literals. For all $j \leq k$, we let $C_j \subseteq B_0 = X \cup \overline{X}$ as the following alphabet:

- Given $x \in X$, we have $x \in C_j$, if and only x is a literal in the disjunction φ_j.
- Given $\overline{x} \in \overline{X}$, we have $\overline{x} \in C_j$, if and only $\neg x$ is a literal in the disjunction φ_j.

Finally, we define $L'_0 = C_1 C_2 \cdots C_k$.

Construction of L_i, L'_i for $i \geq 1$. We assume that L_{i-1}, L'_{i-1} are defined and describe L_i and L'_i. We shall use the two following languages in the construction:

$$T_i = (\#_i x_i (B_{i-1} \setminus \{\overline{x_i}\})^* \$ x_i)^* \quad \text{and} \quad \overline{T_i} = (\#_i \overline{x_i} (B_{i-1} \setminus \{x_i\})^* \$ \overline{x_i})^*$$

The definition of L_i, L'_i from L_{i-1}, L'_{i-1} now depends on whether the quantifier Q_i is existential or universal.

- If Q_i is an existential quantifier (i.e. $Q_i = \exists$):

$$\begin{array}{rcl} L_i & = & (\#_i(x_i + \overline{x_i}) L_{i-1} \$ (x_i + \overline{x_i}))^* \#_i \\ L'_i & = & (\#_i(x_i + \overline{x_i}) L'_{i-1} \$ (x_i + \overline{x_i}))^* \#_i \$ \left(T_i \#_i + \overline{T_i} \#_i\right) \end{array}$$

- If the Q_i is an universal quantifier (i.e. $Q_i = \forall$):

$$\begin{array}{rcl} L_i & = & (\#_i(x_i + \overline{x_i}) L_{i-1} \$ (x_i + \overline{x_i}))^* \#_i \\ L'_i & = & \overline{T_i} \#_i \$ (\#_i(x_i + \overline{x_i}) L'_{i-1} \$ (x_i + \overline{x_i}))^* \#_i \$ T_i \#_i \end{array}$$

Finally, L_Ψ, L'_Ψ are defined as the languages $L_n, L'_n \subseteq (B_n)^*$. It is straightforward to verify from the definition, than given Ψ as input, one may compute NFAs for L_Ψ and L'_Ψ in LogSpace. Consequently, it remains to prove that this construction is the desired reduction. We do so in the following proposition.

▶ **Proposition 20.** *For every quantified Boolean formula Ψ, Ψ is true if and only if L_Ψ is not* ST[3/2]*-separable from L'_Ψ.*

Proposition 20 is proved by considering a stronger result which states properties of all the languages L_i, L'_i used in the construction of L_Ψ, L'_Ψ (the argument is an induction on i). While we postpone the detailed proof to the full version of the paper, let us provide a sketch which presents this stronger result.

Proof of Proposition 20 (sketch). Consider a quantified Boolean formula Ψ. Moreover, let $B_0, \dots, B_n$ and $L_i, L'_i \subseteq (B_i)^*$ as the alphabets and languages defined above. The key idea is to prove a property which makes sense for all languages L_i, L'_i. In the special case when $i = n$, this property implies Proposition 20.

Consider $0 \leq i \leq n$. We write Ψ_i for the sub-formula $\Psi_i := Q_i\ x_i \cdots Q_1\ x_1\ \varphi$ (with the free variables $x_{i+1}, \dots, x_n$). In particular, $\Psi_0 := \varphi$ and $\Psi_n := \Psi$. Moreover, we call "*i-valuation*" a sub-alphabet $V \subseteq B_i$ such that,

1. $\#_1, \dots, \#_i, \$ \in V$ and $x_1, \overline{x_1}, \dots, x_i, \overline{x_i} \in V$, and,
2. for every j such that $i < j \leq n$, one of the two following property holds:
   - $x_j \in V$ and $\overline{x_j} \notin V$, or,
   - $x_j \notin V$ and $\overline{x_j} \in V$.

Clearly, an i-valuation corresponds to a truth assignment for all variables x_j such that $j > i$ (i.e. those that are free in Ψ_i): when the first (resp. second) assertion in Item 2 holds, x_j is assigned to $\top$ (resp. $\bot$). Hence, abusing terminology, we shall say that an i-valuation V *satisfies* Ψ_i if Ψ_i is true when replacing its free variables by the truth values provided by V.

Finally, for $0 \leq i \leq n$, if $V \subseteq B_i$ is an i-valuation, we let $[V] \subseteq V^*$ as the following language. Given $w \in V^*$, we have $w \in [V]$ if and only if for every $j > i$ either $x_j \in \mathsf{alph}(w)$ or $\overline{x_j} \in \mathsf{alph}(w)$ (by definition of i-valuations, exactly one of these two properties must hold). Proposition 20 is now a consequence of the following lemma.

▶ **Lemma 21.** *Consider $0 \leq i \leq n$. Then given an i-valuation V, the two following properties are equivalent:*

1. *Ψ_i is satisfied by V.*
2. *$L_i \cap [V]$ is not* ST[3/2]*-separable from $L'_i \cap [V]$.*

Lemma 21 is proved by induction on i using standard properties of the polynomial closure operation (see [18] for example). The proof is postponed to the full version of the paper. Let us explain why the lemma implies Proposition 20.

Consider the special case of Lemma 21 when $i = n$. Observe that $V = B_n$ is an n-valuation (the second assertion in the definition of n-valuations is trivially true since there are no j such that $n < j \leq n$). Hence, since $\Psi = \Psi_n$ and $L_\Psi, L'_\Psi = L_n, L'_n$, the lemma yields that,

1. Ψ is satisfied by V (i.e. Ψ is true).
2. $L_\Psi \cap [V]$ is not ST[3/2]-separable from $L'_\Psi \cap [V]$.

Moreover, we have $[V] = (B_n)^*$ by definition. Hence, we obtain that Ψ is true if and only if L is not ST[3/2]-separable from L' which concludes the proof of Proposition 20. ◀

5.2 The level two

For the level two, there is a gap between the lower and upper bound that we are able to prove. Specifically, we have the following theorem.

▶ **Theorem 22.** ST[2]*-separation is in* EXPTime *and* PSpace*-hard for both* NFAs *and monoids.*

Similarly to what happened with ST[3/2], the EXPTime upper bound is obtained by building on the techniques used in the previous section. Proving PSpace-hardness is achieved using a reduction from ST[3/2]-separation (which is PSpace-hard by Theorem 19). The reduction is much simpler than what we presented for ST[3/2] above. It is summarized by the following proposition.

▶ **Proposition 23.** *Consider an alphabet A and $H, H' \subseteq A^*$. Let $B = A \cup \{\#, \$\}$ with $\#, \$ \notin A$, $L = \#(H'\#(A^*\$\#)^*)^*H\#(A^*\$\#)^* \subseteq B^*$ and $L' = \#(H'\#(A^*\$\#)^*)^* \subseteq B^*$. The two following properties are equivalent:*

1. *H is* ST[3/2]*-separable from H'.*
2. *L is* ST[2]*-separable from L'.*

Proposition 23 is proved using standard properties of the polynomial and Boolean closure operations. The argument is postponed ot the full version of the paper. It is clear than given as input NFAs for two languages H, H', one may compute NFAs for the languages L, L' defined Proposition 23 in LogSpace. Consequently, the proposition yields the desired LogSpace reduction from ST[3/2]-separation for NFAs to ST[2]-separation for NFAs. This proves that ST[2]-separation is PSpace-hard for NFAs (the result can then be lifted to monoids using Corollary 6) since ST[2] is a variety).

6 Conclusion

We showed several results, all of them raising new questions. First we proved that for many important classes of languages (including all positive varieties), the complexity of separation does not depend on how the input languages are represented. A natural question is whether the technique can be adapted to encompass more classes. In particular, one may define more permissive notions of positive varieties by replacing closure under inverse image by weaker notions. For example, many natural classes are *length increasing positive varieties*: closure under inverse image only has to hold for length increasing morphisms (*i.e.*, morphisms $\alpha : A^* \to B^*$ such that $|\alpha(w)| \geq |w|$ for every $w \in A^*$). For example, the levels of another famous concatenation hiearchy, the dot-depth [1] (whose basis is $\{\emptyset, \{\varepsilon\}, A^+, A^*\}$) are length increasing positive varieties. Can our techniques be adapted for such classes? Let us point out that there exists no example of natural class $\mathcal{C}$ for which separation is decidable and strictly harder for NFAs than for monoids. However, there are classes $\mathcal{C}$ for which the question is open (see for example the class of locally testable languages in [10]).

We also investigated the complexity of separation for levels 1/2 and 1 in finitely based concatenation hierarchies. We showed that when the alphabet is fixed, the problems are respectively in NL and P for any such hierarchy. An interesting follow-up question would be to push these results to level 3/2, for which separation is also known to be decidable in any finitely based concatenation hierarchy [9]. A rough analysis of the techniques used in [9] suggests that this requires moving above P.

Finally, we showed that in the famous Straubing-Thérien hierarchy, ST[3/2]-separation is PSpace-complete and ST[2]-separation is in EXPTime and PSpace-hard. Again, a natural question is to analyze ST[5/2]-separation whose decidability is established in [9].

References

1 Janusz A. Brzozowski and Rina S. Cohen. Dot-depth of star-free events. *Journal of Computer and System Sciences*, 5(1):1–16, 1971.

634 **2** Sang Cho and Dung T. Huynh. Finite automaton aperiodicity is PSPACE-complete. *Theoretical Computer Science*, 88(1):99 – 116, 1991.

636 **3** Wojciech Czerwiński, Wim Martens, and Tomáš Masopust. Efficient separability of regular languages by subsequences and suffixes. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP'13)*, pages 150–161. Springer-Verlag, 2013.

640 **4** James Alexander Green. On the structure of semigroups. *Annals of Mathematics*, 54(1):163–172, 1951.

642 **5** Tomás Masopust. Separability by piecewise testable languages is PTIME-complete. *Theoretical Computer Science*, 711:109–114, 2018.

644 **6** Jean-Éric Pin. The dot-depth hierarchy, 45 years later. In *The Role of Theory in Computer Science - Essays Dedicated to Janusz Brzozowski*, pages 177–202, 2017.

646 **7** Jean-Éric Pin. Mathematical foundations of automata theory. In preparation, 2018. URL: `https://www.irif.fr/~jep/PDF/MPRI/MPRI.pdf`.

648 **8** Jean-Eric Pin and Howard Straubing. Monoids of upper triangular Boolean matrices. In *Semigroups. Structure and Universal Algebraic Problems*, volume 39 of *Colloquia Mathematica Societatis Janos Bolyal*, pages 259–272. North-Holland, 1985.

651 **9** Thomas Place. Separating regular languages with two quantifier alternations. Unpublished, a preliminary version can be found at `https://arxiv.org/abs/1707.03295`, 2018.

653 **10** Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. Separating regular languages by locally testable and locally threshold testable languages. In *Proceedings of the 33rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, FSTTCS'13, pages 363–375, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

658 **11** Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. Separating regular languages by piecewise testable and unambiguous languages. In *Proceedings of the 38th International Symposium on Mathematical Foundations of Computer Science*, MFCS'13, pages 729–740. Springer-Verlag, 2013.

662 **12** Thomas Place and Marc Zeitoun. Separating regular languages with first-order logic. In *Proceedings of the Joint Meeting of the 23rd EACSL Annual Conference on Computer Science Logic (CSL'14) and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'14)*, pages 75:1–75:10. ACM, 2014.

666 **13** Thomas Place and Marc Zeitoun. The tale of the quantifier alternation hierarchy of first-order logic over words. *SIGLOG News*, 2(3):4–17, 2015.

668 **14** Thomas Place and Marc Zeitoun. Separating regular languages with first-order logic. *Logical Methods in Computer Science*, 12(1), 2016.

670 **15** Thomas Place and Marc Zeitoun. Adding successor: A transfer theorem for separation and covering. Unpublished, a preliminary version can be found at `http://arxiv.org/abs/1709.10052`, 2017.

673 **16** Thomas Place and Marc Zeitoun. Going higher in the first-order quantifier alternation hierarchy on words. Unpublished, a preliminary version can be found at `https://arxiv.org/abs/1404.6832`, 2017.

676 **17** Thomas Place and Marc Zeitoun. Separation for dot-depth two. In *Proceedings of the 32th Annual ACM/IEEE Symposium on Logic in Computer Science, (LICS'17)*, pages 202–213. IEEE Computer Society, 2017.

679 **18** Thomas Place and Marc Zeitoun. Generic results for concatenation hierarchies. *Theory of Computing Systems (ToCS)*, 2018. Selected papers from CSR'17.

681 **19** Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8:190–194, 1965.

683 **20** Imre Simon. Piecewise testable events. In *2nd GI Conference on Automata Theory and*
684 *Formal Languages*, pages 214–222, 1975.
685 **21** Howard Straubing. A generalization of the schützenberger product of finite monoids. *The-*
686 *oretical Computer Science*, 13(2):137–150, 1981.
687 **22** Denis Thérien. Classification of finite monoids: The language approach. *Theoretical Com-*
688 *puter Science*, 14(2):195–208, 1981.

A Appendix to Section 3

In this appendix, we present the missing proofs for the statements of Section 3.

A.1 Proof of Proposition 3

We start with Proposition 3 which is used to build morphisms recognizing the languages we associate to NFAs and tagging pairs. Let us recall the statement.

▶ **Proposition 3.** *Given a* NFA $\mathcal{A} = (A, Q, \delta, I, F)$ *and a compatible tagging* P *of size* n*, one may compute in* LogSpace *a monoid morphism* $\alpha : (A \cup \mathbb{E})^* \to M$ *recognizing* $L[\mathcal{A}, P]$ *and such that* $|M| \leq n + |A| \times n^2 \times (|Q|^2 + 2)$*.*

Let $P = (\tau : \mathbb{E}^* \to T, G)$ $(n = |T|)$. We construct the morphism $\alpha : (A \cup \mathbb{E})^* \to M$ recognizing $L[\mathcal{A}, P] \subseteq (A \cup \mathbb{E})^*$. That it has size $|M| \leq n + |A| \times n^2 \times (|Q|^2 + 2)$ and can be computed in LogSpace is immediate from the construction.

Recall that $L[\mathcal{A}, P]$ is defined from an intermediary language $K[\mathcal{A}, P] \subseteq (A \times T)^*$ which is recognized by the NFA $\mathcal{A}[P]$. We first prove the following preliminary result about $K[\mathcal{A}, P]$ which uses the fact that, by construction, all transitions in $\mathcal{A}[P]$ are labeled by distinct letters in $A \times T$.

▶ **Lemma 24.** *The language* $K[\mathcal{A}, P]$ *is recognized by a morphism* $\beta : (A \times T)^* \to N$ *such that monoid* N *has size* $|N| \leq |Q|^2 + 2$*.*

Proof. Recall that $\mathcal{A}[P] = (A \times T, Q, \delta[P], I, F)$ where $\delta[P]$ is obtained by relabeling the transition of $\mathcal{A}$. We let $N = Q^2 \cup \{0_N, 1_N\}$ and equip N with the following multiplication. The elements 0_N and 1_N are respectively a zero and a neutral element. For $(q_1, r_1), (q_2, r_2) \in Q^2$, we define,

$$(q_1, r_1) \cdot (q_2, r_2) = \begin{cases} (q_1, r_2) & \text{if } r_1 = q_2 \\ 0_N & \text{otherwise} \end{cases}$$

We now define a morphism $\beta : (A \times T)^* \to N$. Given $(a, t) \in A \times T$, we know by definition that there exists at most one transition in $\delta[P]$ whose label is (a, t). Therefore, either there is no such transition and we let $\beta((a, t)) = 0_N$ or there exists exactly one pair $(q, r) \in Q^2$ such that $(q, (a, t), r) \in \delta[P]$ and we define $\beta((a, t)) = (q, r)$. One may now verify that β recognizes $L(\mathcal{A}[P]) = K[\mathcal{A}, P]$. ◀

Let us briefly recall how $L[\mathcal{A}, P] \subseteq (A \cup \mathbb{E})^*$ is defined from $K[\mathcal{A}, P]$. We have a map $\mu : (A\mathbb{E}^*)^* \to (A \times T)^*$ defined as follows. Consider $w \in (A\mathbb{E}^*)^*$. Since $A \cap \mathbb{E} = \emptyset$, w admits a unique decomposition $w = a_1 w_1 \cdots a_n w_n$ with $a_1, \ldots, a_n \in A$ and $w_1, \ldots, w_n \in \mathbb{E}^*$. We define, $\mu(w) = (a_1, \tau(w_1)) \cdots (a_n, \tau(w_n))$. Finally, recall that,

$$L[\mathcal{A}, P] = \mathbb{E}^* \cdot \mu^{-1}(K[\mathcal{A}, P]) \subseteq \mathbb{E}^*(A\mathbb{E}^*)^* = (A \cup \mathbb{E})^*$$

We may now define the morphism $\alpha : (A \cup \mathbb{E})^* \to M$. We let $\beta : (A \times T)^* \to N$ as the morphism given by Lemma 24. Consider the following set M:

$$M = T \cup (T \times N \times A \times T)$$

Note that since $|N| \leq |Q|^2 + 2$, we do have $|M| \leq n + |A| \times n^2 \times (|Q|^2 + 2)$ as desired. We equip M with the following multiplication. Since M is defined as a union there are two kinds of elements which means that we have to consider four cases:

- If $t, t' \in T$, then their multiplication as element of M is the one in T, i.e. tt'.
- If $t \in T$ and $(t_1, s, a, t_2) \in T \times N \times A \times T$, we let,

$$\begin{array}{rcl} t \cdot (t_1, s, a, t_2) & = & (tt_1, s, a, t_2) \\ (r, t_1, s, a, t_2) \cdot t & = & (t_1, s, a, t_2 t) \end{array}$$

- If $(t_1, s, a, t_2), (t'_1, s', a', t'_2) \in T \times N \times A \times T$, we let,

$$(t_1, s, a, t_2) \cdot (t'_1, s', a', t'_2) = (t_1, s\beta((a, t_2 t'_1))s', a', t'_2)$$

One may verify that this multiplication is associative and that $1_T \in T$ is a neutral element for M. Finally, we define a morphism $\alpha : (A \cup \mathbb{E})^* \to M$ as follows. For $a \in A$, we let $\alpha(a) = (1_T, 1_N, a, 1_T) \in T \times N \times A \times T$ and for $b \in \mathbb{E}$, we let $\alpha(b) = \tau(b) \in T$. The following fact can be verified from the definition of α.

▶ **Fact 25.** *Consider a word $u \in (A \cup \mathbb{E})^*$. Then, one of the two following properties holds:*

1. *$u \in \mathbb{E}^*$ and $\alpha(u) = \tau(u) \in T$.*
2. *$u = u_0 u_1 a u_2$ with $u_0 \in \mathbb{E}^*$, $u_1 \in (A\mathbb{E}^*)^*$, $a \in A$ and $u_2 \in \mathbb{E}^*$ and we have,*

$$\alpha(u) = (\tau(u_0), \beta(\mu(u_1)), a, \tau(u_2))$$

It remains to verify that α recognizes $L[\mathcal{A}, P]$. Since $K[\mathcal{A}, P]$ is recognized by β, we have $H \subseteq N$ such that $K[\mathcal{A}, P] = \beta^{-1}(H)$. We define $H' \subseteq M$ as the following set:

$$H' = \begin{cases} \{(t_1, s, a, t_2) \in T \times N \times A \times T \mid s\beta((a, t_2)) \in H\} & \text{if } 1_N \notin H \\ \{(t_1, s, a, t_2) \in T \times N \times A \times T \mid s\beta((a, t_2)) \in H\} \cup T & \text{if } 1_N \in H \end{cases}$$

Since $L[\mathcal{A}, P] = \mathbb{E}^* \cdot \mu^{-1}(K[\mathcal{A}, P])$ by definition, it can be verified from Fact 25 that $L[\mathcal{A}, P] = \alpha^{-1}(H')$ which concludes the proof.

A.2 Proof of Proposition 4

We first recall Proposition 4.

▶ **Proposition 4.** *Let $\mathcal{C}, \mathcal{D}$ be quotienting lattices such that $\mathcal{D}$ extends $\mathcal{C}$. Consider two* NFAs *$\mathcal{A}_1$ and $\mathcal{A}_2$ over some alphabet A and let P be a compatible tagging that fools $\mathcal{D}$. Then, $L(\mathcal{A}_1)$ is $\mathcal{C}(A)$-separable from $L(\mathcal{A}_2)$ if and only if $L[\mathcal{A}_1, P]$ is $\mathcal{D}(A \cup \mathbb{E})$-separable from $L[\mathcal{A}_2, P]$.*

We fix $\mathcal{A}_1 = (A, Q_1, \delta_1, I_1, F_1)$ and $\mathcal{A}_2 = (A, Q_2, \delta_2, I_2, F_2)$ for the proof. Moreover, we let $P = (\tau : \mathbb{E}^* \to T, G)$ as the tagging pair which fools $\mathcal{D}$.

There are two directions to prove. First, we assume that $L(\mathcal{A}_1)$ is $\mathcal{C}$-separable from $L(\mathcal{A}_2)$. We prove that $L[\mathcal{A}_1, P]$ is $\mathcal{D}$-separable from $L[\mathcal{A}_2, P]$. Note that this direction is independent from the hypothesis that P fools $\mathcal{D}$. Let $K \in \mathcal{C}(A)$ be a separator for $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$: $L(\mathcal{A}_1) \subseteq K$ and $L(\mathcal{A}_2) \cap K = \emptyset$. Consider the morphism $\gamma : (A \cup \mathbb{E})^* \to A^*$ defined by $\gamma(a) = a$ for $a \in A$ and $\gamma(b) = \varepsilon$ for $b \in \mathbb{E}$. Since $\mathcal{D}$ is an extension of $\mathcal{C}$, we have $\gamma^{-1}(K) \in \mathcal{D}(A \cup \mathbb{E})$ by definition. Moreover, it is straightforward to verify from the definitions of γ, $L[\mathcal{A}_1, P]$ and $L[\mathcal{A}_2, P]$ that $\gamma^{-1}(K)$ separates $L[\mathcal{A}_1, P]$ from $L[\mathcal{A}_2, P]$ which concludes this direction of the proof.

Assume now that $L[\mathcal{A}_1, P]$ is $\mathcal{D}$-separable from $L[\mathcal{A}_2, P]$. We show that $L(\mathcal{A}_1)$ is $\mathcal{C}$-separable from $L(\mathcal{A}_2)$. Let $K \in \mathcal{D}(A \cup \mathbb{E})$ which separates $L[\mathcal{A}_1, P]$ from $L[\mathcal{A}_2, P]$. Clearly, $K \in \mathit{Bool}(\mathcal{D})(A \cup \mathbb{E})$. Moreover, since $\mathcal{D}$ is a quotienting lattice, one may verify that $\mathit{Bool}(\mathcal{D})$ is a quotienting Boolean algebra (quotients commute with Boolean operations). Therefore,

it follows from standard results about quotienting Boolean algebras that there exists a morphism $\alpha : (A \cup \mathbb{E})^* \to M$ into a finite monoid M which recognizes K and such that every language recognized by α belongs to $Bool(\mathcal{D})$ (it suffices to choose α as the "syntactic morphism" of K, see [7] for details). By definition of α and since P fools $\mathcal{D}$, we get the following fact.

▶ **Fact 26.** *There exists $s \in M$ such that for every $t \in G$, we have $w_t \in \mathbb{E}^*$ satisfying $\alpha(w_t) = s$ and $\tau(w_t) = t$.*

Let $u = w_t \in \mathbb{E}^*$ for some arbitrary $t \in G$ and consider the morphism $\lambda_u : A^* \to (A \cup \mathbb{E})^*$ defined by $\gamma(a) = au \in (A \cup \mathbb{E})^*$ for every $a \in A$. Finally, we let $K' = \lambda_u^{-1}(K)$. Since $K \in \mathcal{D}(A \cup \mathbb{E})$ and $\mathcal{D}$ is an extension of $\mathcal{C}$, it is immediate that $K' \in \mathcal{C}(A)$. We now show that K' separates $L(\mathcal{A}_1)$ from $L(\mathcal{A}_2)$ which concludes the argument.

We concentrate on proving that $L(\mathcal{A}_1) \subseteq K'$. That $L(\mathcal{A}_2) \cap K' = \emptyset$ is showed symmetrically and left to the reader. Consider some word $v = a_1 \cdots a_n \in L(\mathcal{A}_1)$. We show that $v \in K'$. By definition of $L[\mathcal{A}_1, P]$, it is straightforward to verify that there exists $t_1, \ldots, t_n \in G$ (each depending on the whole word v) such that $a_1 w_{t_1} \cdots a_n w_{t_n} \in L[\mathcal{A}_1, P]$. Moreover, by definition in Fact 26, we know that $\alpha(w_t) = \alpha(u) = s$ for every $t \in G$. Consequently, we get,

$$\alpha(a_1 w_{t_1} \cdots a_n w_{t_n}) = \alpha(a_1 u \cdots a_n u) = \alpha(\lambda_u(v))$$

Since α recognizes $L[\mathcal{A}_1, P]$ which contains $a_1 w_{t_1} \cdots a_n w_{t_n}$, it follows that $\lambda_u(v) \in L[\mathcal{A}_1, P]$ as well. Hence, since $L[\mathcal{A}_1, P] \subseteq K$, we obtain that $\lambda_u(v) \in K$. Finally, this yields $v \in \lambda_u^{-1}(K) = K'$, finishing the proof.

A.3 Proof of Lemma 5

We first recall the statement of Lemma 5.

▶ **Lemma 5.** *Let $\mathcal{C}$ be a positive variety. Then, $\mathcal{C}$ is an extension of itself. Moreover, if $Bool(\mathcal{C}) \neq \mathrm{REG}$, then $\mathcal{C}$ is smooth.*

We fix the positive variety $\mathcal{C}$ for the proof. Clearly, $\mathcal{C}$ is an extension of itself since positive varieties are closed under inverse image by definition. We now assume that $Bool(\mathcal{C}) \neq \mathrm{REG}$ and show that $\mathcal{C}$ is smooth: given as input $k \in \mathbb{N}$, one may compute in LogSpace (with respect to k) a tagging of rank at least k and which fools $\mathcal{C}$. We describe how to construct a tagging of rank k and size polynomial in k, that it can be computed in LogSpace is straightforward to verify and left to the reader. Furthermore, we consider the special case when $k = 2^h$ for some $h \geq 1$ (when k is not of this form, it suffices to consider the least h such that $k \leq 2^h$). The construction is based on the following preliminary lemma.

▶ **Lemma 27.** *There exist constants $\ell, m \in \mathbb{N}$ such that for every $h \geq 1$, there exists a morphism $\gamma : B^* \to T$ and $F \subseteq T$ such that,*

1. *$B \leq h \times \ell$, $|T| \leq m^h$ and $|F| \geq 2^h$.*
2. *for every alphabet A and every morphism $\alpha : (A \cup B)^* \to M$ into a finite monoid M, if all languages recognized by α belongs to $Bool(\mathcal{C})(A \cup B)$, then, there exists $s \in M$, such that for every $t \in T$, we have $w_t \in B^*$ which satisfies $\alpha(w_t) = s$ and $\tau(w_t) = t$.*

Before we prove Lemma 27, let us use it to finish the construction of smooth taggings. We fix $h \geq 1$ and build a tagging of rank 2^h and size polynomial in 2^h. Let $\gamma : B^* \to T$ and $F \subseteq T$ be as defined in Lemma 27. We fix some binary encoding of the alphabet B over the

two letter alphabet $\mathbb{E}$ given by the morphism $\eta : B^* \to \mathbb{E}*$: for every $b \in B$, $\eta(b)$ is distinct word of length $log_2(|B|)$.

It is straightforward to build a morphism $\tau : \mathbb{E}^* \to T'$ which recognizes the languages $\eta(\gamma^{-1}(s))$ for $s \in T$. Moreover, one may verify that it is possible to do so with a monoid T' of size polynomial with respect to $|T|$ and $|B|$. Therefore the size of T' is polynomial with respect to 2^h since $B \leq h \times m$, $|T| \leq m^h$. One may now verify from our hypothesis on γ that there exists $F' \subseteq T'$ such that $|F'| \geq 2^h$ and $(\tau : \mathbb{E}^* \to T', F')$ fools $\mathcal{C}$. This concludes the main proof. It remains to handle Lemma 27.

Proof of Lemma 27. We start by proving the following fact which handles the special case when $h = 1$. We shall use this fact to define the constants $\ell, m \in \mathbb{N}$.

▶ **Fact 28.** *There exists a morphism $\eta : D^* \to R$ and $G \subseteq R$ such that $|G| = 2$ and for every alphabet A and every morphism $\alpha : (A \cup D)^* \to M$ into a finite monoid M, if all languages recognized by α belongs to $Bool(\mathcal{C})(A \cup D)$, then, there exists $s \in M$, such that for every $r \in R$, we have $w_r \in D^*$ which satisfies $\alpha(w_r) = s$ and $\eta(w_t) = t$.*

Proof. Since $Bool(\mathcal{C}) \neq$ REG, there exist an alphabet D and a regular language $L \subseteq D^*$ such that $L \notin Bool(\mathcal{C})(D)$. Since L is regular, we have a morphism $\eta : D^* \to R$ into a finite monoid R and $XF \subseteq R$ such that $L = \eta^{-1}(X)$. Since $L \notin Bool(\mathcal{C})$, it is not $Bool(\mathcal{C})$-separable from $D^* \setminus L = \eta^{-1}(R \setminus X)$. This implies the existence of $r \in X$ and $r' \in R \setminus X$ such that $\eta^{-1}(r)$ is not $Bool(\mathcal{C})$-separable from $\eta^{-1}(r')$. We let $G = \{r, r'\}$. It remains to show the property described in the fact is satisfied.

Consider a morphism $\alpha : (A \cup D)^* \to M$ such that every language recognized by α belongs to $Bool(\mathcal{C})(A \uplus D)$. We have to exhibit $s \in M$ and $w, w' \in D^*$ such that $\alpha(w) = \alpha(w') = s$, $\eta(w) = r$ and $\eta(w') = r'$. Let $\beta : D^* \to M$ be the restriction of α to D^*. Since $Bool(\mathcal{C})$ is a variety, one may verify that every language recognized by β belongs to $Bool(\mathcal{C})(D)$. Since $\eta^{-1}(r) \subseteq D^*$ is not $Bool(\mathcal{C})$-separable from $\eta^{-1}(r') \subseteq D^*$, it follows that there exists $s \in M$ such that $\beta^{-1}(s)$ intersects both $\eta^{-1}(r)$ and $\eta^{-1}(r')$ (otherwise a separator in $Bool(\mathcal{C})$ would be recognized by β). This exactly says that we have $w, w' \in D^*$ such that $\beta(w) = \alpha(w) = \beta(w') = \alpha(w') = s$, $\eta(w) = r$ and $\eta(w') = r'$, finishing the proof. ◀

We fix the tagging $\eta : D^* \to R$ and G for the remainder of the argument. We define $\ell = |D|$ and $m = |R|$. We may now prove the Lemma 27. We proceed by induction on $h \geq 1$.

The case $h = 1$ has already been handled with Fact 26. Assume now that $h \geq 2$. Induction to $h - 1$ yields a morphism $\gamma' : (B')^* \to T'$ and $F' \subseteq T'$ satisfying the two assertions in the lemma. Recall that $Bool(\mathcal{C})$ is a variety by hypothesis. Hence, it is closed under bijective renaming of letters and we may assume without loss of generality that $D \cap B' = \emptyset$. We define the alphabet B as the disjoint union $B = B' \cup D$. Moreover, we let T as the monoid $T = T' \times R$ equipped with the componentwise multiplication. We let $\gamma : B^* \to T$ as the morphism such for every $b \in B$,

$$\gamma(b) = \begin{cases} (\gamma'(b), 1_R) & \text{if } b \in B' \\ (1_{T'}, \eta(b)) & \text{if } b \in D \end{cases}$$

Finally, we let $F = F' \times G$. Observe that by definition, we have $|F| = 2 \times |F'| \geq 2^h$. Moreover, $|B| = |D| + |B'| \leq h \times \ell$ and $|T| = |T'| \times |R| \leq m^h$. It remains to show that the second assertion in Lemma 27 holds.

We consider an alphabet and a morphism $\alpha : (A \cup B)^* \to M$ such that every language recognized by α belong to $Bool(\mathcal{C})(A \cup B)$. We have to exhibit $s \in M$ such for every $t \in F$,

there exists $w_t \in B^*$ satisfying $\alpha(w_t) = s$ and $\gamma(w_t) = t$. By hypothesis on η and γ', we have the following fact.

▶ **Fact 29.** *We have two elements $s_{B'}, s_D \in M$ which satisfy the following properties:*
- *for every $t' \in F'$, we have $w_{t'} \in (B')^*$ such that $\alpha(w_{t'}) = s_{B'}$ and $\gamma'(w_{t'}) = t'$.*
- *for every $r \in G$, we have $w_r \in D^*$ such that $\alpha(w_r) = s_D$ and $\eta(w_r) = r$.*

Proof. We prove the existence of $s_{B'}$, the argument for s_D is symmetrical. Recall that $B = B' \cup D$ and let $\beta : (A \cup B')^* \to M$ be the restriction of α to $(A \cup B')^*$. Since $Bool(\mathcal{C})$ is a variety, and all languages recognized by α belong to $Bool(\mathcal{C})(A \cup B)$, it straightforward to verify that all languages recognized by β belong to $Bool(\mathcal{C})(A \cup B')$. Hence, since by hypothesis on $\gamma' : (B')^* \to T'$ and F', we obtain $s_{B'} \in M$ such that for every $t' \in F'$, we have $w_{t'} \in (B')^*$ such that $\alpha(w_{t'}) = \beta(w_{t'}) = s_{B'}$ and $\gamma'(w_{t'}) = t'$. ◀

We define $s = s_{B'}s_D$. It remains to show that s satisfies the desired property. Consider $t \in F = F' \times G$. We have $t = (t', r)$ with $t' \in F'$ and $r \in G$. Let $w_t = w_{t'}w_r$. By definition of γ, since $w_{t'} \in (B')^*$ and $w_r \in D^*$, we have,

$$\gamma(w_t) = \gamma(w_{t'})\gamma(w_r) = (\gamma'(w_{t'}), 1_R) \cdot (1_{T'}, \eta(w_r)) = (t', 1_R) \cdot (1_{T'}, r) = (t', r) = t$$

This concludes the proof. ◀

A.4 Proof of Lemma 8

We now prove Lemma 8. Let us first recall the statement.

▶ **Lemma 8.** *Let $\mathcal{C}$ be a finite quotienting Boolean algebra. For every $n \in \frac{1}{2}\mathbb{N}$, $\mathcal{C}_{\mathbb{E}}[n]$ is smooth and an extension of $\mathcal{C}[n]$.*

We fix the finite quotienting Boolean algebra $\mathcal{C}$ for the proof. We start by proving that $\mathcal{C}_{\mathbb{E}}[n]$ is smooth for every $n \in \frac{1}{2}\mathbb{N}$.

Let $k \in \mathbb{N}$, we describe a tagging of rank k. we let $T_k = \{t_0, \dots, t_{k-1}\}$ as the monoid whose multiplication is defined by $t_i t_j = t_{i+j \mod k}$ for $i, j \leq k-1$ (i.e. T is isomorphic to $\mathbb{Z}/k\mathbb{Z}$). We now consider the morphism $\tau_k : \mathbb{E}^* \to T_k$ defined by $\beta(0) = \beta(1) = t_1$ (i.e. τ_k counts the length of words modulo k). Clearly the tagging $(\tau_k : \mathbb{E}* \to T_k, T_k)$ has rank k and can be computed in LogSpace. Moreover, the following lemma can be verified from the definition of $\mathcal{C}_{\mathbb{E}}$ and that of concatenation hierarchies (the proof is left to the reader).

▶ **Lemma 30.** *For every $k \in \mathbb{N}$ and every $n \in \frac{1}{2}\mathbb{N}$, the tagging $(\tau_k : \mathbb{E}* \to T_k, T_k)$ fools $\mathcal{C}_{\mathbb{E}}[n]$.*

Altogether, we obtain that $\mathcal{C}_{\mathbb{E}}[n]$ is smooth for every $n \in \frac{1}{2}\mathbb{N}$. It remains to show that $\mathcal{C}_{\mathbb{E}}[n]$ is an extension of $\mathcal{C}[n]$ for every $n \in \frac{1}{2}\mathbb{N}$. Both conditions involved in extension are verified using induction on n (this amounts to proving that they are preserved by polynomial and Boolean closure). The arguments are straightforward and left to the reader.

B Appendix to Section 4

In this appendix we present the missing proofs of Section 4. Let us first take care of Lemma 10. Recall that in this section, an arbitrary alphabet A and a finite quotienting Boolean algebra $\mathcal{C}$ are fixed.

B.1 Proof of Lemma 10

Let us first recall the statement of Lemma 10

▶ **Lemma 10.** *Given two morphisms recognizing regular languages $L_1, L_2 \subseteq A^*$ as input, one may compute in* LogSpace *a $\mathcal{C}$-compatible morphism which recognizes both L_1 and L_2.*

We let $\alpha_1 : A^* \to M_1$ and $\alpha_2 : A^* \to M_2$ as the morphisms recognizing L_1 and L_2. Recall that the relation $\sim_{\mathcal{C}}$ associated to $\mathcal{C}$ is a congruence over A^* for word concatenation ($\sim_{\mathcal{C}}$ compares words which belong to the same languages in $\mathcal{C}$). Therefore, the quotient set $A^*/\sim_{\mathcal{C}}$ is a monoid (we write "$\bullet$" for its multiplication) and the map $w \mapsto [w]_{\mathcal{C}}$ which maps each word to its $\sim_{\mathcal{C}}$-class is a monoid morphism.

We let $M = M_1 \times M_2 \times (A^*/\sim_{\mathcal{C}})$ as the monoid equipped with the componentwise multiplication. Moreover, we let $\beta : A^* \to M$ as the morphism defined by $\beta(w) = (\alpha_1(w), \alpha_2(w), [w]_{\mathcal{C}})$. Clearly, β recognizes both L_1 and L_2. Moreover, β is $\mathcal{C}$-compatible: given $s = (s_1, s_2, D) \in M$, it suffices to define $[s]_{\mathcal{C}} = D$. It then immediate that the two axioms in the definition of $\mathcal{C}$-compatibility are satisfied:

- Given $w \in A^*$ we $[\beta(w)]_{\mathcal{C}} = [w]_{\mathcal{C}}$.
- Given $s, s' \in M$ $[ss']_{\mathcal{C}} = [s]_{\mathcal{C}} \bullet [s']_{\mathcal{C}}$.

Finally, it is clear that β ca be computed in LogSpace from α_1 and α_2.

▶ Remark. It is important here that the alphabet A is fixed. This implies that the monoid $A^*/\sim_{\mathcal{C}}$ is a constant. When A is a parameter, it may not be possible to compute β in LogSpace (this depends on $\mathcal{C}$).

B.2 Proof of Proposition 12

We actually prove a statement which is slightly stronger than Proposition 12 (this is required to use induction in the proof). It is as follows.

▶ **Proposition 31.** *Let $h, m \in \mathbb{N}$ be constants. Consider two $\mathcal{C}$-compatible morphisms $\alpha : A^* \to M$ and $\beta : A^* \to N$ and a good subset $S \subseteq N$. Given $s \in M$ and $T \in 2^N$ such that $|T| \leq m$, one may test in* NL *with respect to $|M|$ and $|N|$ whether there exists an (α, β, S)-tree of operational height at most h and with root label (s, T).*

Clearly, Proposition 12 is the special case of Proposition 31 when $m = 1$. Hence, we may concentrate on proving Proposition 31.

Consider two $\mathcal{C}$-compatible morphisms $\alpha : A^* \to M$ and $\beta : A^* \to N$ and a good subset $S \subseteq N$. Given $h, m \in \mathbb{N}$, we shall write $X_{h,m} \subseteq M \times 2^N$ for the set of all elements $(s, T) \in M \times 2^N$ such that $|T| \leq m$ and (s, T) is the root label of an (α, β, S)-tree of operational height is a most h.

We have to show that when h and m are fixed, one may test in NL with respect to $|M|$ and $|N|$ whether some input pair $(s, T) \in M \times 2^N$ belongs to $X_{h,m}$. We proceed by induction on h.

When $h = 0$, (α, β, S)-trees of operational height 0 contain only leaves and binary nodes. Therefore, one may verify from the definition that their labels are always of the form $(\alpha(w), \{\beta(w)\})$ for some $w \in A^*$. Consequently, the problem of deciding whether (s, T) belongs to $X_{h,m}$ amounts to verifying that T is a singleton $\{t\}$ and that there exists $w \in A^*$ such that $\alpha(w) = s$ and $\beta(w) = t$. This is easily achieved in NL.

We now assume that $h \geq 1$. We introduce an auxiliary set $Y_{h,m} \subseteq M \times 2^N$. Given $(s, T) \in M \times 2^N$, we have $(s, T) \in Y_{h,m}$ when $|T| \leq m$ and one of the two following conditions holds:

- $(s, T) \in X_{h-1,m}$, or,
- (s, T) is the root label of an (α, β, S)-tree having operational height h and whose root is an S-operation node (i.e. the unique child of the root has operational height $h-1$).

By induction on h, we have the following lemma.

▶ **Lemma 32.** *Let $s \in M$ and $T \in 2^N$, one may test in* NL *with respect to $|M|$ and $|N|$ whether $(s, T) \in Y_{h,m}$*

Proof. It suffices to verify that given as input $(s, T) \in Y_{h,m}$ such that $|T| \leq m$, one may check in NL whether one of the two conditions in the definition of $Y_{h,m}$ is satisfied. Testing whether $(s, T) \in X_{h-1,m}$ can be achieved in NL by induction on $h-1$. For the second condition, we know that the two following properties are equivalent:

- (s, T) is the root label of an (α, β, S)-tree having operational height at h and whose root is an S-operation node.
- there exists an (α, β, S)-tree having operational height $h-1$ whose root label (e, E) is an idempotent satisfying:

$$e = s \quad \text{and} \quad T \subseteq E \cdot \{t \in S \mid [e]_{\mathcal{C}} = [t]_{\mathcal{C}} \in S\} \cdot E$$

Since $|T| \leq m$, it is straightforward to verify that the second assertion is satisfied if and only if E can be chosen such that $|E| \leq 2m$ (i.e. $(e, E) \in X_{h-1,2m}$). Hence, the second conditions can be checked in NL by induction which concludes the proof. ◀

Moreover, the next lemma is immediate from the definition of (α, β, S)-trees of operational height h and a pigeon-hole principle argument.

▶ **Lemma 33.** *Let $(s, T) \in M \times 2^N$. Then, $(s, T) \in X_{h,m}$ if and only if there exists $\ell \leq |M| \times |N|^m$ and ℓ elements $(r_1, T_1), \ldots, (r_\ell, T_\ell) \in Y_{h,m}$ such that,*

$$s = r_1 \cdots r_\ell \quad \text{and} \quad \{t_1, \ldots, t_m\} \subseteq T_1 \cdots T_\ell$$

It is now immediate from Lemma 32 and 33 that one may test in NL with respect to $|M|$ and $|N|$ whether some input pair $(s, T) \in M \times 2^N$ belongs to $X_{h,m}$. This concludes the proof.

B.3 Proof of Proposition 13

Let us first recall the statement of Proposition 13.

▶ **Proposition 13.** *Let $h \in \mathbb{N}$ be the $\mathcal{J}$-depth of $A^*/\sim_{\mathcal{C}}$. Consider two $\mathcal{C}$-compatible morphisms $\alpha : A^* \to M$ and $\beta : A^* \to N$, and a good subset $S \subseteq N$. Then, for every $(s, T) \in M \times 2^N$, the following properties are equivalent:*

1. *(s, T) is the root label of some (α, β, S)-tree.*
2. *(s, T) is the root label of some (α, β, S)-tree whose operational height is at most h.*

We fix h as the $\mathcal{J}$-depth of $A^*/\sim_{\mathcal{C}}$. Moreover, we let $\alpha : A^* \to M$ and $\beta : A^* \to N$ as two $\mathcal{C}$-compatible morphisms and fix $S \subseteq N$ as a good subset. The direction 2) $\Rightarrow$ 1) in Proposition 13 is trivial. Therefore, we concentrate on proving that 1) $\Rightarrow$ 2). Given $(s, T) \in M \times 2^N$ and a (α, β, S)-tree $\mathbb{T}$ whose root label is (s, T), we explain how to construct a second tree with the same root label and whose operational height is bounded by h.

For the proof, we call *operational size* of an (α, β, S)-tree the total number of operation nodes it contains (clearly, this number is always larger than the operational height). The result is a consequence of the following lemma.

▶ **Lemma 34.** *Consider an (α, β, S)-tree $\mathbb{T}$ and assume that it contains a branch with two distinct operation nodes x and x' whose labels (s, T) and (s', T') satisfy $[s]_{\mathcal{C}} = [s']_{\mathcal{C}}$. Then, there exists a second tree $\mathbb{T}'$ with strictly smaller operational size than $\mathbb{T}$ and with the same root label.*

Starting from an arbitrary (α, β, S)-tree $\mathbb{T}$, one may use Lemma 34 recursively to build $\mathbb{T}'$ which has the same label as $\mathbb{T}$ and such that for any two operation nodes x and x' on the same branch of $\mathbb{T}'$, their labels (s, T) and (s', T') satisfy $[s]_{\mathcal{C}} \neq [s']_{\mathcal{C}}$. Clearly, this tree $\mathbb{T}'$ has operational height bounded by h (by definition of h as the $\mathcal{J}$-depth of $A^*/\sim_{\mathcal{C}}$). This concludes the proof for the implication 1) $\Rightarrow$ 2) in Proposition 13.

We now concentrate on proving Lemma 34. We let $\mathbb{T}$ and $x \neq x'$ the nodes defined in the lemma. Since x, x' are on the same branch, one is an ancestor of the other. By symmetry, we assume that x is an ancestor of x'. We let $\mathbb{S}$ as the subtree of $\mathbb{T}$ which is rooted in x. We let (s, T) as the label $(s, T) = lab(\mathbb{S}) = lab(x)$. We build a new tree $\mathbb{S}'$ with the same label as $\mathbb{S}$ and strictly smaller operational size. It will then be simple to build the desired tree $\mathbb{T}'$ by replacing the subtree $\mathbb{S}$ with $\mathbb{S}'$ in $\mathbb{T}$.

Given two nodes z, z' of $\mathbb{S}$, we write $z < z'$ to denote the fact that z is a (strict) ancestor of z'. By hypothesis, we have $x < x'$, hence we may consider the sequence of operations nodes which are between the two. We let $x_1, \ldots, x_k$ as the sequence of all nodes which satisfy the following properties:

- For all i, x_i is an operation node.
- $x = x_k < \cdots < x_1 = x'$.

Note that since $x_k = x$ and $x_1 = x'$, we have $k \geq 2$. For all $i \geq 1$, we let (f_i, T_i) as label of x_i. By definition of operation nodes, $f_i \in M$ must be an idempotent. Moreover, $(f_k, T_k) = (s, T)$ is the label of $\mathbb{S}$ and we know by hypothesis that $[f_1]_{\mathcal{C}} = [f_k]_{\mathcal{C}}$. Finally, consider the unique child of x_1 and let (e, E) be the label of this child (which is an idempotent of $M \times 2^N$ since x_1 is an operation node). Recall that by definition of operation nodes, we have $e = f_1$ and $T_1 \subseteq E \cdot \{t \in S \mid [e]_{\mathcal{C}} = [t]_{\mathcal{C}}\} \cdot E$.

We now classify the nodes within $\mathbb{S}$ in several categories. We call *backbone* of $\mathbb{S}$ the path made of all (strict) ancestors of x_1. Since x_k is the root, there are $k - 1 \geq 1$ operation nodes on the backbone (the nodes $x_2, \ldots, x_k$). Furthermore, we call *lower nodes* all nodes within the subtree rooted in x_1 (including x_1). We denote by m the number operation nodes which are lower nodes. Finally, all nodes which are neither backbone nor lower nodes are called *side nodes*. Observe that any side node z has a closest ancestor y on the backbone which has to be a binary node. We say that z is a *left (resp. right) side node* when it belongs to the subtree whose root is the left (resp. right) child of y. Finally, we associate a *rank* to each side node z: the rank of z is the smallest $i \leq k$ such that x_i is an ancestor of z (i must exist since x_k is the root). For all $i \leq k$, we write ℓ_i (resp. r_i) the number of operation nodes which are left (resp. right) side nodes of rank i. We illustrate these definitions in Figure 1.

Observe that by definition, backbone nodes, lower nodes and side nodes account for all nodes in the tree. Thus, we have the following fact.

▶ **Fact 35.** *The total number of operation nodes in $\mathbb{S}$ is,*

$$k - 1 + m + \ell_1 + \cdots \ell_k + r_1 + \cdots + r_k$$

Essentially, the desired tree $\mathbb{S}'$ is built by removing all backbone nodes from $\mathbb{S}$ and replacing them with binary nodes. Thus, we obtain a tree $\mathbb{S}'$ whose operational size is $m + \ell_1 + \cdots \ell_k + r_1 + \cdots + r_k$ which is strictly smaller than that of $\mathbb{S}$ since $k - 1 \geq 1$. We use an inductive construction which is formalized in the following lemma.

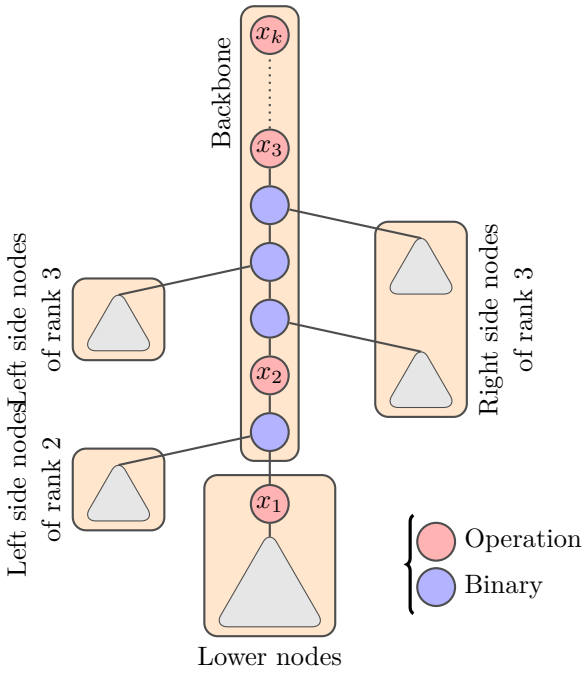

Figure 1 Classification of the nodes in $\mathbb{S}$ (here, there are no right side nodes of rank 2).

▶ **Lemma 36.** *For every $i \leq k$, there exist two (α, β, S)-trees $\mathbb{U}_i$ and $\mathbb{V}_i$ of labels (u_i, U_i) and (v_i, V_i) with operational heights $\ell_1 + \cdots + \ell_i$ and $r_1 + \cdots + r_i$ respectively. Moreover, there exist $u'_i, v'_i \in M$ satisfying the following two conditions:*

1. *For $q \in \{u_i, u'_i\}$ and $r \in \{v_i, v'_i\}$, $f_i = qer$.*
2. *$T_i \subseteq U_i E \cdot \{t \in S \mid [t]_{\mathcal{C}} = [ev'_i f_i u'_i e]_{\mathcal{C}}\} \cdot EV_i$.*

Before we show Lemma 36, we use it to build the desired tree $\mathbb{S}'$ and finish the proof of Lemma 34. Recall that we need $\mathbb{S}'$ to have label $lab(\mathbb{S}) = (s, T) = (f_k, T_k)$. We apply Lemma 36 in the special case when $i = k$. This yields two (α, β, S)-trees $\mathbb{U}_k$ and $\mathbb{V}_k$ with labels (u_k, U_k) and (v_k, V_k) which have operational heights $\ell_1 + \cdots + \ell_i$ and $r_1 + \cdots + r_i$. Moreover, we let $u'_k, v'_k \in M$ which satisfy the two assertions in the lemma.

It follows from the first assertion in Lemma 36 that $u_k e v_k = v'_k e u'_k = f_k = s$. This implies the following fact.

▶ **Fact 37.** $[e]_{\mathcal{C}} = [ev'_k f_k u'_k e]_{\mathcal{C}}$.

Proof. By definition of $\mathcal{C}$-compatible morphisms we have,

$$[ev'_k f_k u'_k e]_{\mathcal{C}} = [e]_{\mathcal{C}} \bullet [v'_k]_{\mathcal{C}} \bullet [f_k]_{\mathcal{C}} \bullet [u'_k]_{\mathcal{C}} \bullet [e]_{\mathcal{C}}$$

Therefore, since $[f_k]_{\mathcal{C}} = [e]_{\mathcal{C}}$, it suffices to prove that, $[e]_{\mathcal{C}} = [e]_{\mathcal{C}} \bullet [v'_k]_{\mathcal{C}} \bullet [e]_{\mathcal{C}} \bullet [u'_k]_{\mathcal{C}} \bullet [e]_{\mathcal{C}}$.

By the first assertion in Lemma 36, we have $e = f_k = u'_k e v'_k$. Hence, $[e]_{\mathcal{C}} = [u'_k]_{\mathcal{C}} \bullet [e]_{\mathcal{C}} \bullet [v'_k]_{\mathcal{C}}$. Moreover, since e is idempotent of M, $[e]_{\mathcal{C}} = [ee]_{\mathcal{C}} = [e]_{\mathcal{C}} \bullet [e]_{\mathcal{C}}$ is an idempotent of $A^*/\sim_{\mathcal{C}}$. This yields,

$$\begin{aligned}
[e]_{\mathcal{C}} &= [e]_{\mathcal{C}} \bullet [u'_k]_{\mathcal{C}} \bullet [e]_{\mathcal{C}} \bullet [v'_k]_{\mathcal{C}} \bullet [e]_{\mathcal{C}} \\
[e]_{\mathcal{C}} &= ([e]_{\mathcal{C}} \bullet [u'_k]_{\mathcal{C}})^{\omega} \bullet [e]_{\mathcal{C}} \bullet ([v'_k]_{\mathcal{C}} \bullet [e]_{\mathcal{C}})^{\omega} \\
[e]_{\mathcal{C}} &= [e]_{\mathcal{C}} \bullet ([v'_k]_{\mathcal{C}} \bullet [e]_{\mathcal{C}})^{\omega} \\
[e]_{\mathcal{C}} &= [e]_{\mathcal{C}} \bullet [v'_k]_{\mathcal{C}} \bullet [e]_{\mathcal{C}} \bullet ([v'_k]_{\mathcal{C}} \bullet [e]_{\mathcal{C}})^{\omega - 1}
\end{aligned}$$

We may now replace the second copy of $[e]_{\mathcal{C}}$ in the above with $[e]_{\mathcal{C}} \cdot [u'_k]_{\mathcal{C}} \cdot [e]_{\mathcal{C}} \cdot [v'_k]_{\mathcal{C}} \cdot [e]_{\mathcal{C}}$ which yields,

$$[e]_{\mathcal{C}} = [e]_{\mathcal{C}} \cdot [v'_k]_{\mathcal{C}} \cdot [e]_{\mathcal{C}} \cdot [u'_k]_{\mathcal{C}} \cdot [e]_{\mathcal{C}} \cdot ([v'_k]_{\mathcal{C}} \cdot [e]_{\mathcal{C}})^{\omega}$$

Finally, since $[e]_{\mathcal{C}} = [e]_{\mathcal{C}} \cdot ([v'_k]_{\mathcal{C}} \cdot [e]_{\mathcal{C}})^{\omega}$, this yields $[e]_{\mathcal{C}} = [e]_{\mathcal{C}} \cdot [v'_k]_{\mathcal{C}} \cdot [e]_{\mathcal{C}} \cdot [u'_k]_{\mathcal{C}} \cdot [e]_{\mathcal{C}}$ as desired. ◀

In view of Fact 37 and the second assertion in Lemma 36, we obtain that,

$$T_k \subseteq U_k E \cdot \{t \in S \mid [t]_{\mathcal{C}} = [e]_{\mathcal{C}}\} \cdot E V_k \tag{1}$$

Finally, we have a tree of root label (e, E) whose operational size is $m-1$: the child of x_1. Hence, using one operation node, we may build a tree of operational size m whose root label is:

$$(e, E \cdot \{t \in S \mid [t]_{\mathcal{C}} = [e]_{\mathcal{C}}\})$$

Finally, by (1), we may combine this tree with $\mathbb{U}_k$ and $\mathbb{V}_k$ using two binary nodes to get a tree $\mathbb{S}'$ whose root label is:

$$(s, T) = (f_k, T_k) = (u_k e v_k, T_k)$$

By definition, this tree $\mathbb{S}'$ has operational size $m+m+\ell_1+\cdots+\ell_k+r_1+\cdots+r_k$. As desired, this is strictly smaller than $\mathbb{S}$ (its operational size is $k-1+m+\ell_1+\cdots\ell_k+r_1+\cdots+r_k$ by Fact 35 and $k-1 \geq 1$). This terminates the proof of Lemma 34.

It now remains to prove Lemma 36. We proceed by induction on i. When $i=1$, since x_1 is an operation node whose unique child has label (e, E), we have $f_1 = e$ and $T_1 \subseteq E \cdot \{t \in S \mid [e]_{\mathcal{C}} = [t]_{\mathcal{C}}\} \cdot E$. We define both $\mathbb{U}_1$ and $\mathbb{V}_1$ as the same tree made of a single leaf whose label is $(1_M, \{1_N\}) = (\alpha(\varepsilon), \{\beta(\varepsilon)\})$. It is then simple to verify that the two assertions in the lemma are satisfied for $u'_1 = v'_1 = 1_M$.

We now assume that $i \geq 2$. By definition, x_i has a unique child whose label is an idempotent (f_i, F_i) such that,

$$T_i \subseteq F_i \cdot \{t \in S \mid [f_i]_{\mathcal{C}} = [t]_{\mathcal{C}}\} \cdot F_i$$

We use the following fact to choose our new trees $\mathbb{U}_i, \mathbb{V}_i$.

▶ **Fact 38.** *There exist two (α, β, S)-trees $\mathbb{P}$ and $\mathbb{Q}$ whose operational sizes are respectively bounded by ℓ_i and r_i and whose labels (p, P) and (q, Q) satisfy the following two properties,*

- $f_i = p \cdot f_{i-1} \cdot q$
- $F_i \subseteq P T_{i-1} Q$

Proof. We build $\mathbb{P}$ (resp. $\mathbb{Q}$) by combining all subtrees made of left (resp. right) side nodes of rank i into a single one using binary nodes only. In the degenerate case when there are no left (resp. right) side nodes $\mathbb{P}$ (resp. $\mathbb{Q}$) is a single leaf with label $(1_M, \{1_N\})$.

Let us describe this construction in more details when the set of left and right side nodes of rank i are nonempty Consider all nodes between x_i and x_{i-1} (which are all binary by definition). For each such node, one child is an ancestor of x_{i-1} (or x_{i-1} itself) and the other is a side node. We define,

- $x_i < z_{h_1} < \cdots < z_1 < x_{i-1}$ as all binary nodes whose left children are side nodes (in particular these children and all their descendants are left side nodes of rank i).

- $x_i < z'_{h_2} < \cdots < z'_1 < x_{i-1}$ as all binary nodes whose right children are side nodes (in particular these children and all their descendants are right side nodes of rank i).

We may now define $\mathbb{P}$ and $\mathbb{Q}$. We start with $\mathbb{P}$. For all $j \leq h_1$, we let (p_j, P_j) as the label of the left child of z_j. Clearly, one may combine all subtrees rooted in the left children of the z_j with binary nodes into a single one whose label is,

$$(p, P) = (p_{h_1}, P_{h_1}) \cdot \cdots \cdot (p_1, P_1)$$

By definition, the operational size of $\mathbb{P}$ is ℓ_i: the sum of those for the subtrees we have combined (we only added binary nodes). Symmetrically, one may build $\mathbb{Q}$ of operational size r_i whose label is,

$$(q, Q) = (q_1, Q_1) \cdot \cdots \cdot (q_{h_2}, Q_{h_2})$$

where (q_j, Q_j) is the label of the right child of z'_j for all $j \leq h_2$. One may now verify from the definition that the two assertions in the fact are satisfied. ◀

We are now ready to define our new trees $\mathbb{U}_i$ and $\mathbb{V}_i$. We first use induction to obtain two trees $\mathbb{U}_{i-1}$ and $\mathbb{V}_{i-1}$ of labels (u_{i-1}, U_{i-1}) and (v_{i-1}, V_{i-1}) which satisfy the conditions of Lemma 36 for $i-1$. We define,

- $\mathbb{U}_i$ as the tree of label $(u_i, U_i) = (p \cdot u_{i-1}, PU_{i-1})$ obtained by combining $\mathbb{P}$ and $\mathbb{U}_{i-1}$ with a single binary node.
- $\mathbb{V}_i$ as the tree of label $(v_i, V_i) = (v_{i-1} \cdot q, V_{i-1}S)$ obtained by combining $\mathbb{V}_{i-1}$ and $\mathbb{Q}$ with a single binary node.

It remains to prove that this definition for the trees $\mathbb{U}_i$ and $\mathbb{V}_i$ satisfies the conditions in Lemma 36. By definition, the operational size of $\mathbb{U}_i$ is the sum of that of $\mathbb{P}$ (i.e. ℓ_i by definition in Fact 38) with that of $\mathbb{U}_{i-1}$ (i.e. $\ell_1 + \cdots \ell_{i-1}$ since we obtained $\mathbb{U}_{i-1}$ by induction). This exactly says that the operational size of $\mathbb{U}_i$ is $\ell_1 + \cdots \ell_i$ as desired. Symmetrically, one may verify that the operational size of $\mathbb{V}_i$ is $r_1 + \cdots + r_i$.

We now have to find $u'_i, v'_i \in M$ which satisfy the two assertions in the lemma. Since we obtained $\mathbb{U}_{i-1}$ and $\mathbb{V}_{i-1}$ by induction, we also have $u'_{i-1}, v'_{i-1} \in \mathbf{L}$ which satisfy these two assertions for $i-1$. We define,

$$u'_i = pf_{i-1}u'_{i-1} \quad \text{and} \quad v'_i = v'_{i-1}f_{i-1}q$$

It remains to verify that the two assertions in Lemma 36 hold for this choice of u'_i, v'_i. We begin with the first one.

Assertion 1. We have four equalities to verify. Since the argument is similar for all four, we concentrate on $f_i = u_i e v_i$ and $f_i = u'_i e v'_i$ whose proofs encompass all arguments. By Fact 38, we know that $f_i = pf_{i-1}q$. Moreover, since $f_{i-1} = u_{i-1}ev_{i-1}$ by the inductive definition of u_{i-1} and v_{i-1}, we get,

$$f_i = pu_{i-1}ev_{i-1}q = u_i e v_i$$

Furthermore, f_{i-1} is idempotent. Thus, $f_i = pf_{i-1}q = p(f_{i-1})^3 q$ and since by construction of u'_{i-1} and v'_{i-1}, we have $f_{i-1} = u'_{i-1}ev'_{i-1}$, we obtain,

$$f_i = pf_{i-1}u'_{i-1}ev'_{i-1}f_{i-1}q = u'_i e v'_i$$

Assertion 2. We finish with the second assertion which is the most involved. In particular, this is where we use the fact that S is good. We need to show that,

$$T_i \subseteq U_i E \cdot \{t \in S \mid [t]_{\mathcal{C}} = [ev_i' f_i u_i' e]_{\mathcal{C}}\} \cdot EV_i$$

We start with a simple fact.

▶ **Fact 39.** *For any $(s,T) \in M \times 2^N$ which is the label of an (α,β,S)-tree, we have $T \subseteq \{t \in S \mid [t]_{\mathcal{C}} = [s]_{\mathcal{C}}\}$.*

Proof. This is immediate by induction on the height of (α,β,S)-trees using the hypothesis that S is good. ◀

We now start the proof. By definition, (f_i, T_i) is the label of the operation node x_i whose child has label (f_i, F_i). Hence, $T_i \subseteq F_i \cdot \{t \in S \mid [t]_{\mathcal{C}} = [f_i]_{\mathcal{C}}\} \cdot F_i$ and it follows from the second item in Fact 38 that,

$$T_i \subseteq PT_{i-1}Q \cdot \{t \in S \mid [t]_{\mathcal{C}} = [f_i]_{\mathcal{C}}\} \cdot PT_{i-1}Q$$

The result is now a consequence of the two following inclusions:

$$\begin{array}{rcl} PT_{i-1}Q & \subseteq & U_i E \cdot \{t \in S \mid [t]_{\mathcal{C}} = [ev_i']_{\mathcal{C}}\} \\ PT_{i-1}Q & \subseteq & \{t \in S \mid [t]_{\mathcal{C}} = [u_i' e]_{\mathcal{C}}\} \cdot EV_i \end{array} \tag{2}$$

Indeed, one may combine these two inequalities with the previous one using the hypothesis that S is good to obtain the desired inclusion:

$$\begin{array}{rcl} T_i & \subseteq & U_i E \cdot \{t \in S \mid [t]_{\mathcal{C}} = [ev_i']_{\mathcal{C}}\} \cdot \{t \in S \mid [t]_{\mathcal{C}} = [f_i]_{\mathcal{C}}\} \cdot \{t \in S \mid [t]_{\mathcal{C}} = [u_i' e]_{\mathcal{C}}\} \cdot EV_i \\ & \subseteq & U_i E \cdot \{t \in S \mid [t]_{\mathcal{C}} = [ev_i' f_i u_i' e]_{\mathcal{C}}\} \cdot EV_i \end{array}$$

It remains to prove the two inequalities in (2). As they are based on symmetrical arguments, we concentrate on the first one and leave the other to the reader. Since we built U_{i-1} and V_{i-1} with induction, we have,

$$T_{i-1} \subseteq U_{i-1} E \cdot \{t \in S \mid [t]_{\mathcal{C}} = [ev_{i-1}' f_{i-1} u_{i-1}' e]_{\mathcal{C}}\} \cdot EV_{i-1}$$

By Fact 39, $E \subseteq \{t \in S \mid [t]_{\mathcal{C}} = [e]_{\mathcal{C}}\}$ and $V_{i-1} \subseteq \{t \in S \mid [t]_{\mathcal{C}} = [v_{i-1}]_{\mathcal{C}}\}$. Hence, using the fact that S is good, we may simplify the above inclusion as follows:

$$T_{i-1} \subseteq U_{i-1} E \cdot \{t \in S \mid [t]_{\mathcal{C}} = [ev_{i-1}' f_{i-1} u_{i-1}' e v_{i-1}]_{\mathcal{C}}\}$$

Since u_{i-1}' and v_{i-1} were built by induction, we know that $u_{i-1}' e v_{i-1} = f_{i-1}$. Hence, since f_{i-1} is an idempotent,

$$T_{i-1} \subseteq U_{i-1} E \cdot \{t \in S \mid [t]_{\mathcal{C}} = [ev_{i-1}' f_{i-1}]_{\mathcal{C}}\}$$

Using Fact 39 again, we have $Q \subseteq \{t \in S \mid [t]_{\mathcal{C}} = [q]_{\mathcal{C}}\}$. Thus, using the hypothesis that S is good together with the fact that $v_i' = v_{i-1}' f_{i-1} q$ by definition, this yields the following,

$$\begin{array}{rcl} T_{i-1}Q & \subseteq & U_{i-1} E \cdot \{t \in S \mid [t]_{\mathcal{C}} = [ev_{i-1}' f_{i-1} q]_{\mathcal{C}}\} \\ & \subseteq & U_{i-1} E \cdot \{t \in S \mid [t]_{\mathcal{C}} = [ev_i']_{\mathcal{C}}\} \end{array}$$

Finally, since $U_i = PU_{i-1}$ by definition, we have

$$\begin{array}{rcl} PT_{i-1}Q & \subseteq & PU_{i-1} E \cdot \{t \in S \mid [t]_{\mathcal{C}} = [ev_i']_{\mathcal{C}}\} \\ & \subseteq & U_i E \cdot \{t \in S \mid [t]_{\mathcal{C}} = [ev_i']_{\mathcal{C}}\} \end{array}$$

This conclude the proof of Lemma 36.

C Appendix to Section 5

This section provides the missing proofs in Section 5. We start by introducing additional terminology and preliminary results that we shall need to present these proofs.

C.1 Stratifications

We present a stratification of $\mathrm{ST}[3/2] = \mathit{Pol}(\mathrm{AT})$ into finite quotienting lattices. It was introduced in [17]. We refer the reader to [17] for the proofs of the statements presented here.

For any natural number $k \in \mathbb{N}$, we define a finite quotienting lattice $\mathit{Pol}_k(\mathrm{AT}) \subseteq \mathit{Pol}(\mathrm{AT})$. The definition uses induction on k:

- When $k = 0$, we simply define $\mathit{Pol}_0(\mathrm{AT}) = \mathrm{AT}$.
- When $k \geq 1$, we define $\mathit{Pol}_k(\mathrm{AT})$ as the smallest lattice which contains $\mathit{Pol}_{k-1}(\mathrm{AT})$ and such for any $L_1, L_2 \in \mathit{Pol}_{k-1}(\mathrm{AT})$ and any $a \in A$,

$$L_1 a L_2 \in \mathit{Pol}_k(\mathrm{AT})$$

One may verify from the definitions that for every $k \in \mathbb{N}$, $\mathit{Pol}_k(\mathrm{AT})$ is a finite quotienting lattice and that $\mathit{Pol}_k(\mathrm{AT}) \subseteq \mathit{Pol}_{k+1}(\mathrm{AT})$. Moreover, by definition of $\mathit{Pol}(\mathrm{AT})$, we have,

$$\mathrm{ST}[3/2] = \mathit{Pol}(\mathrm{AT}) = \bigcup_{k \geq 0} \mathit{Pol}_k(\mathrm{AT})$$

Given any alphabet A, we associate preorder relations to the strata $\mathit{Pol}_k(\mathrm{AT})$. For every $k \in \mathbb{N}$ and $u, v \in A^*$, we write $u \leqslant_k v$ when the following condition is satisfied,

$$\text{For every } L \in \mathit{Pol}_k(\mathrm{AT})(A), \quad u \in L \Rightarrow v \in L$$

It is immediate by definition that $\leqslant_k$ is a preorder relation on A^*. The key point is that we may use it to characterize separability for $\mathit{Pol}(\mathrm{AT}) = \mathrm{ST}[3/2]$.

▶ **Lemma 40.** *Let A be an alphabet and $L, L' \subseteq A^*$ two languages. Then, the two following properties are equivalent:*

1. *L is **not** $\mathrm{ST}[3/2]$-separable from L'.*
2. *For every $k \in \mathbb{N}$, there exists $w \in L$ and $w' \in L'$ such that $w \leqslant_k w'$.*

Moreover, we may also use $\leqslant_k$ to characterize separability for $\mathit{BPol}(\mathrm{AT}) = \mathrm{ST}[2]$.

▶ **Lemma 41.** *Let A be an alphabet and $L, L' \subseteq A^*$ two languages. Then, the two following properties are equivalent:*

1. *L is **not** $\mathrm{ST}[2]$-separable from L'.*
2. *For every $k \in \mathbb{N}$, there exists $w \in L$ and $w' \in L'$ such that $w \leqslant_k w'$ and $w' \leqslant_k w$.*

We finish the presentation with three properties of the relations $\leqslant_k$. The first one is simple and states that they are compatible with word (this is because the strata $\mathit{Pol}_k(\mathrm{AT})$ are closed under quotients).

▶ **Lemma 42.** *Let A be an alphabet and $k \in \mathbb{N}$. For every $u_1, u_2; v_1, v_2 \in A^*$ such that $u_1 \leqslant_k v_1$ and $u_2 \leqslant_k v_2$, we have $u_1 u_2 \leqslant_k v_1 v_2$.*

The second lemma holds because $\mathit{Pol}(\mathrm{AT})$ is a sub-class of the star-free languages. It is as follows.

▶ **Lemma 43.** *Let A be an alphabet and $k \in \mathbb{N}$. Consider $h_1, h_2 \geq 3^{k+1} - 1$ and any $u \in A^*$. Then, we have $u^{h_1} \leqslant_k u^{h_2}$.*

Finally, the third lemma states a characteristic property of $Pol(\mathrm{AT})$. The proof is rather technical (see [17] for details). Given an alphabet A and a word $w \in A^*$, we write $\mathsf{alph}(w)$ for the *alphabet* of w, i.e. the least sub-alphabet $B \subseteq A$ such $w \in B^*$.

▶ **Lemma 44.** *Let A be an alphabet and $k \in \mathbb{N}$. Consider $h, h_1, h_2 \geq 3^{k+1} - 1$ and any $u, v \in A^*$ such that $\mathsf{alph}(v) \subseteq \mathsf{alph}(u)$, we have $u^h \leqslant_k u^{h_1} v u^{h_2}$.*

C.2 Upper bound in Theorem 19

We explain why ST[3/2]-separation is in PSpace for monoids (as usual, the result may then be lifted to NFAs using Corollary 6). The argument reuses the results of Section 4 and Appendix B, and the fact that $\mathrm{ST}[3/2] = Pol(\mathrm{AT})$. In particular, we adapt Theorem 11 to this setting. We start with some preliminary observations about the class AT.

By definition of AT, it is straightforward to verify that the equivalence $\sim_{\mathrm{AT}}$ compares words with the same alphabet. For $u, v \in A^*$, we have $u \sim_{\mathrm{AT}} v$ if and only if $\mathsf{alph}(u) = \mathsf{alph}(v)$. Therefore, the monoid $A^*/\!\sim_{\mathrm{AT}}$ corresponds to 2^A (the set of sub-alphabets) equipped with union as the multiplication. Moreover, for every $w \in A^*$, we have $[\mathrm{AT}]_w = \mathsf{alph}(w)$.

We shall consider AT-compatible morphisms. If $\alpha : A^* \to M$ is AT-compatible, given $s \in M$, we shall write $\mathsf{alph}(s)$ for $[\mathrm{AT}]_s$. We reuse the notion of (α, β, S)-trees which we introduced in Section 4 (here, we use them in the special case when $\mathcal{C} = \mathrm{AT}$). Consider an alphabet A and two AT-compatible morphisms $\alpha : A^* \to M$ and $\beta : A^* \to N$. Given a pair $(s, T) \in M \times 2^N$, we say that (s, T) is alphabet safe when $\mathsf{alph}(s) = \mathsf{alph}(t)$ for every $t \in T$. The following lemma follows from definitions.

▶ **Lemma 45.** *Consider an alphabet A and two* AT*-compatible morphisms $\alpha : A^* \to M$ and $\beta : A^* \to N$. Moreover, let $S \subseteq N$ be a good subset of N. Then, every $(s, T) \in M \times 2^N$ which is the root label of some (α, β, S)-tree is alphabet safe.*

Note that in the Appendix, the alphabet is one of our parameters which means that the size of the monoid $A^*/\!\sim_{\mathrm{AT}} = 2^A$ may not be constant. Consequently, building AT-compatible morphisms is costly. Hence, we shall have to manipulate the construction explicitly. Given an arbitrary morphism $\alpha : A^* \to M$ into a finite monoid M, we write α_{AT} for the AT-compatible morphism $\alpha_{\mathrm{AT}} : A^* \to M \times 2^A$ defined by $\alpha_{\mathrm{AT}}(w) = (\alpha(w), \mathsf{alph}(w))$.

We may now adapt Theorem 11 to this setting. This is the key result for proving that ST[3/2]-separation is in PSpace for monoids.

▶ **Proposition 46.** *Consider two morphisms $\alpha : A^* \to M$ and $\beta : A^* \to N$. Moreover, let $\alpha_{\mathrm{AT}} : A^* \to M \times 2^A$ and $\beta_{\mathrm{AT}} : A^* \to N \times 2^A$ be the corresponding* AT*-compatible morphisms. Finally, let $S \subseteq N \times 2^A$ be a good subset of $N \times 2^A$ for β_{AT}.*

Given an alphabet safe pair $(s, T) \in (M \times 2^A) \times 2^{N \times 2^A}$, one may test in PSpace *with respect to $|A|$, $|M|$ and $|N|$ whether there exists an $(\alpha_{\mathrm{AT}}, \beta_{\mathrm{AT}}, S)$-tree with root label (s, T).*

Proof sketch. By Lemma 45, the set of possible labels for nodes in $(\alpha_{\mathrm{AT}}, \beta_{\mathrm{AT}}, S)$-trees has size at most $|M| \times 2^{|N|} \times 2^{|A|}$ (this is the size of the set of all alphabet safe pairs in $(M \times 2^A) \times 2^{N \times 2^A}$). This observation yields an EXPTime least fixpoint algorithm for computing the set of all root labels of $(\alpha_{\mathrm{AT}}, \beta_{\mathrm{AT}}, S)$-tree with root label (s, T).

This can be improved to PSpace by observing that it suffices to consider $(\alpha_{\mathrm{AT}}, \beta_{\mathrm{AT}}, S)$-trees whose heights are polynomially bounded with respect to $|A|$, $|M|$ and $|N|$. This is a

simple consequence of Proposition 13 since the $\mathcal{J}$-depth of $A^*/\sim_{\mathrm{AT}} = 2^A$ is easily verified to be $|A| + 1$. ◀

Since $\mathrm{ST}[3/2] = Pol(\mathrm{AT})$, it is now simple to combine Theorem 14 with Proposition 46 to get a PSpace algorithm for ST[3/2]-separation which concludes the proof.

C.3 Proof of Lemma 21

Let us recall the statement of Lemma 21 (we refer the reader to Section 5 for the definition of the relevant notations).

▶ **Lemma 21.** *Consider $0 \leq i \leq n$. Then given an i-valuation V, the two following properties are equivalent:*

1. *Ψ_i is satisfied by V.*
2. *$L_i \cap [V]$ is not $\mathrm{ST}[3/2]$-separable from $L'_i \cap [V]$.*

We proceed by induction on $0 \leq i \leq n$. Let us start with the base case $i = 0$. In that case, Ψ_0 is the quantifier-free formula φ. Consider some 0-valuation $V \subseteq (B_0)^*$. One may verify the following fact from the definitions of $L' \subseteq (B_0)^*$ and $[V]$.

▶ **Fact 47.** *The two following properties are equivalent:*

1. *Ψ_0 is satisfied by V.*
2. *$L'_0 \cap [V] \neq \emptyset$.*

Since $L_0 = (B_0)^*$ by definition, we have $L_0 \cap [V] = [V]$. Hence, it is immediate that $L_0 \cap [V] = [V]$ is not $\mathrm{ST}[3/2]$-separable from $L'_0 \cap [V]$ if and only if $L'_0 \cap [V] \neq \emptyset$. Combined with Fact 47, this yields Lemma 21 in the case $i = 0$.

We now assume that $i \geq 1$. There are two cases depending on whether the quantifier Q_i is existential or universal (this is expected since the definitions of L_i and L'_i depend on this parameter). Since these two cases are similar, we handle the one when Q_i is existential and leaver the other to the reader. Consider an i-valuation $V \subseteq (B_i)^*$. We have to show that the two following properties are equivalent:

1. Ψ_i is satisfied by V.
2. $L_i \cap [V]$ is not $\mathrm{ST}[3/2]$-separable from $L'_i \cap [V]$.

Let us start with some terminology that we shall use for both directions. We let $V_\perp$ and $V_\top$ as the following $(i-1)$-valuations built from V:

$$V_\top = V \setminus \{\#_i, \overline{x_i}\} \subseteq B_{i-1} \quad \text{and} \quad V_\perp = V \setminus \{\#_i, x_i\} \subseteq B_{i-1}$$

We may now prove the equivalence. There are two directions to show.

Direction 1) ⇒ 2)**.** Assume that Ψ_i is satisfied by V. We show that $L_i \cap [V]$ is not $\mathrm{ST}[3/2]$-separable from $L'_i \cap [V]$. We use Lemma 40: given an arbitrary $k \in \mathbb{N}$, we have to exhibit $w \in L_i \cap [V]$ and $w' \in L'_i \cap [V]$ such that $w \leqslant_k w'$. We fix k for the proof.

Recall that by hypothesis, we have $\Psi_i = \exists x_i\ \Psi_{i-1}$. Hence, since Ψ_i is satisfied by V, the definitions yield that either $V_\top$ or $V_\perp$ satisfies Ψ_{i-1}. By symmetry, we assume that we are in the former case: $V_\top$ satisfies Ψ_{i-1}. By induction hypothesis this implies that $L_{i-1} \cap [V_\top]$ is not $\mathrm{ST}[3/2]$-separable from $L'_{i-1} \cap [V_\top]$. Consequently, Lemma 40 yields $u \in L_{i-1} \cap [V_\top]$ and $u' \in L'_{i-1} \cap [V_\top]$ such that $u \leqslant_k u'$. Note that by definition of $V_\top$, we have $u, u' \in (B_{i-1} \setminus \{\overline{x_i}\})^*$. We define,

$$\begin{aligned} w &= (\#_i x_i u \$ x_i)^{3^{k+1}} \#_i \\ y &= (\#_i x_i u \$ x_i)^{3^{k+1}} \#_i \$ (\#_i x_i u \$ x_i)^{3^{k+1}} \#_i \\ w' &= (\#_i x_i u' \$ x_i)^{3^{k+1}} \#_i \$ (\#_i x_i u' \$ x_i)^{3^{k+1}} \#_i \end{aligned}$$

Clearly, $\mathsf{alph}(\#_i\$) \subseteq \mathsf{alph}(\#_i x_i u \$ x_i)$. Therefore, Lemma 44 yields that $w \leqslant_k y$. Moreover, since $u \leqslant_k u'$, we get from Lemma 42 that $y \leqslant_k w'$. By transitivity, we get $w \leqslant_k w'$. Finally, one may verify from the definition of L_i and L'_i that $w \in L_i \cap [V]$ and $w' \in L'_i \cap [V]$. Therefore, Lemma 40 yields that $L_i \cap [V]$ is not ST[3/2]-separable from $L'_i \cap [V]$ as desired.

Direction 2) $\Rightarrow$ 1)**.** We actually prove the contrapositive of this implication. Assuming that Ψ_i is not satisfied by V, we show that $L_i \cap [V]$ is ST[3/2]-separable from $L'_i \cap [V]$. Since $\Psi_i = \exists x_i\ \Psi_{i-1}$, our hypothesis yields that Ψ_{i-1} is neither satisfied by $V_\top$ nor by $V_\bot$. Therefore, induction yields the two following properties:

1. $L_{i-1} \cap [V_\top]$ is ST[3/2]-separable from $L'_{i-1} \cap [V_\top]$. We let $K_\top \in \mathrm{ST}[3/2]$ as a separator. Note that since $[V_\top] \in \mathrm{ST}[3/2]$ (actually $[V_\top] \in \mathrm{AT}$), we may assume without loss of generality that $K_\top \subseteq [V_\top]$.
2. $L_{i-1} \cap [V_\bot]$ is ST[3/2]-separable from $L'_{i-1} \cap [V_\bot]$. We let $K_\bot \in \mathrm{ST}[3/2]$ as a separator. Again, we may assume without loss of generality that $K_\top \subseteq [V_\bot]$.

We now define a language $K \in \mathrm{ST}[3/2]$ from $K_\top$ and $K_\bot$. We then show that it separates $L_i \cap [V]$ from $L'_i \cap [V]$. We let,

$$
K = \begin{array}{ll}
 & \{\#_i\} \\
\cup & A^*\#_i((A^*x_iA^* \cap A^*\overline{x_i}A^*) \setminus (A^*\#_iA^*))\#_i \\
\cup & \#_i x_i K_\top \$ x_i \#_i (A \setminus \{\overline{x_i}\})^* \\
\cup & A^*\#_i((A^*\overline{x_i}A^*) \setminus (A^*\#_iA^*))\#_i x_i K_\top \$ x_i \#_i (A \setminus \{\overline{x_i}\})^* \\
\cup & \#_i \overline{x_i} K_\bot \$ \overline{x_i} \#_i (A \setminus \{x_i\})^* \\
\cup & A^*\#_i((A^*x_iA^*) \setminus (A^*\#_iA^*))\#_i \overline{x_i} K_\bot \$ \overline{x_i} \#_i (A \setminus \{x_i\})^*
\end{array}
$$

It is straightforward to verify that $K \in \mathit{Pol}(\mathrm{AT}) = \mathrm{ST}[3/2]$. It remains to verify that K separates $L_i \cap [V]$ from $L'_i \cap [V]$.

We first show that $L_i \cap [V] \subseteq K$. Consider a word $w \in L_i \cap [V]$, we show that $w \in K$. Recall that we have $L_i = (\#_i(x_i + \overline{x_i})L_{i-1}\$(x_i + \overline{x_i}))^*\#_i$. Consequently, there exists $k \geq 0$ and $w_1, \ldots, w_k \in (x_i + \overline{x_i})L_{i-1}\$(x_i + \overline{x_i})$ such that,

$$w = \#_i w_1 \cdots \#_i w_k \#_i$$

Observe first that if $k = 0$, then $w = \#_i \in K$ and we are finished. Assume now that $k = 1$. By definition of K, when $w_k \in (A^*x_iA^* \cap A^*\overline{x_i}A^*) \setminus (A^*\#_iA^*)$, we also have $w \in K$. Therefore, we assume that $w_k \notin (A^*x_iA^* \cap A^*\overline{x_i}A^*) \setminus (A^*\#_iA^*)$. Since $w_k \in (x_i + \overline{x_i})L_{i-1}\$(x_i + \overline{x_i})$, the letter $\#_i$ cannot occur in w_k (by definition of L_{i-1}). Hence, our hypothesis on w_k implies one of the two following properties holds:

- $x_i \in \mathsf{alph}(w_k)$ and $\overline{x_i} \notin \mathsf{alph}(w_k)$, or,
- $\overline{x_i} \in \mathsf{alph}(w_k)$ and $x_i \notin \mathsf{alph}(w_k)$.

By symmetry, we handle the case when the first property holds and leave the other to the reader. We now assume that $x_i \in \mathsf{alph}(w_k)$ and $\overline{x_i} \notin \mathsf{alph}(w_k)$.

There are two sub-cases depending on whether $\overline{x_i} \in \mathsf{alph}(w)$ or not. Assume first that $\overline{x_i} \notin \mathsf{alph}(w)$. Since $w_1 \in (x_i + \overline{x_i})L_{i-1}\$(x_i + \overline{x_i})$, it follows that $w_1 = x_i u \$ x_i$ where $u \in L_{i-1}$. Moreover, recall that $w \in [V]$ by definition which implies that $u \in [V]$. Moreover, $\mathsf{alph}(u)$ contains neither $\overline{x_i}$ nor $\#_i$ (the latter holds by definition of L_{i-1}). Altogether, this yields that $u \in L_{i-1} \cap [V_\top]$ and therefore $u \in K_\top$ by definition of $K_\top$. It follows that $w_1 \in x_i K_\top \$ x_i$ which implies that $w \in \#_i x_i K_\top \$ x_i \#_i (A \setminus \{\overline{x_i}\})^* \subseteq K$ which concludes this case.

Finally, assume that $\overline{x_i} \in \mathsf{alph}(w)$. Therefore, there exists some factor w_j for $j \leq k$ such that $\overline{x_i} \in \mathsf{alph}(w_j)$. We consider the rightmost one. Note that we have $j < k$ by hypothesis

on w_k. By definition, we know that $\overline{x_i} \notin \mathsf{alph}(\#_i w_{j+1} \cdots \#_i w_k \#_i)$. We may now reuse the argument of the previous case to obtain that,

$$\#_i w_{j+1} \cdots \#_i w_k \#_i \in \#_i x_i K_\top \$ x_i \#_i (A \setminus \{\overline{x_i}\})^*$$

Moreover, by definition of w_j, we have $w_j \in (A^* \overline{x_i} A^*) \setminus (A^* \#_i A^*)$. Therefore, we obtain,

$$w \in A^* \#_i ((A^* \overline{x_i} A^*) \setminus (A^* \#_i A^*)) \#_i x_i K_\top \$ x_i \#_i (A \setminus \{\overline{x_i}\})^* \subseteq K$$

This concludes the proof that $L_i \subseteq K$.

It remains to show that $L'_i \cap [V] \cap K = \emptyset$. We proceed by contradiction and assume that there exists $w \in L'_i \cap [V] \cap K$. Recall that by definition, we have

$$\begin{aligned} T_i = (\#_i x_i (B_{i-1} \setminus \{\overline{x_i}\}) \$ x_i)^* \quad \text{and} \quad \overline{T_i} = (\#_i \overline{x_i} (B_{i-1} \setminus \{x_i\}) \$ \overline{x_i})^* \\ L'_i = (\#_i (x_i + \overline{x_i}) L'_{i-1} \$ (x_i + \overline{x_i}))^* \#_i \$ \left(T_i \#_i \cup \overline{T_i} \#_i\right) \end{aligned}$$

Therefore, since $w \in L'_i$, we have $w = u \#_i \$ v \#_i$ with $u \in (\#_i (x_i + \overline{x_i}) L'_{i-1} \$ (x_i + \overline{x_i}))^*$ and $v \in T_i \cup \overline{T_i}$. By symmetry, we shall assume that $v \in T_i$. We obtain that $k, \ell \geq 0$ and $u_1, \ldots, u_k \in (x_i + \overline{x_i}) L'_{i-1} \$ (x_i + \overline{x_i})$ and $v_1, \ldots, v_\ell \in \#_i x_i (B_{i-1} \setminus \{\overline{x_i}\}) \$ x_i$ such that,

$$u = \#_i u_1 \cdots \#_i u_k \quad \text{and} \quad v = \#_i v_1 \cdots \#_i v_\ell$$

Since K is defined as a union, w belongs to some member of this union. We treat each case independently. If $w \in \{\#_i\}$, we have a contradiction since w contains the letter \$ by definition.

Assume now that $w \in A^* \#_i ((A^* x_i A^* \cap A^* \overline{x_i} A^*) \setminus (A^* \#_i A^*)) \#_i$. If $\ell = 0$, this means that $\$ \in (A^* x_i A^* \cap A^* \overline{x_i} A^*) \setminus (A^* \#_i A^*)$ which is a contradiction. Otherwise $\ell \geq 1$ and we obtain that $v_\ell \in (A^* x_i A^* \cap A^* \overline{x_i} A^*) \setminus (A^* \#_i A^*)$. This is also a contradiction since $v_\ell \in \#_i x_i (B_{i-1} \setminus \{\overline{x_i}\}) \$ x_i$ and cannot contain the letter $\overline{x_i}$.

We now treat the case when $w \in \#_i x_i K_\top \$ x_i \#_i (A \setminus \{\overline{x_i}\})^*$. If $k = 0$, this implies that $\$ \in x_i K_\top \$ x_i$ which is a contradiction. Otherwise, we have $u_1 \in x_i K_\top \$ x_i$. Recall that $u_1 \in (x_i + \overline{x_i}) L_{i-1} \$ (x_i + \overline{x_i})$. Therefore, $u_1 \in x_i L'_{i-1} \$ x_i$ which implies that $L'_{i-1} \cap K_\top \neq \emptyset$. Furthermore, since $K_\top \subseteq [V_\top]$ by definition, we get that $L'_{i-1} \cap [V_\top] \cap K_\top \neq \emptyset$. This contradicts the definition of $K_\top$. One may handle the case when $w \in \#_i \overline{x_i} K_\bot \$ \overline{x_i} \#_i (A \setminus \{x_i\})^*$ symmetrically using the definition of $K_\bot$.

We turn to the case when $w \in A^* \#_i ((A^* \overline{x_i} A^*) \setminus (A^* \#_i A^*)) \#_i x_i K_\top \$ x_i \#_i (A \setminus \{\overline{x_i}\})^*$. Since the factors v_j cannot contain the letter $\overline{x_i}$, it follows that there exists $j \leq k$ such that $u_j \in (A^* \overline{x_i} A^*) \setminus (A^* \#_i A^*)$ and,

$$\#_i u_{j+1} \cdots \#_i u_k \#_i \$ v \#_i \in \#_i x_i K_\top \$ x_i \#_i (A \setminus \{\overline{x_i}\})^*$$

One may now reuse the argument of the previous case to derive a contradiction. Finally, one may handle that case when $w \in A^* \#_i ((A^* x_i A^*) \setminus (A^* \#_i A^*)) \#_i \overline{x_i} K_\bot \$ \overline{x_i} \#_i (A \setminus \{x_i\})^*$ symmetrically which concludes the proof.

C.4 Proof of Theorem 16

It is straightforward to verify from Proposition 46 and Theorem 16 that ST[2]-separation is in EXPTime for monoids (since ST[2] is a variety, this is also the case for NFAs by Corollary 6). We focus on proving that ST[2]-separation is PSpace-hard for NFAs (again this is lifted to monoids with Corollary 6). As explained in the main paper, this boils down to proving Proposition 23.

▶ **Proposition 23.** *Consider an alphabet A and $H, H' \subseteq A^*$. Let $B = A \cup \{\#, \$\}$ with $\#, \$ \notin A$, $L = \#(H'\#(A^*\$\#)^*)^*H\#(A^*\$\#)^* \subseteq B^*$ and $L' = \#(H'\#(A^*\$\#)^*)^* \subseteq B^*$. The two following properties are equivalent:*

1. *H is $\mathrm{ST}[3/2]$-separable from H'.*
2. *L is $\mathrm{ST}[2]$-separable from L'.*

We start with the direction 1) $\Rightarrow$ 2). Assume that H is $\mathrm{ST}[3/2]$-separable from H' and let $K \subseteq A^*$ be a separator in $\mathrm{ST}[3/2]$. Consider the following language $S \subseteq B^*$:

$$S = B^*\#K\#B^*$$

Clearly, $S \in \mathrm{ST}[3/2] \subseteq \mathrm{ST}[2]$. Moreover, since $L = \#(H'\#(A^*\$\#)^*)^*H\#(A^*\$\#)^*$ and $H \subseteq K$ by definition of K, we have $L \subseteq S$. Finally, we have $H' \cap K = \emptyset$ by definition of K. Moreover, $L' = \#(H'\#(A^*\$\#)^*)^*$. Since $\#, \$ \notin A$, given $w \in L'$, the only factors of w belonging to $\#A^*\#$ actually belong to $\#H'\#$. Therefore, since $K \subseteq A^*$, we get $L' \cap K = \emptyset$ which concludes the proof for the direction 1) $\Rightarrow$ 2).

We turn to the direction 2) $\Rightarrow$ 1). Actually, we prove the contrapositive. Assuming that H is not $\mathrm{ST}[3/2]$-separable from H', we show that L is not $\mathrm{ST}[2]$-separable from L'. By Lemma 41, we have to show that for every $k \in \mathbb{N}$, there exists $w \in L$ and $w' \in L'$ such that $w \leqslant_k w'$ and $w' \leqslant_k w$. we fix k for the proof.

Since H is not $\mathrm{ST}[3/2]$-separable from H', Lemma 41 yields $u \in H$ and $u' \in H'$ such that $u \leqslant_k u'$. We define,

$$\begin{array}{rcl} w & = & \#(u'\#(u\$\#)^{3^{k+1}})^{3^{k+1}}u\#(u\$\#)^{3^{k+1}} \\ w' & = & \#(u'\#(u\$\#)^{3^{k+1}})^{3^{k+1}} \end{array}$$

Since $u \in H$ and $u' \in H'$, it is clear from the definitions of L and L' that $w \in L$ and $w' \in L'$. It remains to show that $w \leqslant_k w'$ and $w' \leqslant_k w$. We start with the former.

Since $u \leqslant_k u'$, we may use Lemma 42 to obtain the following inequality:

$$w \leqslant_k \#(u'\#(u\$\#)^{3^{k+1}})^{3^{k+1}}u'\#(u\$\#)^{3^{k+1}} = \#(u'\#(u\$\#)^{3^{k+1}})^{3^{k+1}+1}$$

Moreover, it is immediate from Lemma 43 that we have,

$$\#(u'\#(u\$\#)^{3^{k+1}})^{3^{k+1}+1} \leqslant_k w'$$

By transitivity, this yields $w \leqslant_k w'$.

We finish with the converse inequality. Clearly, $\mathsf{alph}(u\#) \subseteq \mathsf{alph}(u\$\#)$. Therefore, Lemma 44 yields that,

$$(u\$\#)^{3^{k+1}} \leqslant_k (u\$\#)^{3^{k+1}}u\#(u\$\#)^{3^{k+1}}$$

We may apply Lemma 42 to obtain,

$$\#(u'\#(u\$\#)^{3^{k+1}})^{3^{k+1}} \leqslant_k \#(u'\#(u\$\#)^{3^{k+1}})^{3^{k+1}}u\#(u\$\#)^{3^{k+1}}$$

This exactly says that $w' \leqslant_k w$, finishing the proof.