Separation for Dot-Depth Two

Thomas Place and Marc Zeitoun
LaBRI, Bordeaux University, Bordeaux, France
Email: {tplace,mz}@labri.fr

***Abstract*—The dot-depth hierarchy of Brzozowski and Cohen is a classification of all first-order definable languages. It rose to prominence following the work of Thomas, who established an exact correspondence with the quantifier alternation hierarchy of first-order logic: each level contains languages that can be defined with a prescribed number of quantifier blocks. One of the most famous open problems in automata theory is to obtain membership algorithms for all levels in this hierarchy.**

For a fixed level, the membership problem asks whether an input regular language belongs to this level. Despite a significant research effort, membership by itself has only been solved for low levels. Recently, a breakthrough was made by replacing membership with a more general problem called *separation*. This problem asks whether, for two input languages, there exists a third language in the investigated level containing the first language and disjoint from the second. The motivation for looking at separation is threefold: (1) while more difficult, it is more rewarding; (2) being more general, it provides a more convenient framework, and (3) all recent membership algorithms are actually reductions to separation for lower levels.

This paper presents a separation algorithm for dot-depth 2. A crucial point is that while dot-depth 2 is our main application, we prove a much more general theorem. Indeed, dot-depth belongs to a family of hierarchies which all share the same generic construction process: starting from an initial class of languages called the basis, one applies generic operations to build new levels. We prove that for any such hierarchy whose basis is a finite class, level 1 has decidable separation. In the special case of dot-depth, this generic result can easily be lifted to level 2.

I. Introduction

Concatenation hierarchies. Many fundamental problems about regular languages [19] led to considerable advances, not only in automata theory but also in logic and algebra, thanks to the discovery of deep connections between these areas that led to the problems' solutions. Even if some of these questions are now well understood, a few others remain wide open, despite a wealth of research work spanning several decades. This is the case for the fascinating dot-depth problem [20], which has two elementary formulations: a language-theoretic one and a logical one. The language-theoretic one is the older of the two. It takes its roots in a theorem of Schützenberger [30] (see also [11], [9]), which gives an algorithm to decide whether a regular language is star-free, *i.e.*, can be expressed using union, complement and concatenation, but without Kleene star. This celebrated result was highly influential for three reasons:

- First, Schützenberger precisely formalized the objective of "understanding the expressive power of a formalism" through a decision problem called *membership*, which asks whether an input language belongs to a class under study.
- Next, he developed a methodology for tackling it, which he applied to membership for the class of star-free languages.
- Finally, McNaughton and Papert [16] established that star-free languages are exactly the first-order definable ones.

This work highlighted the robustness of the notion of regularity, underlining the ties between automata theory and logic, and revealing new links with algebra. It also established membership as the reference problem for investigating classes of languages.

Schützenberger's theorem led Brzozowski and Cohen to define the dot-depth hierarchy [5], an infinite classification of all star-free languages counting the number of alternations between concatenations and complements needed to define them. This definition is a particular instance of a generic construction process, which was formalized later and named *concatenation hierarchies*. Any such hierarchy has a single parameter: a "level 0 class" (its *basis*). Then, one uses two operations, polynomial and Boolean closure, to build two kinds of classes: half levels $\frac{1}{2}, \frac{3}{2}, \frac{5}{2} \dots$ and full levels $1, 2, 3 \dots$. Given a class of languages $\mathcal{C}$, its *polynomial closure* $Pol(\mathcal{C})$ is the smallest class of languages containing $\mathcal{C}$ and closed under union and marked product $K, L \mapsto KaL$, where a is a letter. Its *Boolean closure* $Bool(\mathcal{C})$ is the smallest class containing $\mathcal{C}$ and closed under union and complement. For any full level n, the next half and full levels are built as follows:

- Level $n + \frac{1}{2}$ is the polynomial closure of level n.
- Level $n + 1$ is the Boolean closure of level $n + \frac{1}{2}$.

Thus, a concatenation hierarchy is fully determined by its basis. In the paper, we are interested in hierarchies with a *finite* basis.

The most prominent hierarchies of this kind in the literature are the dot-depth and the Straubing-Thérien hierarchy [33], [35]. They acquired this status when it was discovered [36], [17] that each of them coincides with the quantifier alternation hierarchy within an appropriate variant of first-order logic. These two variants have the same overall expressiveness but slightly different signatures (which impacts the properties that one can define at a given level of their quantifier alternation hierarchies).

These correspondences motivated a research program to solve membership for all levels of both hierarchies, thus also characterizing the alternation hierarchies of first-order logic. However, progress has been slow. The classes that were solved for both variants are only level $\frac{1}{2}$ [2], [22], level 1 [31], [14] and level $\frac{3}{2}$ [2], [22], [12]. See [10] for a survey. Following these results, membership for level 2 remained open for a long time and was named the "dot-depth two problem".

Funded by the DeLTA project (ANR-16-CE40-0007).

978-1-5090-3018-7/17/$31.00 ©2017 European Union

Separation. Recently [25], [23], solutions were found for levels 2, $\frac{5}{2}$ and $\frac{7}{2}$. The key ingredient is a new problem stronger than membership: *separation*, initially introduced in the context of semigroup theory [1]. Rather than asking whether an input language belongs to the class $\mathcal{C}$ under investigation, the $\mathcal{C}$-separation problem takes as input *two* languages, and asks whether there exists a third one *from* $\mathcal{C}$ containing the first and disjoint from the second. While the interest in separation is recent, it has quickly replaced membership as the central question. A first practical reason is that separation proved itself to be a key ingredient in obtaining all recent membership results. See [27] for an overview. A striking example is provided by a crucial theorem of [25]. It establishes a generic reduction from $Pol(\mathcal{C})$-separation to $\mathcal{C}$-membership which holds for any class $\mathcal{C}$. Combined with a separation algorithm for level $\frac{3}{2}$ and a little extra work, this yields a membership algorithm for level $\frac{5}{2}$.

However, the main reason is deeper. The primary motivation for considering such problems is to thoroughly understand the classes under investigation. In this respect, while harder, separation is also far more rewarding than membership. On one hand, a membership algorithm for a class $\mathcal{C}$ only applies to languages of $\mathcal{C}$: it can detect them and build a description witnessing membership. On the other hand, a separation algorithm for $\mathcal{C}$ is universal: it applies to ***any*** language. Indeed, one may view separation as an approximation problem: given an input pair (L_1, L_2) one wants to over-approximate L_1 by a language in $\mathcal{C}$, and L_2 serves to specify what a satisfying approximation is. This is why we look at separation: it yields a more robust understanding of the classes than membership.

The state of the art for separation is the following: it was shown to be decidable for levels $\frac{1}{2}$, 1, $\frac{3}{2}$ and $\frac{5}{2}$ in the Straubing-Thérien hierarchy [8], [24], [25], [23]. These results can be lifted to dot-depth using a generic transfer theorem [26]. Notice the gap between levels $\frac{3}{2}$ and $\frac{5}{2}$: no algorithm is known for level 2. This is explained by the fact that obtaining separation algorithms presents very different challenges for half levels and for full levels. Indeed, it turns out that most separation algorithms rely heavily on closure under marked concatenation, which holds for half levels by definition, but not for full levels.

Contributions. Our main result is a separation algorithm for level 2 in the Straubing-Thérien hierarchy. Furthermore, by the aforementioned transfer theorem [26], this can be lifted to separation for dot-depth 2. A crucial point is that this separation result is actually an instance of a generic theorem, which applies to any finite class $\mathcal{C}$ satisfying a few standard properties (namely closure under Boolean operations and quotients). It states that for such a class $\mathcal{C}$, both $Pol(\mathcal{C})$ and $Bool(Pol(\mathcal{C}))$ have decidable separation. This has two important consequences,

- In *any* hierarchy whose basis is such a class, levels $\frac{1}{2}$ and 1 both have decidable separation.
- In the specific case of the Straubing-Thérien hierarchy, this extends to levels $\frac{3}{2}$ and 2, since they are also levels $\frac{1}{2}$ and 1 in another concatenation hierarchy of finite basis [21].

Being generic, this approach yields separation algorithms for a whole family of classes. Moreover, it serves to pinpoint the key hypotheses which are critical in order to solve separation for dot-depth 2. Let us also stress that we obtain new direct proofs that separation is decidable for the levels 1 in the dot-depth and Straubing-Thérien hierarchies. This is of particular interest for dot-depth 1 since the previous solution was indirect, as it relied on a transfer result from [26]. Moreover, while the separation algorithm for level 2 in the Straubing-Thérien hierarchy relies on a (nontrivial) generalization of the work of [25] for level $\frac{3}{2}$, the arguments are new after this point.

Finally, our approach is amenable to complexity analysis. Due to space limitations, we leave this development for further work. Let us just outline here the main results. When the alphabet is fixed, one obtains a generic PTIME upper bound for both algorithms (*i.e.*, for levels $\frac{1}{2}$ and 1) if the inputs are given by nondeterministic finite automata. When the alphabet is taken into account, this upper bound still holds for levels $\frac{1}{2}$ and 1 of both the Straubing-Thérien and dot-depth hierarchies. While this was known for the former [8], [24], this is a new result for the latter. Finally, for levels $\frac{3}{2}$ and 2 in the Straubing-Thérien hierarchy, we obtain a PSPACE upper bound.

Organization. The paper is organized as follows. Section II gives preliminary definitions. We introduce concatenation hierarchies and state our generic separation theorem in Section III. The remainder of the paper is devoted to its proof. In Sections IV and V, we outline our general approach and introduce the framework we use. Our algorithms for $Pol(\mathcal{C})$- and $Bool(Pol(\mathcal{C}))$-separation (when $\mathcal{C}$ is finite) are given in Sections VI and VII, respectively. Finally, Section VIII presents the main ideas used for proving the correctness of our $Bool(Pol(\mathcal{C}))$-separation algorithm. Due to lack of space, some proofs are postponed to the full version of the paper.

II. PRELIMINARIES

In this section, we provide standard definitions for the objects investigated in the paper and we state our main result.

A. Words, Languages and Classes

For the whole paper, we fix a finite alphabet A. We denote by A^* the set of all words over A, including the empty word ε. We let $A^+ = A^* \setminus \{\varepsilon\}$. If $u, v \in A^*$ are words, we write $u \cdot v$ or uv the word obtained by concatenating u and v.

A subset of A^* is called a *language*. We denote the singleton language $\{u\}$ by u. It is standard to extend the concatenation operation to languages: given $K, L \subseteq A^*$, we denote by KL the language $KL = \{uv \mid u \in K \text{ and } v \in L\}$. Moreover, we will 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 $a \in A$.

A *class of languages* is simply a set of languages. All classes that we consider are included in the class of *regular languages*. These are the languages that can be equivalently defined by monadic second-order logic, finite automata or finite monoids.

B. Separation for Hierarchies of First-Order Languages

Our goal is to investigate classes corresponding to fragments within the quantifier alternation hierarchy of first-order logic. Let us recall these notions.

A word $w \in A^*$ may be viewed as a logical structure made of a linearly ordered sequence of positions. Each position carries a label in A and can be quantified. We denote by "$<$" the (strict) linear order over these positions. We consider first-order logic, denoted by FO($<$), using the following predicates:

- For each $a \in A$, a unary predicate P_a selecting positions labeled with an "a".
- A binary predicate "$<$" for the linear order.

To every first-order sentence φ, one may associate the language $\{w \in A^* \mid w \models \varphi\}$ of words that satisfy φ. Hence, FO($<$) defines a class of languages: the class of all languages that can be defined using an FO($<$) sentence. It is usual to abuse notation and to denote this class by FO($<$) as well.

One may classify FO($<$) sentences by counting their number of quantifier alternations. For $n \in \mathbb{N}$, a sentence is said to be $\Sigma_n(<)$ (resp. $\Pi_n(<)$) if its prenex normal form has either:

- *Exactly* n blocks of quantifiers, the leftmost one being an "$\exists$" (resp. a "$\forall$") block, or
- *Strictly less* than n blocks of quantifiers.

For example, a formula whose prenex normal form is

$$\exists x_1 \exists x_2\ \forall x_3\ \exists x_4\ \varphi(x_1, x_2, x_3, x_4) \quad (\varphi \text{ quantifier-free})$$

is $\Sigma_3(<)$. In general, the negation of a $\Sigma_n(<)$ sentence is not a $\Sigma_n(<)$ sentence (it is $\Pi_n(<)$), and the corresponding classes of languages are not closed under complement. It is therefore relevant to define $\mathcal{B}\Sigma_n(<)$ sentences as Boolean combinations of $\Sigma_n(<)$ and $\Pi_n(<)$ sentences. This gives a strict hierarchy of classes of languages [6] depicted in Figure 1, where again, slightly abusing notation, each level denotes the class of languages defined by the corresponding set of formulas.

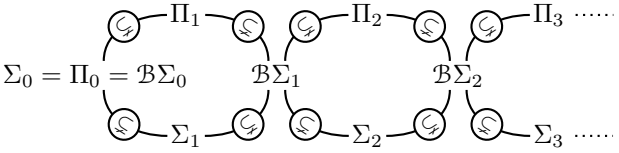

Fig. 1. Quantifier Alternation Hierarchy

As we explained, our objective is to obtain algorithms for the separation problem associated to classes in this hierarchy. We briefly recall it now. Let $\mathcal{C}$ be a class of languages. Given two languages L_0 and L_1, we say that L_0 is $\mathcal{C}$-separable from L_1 when there exists a third language $K \in \mathcal{C}$ such that $L_0 \subseteq K$ and $L_1 \cap K = \emptyset$. In that case, we say that K is a *($\mathcal{C}$-)separator*.

Remark 1. *Observe that when $\mathcal{C}$ is not closed under complement (for example when $\mathcal{C} = \Sigma_n(<)$ where $n \geq 1$), the definition is not symmetrical: L_0 may be $\mathcal{C}$-separable from L_1 even when L_1 is not $\mathcal{C}$-separable from L_0.*

The separation problem for $\mathcal{C}$ is as follows:

INPUT: Two regular languages L_0 and L_1.
OUTPUT: Is L_0 $\mathcal{C}$-separable from L_1?

Separation is a mathematical tool designed to investigate classes of languages: given a fixed class $\mathcal{C}$, obtaining a $\mathcal{C}$-separation algorithm usually requires a solid understanding of $\mathcal{C}$. In particular, a typical objective when considering separation is to not only get an algorithm that decides it, but also a generic method for computing a separator, when it exists.

For the hierarchy, separation algorithms are currently known for FO($<$) itself [28] and for the levels $\Sigma_1(<)$ [8], $\mathcal{B}\Sigma_1(<)$ [8], [24], $\Sigma_2(<)$ [25] and $\Sigma_3(<)$ [23]. As announced, a corollary of our main result is an algorithm for $\mathcal{B}\Sigma_2(<)$.

Theorem 2. *Separation is decidable for $\mathcal{B}\Sigma_2(<)$.*

We actually prove a more general theorem, which establishes that separation is decidable for several classes—$\mathcal{B}\Sigma_2(<)$ being just one among them. We present this theorem in Section III.

Remark 3. *Together with the decidability of separation for $\Sigma_3(<)$ [23], Theorem 2 is the most advanced result of this kind for the hierarchy. While both proofs build on the ideas used in [25] to decide $\Sigma_2(<)$-separation, they do so in orthogonal directions. As a result, they are very different. The reason is that our techniques rely heavily on the concatenation operation, and while $\Sigma_3(<)$ is closed under concatenation, $\mathcal{B}\Sigma_2(<)$ is not.*

Let us finish with an interesting corollary of Theorem 2. It is standard to consider another quantifier alternation hierarchy, called *enriched hierarchy*. It is associated to a variant of first-order logic allowing the following additional predicates:

- "$+1$" interpreted as the successor relation,
- "min" and "max", which are unary and select the leftmost and rightmost positions in a word,
- "ε", which is nullary and holds for the empty word only.

While adding these new predicates does not change FO($<$) as a whole (since they can be defined from "$<$"), this changes the hierarchy, as defining them costs quantifier alternations.

We denote by $\Sigma_n(<, +1)$ and $\mathcal{B}\Sigma_n(<, +1)$ the levels in this enriched hierarchy. The languages in $\mathcal{B}\Sigma_n(<, +1)$ are called the *languages of dot-depth n* (which explains the title of the paper). It is known that for any level of the enriched hierarchy, separation reduces to the same problem for the corresponding level in the first hierarchy [26]. Hence, the decidability of $\mathcal{B}\Sigma_2(<, +1)$-separation is a corollary of Theorem 2.

Corollary 4. *Separation is decidable for $\mathcal{B}\Sigma_2(<, +1)$.*

III. A General Separation Theorem

As explained above, Theorem 2 is a corollary of general statement, which applies to many classes. In this section, we present this result and we connect it to Theorem 2.

A. Statement of the Main Result

Recall that classes we consider ***consist of regular languages only***. This will be understood from now on. Furthermore, our results apply to classes satisfying standard closure properties:

- A *lattice* is a class of languages $\mathcal{C}$ closed under finite union and intersection, and such that $\emptyset \in \mathcal{C}$ and $A^* \in \mathcal{C}$.
- A *Boolean algebra* is a lattice closed under complement.

- We say that a class $\mathcal{C}$ is *quotienting* when it is closed under quotients, *i.e.*, when for all $L \in \mathcal{C}$ and all $w \in A^*$,

$$w^{-1}L \stackrel{\text{def}}{=} \{u \in A^* \mid wu \in L\} \text{ and}$$
$$Lw^{-1} \stackrel{\text{def}}{=} \{u \in A^* \mid uw \in L\}$$

both belong to $\mathcal{C}$.

Our generic theorem states that separation is decidable for any class built from a *finite* quotienting Boolean algebra by applying two operations (at most once for each operation): polynomial and Boolean closure. Given a class $\mathcal{C}$, its *Boolean closure* $Bool(\mathcal{C})$ is the smallest Boolean algebra containing $\mathcal{C}$. Moreover, the *polynomial closure* $Pol(\mathcal{C})$ of $\mathcal{C}$ is the smallest *lattice* containing $\mathcal{C}$ and closed under marked concatenation:

$$\text{for all } K, L \in Pol(\mathcal{C}) \text{ and all } a \in A, \quad KaL \in Pol(\mathcal{C}).$$

We shall write $BPol(\mathcal{C})$ for $Bool(Pol(\mathcal{C}))$. The following easy fact entails that under mild hypotheses, being quotienting is preserved under polynomial and Boolean closure.

Fact 5. *Let $\mathcal{C}$ be a quotienting Boolean algebra. Then,*

1) *$Pol(\mathcal{C})$ is a quotienting lattice and is closed under concatenation and marked concatenation.*
2) *$BPol(\mathcal{C})$ is a quotienting Boolean algebra.*

Observe that in contrast to $Pol(\mathcal{C})$, in general, $BPol(\mathcal{C})$ is ***not*** closed under concatenation. This should be emphasized, as our techniques for solving separation rely on this operation: Boolean closure is usually much harder to deal with than polynomial closure. We can now state our main theorem.

Theorem 6. *Let $\mathcal{C}$ be a finite quotienting Boolean algebra. Then separation is decidable for both $Pol(\mathcal{C})$ and $BPol(\mathcal{C})$.*

The proof of Theorem 6 spans the four remaining sections. Before we present this proof, let us first give a few applications of the theorem (such as the decidability of $\mathcal{B}\Sigma_2(<)$-separation).

B. Applications of the Main Result

Classes of the form $Pol(\mathcal{C})$ and $BPol(\mathcal{C})$ are important as they serve to build natural hierarchies of classes of languages, called *concatenation hierarchies*. Let us briefly recall what they are (see [18], [22], [20], [34] for details). Each such hierarchy depends on a single parameter: a quotienting Boolean algebra of regular languages $\mathcal{C}$, called its *basis*. Once the basis is chosen, the construction is uniform. Languages are classified into levels of two kinds: full levels (denoted by $0, 1, 2, \dots$) and half levels (denoted by $\frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \dots$):

- Level 0 is the basis (*i.e.*, our parameter class $\mathcal{C}$).
- Each *half level* $n+\frac{1}{2}$, for $n \in \mathbb{N}$, is the *polynomial closure* of the previous full level, *i.e.*, of level n.
- Each *full level* $n+1$, for $n \in \mathbb{N}$, is the *Boolean closure* of the previous half level, *i.e.*, of level $n+\frac{1}{2}$.

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

Fig. 2. A concatenation hierarchy

Hence, a reformulation of Theorem 6 is that for any concatenation hierarchy whose basis is *finite*, separation is decidable for its levels $\frac{1}{2}$ and 1. There are two famous examples:

- The *Straubing-Thérien* hierarchy [33], [35], whose basis is the class $\{\emptyset, A^*\}$.
- The *dot-depth* hierarchy of Brzozowski and Cohen [5], whose basis is the class $\{\emptyset, \{\varepsilon\}, A^+, A^*\}$.

It was shown [36], [17] that these two hierarchies correspond respectively to the original and enriched quantifier alternation hierarchies defined in the previous section: level n corresponds to $\mathcal{B}\Sigma_n$ and level $n+\frac{1}{2}$ to Σ_{n+1}. Actually, such correspondences are not coincidental, as stated in the next proposition.

Proposition 7. *For any quotienting Boolean algebra of regular languages $\mathcal{C}$, there is a natural set $S_\mathcal{C}$ of first order predicates, such that the corresponding quantifier alternation levels $\mathcal{B}\Sigma_n(S_\mathcal{C})$ and $\Sigma_{n+1}(S_\mathcal{C})$ are exactly levels n and $n+\frac{1}{2}$, resp., in the concatenation hierarchy of basis $\mathcal{C}$.*

Proof sketch. For a word $w = a_1 a_2 \cdots a_n$ ($n \geq 0$, $a_i \in A$) and $i, j \in \{1, \dots, n\}$, we let $w[i, j]$ be the word $a_i a_{i+1} \cdots a_j$. Likewise, we write $w[i, j[= w[i, j-1]$, $w]i, j] = w[i+1, j]$ and $w]i, j[= w[i+1, j-1]$. For each language $L \in \mathcal{C}$, we define four predicates I_L, P_L, S_L, W_L. Assume that the interpretation of the free variables x, y are $i, j \in \{1, \dots, n\}$, respectively. Then,

- $w \models I_L(x, y)$ when $w]i, j[$ belongs to L.
- $w \models P_L(y)$ when $w[1, j[$ belongs to L.
- $w \models S_L(x)$ when $w]i, n]$ belongs to L.
- $w \models W_L$ when w belongs to L.

In this notation, I stands for infix, P for prefix, S for suffix and W for word. We denote by $S_\mathcal{C}$ the signature consisting of the predicates I_L, P_L, S_L, W_L ($L \in \mathcal{C}$) in addition to $<$ and P_a ($a \in A$). Using arguments similar to the ones of [36], one can verify that levels n and $n+\frac{1}{2}$ in the concatenation hierarchy of basis $\mathcal{C}$ correspond respectively to the fragments $\mathcal{B}\Sigma_n(S_\mathcal{C})$ and $\Sigma_{n+1}(S_\mathcal{C})$ of first-order logic over this signature. □

Theorem 6 already provides a new proof for the decidability of $\mathcal{B}\Sigma_1$-separation, and in particular a new self-contained proof of separation for $\mathcal{B}\Sigma_1(<, +1)$, thus subsuming the nontrivial decidability result for membership of Knast [14]. Regarding the hierarchies, it actually proves more, as $\mathcal{B}\Sigma_2(<)$ is also level 1 in another concatenation hierarchy whose basis is finite: let us denote by AT the class of languages consisting of all Boolean combinations of languages A^*aA^*, for some $a \in A$.

Remark 8. *Though this terminology is not standard, "*AT*" stands for "alphabet testable": $L \in$ AT iff membership of a word w in L depends only on the set of letters occurring in w.*

The following lemma is a direct consequence of a result of [21].

Lemma 9. *$\Sigma_2(<) = Pol(\text{AT})$ and $\mathcal{B}\Sigma_2(<) = BPol(\text{AT})$.*

It is easy to verify that AT is a finite quotienting Boolean algebra. Hence, Theorem 2 is now an immediate consequence of Lemma 9 and Theorem 6: $\mathcal{B}\Sigma_2(<)$-separation is decidable.

Remark 10. *We also obtain from Lemma 9 that $\Sigma_2(<)$-separation is decidable, which reproves a result of [25]. In fact, the proof that $Pol(\mathcal{C})$-separation is decidable is a (nontrivial) generalization of the corresponding result in [25] for $\Sigma_2(<)$. Moreover, since membership reduces to separation, we obtain a proof for $\mathcal{B}\Sigma_2(<)$-membership based on new arguments.*

IV. Separation for Boolean Closures

We now start the proof of Theorem 6. An important remark is that we focus on $BPol(\mathcal{C})$-separation: the algorithm for $Pol(\mathcal{C})$ is obtained as a byproduct. In this section we devise a general approach to $Bool(\mathcal{D})$-separation when $\mathcal{D}$ is an arbitrary lattice. We will develop it when $\mathcal{D} = Pol(\mathcal{C})$ in Sections VI to VIII. The crux of this approach is to reduce $Bool(\mathcal{D})$-separation to another decision problem (harder than separation, unfortunately) for the simpler class $\mathcal{D}$. We first present this reduction and then explain how to tackle this new decision problem.

A. Generalized Separation

Let us fix a lattice $\mathcal{D}$. Given an integer $n \geq 1$, a tuple $(L_1, \ldots, L_n)$ of n languages is called a n-tuple. We define a generalized notion of $\mathcal{D}$-separation that applies to n-tuples (the case $n = 2$ boils down to classical separation). Our goal is to connect $Bool(\mathcal{D})$-separation to this generalized notion for $\mathcal{D}$.

We start with a notation. Given an integer $n \geq 1$, an n-tuple $(H_1, \ldots, H_n)$ and a single language K, we write:

$$(H_1, \ldots, H_n) \cap K \stackrel{\text{def}}{=} (H_1 \cap K, \ldots, H_n \cap K).$$

We now generalize $\mathcal{D}$-separation to n-tuples. Let $(L_1, \ldots, L_n)$ be an n-tuple. We use induction on n to define whether or not $(L_1, \ldots, L_n)$ is $\mathcal{D}$-separable.

- If $n = 1$, (L_1) is $\mathcal{D}$-separable when $L_1 = \emptyset$.
- If $n \geq 2$, $(L_1, \ldots, L_n)$ is $\mathcal{D}$-separable when there exists $K \in \mathcal{D}$ such that $L_1 \subseteq K$ and $(L_2, \ldots, L_n) \cap K$ is $\mathcal{D}$-separable. We call K a *($\mathcal{D}$-)separator* of $(L_1, \ldots, L_n)$.

Remark 11. *For $n = 2$, we recover the classical notion: (L_1, L_2) is $\mathcal{D}$-separable iff there exists $K \in \mathcal{D}$ such that $L_1 \subseteq K$ and $L_2 \cap K = \emptyset$.*

This generalized notion makes sense for any lattice $\mathcal{D}$. However, it is tailored to be used with those that are not closed under complement as a way to investigate $Bool(\mathcal{D})$-separation. Let us explain this reduction. We start with a simple observation: long n-tuples are "easier" to separate than short ones.

Lemma 12. *Let $n \geq m \geq 1$, let $(L_1, \ldots, L_n)$ be an n-tuple and $i_1, \ldots, i_m \in \mathbb{N}$ such that $1 \leq i_1 < i_2 < \cdots < i_m \leq n$. If $(L_{i_1}, \ldots, L_{i_m})$ is $\mathcal{D}$-separable, then so is $(L_1, \ldots, L_n)$.*

Given two languages H, L and $p \geq 1$, we write $(H, L)^p$ for the $2p$-tuple built by concatenating p copies of (H, L) (for example $(H, L)^3 = (H, L, H, L, H, L)$). A particular consequence of Lemma 12 is that for all p, the tuple $(H, L)^{p+1}$ is "more likely" to be $\mathcal{D}$-separable than $(H, L)^p$. Hence, a natural problem is to ask whether there exists $p \geq 1$ such that $(H, L)^p$ is $\mathcal{D}$-separable. It turns out that this problem is equivalent to $Bool(\mathcal{D})$-separation.

Theorem 13. *Let $\mathcal{D}$ be a lattice and let L_1, L_2 be two languages. The following properties are equivalent:*

1) *L_1 is $Bool(\mathcal{D})$-separable from L_2.*
2) *There exists $p \geq 1$ such that $(L_1, L_2)^p$ is $\mathcal{D}$-separable.*

Theorem 13 applied to $\mathcal{D} = Pol(\mathcal{C})$ provides the intended reduction: for a finite quotienting Boolean algebra $\mathcal{C}$, we are now faced with a problem on $Pol(\mathcal{C})$ instead of $BPol(\mathcal{C})$-separation. Let us prove the direction $2) \Rightarrow 1)$ (the converse is postponed to the full version of the paper). Our approach is constructive: one combines separators in $\mathcal{D}$ witnessing that $(L_1, L_2)^p$ is $\mathcal{D}$-separable for some p into a $Bool(\mathcal{D})$-separator of L_1 and L_2.

Proof. Assume that $(L_1, L_2)^p$ is $\mathcal{D}$-separable for some $p \geq 1$. Using induction on p, we prove that L_1 is $Bool(\mathcal{D})$-separable from L_2. When $p = 1$, L_1 is $\mathcal{D}$-separable from L_2 and since $\mathcal{D} \subseteq Bool(\mathcal{D})$, the result is trivial. Assume that $p \geq 2$. By definition, we have $K, K' \in \mathcal{D}$ such that $L_1 \subseteq K$, $L_2 \cap K \subseteq K'$ and $(L_1, L_2)^{p-1} \cap K \cap K'$ is $\mathcal{D}$-separable. Using induction, we then obtain a language $G \in Bool(\mathcal{D})$ separating $L_1 \cap K \cap K'$ from $L_2 \cap K \cap K'$. Consider the following language:

$$H = (K \cap G) \cup (K \setminus K').$$

Clearly, $H \in Bool(\mathcal{D})$. We prove that H separates L_1 from L_2.

We begin with $L_1 \subseteq H$. Let $w \in L_1$, we prove that $w \in H$. Clearly, $w \in K$ since $L_1 \subseteq K$. Moreover, either $w \in K'$ and therefore $w \in K \cap G$ since $L_1 \cap K \cap K' \subseteq G$, or $w \notin K'$ and therefore $w \in K \setminus K'$. Altogether, we conclude that $w \in H$.

It remains to prove that $L_2 \cap H = \emptyset$. Let $w \in L_2$, we prove that $w \notin H$. There are two cases depending on whether $w \in K$. If $w \notin K$, then clearly $w \notin K \cap G$ and $w \notin K \setminus K'$, hence $w \notin H$. Otherwise, $w \in L_2 \cap K \subseteq K'$. Therefore, $w \notin K \setminus K'$ and $w \notin K \cap G$ since $L_2 \cap K \cap K' \cap G = \emptyset$ by the choice of G. We conclude that $w \notin H$, which terminates the proof. □

Now that $Bool(\mathcal{D})$-separation is reduced to a new problem on the more manageable class $\mathcal{D}$, we present our approach.

B. Non-separability and Alternating Pairs

Theorem 13 gives a means to tackle $BPol(\mathcal{C})$-separation by reducing it to generalized separation for the simpler class $Pol(\mathcal{C})$. We now describe our strategy to solve this new problem.

$\mathcal{D}$-tied n-tuples. A first key point is that we shall actually work with the *complement* of separation: we try to decide whether languages are ***not*** $BPol(\mathcal{C})$-separable. We say that:

- An n-tuple $(L_1, \ldots, L_n)$ is *$\mathcal{D}$-tied* if it is ***not*** $\mathcal{D}$-separable.
- A pair (L_1, L_2) is *$\mathcal{D}$-alternating* if $(L_1, L_2)^p$ is $\mathcal{D}$-tied for all $p \geq 1$.

Theorem 13 can then be reformulated as follows.

Corollary 14. *Let $\mathcal{D}$ be a lattice and let L_1, L_2 be two languages. The following properties are equivalent:*

1) *L_1 is **not** $Bool(\mathcal{D})$-separable from L_2.*
2) *(L_1, L_2) is $\mathcal{D}$-alternating.*

This new viewpoint is just symmetrical with the previous one. However, it has the advantage that $\mathcal{D}$-tied n-tuples have better properties than $\mathcal{D}$-separable n-tuples (see Section VI). This is our motivation for now focusing on Item 2 of Corollary 14.

From input pairs to input multisets. A second key point is that to decide whether two languages L_1, L_2 are $Bool(\mathcal{D})$-separable, one never deals with L_1, L_2 alone. Instead, we work with a finite ***multiset of languages*** $\mathbf{L}$, built from L_1, L_2.

Remark 15. *We speak of "multiset" of languages to allow several copies of the same language in* $\mathbf{L}$*. This is needed in practice, see for example the languages* $L_{q,r}$ *below.*

Typically, if $\mathcal{A}$ is an automaton recognizing both L_1 and L_2 (thanks to two sets of accepting states), then we solve separation for *all* pairs of languages of the form $L_{q,r} = \{w \mid q \xrightarrow{w} r\}$, for q, r states of $\mathcal{A}$. The benefit of doing so is that this multiset of languages has an algebraic structure, upon which our separation algorithms rely crucially. Therefore, we now consider as input a whole ***finite multiset*** of languages $\mathbf{L}$.

Motivated by these two key points, given a finite multiset of languages $\mathbf{L}$ and an integer $n \geq 1$, we introduce:

- $\mathcal{T}^n_{\mathcal{D}}[\mathbf{L}]$, the set of all n-tuples in $\mathbf{L}^n$ which are $\mathcal{D}$-tied.
- $\mathcal{A}_{\mathcal{D}}[\mathbf{L}]$, the set of all pairs in $\mathbf{L}^2$ which are $\mathcal{D}$-alternating.

Our new goal is now to solve this problem for $\mathcal{D} = Pol(\mathcal{C})$:

INPUT: A finite multiset of languages $\mathbf{L}$.
OUTPUT: The set $\mathcal{A}_{\mathcal{D}}[\mathbf{L}]$.

Indeed, by Corollary 14 applied to $\mathcal{D} = Pol(\mathcal{C})$, this is equivalent to computing pairs that are not $BPol(\mathcal{C})$-separable.

C. Computing Alternating Pairs: Proof Outline

To compute the set of all $Pol(\mathcal{C})$-alternating pairs from a multiset $\mathbf{L}$ of languages, we design an algorithm in two steps.

Step 1: computing $Pol(\mathcal{C})$-tied tuples. We first obtain an algorithm solving the following simpler problem:

INPUT: A finite multiset of languages $\mathbf{L}$ and $n \geq 1$.
OUTPUT: The set $\mathcal{T}^n_{Pol(\mathcal{C})}[\mathbf{L}]$.

Remark 16. *The correctness proof is constructive in the following sense. When a tuple* $(L_1, \ldots, L_n)$ *is not computed (*i.e.*, is* $Pol(\mathcal{C})$*-separable), we have a generic method for computing a separator in* $Pol(\mathcal{C})$*.*

Remark 17. *The special case* $n = 2$ *is a* $Pol(\mathcal{C})$*-separation algorithm. Thus, we get the first part of Theorem 6. Moreover, by the previous remark, we also get a generic method for building separators in* $Pol(\mathcal{C})$*, when they exist.*

Remark 18. *By Theorem 13, this gives a semi-algorithm for* $BPol(\mathcal{C})$*-separation as well as a generic method for computing separators in* $BPol(\mathcal{C})$*. If there exists* $p \geq 1$ *such that* $(L_1, L_2)^p$ *is* $Pol(\mathcal{C})$*-separable (*i.e.*, not* $Pol(\mathcal{C})$*-tied), one may find it with the above algorithm and compute a separator in* $Pol(\mathcal{C})$*. One may then use induction to build a separator of* L_1, L_2 *in* $BPol(\mathcal{C})$ *by following the proof of Theorem 13.*

A point that will be crucial for Step 2 is that our algorithm is recursive: for $n \geq 2$, computing the $Pol(\mathcal{C})$-tied tuples in $\mathbf{L}^n$ requires us to have computed those in $\mathbf{L}^{n-1}$ beforehand (the case $n = 1$ is simple, as (L) is $Pol(\mathcal{C})$-tied iff $L \neq \emptyset$).

Each induction step is based on a *least fixpoint procedure*. For $n \geq 2$, we compute the set of all $Pol(\mathcal{C})$-tied tuples in $\mathbf{L}^n$, first starting from a subset of "trivial" ones, and using operations to add new ones until a fixpoint is reached. Again, implementing one of these operations requires us to know all $Pol(\mathcal{C})$-tied tuples in $\mathbf{L}^{n-1}$ (this is where we use recursion).

Remark 19. *While we only aim at computing* $Pol(\mathcal{C})$*-tied tuples, our fixpoint computation requires us to compute more information. This is not surprising as it happens in many separation algorithms, as for* FO$(<)$ *[28] or for* $\Sigma_2(<)$ *[25].*

Remark 20. *While the presentation is very different and the generalization is nontrivial, the ideas used for this step are built upon those used in [25] for solving* $\Sigma_2(<)$*-separation (which is a special case:* $\Sigma_2(<) = Pol(\mathrm{AT})$ *by Lemma 9).*

Step 2: computing $Pol(\mathcal{C})$-alternating pairs. We present our final algorithm, which computes all $Pol(\mathcal{C})$-alternating pairs in an input multiset. By Corollary 14, this solves $BPol(\mathcal{C})$-separation. The correctness of this procedure relies on a difficult analysis of the least fixpoint procedure of Step 1. However, the algorithm itself is simple, let us now describe it.

Let $\mathcal{D} = Pol(\mathcal{C})$. Recall that (L_1, L_2) being $\mathcal{D}$-alternating means $(L_1, L_2)^p$ being $\mathcal{D}$-tied for every $p \geq 1$. In particular, a $\mathcal{D}$-alternating pair is a $\mathcal{D}$-tied 2-tuple: $\mathcal{A}_{\mathcal{D}}[\mathbf{L}] \subseteq \mathcal{T}^2_{\mathcal{D}}[\mathbf{L}]$. The algorithm relies on a greatest fixpoint: it computes a decreasing sequence of finite sets,

$$\mathcal{T}^2_{\mathcal{D}}[\mathbf{L}] = \mathbf{T}_0 \supseteq \mathbf{T}_1 \supseteq \mathbf{T}_2 \supseteq \cdots,$$

until a fixpoint is reached: $\mathbf{T}_p = \mathbf{T}_{p+1}$ entails $\mathcal{A}_{\mathcal{D}}[\mathbf{L}] = \mathbf{T}_p$.

Since $\mathcal{T}^2_{\mathcal{D}}[\mathbf{L}]$ can be computed from Step 1, it remains to explain how to compute $\mathbf{T}_{i+1}$ from $\mathbf{T}_i$. We rely again on the algorithm of Step 1, which computes inductively the set of $\mathcal{D}$-tied n-tuples in $\mathbf{L}^n$ from the set of $\mathcal{D}$-tied $(n-1)$-tuples obtained at the previous step. In particular, to compute the set $\mathcal{T}^3_{\mathcal{D}}[\mathbf{L}]$ of $\mathcal{D}$-tied 3-tuples, the algorithm is fed with $\mathcal{T}^2_{\mathcal{D}}[\mathbf{L}]$. To compute $\mathbf{T}_{i+1}$ from $\mathbf{T}_i$, we feed the very same algorithm with $\mathbf{T}_i \subseteq \mathcal{T}^2_{\mathcal{D}}[\mathbf{L}]$ instead of $\mathcal{T}^2_{\mathcal{D}}[\mathbf{L}]$. This produces a set of 3-tuples. At last, we define $\mathbf{T}_{i+1}$ as the set of pairs (L_1, L_2) of $\mathbf{T}_i$ such that (L_1, L_2, L_1) appears as one of these produced 3-tuples.

V. Reduction to Tame and $\mathcal{C}$-compatible Inputs

Our general approach to separation decisively exploits the fact that our input languages are *regular*. Here, we explain what we gain from this assumption. Recall from the previous section that in order to decide a separation problem, one actually deals with a whole ***multiset*** *of languages* $\mathbf{L}$ built from the inputs L_1, L_2 of our original separation question. One then solves separability for *all* pairs of languages in $\mathbf{L} \times \mathbf{L}$. The special algebraic structure of $\mathbf{L}$ is what we gain from this generalization, and it is crucial for our algorithms to work. In this section, we present the structural properties of these sets.

Our algorithms are restricted to input multisets having *two* special properties, of different importance. The first one appears

in most separation algorithms and was introduced in [29]: our inputs must be *tame*. On the other hand, the second one is specific to the classes that we consider (*i.e.*, $Pol(\mathcal{C})$ and $BPol(\mathcal{C})$ for some finite $\mathcal{C}$): our inputs must be $\mathcal{C}$-*compatible*.

We present these properties and explain why we may restrict our algorithms to sets fulfilling them without loss of generality.

A. Tame Multisets

A multiset of languages is said to be *tame* when it has a *partial semigroup structure*. Let us first define *partial semigroups*. These are sets S equipped with a partial multiplication (*i.e.*, st may not be defined for some $s, t \in S$). Moreover, for all $r, s, t \in S$, the three following conditions must be equivalent: a) rs and st are defined, b) $(rs)t$ is defined and c) $r(st)$ is defined. In that case, $(rs)t = r(st)$. Additionally, an *idempotent* is any element e of S such that ee is defined and equal to e. It is folklore that for any finite partial semigroup there exists a number $\omega(S)$ (denoted ω when S is understood) such for any $s \in S$, if ss is defined, then s^ω is idempotent.

Let $\mathbf{L}$ be a finite multiset of languages. A *tame multiplication* for $\mathbf{L}$ is a partial semigroup multiplication "$\odot$" over $\mathbf{L}$ (we use this notation to avoid confusion with language concatenation) satisfying the following properties:

(1) For all $L, L' \in \mathbf{L}$, if $L \odot L'$ is defined, then $LL' \subseteq L \odot L'$.
(2) For all $H \in \mathbf{L}$ and all words $w \in H$, if w can be decomposed as $w = uu'$, then there exist $L, L' \in \mathbf{L}$ such that $u \in L$, $u' \in L'$ and $H = L \odot L'$.

The multiset $\mathbf{L}$ is *tame* when it can be equipped with a *tame multiplication*. When working with tame multisets, we implicitly assume that we have the multiplication "$\odot$" in hand. The notion is designed to capture the following typical example.

Example 21. *Given a nondeterministic finite automaton $\mathcal{A} = (Q, I, F, \delta)$, let $\mathbf{L}_\mathcal{A} = \{L_{q,r} \mid (q, r) \in Q^2\}$, where $L_{q,r}$ denotes the language $\{w \mid q \xrightarrow{w} r\}$. Note that this is a multiset: if $(q, r) \neq (q', r')$, we count $L_{q,r}$ and $L_{q',r'}$ as two elements in $\mathbf{L}_\mathcal{A}$, even if they are the same language. This multiset is tame for the following multiplication: for $q, r, s, t \in Q$, $L_{q,r} \odot L_{s,t}$ is undefined if $r \neq s$ and equal to $L_{q,t}$ if $r = s$.*

B. $\mathcal{C}$-compatible Multisets

We further restrict our algorithms to $\mathcal{C}$-compatible inputs. This notion depends on an arbitrary finite quotienting Boolean algebra $\mathcal{C}$ that we fix for the definition. Contrary to tameness, this notion is specific to separation for $Pol(\mathcal{C})$ and $BPol(\mathcal{C})$.

First note that one may associate a canonical equivalence $\sim_\mathcal{C}$ over A^* to $\mathcal{C}$: two words are equivalent when they belong to the same languages in $\mathcal{C}$. Given $w, w' \in A^*$,

$$w \sim_\mathcal{C} w' \quad \text{if and only if} \quad \forall L \in \mathcal{C},\ w \in L \Leftrightarrow w' \in L.$$

Given a word w, we denote by $[w]_\mathcal{C}$ its $\sim_\mathcal{C}$-equivalence class.

Example 22. *When $\mathcal{C}$ is the class* AT *of alphabet testable languages, $w \sim_{\text{AT}} w'$ when w and w' have the same alphabet (the alphabet of w is the smallest $B \subseteq A$ such that $w \in B^*$). Hence, in that case, $[w]_{\text{AT}}$ corresponds to the alphabet of w.*

Because $\mathcal{C}$ is both finite and a quotienting Boolean algebra, the equivalence $\sim_\mathcal{C}$ has the following convenient properties.

Lemma 23. *The equivalence $\sim_\mathcal{C}$ has finite index and the languages of $\mathcal{C}$ are exactly the unions of equivalence classes. Moreover, $\sim_\mathcal{C}$ is a congruence for the concatenation operation (if $u \sim_\mathcal{C} u'$ and $v \sim_\mathcal{C} v'$ then $uv \sim_\mathcal{C} u'v'$).*

We are now ready to define $\mathcal{C}$-compatibility. Let $\mathbf{L}$ be any finite multiset of languages. We say that $\mathbf{L}$ *is* $\mathcal{C}$-*compatible* when all $L \in \mathbf{L}$ satisfy the two following conditions:

1) L is non-empty.
2) L is included in some equivalence class of $\sim_\mathcal{C}$.

Observe that if $\mathbf{L}$ is $\mathcal{C}$-compatible, then given any $L \in \mathbf{L}$, one can define the $\mathcal{C}$-*type* of L as the unique equivalence class containing L, denoted by $[L]_\mathcal{C}$. That is, $[L]_\mathcal{C} = [w]_\mathcal{C}$ for any word $w \in L$. Notice that in particular, $L \subseteq [L]_\mathcal{C} \in \mathcal{C}$.

C. Reduction to Tame and $\mathcal{C}$-Compatible Multisets

While our separation algorithms are restricted to inputs that are both tame and $\mathcal{C}$-compatible, it turns out that we may always reduce the general case to this one. The reason stems from the notion of *extension*, introduced in [29]. Given two multisets of languages $\mathbf{H}$ and $\mathbf{L}$, we say that $\mathbf{H}$ *extends* $\mathbf{L}$ when any language in $\mathbf{L}$ is a union of languages in $\mathbf{H}$: for any language $L \in \mathbf{L}$, there exists $\mathbf{H}' \subseteq \mathbf{H}$ with $L = \bigcup_{H \in \mathbf{H}'} H$.

Lemma 24. *Let $\mathcal{D}$ be a lattice and let $\mathbf{H}, \mathbf{L}$ be two multisets of languages. Assume that $\mathbf{H}$ extends $\mathbf{L}$. Then, for any $L_1, L_2 \in \mathbf{L}$, the following are equivalent:*

1) L_1 is $\mathcal{D}$-separable from L_2.
2) For all $H_1, H_2 \in \mathbf{H}$ such that $H_1 \subseteq L_1$ and $H_2 \subseteq L_2$, H_1 is $\mathcal{D}$-separable from H_2.

Lemma 25. *Given as input a finite multiset of regular languages $\mathbf{L}$, one may construct a finite tame and $\mathcal{C}$-compatible multiset $\mathbf{H}$ extending $\mathbf{L}$.*

Lemmas 24 and 25 are proved similarly as results of [29] (the reduction to a tame set is essentially given in Example 21).

Remark 26. *The construction in Lemma 25 entails a polynomial blow-up in size. Consider $\mathbf{L} = \{L_1, \ldots, L_n\}$ and assume that each L_i is recognized by a nondeterministic finite automaton $\mathcal{A}_i$. Then, the size of the tame and $\mathcal{C}$-compatible multiset $\mathbf{H}$ extending $\mathbf{L}$ is bounded by $(|\mathcal{A}_1|^2 + \cdots + |\mathcal{A}_n|^2) \times |\mathcal{C}|$ (where $|\mathcal{A}_i|$ stands for the number of states in $\mathcal{A}_i$).*

In view of Lemmas 24 and 25, we may restrict ourselves to tame and $\mathcal{C}$-compatible multisets without loss of generality. Indeed, in order to decide whether L_1 is $\mathcal{D}$-separable from L_2 when L_1, L_2 are regular, one may proceed as follows:

1) Build a tame multiset $\mathbf{H}$ extending $\{L_1, L_2\}$ (Lemma 25).
2) Decide $\mathcal{D}$-separability for *all* pairs of languages in $\mathbf{H} \times \mathbf{H}$.
3) The answer for L_1 and L_2 is then given by Lemma 24.

Therefore, we may now assume without loss of generality that our input is a tame and $\mathcal{C}$-compatible multiset $\mathbf{L}$ and that our objective is to get an algorithm that decides generalized $Pol(\mathcal{C})$-separability for *all* pairs in $\mathbf{L} \times \mathbf{L}$.

VI. Solving Generalized Separation for $Pol(\mathcal{C})$

This section is devoted to the first step in our quest for a $BPol(\mathcal{C})$-separation algorithm. Given a fixed finite quotienting Boolean algebra $\mathcal{C}$, our objective is to present an algorithm taking as input a tame and $\mathcal{C}$-compatible multiset $\mathbf{L}$ together with $n \geq 1$, and that outputs the set $\mathcal{T}^n_{Pol(\mathcal{C})}[\mathbf{L}]$. In particular, the case $n = 2$ solves $Pol(\mathcal{C})$-separation.

As explained in Remark 19, we need to compute some extra information in order to carry out this computation. Given an arbitrary lattice $\mathcal{D}$, we first present the notion of $\mathcal{D}$-*tied* n-*join*, which precisely encodes the needed information. Then, we describe our algorithm which computes all $\mathcal{D}$-tied n-joins, when $\mathcal{D} = Pol(\mathcal{C})$.

A. $\mathcal{D}$-tied Joins

Let us fix an arbitrary lattice $\mathcal{D}$. As explained, in order to inductively compute the set $\mathcal{T}^n_{\mathcal{D}}[\mathbf{L}]$ of $\mathcal{D}$-tied n-tuples when $\mathcal{D} = Pol(\mathcal{C})$, we actually need to compute more information. To describe this information, we lift the notion of $\mathcal{D}$-separability to objects called n-joins. An n-*join* is a pair $(H, \mathbf{H})$ where H is a language and $\mathbf{H}$ is a finite ***set*** of n-tuples.

We say that an n-join $(H, \mathbf{H})$ is $\mathcal{D}$-*separable* when there exists a *finite* set of languages $\mathbf{K} \subseteq \mathcal{D}$ such that:

- $H \subseteq \bigcup_{K \in \mathbf{K}} K$
- For all $K \in \mathbf{K}$, there exists $(H_1, \ldots, H_n) \in \mathbf{H}$ such that $(H_1, \ldots, H_n) \cap K$ is $\mathcal{D}$-separable.

Remark 27. *Let us mention that this definition is inspired by that of another problem: pointed covering, introduced in [29].*

We say that $\mathbf{K}$ is a *separating cover* of $(H, \mathbf{H})$. We first explain the connection with separability for n-tuples. It corresponds to the case where the set $\mathbf{H}$ is a singleton $\{(H_1, \ldots, H_n)\}$.

Lemma 28. *Let $n \geq 1$ and consider an $(n+1)$-tuple $(L_1, \ldots, L_{n+1})$. The two following properties are equivalent:*

1) *The $(n+1)$-tuple $(L_1, \ldots, L_{n+1})$ is $\mathcal{D}$-separable.*
2) *The n-join $(L_1, \{(L_2, \ldots, L_{n+1})\})$ is $\mathcal{D}$-separable.*

Proof. If $(L_1, \ldots, L_{n+1})$ is $\mathcal{D}$-separable, then there is a separator $K \in \mathcal{D}$. One may verify that $(L_1, \{(L_2, \cdots, L_{n+1})\})$ is $\mathcal{D}$-separable for the separating cover $\mathbf{K} = \{K\} \subseteq \mathcal{D}$.

Conversely, assume that $(L_1, \{(L_2, \cdots, L_{n+1})\})$ is $\mathcal{D}$-separable and let $\mathbf{K} = \{K_1, \ldots, K_m\} \subseteq \mathcal{D}$ be the separating cover. One may verify that $K = K_1 \cup \cdots \cup K_m \in \mathcal{D}$ is a separator of $(L_1, \ldots, L_{n+1})$, which terminates the proof. □

As for n-tuples, we mainly work with n-joins that are ***not*** $\mathcal{D}$-separable, which we call the $\mathcal{D}$-*tied n-joins*. Moreover, if $\mathbf{L}$ is a finite multiset of languages and $n \geq 1$, we write $\mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$ the set of all $\mathcal{D}$-tied n-joins in $\mathbf{L} \times 2^{\mathbf{L}^n}$.

Let us briefly recall our motivation for introducing n-joins. What we want in *Step 1* of our quest for a $BPol(\mathcal{C})$-separation algorithm is a least fixpoint procedure computing the set $\mathcal{T}^{n+1}_{\mathcal{D}}[\mathbf{L}]$ for every $n \geq 1$. By Lemma 28, one may view this set as a subset of $\mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$. More precisely, $\mathcal{T}^{n+1}_{\mathcal{D}}[\mathbf{L}]$ is the set of all $(n+1)$-tuples $(L_1, \ldots, L_{n+1}) \in \mathbf{L}^{n+1}$ such that $(L_1, \{(L_2, \cdots, L_{n+1})\}) \in \mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$. It turns out that our fixpoint computation requires us to compute the whole set $\mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$. Even though we are only interested in computing a strict subset of it, namely $\mathcal{T}^{n+1}_{\mathcal{D}}[\mathbf{L}]$, the extra elements of $\mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$ may be required as intermediaries in this computation.

We finish the section by presenting properties of the set $\mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$ which are crucial in the algorithm. Two are generic to all lattices $\mathcal{D}$, one requires $\mathcal{D}$ to be a quotienting lattice, and the final one is specific to the case $\mathcal{D} = Pol(\mathcal{C})$.

The first property states that there are always "trivial" n-joins. Given $n \geq 1$, we define $\mathcal{J}^n_{triv}[\mathbf{L}] \subseteq \mathbf{L} \times 2^{\mathbf{L}^n}$ as the set of all n-joins $(S, \mathbf{S})$ such that the intersection of all languages in $(S, \mathbf{S})$ is nonempty. More precisely, $(S, \mathbf{S}) \in \mathcal{J}^n_{triv}[\mathbf{L}]$ when there exists $w \in A^*$ such that (1) the word w belongs to S, and (2) for all $(S_1, \ldots, S_n) \in \mathbf{S}$, the word w belongs to $\bigcap_i S_i$.

Lemma 29. *Let $n \geq 1$ and $\mathbf{L}$ be a finite multiset of languages. Then, $\mathcal{J}^n_{triv}[\mathbf{L}] \subseteq \mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$.*

Proof. Let $(S, \mathbf{S}) \in \mathcal{J}^n_{triv}[\mathbf{L}]$. We prove that $(S, \mathbf{S})$ is $\mathcal{D}$-tied. Consider $\mathbf{K} \subseteq \mathcal{D}$ such that $S \subseteq \bigcup_{K \in \mathbf{K}} K$, we have to find $K \in \mathbf{K}$ such that $(S_1, \ldots, S_n) \cap K$ is $\mathcal{D}$-tied for all $(S_1, \ldots, S_n) \in \mathbf{S}$. Let w be as defined above for $(S, \mathbf{S})$. Since $w \in S$, we have $w \in K$ for some $K \in \mathbf{K}$. Now, for all $(S_1, \ldots, S_n) \in \mathbf{S}$, we have $w \in \bigcap_{1 \leq i \leq n} (S_i \cap K)$. This entails that the n-tuple $(S_1, \ldots, S_n) \cap K$ is $\mathcal{D}$-tied. □

Another property is closure under *downset*. Let $n \geq 1$ and $(S, \mathbf{S})$ and $(T, \mathbf{T})$ be two n-joins. We write $(S, \mathbf{S}) \subseteq (T, \mathbf{T})$ when $S = T$ and $\mathbf{S} \subseteq \mathbf{T}$. Furthermore, if $\mathbf{L}$ is a finite multiset of languages and $\mathcal{S}$ is a subset of $\mathbf{L} \times 2^{\mathbf{L}^n}$, the downset of $\mathcal{S}$, denoted by $\downarrow \mathcal{S}$, is the following set:

$$\downarrow \mathcal{S} = \{(S, \mathbf{S}) \mid \exists (T, \mathbf{T}) \in \mathcal{S} \text{ such that } (S, \mathbf{S}) \subseteq (T, \mathbf{T})\}.$$

Lemma 30. *Let $n \geq 1$ and $\mathbf{L}$ be a finite multiset of languages. Then, $\mathcal{J}^n_{\mathcal{D}}[\mathbf{L}] = \downarrow \mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$.*

Proof. This amounts to proving that if $(S, \mathbf{S}) \subseteq (T, \mathbf{T})$, then $(T, \mathbf{T})$ $\mathcal{D}$-tied implies $(S, \mathbf{S})$ $\mathcal{D}$-tied. The contrapositive is immediate: if $\mathbf{K} \subseteq \mathcal{D}$ is a separating cover for $(S, \mathbf{S})$, then it must be a separating cover for $(T, \mathbf{T})$. □

The third property, closure under multiplication, requires $\mathcal{D}$ to be a quotienting lattice and the multiset $\mathbf{L}$ to be tame. Recall that this makes $\mathbf{L}$ a partial semigroup for the multiplication "$\odot$". This implies that for any $n \geq 1$, $\mathbf{L}^n$ is a partial semigroup as well for the componentwise multiplication defined as follows: $(L_1, \ldots, L_n) \odot (H_1, \ldots, H_n)$ is defined when $L_i \odot H_i$ is defined for all i and in that case,

$$(L_1, \ldots, L_n) \odot (H_1, \ldots, H_n) = (L_1 \odot H_1, \ldots, L_n \odot H_n).$$

This may be lifted to a true semigroup multiplication for the set $2^{\mathbf{L}^n}$. Given $\mathbf{S}, \mathbf{T} \in 2^{\mathbf{L}^n}$, we define:

$$\mathbf{S} \odot \mathbf{T} = \{\overline{S} \odot \overline{T} \mid \overline{S} \in \mathbf{S}, \overline{T} \in \mathbf{T}\}.$$

It now follows that the set $\mathbf{L} \times 2^{\mathbf{L}^n}$ is a partial semigroup for the componentwise multiplication. In summary, $\mathbf{L}^n$ (which contains $\mathcal{T}^n_{\mathcal{D}}[\mathbf{L}]$) and $\mathbf{L} \times 2^{\mathbf{L}^n}$ (which contains $\mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$) are

both partial semigroups. It turns out that both properties get transferred to $\mathcal{T}^n_{\mathcal{D}}[\mathbf{L}]$ and $\mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$ provided $\mathcal{D}$ is quotienting.

Lemma 31. *Assume that $\mathcal{D}$ is a quotienting lattice and let $\mathbf{L}$ be a tame multiset. For $n \geq 1$, the following properties hold:*

1) *$\mathcal{T}^n_{\mathcal{D}}[\mathbf{L}]$ is closed under multiplication: if $\overline{S}, \overline{T} \in \mathcal{T}^n_{\mathcal{D}}[\mathbf{L}]$, then $\overline{S} \odot \overline{T} \in \mathcal{T}^n_{\mathcal{D}}[\mathbf{L}]$ when defined.*
2) *$\mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$ is closed under multiplication: if $(S, \mathbf{S}), (T, \mathbf{T}) \in \mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$, then $(S, \mathbf{S}) \odot (T, \mathbf{T}) \in \mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$ when defined.*

We finish with our last property which is specific to the case $\mathcal{D} = Pol(\mathcal{C})$ and requires $\mathbf{L}$ to be $\mathcal{C}$-compatible: all languages in a $Pol(\mathcal{C})$-tied n-join carry the same $\mathcal{C}$-type.

Lemma 32. *Let $\mathcal{C}$ be a finite quotienting Boolean algebra, $\mathbf{L}$ a finite $\mathcal{C}$-compatible multiset of languages and $n \geq 1$. For any $(S, \mathbf{S}) \in \mathcal{J}^n_{Pol(\mathcal{C})}[\mathbf{L}]$ and any $(S_1, \ldots, S_n) \in \mathbf{S}$, all S_i share the $\mathcal{C}$-type of S,* i.e., *$[S]_{\mathcal{C}} = [S_1]_{\mathcal{C}} = [S_2]_{\mathcal{C}} = \cdots = [S_n]_{\mathcal{C}}$.*

Proof. The contrapositive is simple. Since $[S]_{\mathcal{C}} \in Pol(\mathcal{C})$, one may verify that when there exists $(S_1, \ldots, S_n) \in \mathbf{S}$ and $i \leq n$ such that $[S]_{\mathcal{C}} \neq [S_i]_{\mathcal{C}}$ (which means $S \cap S_i = \emptyset$ by $\mathcal{C}$-compatiblity), then $\{[S]_{\mathcal{C}}\}$ is a separating cover for $(S, \mathbf{S})$. □

B. Algorithm for Computing $Pol(\mathcal{C})$-tied Joins

We are now ready to present our algorithm computing $Pol(\mathcal{C})$-tied n-joins where $\mathcal{C}$ is an arbitrary finite quotienting Boolean algebra, which we assume fixed until the end of the section. The procedure takes three inputs: an integer $n \geq 1$, a $\mathcal{C}$-compatible tame multiset $\mathbf{L}$, and a set $\mathbf{T} \subseteq \mathbf{L}^n$ of n-tuples. From these inputs, it uses a least fixpoint to compute a set

$$Sat^n(\mathbf{L}, \mathbf{T}) \subseteq \mathbf{L} \times 2^{\mathbf{L}^n}.$$

We then prove that for the appropriate set $\mathbf{T}$, the set $Sat^n(\mathbf{L}, \mathbf{T})$ is exactly $\mathcal{J}^n_{Pol(\mathcal{C})}[\mathbf{L}]$.

In the algorithm, we will use an operation that restricts a set of n-tuples $\mathbf{T} \subseteq \mathbf{L}^n$ to n-tuples involving only languages of a given $\mathcal{C}$-type, say $[L]_{\mathcal{C}}$ for some $L \in \mathbf{L}$. We denote this restriction $\mathbf{T}|_{[L]_{\mathcal{C}}} \subseteq \mathbf{T}$. Formally:

$$\mathbf{T}|_{[L]_{\mathcal{C}}} = \{(T_1, \ldots, T_n) \in \mathbf{T} \mid [T_i]_{\mathcal{C}} = [L]_{\mathcal{C}} \text{ for all } i \leq n\}.$$

We define $Sat^n(\mathbf{L}, \mathbf{T})$ as the smallest subset of $\mathbf{L} \times 2^{\mathbf{L}^n}$ containing $\mathcal{J}^n_{triv}[\mathbf{L}]$ and closed under the following operations:

1) Downset: $Sat^n(\mathbf{L}, \mathbf{T}) = {\downarrow} Sat^n(\mathbf{L}, \mathbf{T})$.
2) Multiplication: for all $(R, \mathbf{R}), (S, \mathbf{S}) \in Sat^n(\mathbf{L}, \mathbf{T})$ such that $R \odot S$ is defined,
$$(R, \mathbf{R}) \odot (S, \mathbf{S}) \in Sat^n(\mathbf{L}, \mathbf{T}).$$
3) For any $(E, \mathbf{E}) \in Sat^n(\mathbf{L}, \mathbf{T})$ which is ***idempotent***,
$$(E, \mathbf{E} \odot \mathbf{T}|_{[E]_{\mathcal{C}}} \odot \mathbf{E}) \in Sat^n(\mathbf{L}, \mathbf{T}).$$

Observe that given $\mathbf{L}$, n and $\mathbf{T}$ as input, one may easily compute $Sat^n(\mathbf{L}, \mathbf{T})$ with a least fixpoint procedure. We may now state the correctness of our algorithm: for the appropriate $\mathbf{T}$, the set $Sat^n(\mathbf{L}, \mathbf{T})$ is exactly $\mathcal{J}^n_{Pol(\mathcal{C})}[\mathbf{L}]$.

Theorem 33. *Let $\mathbf{L}$ be a multiset of languages which is both tame and $\mathcal{C}$-compatible and let $n \geq 1$. Then,*

$$\mathcal{J}^n_{Pol(\mathcal{C})}[\mathbf{L}] = Sat^n(\mathbf{L}, \mathcal{T}^n_{Pol(\mathcal{C})}[\mathbf{L}]).$$

Theorem 33 yields an inductive procedure for computing $\mathcal{J}^n_{Pol(\mathcal{C})}[\mathbf{L}]$ from a tame and $\mathcal{C}$-compatible multiset $\mathbf{L}$ and $n \geq 1$. Indeed, as explained above, knowing $\mathbf{L}$, n and $\mathcal{T}^n_{Pol(\mathcal{C})}[\mathbf{L}]$ is enough to compute $Sat^n(\mathbf{L}, \mathcal{T}^n_{Pol(\mathcal{C})}[\mathbf{L}])$. We have the inputs $\mathbf{L}$ and n in hand. For the set $\mathcal{T}^n_{Pol(\mathcal{C})}[\mathbf{L}]$, there are two cases:

1) $\mathcal{T}^1_{Pol(\mathcal{C})}[\mathbf{L}]$ is by definition the set of all nonempty languages in $\mathbf{L}$ (which is $\mathbf{L}$ itself by $\mathcal{C}$-compatibility).
2) If $n \geq 2$, one may compute $\mathcal{J}^{n-1}_{Pol(\mathcal{C})}[\mathbf{L}]$ by induction and $\mathcal{T}^n_{Pol(\mathcal{C})}[\mathbf{L}]$ can be extracted from this set by Lemma 28.

This solves *Step 1* in our quest for a $BPol(\mathcal{C})$-separation algorithm: computing $\mathcal{T}^n_{Pol(\mathcal{C})}[\mathbf{L}]$ from $\mathbf{L}$ (tame and $\mathcal{C}$-compatible) and n. As announced, this algorithm is recursive: computing $\mathcal{T}^n_{Pol(\mathcal{C})}[\mathbf{L}]$ requires us to compute $\mathcal{T}^{n-1}_{Pol(\mathcal{C})}[\mathbf{L}]$ beforehand.

Before we move to Step 2, let us make a few additional comments. First, observe that Theorem 33 proves half of Theorem 6: $Pol(\mathcal{C})$-separation is decidable. Indeed, the special case $n = 1$ of Theorem 33 yields an algorithm taking as input a $\mathcal{C}$-compatible tame multiset $\mathbf{L}$ and computing $\mathcal{T}^2_{Pol(\mathcal{C})}[\mathbf{L}]$, *i.e.*, all pairs in $\mathbf{L}^2$ which are not $Pol(\mathcal{C})$-separable. By Lemmas 24, and 25, this is enough to get a $Pol(\mathcal{C})$-separation algorithm.

Corollary 34. *For any finite quotienting Boolean algebra $\mathcal{C}$, the $Pol(\mathcal{C})$-separation problem is decidable.*

Note that the proof of the left to right inclusion in Theorem 33 is constructive. One proves that $Sat^n(\mathbf{L}, \mathcal{T}^n_{Pol(\mathcal{C})}[\mathbf{L}])$ computes all n-joins $\mathcal{J}^n_{Pol(\mathcal{C})}[\mathbf{L}]$ by exhibiting separating covers in $Pol(\mathcal{C})$ for those that are not computed (thus proving that they are $Pol(\mathcal{C})$-separable and therefore outside of $=\mathcal{J}^n_{Pol(\mathcal{C})}[\mathbf{L}]$).

VII. Solving $BPol(\mathcal{C})$-Separation

We now turn to our main result and present a separation algorithm for $BPol(\mathcal{C})$ where $\mathcal{C}$ is an arbitrary finite quotienting Boolean algebra, which we assume fixed for the section. We showed in Corollary 14 that this amounts to finding an algorithm which performs the following computation:

INPUT: A tame and $\mathcal{C}$-compatible multiset $\mathbf{L}$.
OUTPUT: The set $\mathcal{A}_{Pol(\mathcal{C})}[\mathbf{L}] \subseteq \mathbf{L}^2$.

The section is organized in two parts. First, we present an algorithm for the above problem, thus solving $BPol(\mathcal{C})$-separation. Then, we discuss the problem of constructing an actual separator when this algorithm answers positively.

A. Algorithm for $BPol(\mathcal{C})$-Separation

Let us briefly recap what we have so far. Given a tame and $\mathcal{C}$-compatible multiset $\mathbf{L}$, the set $\mathcal{A}_{Pol(\mathcal{C})}[\mathbf{L}]$ consists by definition of all pairs $(L_1, L_2) \in \mathbf{L}^2$ such that $(L_1, L_2)^p \in \mathcal{T}^{2p}_{Pol(\mathcal{C})}[\mathbf{L}]$ for all $p \geq 1$. While we do not have an algorithm for computing $\mathcal{A}_{Pol(\mathcal{C})}[\mathbf{L}]$ yet, we already have a combinatorial description. For any $n \geq 1$, we have a procedure for computing the set

$\mathcal{T}^n_{Pol(\mathcal{C})}[\mathbf{L}]$. Indeed, this is trivial for $n = 1$ and for $n \geq 2$, one may use the procedure Sat^{n-1}: by Theorem 33, we have,

$$\mathcal{J}^{n-1}_{Pol(\mathcal{C})}[\mathbf{L}] = Sat^{n-1}(\mathbf{L}, \mathcal{T}^{n-1}_{Pol(\mathcal{C})}[\mathbf{L}])$$

and $\mathcal{J}^{n-1}_{Pol(\mathcal{C})}[\mathbf{L}]$ encodes the set $\mathcal{T}^n_{Pol(\mathcal{C})}[\mathbf{L}]$ by Lemma 28.

Our algorithm for computing $\mathcal{A}_{Pol(\mathcal{C})}[\mathbf{L}]$ is a greatest fixpoint based on the procedure Sat^2. The crux is the next theorem, whose proof is based on a hard analysis of the procedures Sat^n.

Theorem 35. *Let $\mathbf{L}$ be a tame and $\mathcal{C}$-compatible multiset. There exists a largest subset $\mathbf{T}$ of $\mathcal{T}^2_{Pol(\mathcal{C})}[\mathbf{L}]$ (with respect to inclusion) satisfying the following property:*

$$\text{For all } (L_1, L_2) \in \mathbf{T},\ (L_1, \{(L_2, L_1)\}) \in Sat^2(\mathbf{L}, \mathbf{T}). \quad (1)$$

Moreover, $\mathbf{T} = \mathcal{A}_{Pol(\mathcal{C})}[\mathbf{L}]$.

As announced, Theorem 35 yields a greatest fixpoint algorithm for computing $\mathcal{A}_{Pol(\mathcal{C})}[\mathbf{L}]$ from an input tame and $\mathcal{C}$-compatible multiset $\mathbf{L}$. We start from the set $\mathbf{T} = \mathcal{T}^2_{Pol(\mathcal{C})}[\mathbf{L}]$ (which can be computed, see Theorem 33). Then, we repeat the two following operations until $\mathbf{T}$ satifies (1):

1) Use the procedure Sat^2 to compute the set $\mathbf{T}'$ of all pairs $(L_1, L_2) \in \mathbf{T}$ such that $(L_1, \{(L_2, L_1)\}) \in Sat^2(\mathbf{L}, \mathbf{T})$.
2) Replace $\mathbf{T}$ with $\mathbf{T}'$.

When the computation ends, it follows from Theorem 35 that $\mathbf{T} = \mathcal{A}_{Pol(\mathcal{C})}[\mathbf{L}]$.

Hence, one may compute $\mathcal{A}_{Pol(\mathcal{C})}[\mathbf{L}]$ from any input tame and $\mathcal{C}$-compatible multiset $\mathbf{L}$. By Corollary 14 and Lemmas 24 and 25, this solves $BPol(\mathcal{C})$-separation. Thus, we get the following corollary which completes the proof of Theorem 6.

Corollary 36. *For any finite quotienting Boolean algebra $\mathcal{C}$, the $BPol(\mathcal{C})$-separation problem is decidable.*

The main difficulty in Theorem 35 consists in proving that $\mathcal{A}_{Pol(\mathcal{C})}[\mathbf{L}] \subseteq \mathbf{T}$ (and more precisely that $\mathcal{A}_{Pol(\mathcal{C})}[\mathbf{L}]$ satisfies (1)). We outline this proof in section VIII. The converse inclusion is simpler and presented thoroughly in Section VIII. Before we turn to the proof, let us say a few words about the construction of separators in $BPol(\mathcal{C})$ when they exist.

B. Constructing $BPol(\mathcal{C})$-Separators

We sketch a methodology for constructing separators in $BPol(\mathcal{C})$ when they exist. Assume that we have two input regular languages L_1, L_2 which are known to be $BPol(\mathcal{C})$-separable (this can be decided by Theorem 35 above). We explain how to build $K \in BPol(\mathcal{C})$ separating L_1 from L_2.

We know from Theorem 13 that there exists $p \geq 1$ such that $(L_1, L_2)^p$ is $Pol(\mathcal{C})$-separable. Moreover, it follows from the proof of this theorem that if we have this number p in hand together with $Pol(\mathcal{C})$-separators witnessing the fact that $(L_1, L_2)^p$ is $Pol(\mathcal{C})$-separable, we may use them to construct $K \in BPol(\mathcal{C})$ separating L_1 from L_2. This is possible since we have an algorithm which decides $Pol(\mathcal{C})$-separation for any input tuple (see Theorem 33). Hence, since we know that p exists, we may find it with this algorithm. Finally, the proof of Theorem 33 (presented in the full version) is constructive: once we have p such that $(L_1, L_2)^p$ is $Pol(\mathcal{C})$-separable in hand, we may follow it to build $Pol(\mathcal{C})$-separators witnessing this fact.

VIII. Proof of Theorem 35

This section is devoted to proving Theorem 35. For the whole section, we assume fixed a finite quotienting Boolean algebra $\mathcal{C}$ as well as a tame and $\mathcal{C}$-compatible multiset $\mathbf{L}$. Moreover, for the sake of improved readability, given $n \geq 1$, we write $\mathcal{A}[\mathbf{L}]$ for $\mathcal{A}_{Pol(\mathcal{C})}[\mathbf{L}]$, $\mathcal{T}^n[\mathbf{L}]$ for $\mathcal{T}^n_{Pol(\mathcal{C})}[\mathbf{L}]$ and $\mathcal{J}^n[\mathbf{L}]$ for $\mathcal{J}^n_{Pol(\mathcal{C})}[\mathbf{L}]$.

Theorem 35 states that there exists a largest subset $\mathbf{T}$ of $\mathcal{T}^2[\mathbf{L}]$ which satisfies (1):

$$\text{For all } (L_1, L_2) \in \mathbf{T},\ (L_1, \{(L_2, L_1)\}) \in Sat^2(\mathbf{L}, \mathbf{T}),$$

and that this subset is exactly $\mathcal{A}[\mathbf{L}]$. This is an immediate consequence of the two following propositions.

Proposition 37. *Any subset $\mathbf{T}$ of $\mathcal{T}^2[\mathbf{L}]$ which satisfies (1) is a subset of $\mathcal{A}[\mathbf{L}]$.*

Proposition 38. *$\mathcal{A}[\mathbf{L}]$ is a subset of $\mathcal{T}^2[\mathbf{L}]$ and satisfies (1).*

In this section, we give a complete proof of Proposition 37. Moreover, we introduce the main ideas needed to prove Proposition 38 (which is far more involved) and reduce this proof to two other propositions.

A. Proof of Proposition 37

We begin with a few definitions. Recall that for $(H, L) \in \mathbf{L}^2$ and $p \geq 1$, we write $(H, L)^p$ the $2p$-tuple obtained by concatenating p copies of (H, L). We extend this notation and write $(H, L)^{p+\frac{1}{2}}$ the $(2p+1)$-tuple obtained by concatenating p copies of (H, L) plus one copy of (H). For example,

$$(H, L)^{2+\frac{1}{2}} = (H, L, H, L, H)$$

We now introduce a new notion: n-iterations. Consider a set of pairs $\mathbf{S} \in 2^{\mathbf{L}^2}$. For any $n \geq 2$, we define the n-iteration of $\mathbf{S}$, denoted by $it_n(\mathbf{S})$ as the following subset of $\mathbf{L}^n$:

1) If n is even, then $n = 2p$ for $p \geq 1$ and we define $it_n(\mathbf{S}) = \{(S_1, S_2)^p \mid (S_1, S_2) \in \mathbf{S}\}$.
2) If n is odd, then $n = 2p + 1$ for $p \geq 1$, and we define $it_n(\mathbf{S}) = \{(S_1, S_2)^{p+\frac{1}{2}} \mid (S_1, S_2) \in \mathbf{S}\}$.

Note that $it_n(\mathbf{S}) \subseteq \mathbf{L}^n$. The definition of n-iterations entails the following simple lemma.

Lemma 39. *Let $\mathbf{T} \in 2^{\mathbf{L}^2}$. For any $n \geq 2$, and any $(S, \mathbf{S}) \in Sat^2(\mathbf{L}, \mathbf{T})$, we have $(S, it_n(\mathbf{S})) \in Sat^n(\mathbf{L}, it_n(\mathbf{T}))$.*

Proof. By hypothesis, $(S, \mathbf{S}) \in Sat^2(\mathbf{L}, \mathbf{T})$ can be built from $\mathcal{J}^2_{triv}[\mathbf{L}]$ using downset, multiplication and Operation 3. We use an induction on this construction. If $(S, \mathbf{S}) \in \mathcal{J}^2_{triv}[\mathbf{L}]$, then by definition there exists $w \in S$ such that $w \in S_1 \cap S_2$ for all $(S_1, S_2) \in \mathbf{S}$. It is now immediate by definition of $it_n(\mathbf{S})$ that $w \in S$ and $w \in S_1 \cap \cdots \cap S_n$ for all $(S_1, \ldots, S_n) \in it_n(\mathbf{S})$. Hence, $(S, it_n(\mathbf{S})) \in \mathcal{J}^n_{triv}[\mathbf{L}] \subseteq Sat^n(\mathbf{L}, it_n(\mathbf{T}))$.

Otherwise, there are three cases depending on the last operation of $Sat^2(\mathbf{L}, \mathbf{T})$ used to build $(S, \mathbf{S})$. As all cases

are similar, let us concentrate on Operation 3. In that case, $(S, \mathbf{S}) = (E, \mathbf{E} \odot \mathbf{T}|_{[E]_\mathcal{C}} \odot \mathbf{E})$ for some idempotent $(E, \mathbf{E})$ of $Sat^2(\mathbf{L}, \mathbf{T})$. By induction, $(E, it_n(\mathbf{E})) \in Sat^n(\mathbf{L}, it_n(\mathbf{T}))$. It is immediate from the definition of n-iterations that $(E, it_n(\mathbf{E}))$ is idempotent as well and that:

$$(S, it_n(\mathbf{S})) = (E, it_n(\mathbf{E}) \odot it_n(\mathbf{T})|_{[E]_\mathcal{C}} \odot it_n(\mathbf{E})).$$

Hence, we conclude from Operation 3 in the definition of Sat^n that $(S, it_n(\mathbf{S}))$ belongs to $Sat^n(\mathbf{L}, it_n(\mathbf{T}))$. □

We may now proceed with the proof of Proposition 37. Let $\mathbf{T}$ be a subset of $\mathcal{T}^2[\mathbf{L}]$ satisfying (1). By definition of $\mathcal{A}[\mathbf{L}]$, we have to prove that for all $p \geq 1$ and $(T_1, T_2) \in \mathbf{T}$, $(T_1, T_2)^p \in \mathcal{T}^{2p}[\mathbf{L}]$. This is a consequence of the following result, based on Lemma 39 (note that this is where we use our two hypotheses on $\mathbf{T}$).

Lemma 40. *For all $n \geq 2$, $it_n(\mathbf{T}) \subseteq \mathcal{T}^n[\mathbf{L}]$.*

Given $p \geq 1$ and $(T_1, T_2) \in \mathbf{T}$, we have $(T_1, T_2)^p \in it_{2p}(\mathbf{T})$ by definition of $2p$-iteration. It then immediate from the lemma that $(T_1, T_2)^p \in \mathcal{T}^{2p}[\mathbf{L}]$, which proves Proposition 37.

It remains to prove Lemma 39. The argument is by induction on n. For $n = 2$, we know that $\mathbf{T} \subseteq \mathcal{T}^2[\mathbf{L}]$ by hypothesis. Let us now assume that $n \geq 3$. Let $(T_1, \ldots, T_n) \in it_n(\mathbf{T})$, we prove that $(T_1, \ldots, T_n) \in \mathcal{T}^n[\mathbf{L}]$. By definition $(T_1, T_2) \in \mathbf{T}$. Since $\mathbf{T}$ satisfies (1) by hypothesis, we know that $(T_1, \{(T_2, T_1)\})$ belongs to $Sat^2(\mathbf{L}, \mathbf{T})$. Therefore, we obtain by Lemma 39,

$$(T_1, it_{n-1}(\{(T_2, T_1)\})) \in Sat^{n-1}(\mathbf{L}, it_{n-1}(\mathbf{T})).$$

Moreover, since $(T_1, \ldots, T_n) \in it_n(\mathbf{T})$, it follows from the definition of iteration that $it_{n-1}(\{(T_2, T_1)\}) = \{(T_2, \ldots, T_n)\}$. Finally, since $it_{n-1}(\mathbf{T}) \subseteq \mathcal{T}^{n-1}[\mathbf{L}]$ by induction, we obtain $Sat^{n-1}(\mathbf{L}, it_{n-1}(\mathbf{T})) \subseteq Sat^{n-1}(\mathbf{L}, \mathcal{T}^{n-1}[\mathbf{L}])$. Altogether, we get

$$(T_1, \{(T_2, \ldots, T_n)\}) \in Sat^{n-1}(\mathbf{L}, \mathcal{T}^{n-1}[\mathbf{L}]).$$

Since $Sat^{n-1}(\mathbf{L}, \mathcal{T}^{n-1}[\mathbf{L}]) = \mathcal{J}^{n-1}[\mathbf{L}]$ by Theorem 33, we conclude using Lemma 28 that $(T_1, \ldots, T_n) \in \mathcal{T}^n[\mathbf{L}]$.

B. Proof of Proposition 38

We have to prove that $\mathcal{A}[\mathbf{L}]$ is a subset of $\mathcal{T}^2[\mathbf{L}]$ and satisfies (1). That $\mathcal{A}[\mathbf{L}] \subseteq \mathcal{T}^2[\mathbf{L}]$ is immediate. Indeed, $\mathcal{A}[\mathbf{L}]$ is the set of all $(L_1, L_2) \in \mathbf{L}^2$ such that $(L_1, L_2)^p \in \mathcal{T}^{2p}[\mathbf{L}]$ for all p. Hence, the case $p = 1$ yields that $\mathcal{A}[\mathbf{L}] \subseteq \mathcal{T}^2[\mathbf{L}]$. Therefore, we may concentrate on proving that $\mathcal{A}[\mathbf{L}]$ satisfies (1).

We have to prove that for any $(L_1, L_2) \in \mathcal{A}[\mathbf{L}]$, we have $(L_1, \{(L_2, L_1)\}) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$. The argument is based on a combinatorial analysis of the procedure computing the set $Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$ from $n \geq 1$ and $\mathbf{L}$. We introduce a new object which represent computations of this procedure: *computation trees*. They will be central in the proof.

For any $n \geq 1$, we associate a set of *computation trees of level n*: each tree represents the computation of some n-join in $Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$. More precisely, a computation tree $\mathbb{T}$ of level n is an unranked ordered tree. Each node x in $\mathbb{T}$ must carry a label $lab(x)$ which is an n-join in $\mathbf{L} \times 2^{\mathbf{L}^n}$ and there are four possible kinds of nodes:

- **Leaves**: x has no children and $lab(x) \in \mathcal{J}^n_{triv}[\mathbf{L}]$.
- **Downset**: x has a unique child y and $lab(x) \subseteq lab(y)$.
- **Binary**: x has exactly two children x_ℓ and x_r. Moreover, $lab(x) = lab(x_\ell) \odot lab(x_r)$ (in particular, this multiplication must be defined).
- **Operation**: x has a unique child y. Moreover, the two following conditions must be satisfied:
  1) The label $lab(y)$ is an idempotent $(E, \mathbf{E})$.
  2) $lab(x) = (E, \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_\mathcal{C}} \odot \mathbf{E})$.

If $\mathbb{T}$ is a computation tree, we write $lab(\mathbb{T})$ for the label of its root. Moreover, we define the *operational height of a computation tree* $\mathbb{T}$ as the maximal number g such that $\mathbb{T}$ contains a branch with g operation nodes.

Computation trees of level n are designed to represent computations of the procedure $Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$: leaves correspond to the starting set $\mathcal{J}^n_{triv}[\mathbf{L}]$ and each kind of node corresponds to an operation. We state this fact in the following proposition, which reformulates Theorem 33 on computation trees.

Proposition 41. *Let $n \geq 1$. For any $(S, \mathbf{S}) \in \mathbf{L} \times 2^{\mathbf{L}^n}$. The two following properties are equivalent:*

1) $(S, \mathbf{S}) \in \mathcal{J}^n[\mathbf{L}]$.
2) There exists a computation tree $\mathbb{T}$ of level n such that $(S, \mathbf{S}) = lab(\mathbb{T})$.

We may now come back to Proposition 38. The argument is based on two intermediary results. The first one states that for any computation tree, there exists another one with the same label but with operational height at most $|\mathcal{C}|$ (observe that this bound is independent from the level n of the trees).

Proposition 42. *For any $n \geq 1$ and any computation tree $\mathbb{T}$ of level n, there exists a computation tree $\mathbb{T}'$ with the same label as $\mathbb{T}$ and whose operational height is at most $|\mathcal{C}|$.*

Presenting the second result requires one more definition. Let $\mathbf{S} \subseteq \mathbf{L}^n$ for some $n \geq 1$. For any $p \geq 1$, we define the *p-extraction of* $\mathbf{S}$ as the set $ex_p(\mathbf{S}) \subseteq \mathbf{L}^2$ of all pairs $(H, L) \in \mathbf{L}^2$ such that $(H, L)^p$ is a subsequence of some n-tuple $(S_1, \ldots, S_n) \in \mathbf{S}$, *i.e.*, there exist $1 \leq i_1 < \cdots < i_{2p} \leq n$ such that,

$$(H, L)^p = (S_{i_1}, \ldots, S_{i_{2p}}).$$

Observe that $ex_p(\mathbf{S})$ is necessarily empty when $n < 2p$. We may now present our second proposition.

Proposition 43. *For all $g \in \mathbb{N}$, there exists $p_g \geq 1$ such that for any $n \geq 1$ and any computation tree $\mathbb{T}$ of level n and operational height at most g, if $(S, \mathbf{S}) = lab(\mathbb{T})$, we have:*

$$(S, ex_{p_g}(\mathbf{S})) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}]).$$

Proposition 42 and Proposition 43 are based on involved combinatorial arguments (presented in the full version of the paper). Here, we use them to finish the proof of Proposition 38.

Consider a pair $(L_1, L_2) \in \mathcal{A}[\mathbf{L}]$, we have to show that $(L_1, \{(L_2, L_1)\}) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$. Let $g = |\mathcal{C}|$ and $p_g \geq 1$ be as defined in Proposition 43 for this g.

By definition of $\mathcal{A}[\mathbf{L}]$, we have $(L_1, L_2)^{2p_g+2} \in \mathcal{T}^{2p_g+2}[\mathbf{L}]$. Therefore, it follows from Lemmas 12 and 28 that:

$$(L_1, \{(L_2, L_1)^{p_g}\}) \in \mathcal{J}^{2p_g}[\mathbf{L}].$$

By Proposition 41, we get a computation tree $\mathbb{T}$ of level $2p_g$ whose label is $(L_1, \{(L_2, L_1)^{p_g}\})$. Moreover, since $g = |\mathcal{C}|$, it follows from Proposition 42 that we may choose $\mathbb{T}$ with operational height smaller than g. By choice of p_g in Proposition 43, we have,

$$(L_1, ex_{p_g}(\{(L_2, L_1)^{p_g}\})) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}]).$$

By definition, we have $ex_{p_g}(\{(L_2, L_1)^{p_g}\}) = \{(L_2, L_1)\}$. Hence, we conclude that $(L_1, \{(L_2, L_1)\}) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$ which terminates the proof of Proposition 38.

IX. Conclusion

We proved that separation is decidable for all classes of the form $Pol(\mathcal{C})$ or $BPol(\mathcal{C})$ when $\mathcal{C}$ is a finite quotienting Boolean algebra. In practice, this yields separation algorithms a whole family of classes. The most important one is the level 2 in the Straubing-Thérien hierarchy (which corresponds to the logic $\mathcal{B}\Sigma_2(<)$). Moreover, using a known transfer theorem [26], this result can be lifted to dot-depth 2.

Further work. While we leave the details for further work, let us briefly discuss complexity. Starting from nondeterministic finite automata $\mathcal{A}_1$ and $\mathcal{A}_2$ recognizing two input languages L_1, L_2, one first builds a tame and $\mathcal{C}$-compatible multiset $\mathbf{L}$ extending $\{L_1, L_2\}$ of size $(|\mathcal{A}_1|^2+|\mathcal{A}_2|^2)\times|\mathcal{C}|$. Note that since the size of $\mathbf{L}$ depends on $|\mathcal{C}|$ which might not be a constant (it may depend on the alphabet A, see the class AT defined in Section III). Hence, the complexity of our algorithms depend on $\mathcal{C}$. For example, one may obtain a PSPACE upper bound for $Pol(\mathrm{AT})$ and $BPol(\mathrm{AT})$ (*i.e.*, the levels $\frac{3}{2}$ and 2 in the Straubing-Thérien hierarchy) with respect to $|\mathcal{A}_1|$ and $|\mathcal{A}_2|$.

However, there is one case when performing a generic complexity analysis is possible: when the alphabet is fixed or when $|\mathcal{C}|$ is a constant independent from it. In that case, one may derive PTIME procedures for both $Pol(\mathcal{C})$- and $BPol(\mathcal{C})$-separation (with respect to $|\mathcal{A}_1|$ and $|\mathcal{A}_2|$). An interesting consequence is that this yields a PTIME separation algorithm for dot-depth one (which is the class $BPol(\mathcal{C})$ where $\mathcal{C} = \{\emptyset, \{\varepsilon\}, A^+, A^*\}$, which is of size 4 for any alphabet A).

Another interesting consequence of our results is that since we proved the decidability of $\mathcal{B}\Sigma_2(<)$-separation, the main theorem of [25] is an immediate corollary: $\mathcal{B}\Sigma_2(<)$-membership is decidable. Moreover, the algorithm of [25] was actually based on a characterization theorem: languages in $\mathcal{B}\Sigma_2(<)$ are characterized by a syntactic property of a canonical recognizer (*i.e.*, the syntactic monoid). It turns out that one can also deduce this characterization theorem from our results (this does require a little combinatorial work however). In fact, one may generalize it to all classes $BPol(\mathcal{C})$ when $\mathcal{C}$ is a finite quotienting Boolean algebra.

Finally, the main and most natural follow-up question is much harder: can our results be pushed to higher levels within concatenation hierarchies? For now, we know that given any finite quotienting Boolean algebra $\mathcal{C}$, $Pol(\mathcal{C})$ and $BPol(\mathcal{C})$ have decidable separation. Moreover, one may combine our generic approach together with ideas of [23] to obtain a procedure for $Pol(BPol(\mathcal{C}))$-separation (the main result of [23] is an algorithm for $\Sigma_3(<)$, i.e. $Pol(BPol(\mathrm{AT}))$). Thus, the next relevant levels are $BPol(BPol(\mathcal{C}))$ and $Pol(BPol(BPol(\mathcal{C})))$.

References

[1] J. Almeida, "Some algorithmic problems for pseudovarieties," *Publicationes Mathematicae Debrecen*, vol. 54, 1999.

[2] M. Arfi, "Polynomial operations on rational languages," in *STACS'87*, 1987.

[5] J. A. Brzozowski and R. S. Cohen, "Dot-depth of star-free events," *J. Comput. System Sci.*, vol. 5, no. 1, 1971.

[6] J. A. Brzozowski and R. Knast, "The dot-depth hierarchy of star-free languages is infinite," *J. Comput. System Sci.*, vol. 16, no. 1, 1978.

[8] W. Czerwiński, W. Martens, and T. Masopust, "Efficient separability of regular languages by subsequences and suffixes," in *ICALP'13*, 2013.

[9] V. Diekert and P. Gastin, "First-order definable languages," in *Logic and Automata: History and Perspectives*, vol. 2, Amsterdam Univ., 2008.

[10] V. Diekert, P. Gastin, and M. Kufleitner, "A survey on small fragments of first-order logic over finite words," *Internat. J. Found. Comput. Sci.*, vol. 19, no. 3, 2008.

[11] V. Diekert, M. Kufleitner, G. Rosenberger, and U. Hertrampf, *Discrete Algebraic Methods: Arithmetic, Cryptography, Automata and Groups*. De Gruyter GmbH, 2016.

[12] C. Glaßer and H. Schmitz, "Languages of dot-depth 3/2," in *STACS'00*, 2000.

[14] R. Knast, "A semigroup characterization of dot-depth one languages," *RAIRO Theoret. Informatics and Appl.*, vol. 17, no. 4, 1983.

[16] R. McNaughton and S. A. Papert, *Counter-Free Automata*. MIT Press, 1971.

[17] D. Perrin and J.-É. Pin, "First-order logic and star-free sets," *J. Comput. System Sci.*, vol. 32, no. 3, 1986.

[18] J.-É. Pin, "Bridges for concatenation hierarchies," in *ICALP'98*, 1998.

[19] ——, "Open problems about regular languages, 35 years later," in *The Role of Theory in Computer Science. Essays Dedicated to Janusz Brzozowski*. World Scientific, 2017.

[20] J.-É. Pin, "The dot-depth hierarchy, 45 years later," in *The Role of Theory in Computer Science. Essays Dedicated to Janusz Brzozowski*. World Scientific, 2017.

[21] J.-É. Pin and H. Straubing, "Monoids of upper triangular boolean matrices," in *Semigroups. Structure and Universal Algebraic Problems*, vol. 39, North-Holland, 1985.

[22] J.-É. Pin and P. Weil, "Polynomial closure and unambiguous product," *Theory Comput. Syst.*, vol. 30, no. 4, 1997.

[23] T. Place, "Separating regular languages with two quantifier alternations," in *LICS'15*, 2015.

[24] T. Place, L. van Rooijen, and M. Zeitoun, "Separating regular languages by piecewise testable and unambiguous languages," in *MFCS'13*, 2013.

[25] T. Place and M. Zeitoun, "Going higher in the first-order quantifier alternation hierarchy on words," in *ICALP'14*, 2014.

[26] ——, "Separation and the successor relation," in *STACS 2015*, 2015.

[27] ——, "The tale of the quantifier alternation hierarchy of first-order logic over words," *SIGLOG news*, vol. 2, no. 3, pp. 4–17, 2015.

[28] ——, "Separating regular languages with first-order logic," *Logical Methods in Computer Science*, vol. 12, no. 1, 2016.

[29] ——, "The covering problem: A unified approach for investigating the expressive power of logics," in *MFCS'16*, 2016, 77:1–77:15.

[30] M. P. Schützenberger, "On finite monoids having only trivial subgroups," *Information and Control*, vol. 8, no. 2, 1965.

[31] I. Simon, "Piecewise testable events," in *2nd GI Conference on Automata Theory and Formal Languages*, 1975.

[33] H. Straubing, "A generalization of the Schützenberger product of finite monoids," *Theoret. Comp. Sci.*, vol. 13, no. 2, 1981.

[34] ——, "Semigroups and languages of dot-depth two," *Theoret. Comp. Sci.*, vol. 58, no. 1-3, 1988.

[35] D. Thérien, "Classification of finite monoids: the language approach," *Theoret. Comp. Sci.*, vol. 14, no. 2, 1981.

[36] W. Thomas, "Classifying regular events in symbolic logic," *J. Comput. System Sci.*, vol. 25, no. 3, 1982.

APPENDIX A
OMITTED PROOFS IN SECTION III

In this appendix, we prove the statements of Section III whose proofs were omitted in the main text. We begin with the fact that in a hierarchy whose basis is a quotienting Boolean algebra, all half levels are quotienting lattices closed under concatenation and marked concatenation, and all full levels are quotienting Boolean algebras.

Fact 5. *Let $\mathcal{C}$ be a quotienting Boolean algebra. Then,*

1) *$Pol(\mathcal{C})$ is a quotienting lattice and is closed under concatenation and marked concatenation.*
2) *$BPol(\mathcal{C})$ is a quotienting Boolean algebra.*

Proof. We first prove Item 1). We already know by definition of $Pol(\mathcal{C})$ that it is a lattice closed under marked concatenation. It remains to establish closure under quotient and concatenation.

For closure under quotient, let $L \in Pol(\mathcal{C})$ and $w \in A^*$, and let us show that $w^{-1}L$ and Lw^{-1} both belong to $Pol(\mathcal{C})$. By symmetry, it suffices to prove that $w^{-1}L \in Pol(\mathcal{C})$. By definition of $Pol(\mathcal{C})$, the language L is built from languages of $\mathcal{C}$ using a finite number of times union, intersection and marked concatenation. We argue by induction on this construction. If L belongs to $\mathcal{C}$, then so does $w^{-1}L$ since $\mathcal{C}$ is closed under quotient, so in particular $w^{-1}L \in Pol(\mathcal{C})$. If $L = K \cup H$ with $K, H \in Pol(\mathcal{C})$, then $w^{-1}L = (w^{-1}K) \cup (w^{-1}H)$. By induction, both $w^{-1}K$ and $w^{-1}H$ belong to $Pol(\mathcal{C})$, and since $Pol(\mathcal{C})$ is closed under union, $w^{-1}L$ also belongs to $Pol(\mathcal{C})$. If $L = K \cap H$ with $K, H \in Pol(\mathcal{C})$, we argue similarly using $w^{-1}L = (w^{-1}K) \cap (w^{-1}H)$. Finally, if $L = KaH$, let V be the finite set $\{v \in A^* \mid w \in Kav\}$. Then $w^{-1}L = (w^{-1}K)aH \cup \bigcup_{v \in V} v^{-1}H$. By induction, all languages $w^{-1}K$ and $v^{-1}H$ belong to $Pol(\mathcal{C})$, hence so does L since $Pol(\mathcal{C})$ is closed under union and marked concatenation.

It remains to show that $Pol(\mathcal{C})$ is closed under concatenation. Let $K, L \in Pol(\mathcal{C})$. We can verify that

$$KL = \begin{cases} K \cup \bigcup_{a \in A} Ka(a^{-1}L) & \text{if } \varepsilon \in L, \\ \bigcup_{a \in A} Ka(a^{-1}L) & \text{if } \varepsilon \notin L. \end{cases}$$

Since we already know that $Pol(\mathcal{C})$ is closed under union, quotient and marked concatenation, we obtain $KL \in Pol(\mathcal{C})$, which proves closure under concatenation.

It remains to prove Item 2). By definition, $BPol(\mathcal{C})$ is closed under boolean operations. To show that it is closed under quotient, one can proceed as above, noting that for any $L \subseteq A^*$ and any $w \in A^*$, we have $w^{-1}(A^* \setminus L) = A^* \setminus (w^{-1}L)$. □

Remark 44. *When $\mathcal{C}$ is a quotienting lattice, an alternate definition of $Pol(\mathcal{C})$ is the following: $Pol(\mathcal{C})$ is the smallest class of languages containing $\mathcal{C}$ and closed under union and marked concatenation. It turns out that if $\mathcal{C}$ is a quotienting lattice, this definition coincides with the one used in this paper. In other words, closing a quotienting lattice under union and marked concatenation entails closure under intersection as well.*

Lemma 9. *$\Sigma_2(<) = Pol(\mathrm{AT})$ and $\mathcal{B}\Sigma_2(<) = BPol(\mathrm{AT})$.*

Proof. We already know that $\Sigma_2(<) = Pol(\mathcal{B}\Sigma_1(<))$ from the correspondences between logical hierarchies and concatenation hierarchies [36], [17] (see also Proposition 7). Furthermore, $\mathrm{AT} \subseteq \mathcal{B}\Sigma_1(<)$, since any language of the form A^*aA^* can be expressed by the $\mathcal{B}\Sigma_1(<)$ formula $\exists x P_a(x)$. Therefore, we obtain $Pol(\mathrm{AT}) \subseteq Pol(\mathcal{B}\Sigma_1(<)) = \Sigma_2(<)$. For the converse inclusion, we use the fact that the class $\Sigma_2(<)$ consists of all finite unions of languages of the form $A_0^* a_1 A_1^* \dots a_n A_n^*$ (see [21], [2] for original proofs or [3], [4] for a short and direct proof). Now, for any $B \subseteq A$, the language B^* belongs to AT, since $B^* = A^* \setminus \bigcup_{a \notin B} A^* a A^*$. Therefore, any language of the form $A_0^* a_1 A_1^* \dots a_n A_n^*$ belongs to $Pol(\mathrm{AT})$, whence so does any finite union of such languages, which shows that $\Sigma_2(<) \subseteq Pol(\mathrm{AT})$. □

APPENDIX B
OMITTED PROOFS IN SECTION IV

This appendix presents the proof of Lemma 12 as well as the missing direction in the proof of Theorem 13. Recall that $\mathcal{D}$ denotes an arbitrary lattice.

A. Proof of Lemma 12

Lemma 12. *Let $n \geq m \geq 1$, $(L_1, \dots, L_n)$ an n-tuple and $i_1, \dots, i_m \in \mathbb{N}$ such that $1 \leq i_1 < i_2 < \cdots < i_m \leq n$. If the m-tuple $(L_{i_1}, \dots, L_{i_m})$ is $\mathcal{D}$-separable, then so is $(L_1, \dots, L_n)$.*

We use an induction on $n + m$. If $n = m = 1$, then this is trivial. Otherwise, we have $n + m > 2$ and we distinguish two cases depending on whether $i_1 > 1$ or $i_1 = 1$. Assume first that $i_1 > 1$. We prove that $A^* \in \mathcal{D}$ is a separator for $(L_1, \dots, L_n)$. Clearly, $L_1 \subseteq A^*$ and since $2 \leq i_1 < \cdots < i_m \leq n$, it is immediate by induction hypothesis that $(L_2, \dots, L_n) \cap A^* = (L_2, \dots, L_n)$ is $\mathcal{D}$-separable.

Finally, assume that $i_1 = 1$. There are two sub-cases depending on whether $m = 1$ or not. If $m = 1$, then (L_{i_1}) is $\mathcal{D}$-separable and therefore, $L_{i_1} = \emptyset$. Since $L_1 = L_{i_1} = \emptyset$, it is now simple to verify that $\emptyset \in \mathcal{D}$ is a separator for $(L_1, \dots, L_n)$. Otherwise, $m > 1$ and we get a separator $K \in \mathcal{D}$ for $(L_{i_1}, \dots, L_{i_m})$. We prove that K is a separator of $(L_1, \dots, L_n)$ as well. By definition, $L_1 = L_{i_1} \subseteq K$ and since $(L_{i_2}, \dots, L_{i_m}) \cap K$ is $\mathcal{D}$-separable, we know by induction that $(L_2, \dots, L_n) \cap K$ is $\mathcal{D}$-separable as well since $2 \leq i_2 < \cdots < i_m \leq n$. This terminates the proof.

B. Proof of Theorem 13

Theorem 13. *Let $\mathcal{D}$ be a lattice and let L_1, L_2 be two languages. The following properties are equivalent:*

1) *L_1 is $Bool(\mathcal{D})$-separable from L_2.*
2) *There exists $p \geq 1$ such that $(L_1, L_2)^p$ is $\mathcal{D}$-separable.*

Let us fix $\mathcal{D}$ as an arbitrary lattice for the proof. We already proved the direction 2) ⇒ 1) in the main paper. Hence, we concentrate on the other one: 1) ⇒ 2). Let us start with a preliminary lemma that we will need.

Lemma 45. *Let $n \geq 1$ and $(L_1, \dots, L_n)$ be an n-tuple. Moreover, let $H_1, H_2 \in \mathcal{D}$ such that $(L_1, \dots, L_n) \cap H_1$ and*

$(L_1, \ldots, L_n) \cap H_2$ *are both* $\mathcal{D}$*-separable. Then, the* n*-tuple* $(L_1, \ldots, L_n) \cap (H_1 \cup H_2)$ *is* $\mathcal{D}$*-separable as well.*

Proof. We proceed by induction on n. When $n = 1$, then we have $L_1 \cap H_1 = \emptyset$ and $L_1 \cap H_2 = \emptyset$ by hypothesis. Hence, $L_1 \cap (H_1 \cup H_2) = \emptyset$ and $(L_1) \cap (H_1 \cup H_2)$ is $\mathcal{D}$-separable.

Assume now that $n \geq 2$. By hypothesis, we have separators $K_1, K_2 \in \mathcal{D}$ for $(L_1, \ldots, L_n) \cap H_1$ and $(L_1, \ldots, L_n) \cap H_2$ respectively. We prove that

$$K = (K_1 \cap H_1) \cup (K_2 \cap H_2) \in \mathcal{D}$$

is a separator for $(L_1, \ldots, L_n) \cap (H_1 \cup H_2)$. There are two properties to prove. We begin with $L_1 \cap (H_1 \cup H_2) \subseteq K$. If $w \in L_1 \cap (H_1 \cup H_2)$, one of the two following properties hold:

1) Either $w \in L_1 \cap H_1$ which is included in K_1 by definition of K_1. Hence, in that case $w \in K_1 \cap H_1 \subseteq K$.
2) Or $w \in L_1 \cap H_2$ which is included in K_2 by definition of K_2. Hence, in that case $w \in K_2 \cap H_2 \subseteq K$.

We conclude that $L_1 \cap (H_1 \cup H_2) \subseteq K$. We now need to prove that, $(L_2, \ldots, L_n) \cap (H_1 \cup H_2) \cap K$ is $\mathcal{D}$-separable. By definition of K, we have $(H_1 \cup H_2) \cap K = K$. Hence, it suffices to prove that $(L_2, \ldots, L_n) \cap K$ is $\mathcal{D}$-separable. This is where we use induction. By definition of K_1 and K_2, we know that $(L_2, \ldots, L_n) \cap H_1 \cap K_1$ and $(L_2, \ldots, L_n) \cap H_2 \cap K_2$ are both $\mathcal{D}$-separable. Hence, we obtain from induction that,

$$(L_2, \ldots, L_n) \cap ((H_1 \cap K_1) \cup (H_2 \cap K_2)) \quad \text{is } \mathcal{D}\text{-separable}$$

Since $K = (K_1 \cap H_1) \cup (K_2 \cap H_2)$, this finishes the proof. □

We may now start the proof of 2) ⇒ 1) in Theorem 13. Let L_1, L_2 be two languages and assume that L_1 is $Bool(\mathcal{D})$-separable from L_2. We have to find $p \geq 1$ such that $(L_1, L_2)^p$ is $\mathcal{D}$-separable. The proof is an induction whose parameter is based on a normal form for the separator $K \in Bool(\mathcal{D})$ of L_1 from L_2. We state it in the following lemma.

Lemma 46. *Let* $n \in \mathbb{N}$ *and* L_1, L_2 *be two languages. Furthermore, consider* $2n$ *languages* $K_1, \ldots, K_n, H_1, \ldots, H_n \in \mathcal{D}$ *and the following set* $K \in Bool(\mathcal{D})$*:*

$$K = \bigcup_{i=1}^{n} K_i \setminus H_i.$$

Assume that $L_1 \subseteq K$ *and* $K \cap L_2 = \emptyset$*. Then,* $(L_1, L_2)^{n+1}$ *is* $\mathcal{D}$*-separable.*

Before we show Lemma 46, let use it to finish the proof of 2) ⇒ 1) in Theorem 13. If L_1 is $Bool(\mathcal{D})$-separable from L_2, we have a separator $S \in Bool(\mathcal{D})$ such that $L_1 \subseteq S$ and $S \cap L_2 = \emptyset$. By definition, S is the Boolean combination of languages in $\mathcal{D}$. We put it in disjunctive normal form. Each disjunct is an intersection languages belonging to $\mathcal{D}$, or whose complement belongs to $\mathcal{D}$. Since $\mathcal{D}$ is closed under union and intersection, both $\mathcal{D}$ and $2^{A^*} \setminus \mathcal{D}$ are closed under intersection. Therefore, each disjunct in the disjunctive normal form of S is actually of the form $K \setminus H$, where K, H both belong to $\mathcal{D}$ (for the case where K or H is empty, recall that both $\emptyset$ and A^* belong to $\mathcal{D}$). In summary, there exist $n \in \mathbb{N}$ and $K_1, \ldots, K_n, H_1, \ldots, H_n \in \mathcal{D}$ such that,

$$S = \bigcup_{i=1}^{n} K_i \setminus H_i.$$

It is now immediate from Lemma 46 that $(L_1, L_2)^{n+1}$ is $\mathcal{D}$-separable, which terminates the proof of Theorem 13.

Proof of Lemma 46. We conclude with the proof of Lemma 46. The argument is by induction on n. If $n = 0$, then K is the empty union: $K = \emptyset \in \mathcal{D}$. Hence, since it is a separator of L_1 from L_2, we know that $(L_1, L_2)^1$ is $\mathcal{D}$-separable.

We now assume that $n \geq 1$. Consider the language $K' = K_1 \cup \cdots \cup K_n \in \mathcal{D}$. We prove that it is a separator of $(L_1, L_2)^{n+1}$. Clearly, $K \subseteq K'$, hence $L_1 \subseteq K \subseteq K'$ by hypothesis. It remains to prove that the following $(2n-1)$-tuple is $\mathcal{D}$-separable:

$$(L_2, L_1, L_2, \ldots, L_1, L_2) \cap K'.$$

We prove that $H' = H_1 \cup \cdots \cup H_n$ is a separator. Let us first show that $L_2 \cap K' \subseteq H'$. By hypothesis on K, $L_2 \cap K = \emptyset$. Hence, we know that for all i, $L_2 \cap (K_i \setminus H_i) = \emptyset$ and therefore $L_2 \cap K_i \subseteq H_i$. The union of these inclusions yields $L_2 \cap K' \subseteq H'$ as desired. It now remains to prove that,

$$(L_1, L_2)^n \cap K' \cap H' \quad \text{is } \mathcal{D}\text{-separable} \tag{2}$$

We use induction to show that for all $j \leq n$, $(L_1, L_2)^n \cap K' \cap H_j$ is $\mathcal{D}$-separable. Since $H' = H_1 \cup \cdots \cup H_n$, it will then suffice to apply Lemma 45 to obtain (2). Let $j \leq n$ and consider the following language G_j,

$$G_j = \bigcup_{i \neq j} K_i \setminus H_i$$

We prove that G_j separates $L_1 \cap K' \cap H_j$ from $L_2 \cap K' \cap H_j$. Since it is the union of $n - 1$ languages, it will then be immediate by induction that $(L_1, L_2)^n \cap K' \cap H_j$ is $\mathcal{D}$-separable.

We first prove that $L_1 \cap K' \cap H_j \subseteq G_j$. Since $L_1 \subseteq K$, we have $L_1 \cap K' \cap H_j \subseteq K \cap K' \cap H_j$. Moreover, since $(K_j \setminus H_j) \cap H_j = \emptyset$, we have $K \cap H_j \subseteq G_j$. Hence, we obtain $L_1 \cap K' \cap H_j \subseteq G_j$ as desired. We now prove that $L_2 \cap K' \cap H_j \cap G_j = \emptyset$. This is immediate since $L_2 \cap K = \emptyset$, $L_2 \cap K' \cap H_j \subseteq L_2$ and by definition, $G_j \subseteq K$. □

APPENDIX C
TAME SETS AND $\mathcal{C}$-COMPATIBILITY

In this appendix, we prove Lemmas 24 and 25. In other words we present the reductions to tame and $\mathcal{C}$-compatible inputs.

A. Proof of Lemma 24

Lemma 24. *Let $\mathcal{D}$ be a lattice and let $\mathbf{H}, \mathbf{L}$ be two sets of languages. Assume that $\mathbf{H}$ extends $\mathbf{L}$. Then, for any $L_1, L_2 \in \mathbf{L}$, the following are equivalent:*

1) *L_1 is $\mathcal{D}$-separable from L_2.*
2) *For all $H_1, H_2 \in \mathbf{H}$ such that $H_1 \subseteq L_1$ and $H_2 \subseteq L_2$, H_1 is $\mathcal{D}$-separable from H_2.*

The direction 1) $\Rightarrow$ 2) is immediate since a separator $K \in \mathcal{D}$ of L_1 from L_2 is also a separator of $H_1 \subseteq L_1$ from $H_2 \subseteq L_2$. Hence, we concentrate on the direction 2) $\Rightarrow$ 1).

Assume that for all $H_1, H_2 \in \mathbf{H}$ such that $H_1 \subseteq L_1$ and $H_2 \subseteq L_2$, H_1 is $\mathcal{D}$-separable from H_2, we let $K_{H_1,H_2} \in \mathcal{D}$ be a separator. We define,

$$K = \bigcup_{H_1 \subseteq L_1} \bigcap_{H_2 \subseteq L_2} K_{H_1,H_2}$$

Let us prove that $K \in \mathcal{D}$ separates L_1 from L_2. We begin with $L_1 \subseteq K$. If $w \in L_1$, it must belong to some $H_1 \in \mathbf{H}$ such that that $H_1 \subseteq L_1$ (by definition of extension L_1 is the union of these H_1). Hence, $w \in K_{H_1,H_2}$ for all $H_2 \subseteq L_2$ and therefore $w \in K$. It remains to prove that $L_2 \cap K = \emptyset$. Let $w \in L_2$, we prove that $w \notin K$. By definition of extension, L_2 is the union of all $H_2 \in \mathbf{H}$ such that $H_2 \subseteq L_2$. Hence, for a fixed H_1, w does not belong to K_{H_1,H_2} for all $H_2 \subseteq L_2$. It follows that $w \notin K$.

B. Proof of Lemma 25

Recall that $\mathcal{C}$ denotes an arbitrary finite quotienting Boolean algebra.

Lemma 25. *Given as input a finite set of regular language names $\mathbf{L}$, one may construct a finite tame and $\mathcal{C}$-compatible set $\mathbf{H}$ extending $\mathbf{L}$.*

The extension relation is transitive. Hence, we proceed in two steps. First, we build a tame $\mathbf{G}$ which extends $\mathbf{L}$, then we build a tame and $\mathcal{C}$-compatible $\mathbf{H}$ which extends $\mathbf{G}$. The construction of a tame extension is taken form [29].

Building tame sets. Let $\mathbf{L} = \{L_1, \ldots, L_n\}$ and for all $i \leq n$, let $\mathcal{A}_i = (Q_i, \delta_i, I_i, F_i)$ be a nondeterministic finite automaton recognizing L_i.

We first build for each $i \leq n$ a tame set $\mathbf{G}_i$ extending $\{L_i\}$ and then construct $\mathbf{G}$ extending $\mathbf{L}$ from the $\mathbf{G}_i$. We define $\mathbf{G}_i = \{L_{q,r} \mid (q,r) \in Q_i^2\}$ where $L_{q,r}$ is a name for the language of all words for which there exists a run from state q to state r in $\mathcal{A}_i$.

Remark 47. *Working with language* names *is important here. It may happen that $L_{q,r}$ and $L_{q',r'}$ are different names for the same language.*

It is now simple to verify that $\mathbf{G}_i$ is tame for the following multiplication "$\odot_i$". Let $q, r, s, t \in Q_i$. If $r \neq s$, then $L_{q,r} \odot_i L_{s,t}$ is undefined. Otherwise, $r = s$ and we define $L_{q,r} \odot_i L_{r,t} = L_{q,t}$. Moreover $\mathbf{G}_i$ extends $\{L_i\}$, since L_i is the union of all $L_{q,r}$ with $q \in I_i$ and $r \in F_i$.

We now construct $\mathbf{G}$ extending $\mathbf{L}$. Since we are dealing with language names, we may assume without loss of generality that the sets $\mathbf{G}_i$ are disjoint. We define $\mathbf{G}$ as the disjoint union $\mathbf{G} = \mathbf{G}_1 \uplus \cdots \uplus \mathbf{G}_n$. It is immediate that $\mathbf{G}$ extends $\mathbf{L}$. Finally, one may verify that $\mathbf{G}$ is tame for the following multiplication "$\odot$". Given $G, G' \in \mathbf{G}$, $G \odot G'$ is defined if and only if $G, G' \in \mathbf{G}_i$ for some i and $G \odot_i G'$ is defined. Moreover, in that case, $G \odot G' = G \odot_i G'$.

Building $\mathcal{C}$-compatible sets. We now build $\mathbf{H}$ extending $\mathbf{G}$ which is both tame and $\mathcal{C}$-compatible. Recall that by Lemma 23, the equivalence $\sim_{\mathcal{C}}$ over A has finite index and is a congruence for concatenation. Hence, the quotient set $A^*/\!\sim_{\mathcal{C}}$ is a finite monoid (the neutral element is $[\varepsilon]_{\mathcal{C}}$) and the map $w \mapsto [w]_{\mathcal{C}}$ is a monoid morphism from A^* into $A^*/\!\sim_{\mathcal{C}}$. We denote by "$\bullet$" the monoid multiplication of $A^*/\!\sim_{\mathcal{C}}$ (note that "$\bullet$" is not to be mistaken with language concatenation). In particular, we have the following lemma for tame and $\mathcal{C}$-compatible sets.

Lemma 48. *Let $\mathbf{L}$ be a finite $\mathcal{C}$-compatible tame set. Then, for all $H, L \in \mathbf{L}$ such that $H \odot L$ is defined, we have*

$$[H \odot L]_{\mathcal{C}} = [H]_{\mathcal{C}} \bullet [L]_{\mathcal{C}}.$$

Proof. By $\mathcal{C}$-compatibility of $\mathbf{L}$, no language of $\mathbf{L}$ is empty. Take $u \in H$ and $v \in L$. Then, $uv \in HL \subseteq H \odot L$, whence $[H \odot L]_{\mathcal{C}} = [uv]_{\mathcal{C}}$. Similarly, $[H]_{\mathcal{C}} = [u]_{\mathcal{C}}$ and $[L]_{\mathcal{C}} = [v]_{\mathcal{C}}$. Therefore $[H]_{\mathcal{C}} \bullet [L]_{\mathcal{C}} = [u]_{\mathcal{C}} \bullet [v]_{\mathcal{C}} = [uv]_{\mathcal{C}} = [H \odot L]_{\mathcal{C}}$. □

We may now construct $\mathbf{H}$. Given any $G \in \mathbf{G}$ and any class K in the (finite) quotient set $A^*/\!\sim_{\mathcal{C}}$, we let $H_{G,K}$ as a name for the language $G \cap K$. Finally, we let $\mathbf{H}$ as the set of all language names $H_{G,K}$ that are nonempty for $H \in \mathbf{H}$ and $K \in A^*/\!\sim_{\mathcal{C}}$.

That $\mathbf{H}$ is $\mathcal{C}$-compatible and extends $\mathbf{G}$ is immediate from the definition. Let us define a tame multiplication for $\mathbf{H}$. Let $H_{G_1,K_1}, H_{G_2,K_2} \in \mathbf{H}$. The multiplication $H_{G_1,K_1} \odot H_{G_2,K_2}$ is defined when $G_1 \odot G_2$ is defined in $\mathbf{G}$ and in that case,

$$H_{G_1,K_1} \odot H_{G_2,K_2} = H_{G_1 \odot G_2, K_1 \bullet K_2}$$

One may verify that this multiplication is well defined (*i.e.*, $H_{G_1 \odot G_2, K_1 \bullet K_2}$ is nonempty and therefore belongs to $\mathbf{H}$) and tame. This terminates the proof.

APPENDIX D
STRATIFICATIONS AND PROOF OF LEMMA 31

In this appendix, we present a notion that we will reuse in Appendices E and G for proving Theorem 33: *stratifications*. This notion is designed to prove properties of $\mathcal{D}$-tied n-tuples and n-joins. Here we use it to prove Lemma 31.

A. Stratifications

Let us fix an arbitrary quotienting lattice $\mathcal{D}$ for the definition. A *stratification* of $\mathcal{D}$ is an infinite sequence $\mathcal{D}_0, \ldots, \mathcal{D}_k, \ldots$ of *finite* quotienting lattices such that,

$$\text{For all } k,\ \mathcal{D}_k \subseteq \mathcal{D}_{k+1} \quad \text{and} \quad \mathcal{D} = \bigcup_{k \in \mathbb{N}} \mathcal{D}_k.$$

A simple but important result is that any quotienting lattice admits a stratification (note that our assumption that we are considering regular languages only is important here).

Lemma 49. *Any quotienting lattice of regular languages admits a stratification.*

Proof. Let $\mathcal{D}$ be a quotienting lattice of languages languages. For all $k \in \mathbb{N}$, we let $\mathcal{G}_k$ as the class of all languages $L \in \mathcal{D}$ that are recognized by a nondeterministic finite automaton with less than k states. Clearly, all $\mathcal{G}_k$ are finite (but not quotienting lattices), $\mathcal{G}_k \subseteq \mathcal{G}_{k+1}$ and all $L \in \mathcal{D}$ belong to some $\mathcal{G}_k$ since $\mathcal{D}$ is a class of *regular* languages. Finally, for all $k \in \mathbb{N}$, we let $\mathcal{D}_k$ as the smallest quotienting lattice containing $\mathcal{G}_k$. Note that $\mathcal{D}_k$ remains finite thanks to Myhill-Nerode Theorem (a regular language has finitely many quotients) and the fact that unions and intersections commute with quotients. One may verify that $\mathcal{D}_0, \ldots, \mathcal{D}_k, \ldots$ is a stratification of $\mathcal{D}$. □

Remark 50. *The stratification given by Lemma 49 is generic. In Appendices E and G we work with one that is specific to the classes $Pol(\mathcal{C})$.*

For the remainder of this appendix, we assume fixed an arbitrary stratification $\mathcal{D}_0, \ldots, \mathcal{D}_k, \ldots$ of $\mathcal{D}$. We use it to define preorder relations over A^*. We then use these relations to give an alternate definition of $\mathcal{D}$-tied objects which is simpler to manipulate. Given any $k \geq 0$, and any $u, v \in A^*$, we write:

$$u \leqslant_k v \text{ if and only if for all } L \in \mathcal{D}_k,\ u \in L \Rightarrow v \in L$$

That $\leqslant_k$ is reflexive and transitive is immediate from the definition. Also observe that for all $k \in \mathbb{N}$, since $\mathcal{D}_k \subseteq \mathcal{D}_{k+1}$, the preorder $\leqslant_{k+1}$ refines $\leqslant_k$. Another useful property is that since the classes $\mathcal{D}_k$ are quotienting, $\leqslant_k$ is compatible with word concatenation.

Lemma 51. *For all $k \in \mathbb{N}$, if $u_1 \leqslant_k v_1$ and $u_2 \leqslant_k v_2$, then $u_1 u_2 \leqslant_k v_1 v_2$.*

Proof. Assume that $u_1 \leqslant_k v_1$ and $u_2 \leqslant v_2$. We prove that $u_1 u_2 \leqslant_k v_1 v_2$. Let $L \in \mathcal{D}_k$ and assume that $u_1 u_2 \in L$. We prove that $v_1 v_2 \in L$. Since $u_1 u_2 \in L$, we have $u_2 \in {u_1}^{-1} L$. By closure under quotients, we have ${u_1}^{-1} L \in \mathcal{D}_k$, hence, since $u_2 \leqslant_k v_2$, we obtain that $v_2 \in {u_1}^{-1} L$ and therefore that $u_1 v_2 \in L$. It now follows that $u_1 \in L v_2^{-1}$. Using closure under quotients again, we obtain that $L v_2^{-1} \in \mathcal{D}_k$. Therefore, since $u_1 \leqslant_k v_1$, we conclude that $v_1 \in L v_2^{-1}$ which means that $v_1 v_2 \in L$ as desired. □

The following lemma connects the preorders $\leqslant_k$ to definability in $\mathcal{D}_k$. For a word $w \in A^*$, we let $\uparrow_k w = \{u \mid w \leqslant_k u\}$ be the upper set of $\{w\}$ for $\leqslant_k$, and for a language L, we write $\uparrow_k L = \{u \mid \exists w \in L,\ w \leqslant_k u\} = \bigcup_{w \in L} \uparrow_k w$.

Lemma 52. *Let $k \in \mathbb{N}$ and let L be any language. Then, $L \in \mathcal{D}_k$ if and only if L is an upper set for $\leqslant_k$.*

Proof. If $L \in \mathcal{D}_k$, it is immediate from the definition of $\leqslant_k$ that L is an upper set for $\leqslant_k$: if $u \in L$ and $u \leqslant_k v$, then $v \in L$. Conversely, assume that L is an upper set. We first claim that for any word w, the set $\uparrow_k w$ is the (finite) intersection of all languages in $\mathcal{D}_k$ containing w:

$$\uparrow_k w = \bigcap_{\substack{K \in \mathcal{D}_k \\ w \in K}} K. \tag{3}$$

Indeed, we already know that any language in $\mathcal{D}_k$ is an upper set for $\leqslant_k$. Therefore, if $K \in \mathcal{D}_k$ contains w, then K also contains $\uparrow_k w$, which shows the left-to-right inclusion. Conversely, if u belongs to the above intersection, then u belongs to any language $K \in \mathcal{D}_k$ containing w. By definition of $\leqslant_k$, this means that $w \leqslant_k u$, *i.e.*, that $u \in \uparrow_k w$.

Since $\mathcal{D}_k$ is a finite lattice, (3) shows, in particular, that $\uparrow_k w \in \mathcal{D}_k$. It follows that there are finitely many distinct languages $\uparrow_k w$. Since L is an upper set, we have $L = \uparrow_k L = \bigcup_{w \in L} \uparrow_k w$, which by the above is actually a finite union of languages in $\mathcal{D}_k$. Since $\mathcal{D}_k$ is a lattice, this entails $L \in \mathcal{D}_k$. □

B. Connection with $\mathcal{D}$-tied objects

We now use stratifications to present alternate definitions of $\mathcal{D}$-tied n-tuples and n-joins. This new definitions will often be simpler to manipulate. In particular, we use them here to prove Lemma 31. We begin with n-tuples.

Lemma 53. *For any $n \geq 1$, an n-tuple $(L_1, \ldots, L_n)$ is $\mathcal{D}$-tied if and only if for all $k \in \mathbb{N}$, there exist $w_1 \in L_1, \ldots, w_n \in L_n$ such that $w_1 \leqslant_k \cdots \leqslant_k w_n$.*

Proof. Assume first that $(L_1, \ldots, L_n)$ is $\mathcal{D}$-tied and let $k \in \mathbb{N}$. We exhibit $w_1 \in L_1, \ldots, w_n \in L_n$ such that $w_1 \leqslant_k \cdots \leqslant_k w_n$. The proof is by induction on $n \in \mathbb{N}$. If $n = 1$, then $L_1 \neq \emptyset$ by definition and we have $w_1 \in L_1$ as desired. Otherwise, let $K = \uparrow_k L_1 = \bigcup_{w \in L_1} \uparrow_k w \in \mathcal{D}_k$. Clearly, $L_1 \subseteq K$. Hence, since $(L_1, \ldots, L_n)$ is $\mathcal{D}$-tied, we know that $(L_2, \ldots, L_n) \cap K$ is $\mathcal{D}$-tied as well. By induction, we get $w_2 \in L_2 \cap K, \ldots, w_n \in L_n \cap K$ such that $w_2 \leqslant_k \cdots \leqslant_k w_n$. Finally, by definition of K, since $w_2 \in K$, there exists $w_1 \in L_1$ such that $w_1 \leqslant_k w_2$ which terminates this direction of the proof.

We now assume that for all $k \in \mathbb{N}$, there exist $w_1 \in L_1, \ldots, w_n \in L_n$ such that $w_1 \leqslant_k \cdots \leqslant_k w_n$. Using induction on n again, we prove that $(L_1, \ldots, L_n)$ is $\mathcal{D}$-tied. If $n = 1$, then $L_1 \neq \emptyset$ since $w_1 \in L_1$, which means that it is $\mathcal{D}$-tied. Otherwise, let $K \in \mathcal{D}$ such that $L_1 \subseteq K$. We have to prove that $(L_2, \ldots, L_n) \cap K$ is $\mathcal{D}$-tied. Using induction, this amounts to exhibiting, for each $k \in \mathbb{N}$, words $w_2 \in L_2 \cap K, \ldots, w_n \in L_n \cap K$ such that $w_2 \leqslant_k \cdots \leqslant_k w_n$. So fix some integer $k \in \mathbb{N}$. Since $K \in \mathcal{D}$, there exists $h \in \mathbb{N}$ such that $K \in \mathcal{D}_h$. We define $g = max(k, h)$. By hypothesis, one can find $w_1 \in L_1, \ldots, w_n \in L_n$ such that $w_1 \leqslant_g \cdots \leqslant_g w_n$. Moreover, since $w_1 \in L_1 \subseteq K$ and $K \in \mathcal{D}_h \subseteq \mathcal{D}_g$, it follows from the definition of $\leqslant_g$ that $w_i \in K$ for all i. Hence, $w_2 \in L_2 \cap K, \ldots, w_n \in L_n \cap K$. Finally, since $g \geq k$, we have $w_2 \leqslant_k \cdots \leqslant_k w_n$. □

Lemma 53 will be very convenient when manipulating $\mathcal{D}$-tied tuples. Let us complete it with a simple observation that often makes the writing easier.

Lemma 54. *Let* $\mathbf{L}$ *be a* **finite** *set of languages and let* $n \geq 1$. *Then, there exists* $\ell \in \mathbb{N}$ *such that the following equivalence holds for each* $k \geq \ell$*:* $(L_1, \ldots, L_n) \in \mathbf{L}^n$ *is* $\mathcal{D}$*-tied if and only if there exists* $(w_1, \ldots, w_n) \in L_1 \times \cdots \times L_n$ *such that* $w_1 \leqslant_k \cdots \leqslant_k w_n$.

Proof. Call a tuple of words $(w_1, \ldots, w_n) \in L_1 \times \cdots \times L_n$ a k-*witness* for $(L_1, \ldots, L_n)$ if $w_1 \leqslant_k \cdots \leqslant_k w_n$. Since for all k, $\leqslant_{k+1}$ refines $\leqslant_k$, one may reformulate Lemma 53 as follows: $(L_1, \ldots, L_n) \in \mathbf{L}^n$ is **not** $\mathcal{D}$-tied if and only if there exists k such that $(L_1, \ldots, L_n)$ has no k'-witness, for all $k' \geq k$. It suffices to choose ℓ to be greater that all such integers k when $(L_1, \ldots, L_n)$ ranges over $\mathcal{D}$-separable n-tuples in $\mathbf{L}^n$. The finiteness of $\mathbf{L}^n$ ensures that ℓ is well-defined. □

Remark 55. *The proof of Lemma 54 is not constructive: we know that* ℓ *exists but so far, we have no way of computing it. Actually, the knowledge of* ℓ *is sufficient to decide generalized separation for* $\mathcal{D}$*. Indeed, by choice of* ℓ*, any* n*-tuple of* $(L_1, \ldots, L_n)$ *is* $\mathcal{D}$*-tied iff it is* $\mathcal{D}_\ell$*-tied, which is decidable by finiteness of* $\mathcal{D}_\ell$.

We now generalize Lemma 53 to n-joins.

Lemma 56. *Let* $n \geq 1$ *and consider an* n*-join* $(S, \mathbf{S})$*. Then the two following conditions are equivalent,*

1) $(S, \mathbf{S})$ *is* $\mathcal{D}$*-tied.*
2) *For all* $k \in \mathbb{N}$*, there exists* $w \in S$ *such that for all* $(S_1, \ldots, S_n) \in \mathbf{S}$*, there are words* $w_1 \in S_1, \ldots, w_n \in S_n$ *such that* $w \leqslant_k w_1 \leqslant_k \cdots \leqslant_k w_n$.

Proof. Assume first that $(S, \mathbf{S})$ is $\mathcal{D}$-tied and let $k \in \mathbb{N}$. We exhibit $w \in S$ as described in the lemma. For any $v \in A^*$, we denote again by $\uparrow_k v$ be the upper set $\{u \mid v \leqslant_k u\} \in \mathcal{D}_k$. Moreover, we let $\mathbf{K} = \{\uparrow_k v \mid v \in S\} \subseteq \mathcal{D}_k$. Note that $\mathbf{K}$ is finite since $\mathcal{D}_k$ is. We know that $(S, \mathbf{S})$ is $\mathcal{D}$-tied. Hence, $\mathbf{K}$ cannot be a separating cover. Since $S \subseteq \uparrow_k S = \bigcup_{K \in \mathbf{K}} K$ by definition, it follows that there exists $K \in \mathbf{K}$ such that

$$(S_1, \ldots, S_n) \cap K \text{ is } \mathcal{D}\text{-tied}$$

for any n-tuple $(S_1, \ldots, S_n) \in \mathbf{S}$. By definition, $K = \uparrow_k w$ for some $w \in S$. Let us prove that this word w satisfies Item 2 in the lemma. Let $(S_1, \ldots, S_n) \in \mathbf{S}$. Since $(S_1, \ldots, S_n) \cap \uparrow_k w$ is $\mathcal{D}$-tied, we know from Lemma 53 that there exists $w_1 \in S_1 \cap \uparrow_k w, \ldots, w_n \in S_n \cap \uparrow_k w$ such that $w_1 \leqslant_k \cdots \leqslant_k w_n$. Finally, since $w_1 \in \uparrow_k w$, it follows from the definition of $\uparrow_k w$ that $w \leqslant_k w_1$ which terminates this direction of the proof.

Assume now that the second item holds. We show that $(S, \mathbf{S})$ is $\mathcal{D}$-tied. Consider a finite set $\mathbf{K} \subseteq \mathcal{D}$ such that $S \subseteq \bigcup_{K \in \mathbf{K}} K$, our objective is to find $K \in \mathbf{K}$ such that $(S_1, \ldots, S_n) \cap K$ is $\mathcal{D}$-tied for all $(S_1, \ldots, S_n) \in \mathbf{S}$. Since $\mathbf{K}$ is finite, there exists $h \in \mathbb{N}$ such that all $K \in \mathbf{K}$ belongs to $\mathcal{D}_h$. Moreover, we may use Lemma 54 to obtain $\ell \in \mathbb{N}$ such that for all $k \geq \ell$, if $(S_1, \ldots, S_n) \in \mathbf{S}$ and $K \in \mathbf{K}$, then $(S_1, \ldots, S_n) \cap K$ is $\mathcal{D}$-tied if and only if there exist $w_1 \in L_1 \cap K, \ldots, w_n \in L_n \cap K$ such that $w_1 \leqslant_k \cdots \leqslant_k w_n$. We let $k = max(\ell, h)$ and let $w \in S$ be given by Item 2 for this k. Since $S \subseteq \bigcup_{K \in \mathbf{K}} K$, there exists some language in $\mathbf{K}$ containing w. We let $K \in \mathbf{K}$ be this language.

We now have to prove that for all $(S_1, \ldots, S_n) \in \mathbf{S}$, $(S_1, \ldots, S_n) \cap K$ is $\mathcal{D}$-tied. By definition of w in Item 2, there are $w_1 \in S_1, \ldots, w_n \in S_n$ such that $w \leqslant_k w_1 \leqslant_k \cdots \leqslant_k w_n$. Moreover, since $w \in K$ and $K \in \mathcal{D}_h \subseteq \mathcal{D}_k$, it follows that $w_i \in K$ for all i. Altogether, we have $w_1 \in S_1 \cap K, \ldots, w_n \in S_n \cap K$ such that $w \leqslant_k w_1 \leqslant_k \cdots \leqslant_k w_n$. Since $k \geq \ell$, we get by choice of ℓ that $(S_1, \ldots, S_n) \cap K$ is $\mathcal{D}$-tied. □

We finish with the proof of Lemma 31, which is now simple thanks to the results that we have just established.

Lemma 31. *Assume that* $\mathcal{D}$ *is a quotienting lattice and let* $\mathbf{L}$ *be a tame set. Then, for any* $n \geq 1$*, the two following properties hold:*

1) $\mathcal{T}^n_{\mathcal{D}}[\mathbf{L}]$ *is closed under multiplication: if* $\overline{S}, \overline{T} \in \mathcal{T}^n_{\mathcal{D}}[\mathbf{L}]$*, then* $\overline{S} \odot \overline{T} \in \mathcal{T}^n_{\mathcal{D}}[\mathbf{L}]$ *when defined.*
2) $\mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$ *is closed under multiplication: if* $(S, \mathbf{S}), (T, \mathbf{T}) \in \mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$*, then* $(S, \mathbf{S}) \odot (T, \mathbf{T}) \in \mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$ *when defined.*

Proof. We have two items to prove. We start with $\mathcal{T}^n_{\mathcal{D}}[\mathbf{L}]$. Let $(S_1, \ldots, S_n), (T_1, \ldots, T_n) \in \mathcal{T}^n_{\mathcal{D}}[\mathbf{L}]$. We use Lemma 53 to prove that $(S_1, \ldots, S_n) \odot (T_1, \ldots, T_n) \in \mathcal{T}^n_{\mathcal{D}}[\mathbf{L}]$. Let $k \in \mathbb{N}$. By hypothesis, we obtain words $u_1 \in S_1, \ldots, u_n \in S_n$ and $v_1 \in T_1, \ldots, v_n \in T_n$ satisfying the inequalities $u_1 \leqslant_k \cdots \leqslant_k u_n$ and $v_1 \leqslant_k \cdots \leqslant_k v_n$. Since $\leqslant_k$ is compatible with concatenation (see Lemma 51), we get that $u_1 v_1 \leqslant_k \cdots \leqslant_k u_n v_n$. Finally, by definition of tame multiplications, $S_i T_i \subseteq S_i \odot T_i$ for all i and therefore $u_i v_i \in S_i \odot T_i$. We conclude as desired that $(S_1, \ldots, S_n) \odot (T_1, \ldots, T_n) \in \mathcal{T}^n_{\mathcal{D}}[\mathbf{L}]$.

We turn to the second item. Let $(S, \mathbf{S}), (T, \mathbf{T}) \in \mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$. We prove that $(S \odot T, \mathbf{S} \odot \mathbf{T}) \in \mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$ by showing that it satisfies Item 2 in Lemma 56. Let $k \in \mathbb{N}$. Since $(S, \mathbf{S}), (T, \mathbf{T}) \in \mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$, we get $u \in S$ and $v \in T$ such that,

- For all $(S_1, \ldots, S_n) \in \mathbf{S}$, there are $u_1 \in S_1, \ldots, u_n \in S_n$ such that $u \leqslant_k u_1 \leqslant_k \cdots \leqslant_k u_n$.
- For all $(T_1, \ldots, T_n) \in \mathbf{T}$, there are $v_1 \in T_1, \ldots, v_n \in T_n$ such that $v \leqslant_k v_1 \leqslant_k \cdots \leqslant_k v_n$.

Consider $w = uv \in ST \subseteq S \odot T$. For any $(R_1, \ldots, R_n) \in \mathbf{S} \odot \mathbf{T}$, there exists $(S_1, \ldots, S_n) \in \mathbf{S}$ and $(T_1, \ldots, T_n) \in \mathbf{T}$ such that $R_i = S_i \odot T_i$ for all i. By the two items above, we get $u_1 \in S_1, \ldots, u_n \in S_n$ such that $u \leqslant_k u_1 \leqslant_k \cdots \leqslant_k u_n$ and $v_1 \in T_1, \ldots, v_n \in T_n$ such that $v \leqslant_k v_1 \leqslant_k \cdots \leqslant_k v_n$. It is now immediate that for all i, $u_i v_i \in S_i T_i \subseteq S_i \odot T_i$ and by Lemma 51 that,

$$w = uv \leqslant_k u_1 v_1 \leqslant_k \cdots \leqslant_k u_n v_n$$

We conclude that $(S \odot T, \mathbf{S} \odot \mathbf{T}) \in \mathcal{J}^n_{\mathcal{D}}[\mathbf{L}]$ again by Lemma 56, which terminates the proof. □

APPENDIX E
SOUNDNESS IN THEOREM 33

In this section, we prove that our algorithm for computing $Pol(\mathcal{C})$-tied n-joins is sound. Recall that we assume fixed an arbitrary finite quotienting Boolean algebra $\mathcal{C}$. Moreover, for all $n \geq 1$, $\mathcal{T}^n[\mathbf{L}]$ denotes $\mathcal{T}^n_{Pol(\mathcal{C})}[\mathbf{L}]$ and $\mathcal{J}^n[\mathbf{L}]$ denotes

$\mathcal{J}^n_{Pol(\mathcal{C})}[\mathbf{L}]$. Our objective is to prove the right to left inclusion in Theorem 33. For any $n \geq 1$ and any tame and $\mathcal{C}$-compatible multiset $\mathbf{L}$, we prove,

$$Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}]) \subseteq \mathcal{J}^n[\mathbf{L}]$$

By definition of Sat^n this amounts to proving that $\mathcal{J}^n[\mathbf{L}]$ contains $\mathcal{J}^n_{triv}[\mathbf{L}]$ and is closed under downset, multiplication and Operation 3. Since $Pol(\mathcal{C})$ is a quotienting lattice (see Fact 5), we already proved that the three first properties hold in Lemmas 29, 30 and 31. Therefore, we may concentrate on proving that $Pol(\mathcal{C})$ is closed under Operation 3.

Our argument is based on a special stratification (as defined in Appendix D) of $Pol(\mathcal{C})$. We begin with its definition and then use it to prove the result. In particular, note that we reuse the stratification in Appendix G.

A. Stratifying $Pol(\mathcal{C})$

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

- When $k = 0$, we simply define $Pol_0(\mathcal{C}) = \mathcal{C}$.
- When $k \geq 1$, we define $Pol_k(\mathcal{C})$ as the smallest lattice which contains $Pol_{k-1}(\mathcal{C})$ and such for any $L_1, L_2 \in Pol_{k-1}(\mathcal{C})$ and any $a \in A$,

$$L_1 a L_2 \in Pol_k(\mathcal{C})$$

Observe that by definition, it is immediate that $Pol_k(\mathcal{C})$ is a finite lattice and that $Pol_k(\mathcal{C}) \subseteq Pol_{k+1}(\mathcal{C})$ for any $k \in \mathbb{N}$. Moreover, since $Pol(\mathcal{C})$ is by definition the smallest lattice containing $\mathcal{C}$ and closed under marked concatenation,

$$Pol(\mathcal{C}) = \bigcup_{k \geq 0} Pol_k(\mathcal{C})$$

Hence, it suffices to prove that all $Pol_k(\mathcal{C})$ are quotienting to obtain that this is a stratification of $Pol(\mathcal{C})$.

Lemma 57. *For any $k \in \mathbb{N}$, $Pol_k(\mathcal{C})$ is a quotienting lattice.*

Proof. We use induction on k to prove closure under left quotients (the proof for right quotients is symmetric). When $k = 0$, this is by hypothesis on $\mathcal{C}$ since $Pol_0(\mathcal{C}) = \mathcal{C}$. Assume now that $k \geq 1$. Let $L \in Pol_k(\mathcal{C})$ and $w \in A^*$, we prove that $w^{-1}L \in Pol_k(\mathcal{C})$. By definition of $Pol_k(\mathcal{C})$ and since quotients commute with unions and intersections, we only have two cases to consider: $L \in Pol_{k-1}(\mathcal{C})$ and $L = L_1 a L_2$ with $L_1, L_2 \in Pol_{k-1}(\mathcal{C})$. When $L \in Pol_{k-1}(\mathcal{C})$, this by induction. Assume now that $L = L_1 a L_2$ with $L_1, L_2 \in Pol_{k-1}(\mathcal{C})$ and let $K \subseteq A^*$ be the finite set of all words v such $w = uav$ for some $u \in L_1$. One may verify that:

$$w^{-1}L = (w^{-1}L_1)aL_2 \cup \bigcup_{v \in K} v^{-1}L_2$$

Since $w^{-1}L_1$ and all languages $v^{-1}L_2$ belong to $Pol_{k-1}(\mathcal{C})$ by induction, we conclude that $w^{-1}L$ is a union of languages in $Pol_k(\mathcal{C})$ and thus belongs to $Pol_k(\mathcal{C})$ itself. □

We work with this stratification for the remainder of the paper (more precisely, we use it here and in Appendix G). In particular, we now write $\leqslant_k$ to denote the preorder associated to $Pol_k(\mathcal{C})$ for some $k \in \mathbb{N}$ (see Appendix D for details). Recall that we proved in Lemma 51 that for all $k \in \mathbb{N}$, $\leqslant_k$ is compatible with multiplication.

We now prove a property which is specific to this particular stratification and central for the soundness proof. We start with a simple fact which defines a constant (depending on $\mathcal{C}$) that we need in order to state the property.

Fact 58. *There exists $p \in \mathbb{N}$ such that for any u, $u^p \sim_{\mathcal{C}} u^{2p}$.*

Proof. Recall that $\sim_{\mathcal{C}}$ is a congruence of finite index for concatenation (see Lemma 23). Hence, the quotient set $A^*/\sim_{\mathcal{C}}$ is a finite semigroup and it has an idempotent power ω. It suffices to choose p as this power. □

We denote by p the constant defined in Fact 58 for the remainder of Appendix E. We may state our property.

Lemma 59. *For any $k \in \mathbb{N}$, any $h, h_1, h_2 \geq 3^{k+1} - 1$ and any $u, v \in A^*$ such that $u^p \sim_{\mathcal{C}} v$, we have $u^{ph} \leqslant_k u^{ph_1} v u^{ph_2}$.*

Proof. Let $k \in \mathbb{N}$, $h, h_1, h_2 \geq 3^{k+1}-1$ and $u, v \in A^*$ such that $u^p \sim_{\mathcal{C}} v$. We prove that $u^{ph} \leqslant_k u^{ph_1} v u^{ph_2}$ using induction on k. When $k = 0$, we have $h, h_1, h_2 \geq 2$ and $u^p \sim_{\mathcal{C}} v$. By definition of p in Fact 58 and since $\sim_{\mathcal{C}}$ is compatible with concatenation, we know that $u^{ph} \sim_{\mathcal{C}} u^{ph_1} u^p u^{ph_2}$. Moreover, since $u^p \sim_{\mathcal{C}} v$, we have $u^{ph} \sim_{\mathcal{C}} u^{ph_1} v u^{ph_2}$. Finally, since $Pol_0(\mathcal{C}) = \mathcal{C}$, this implies $u^{ph} \leqslant_0 u^{ph_1} v u^{ph_2}$.

We now assume that $k \geq 1$. Our goal is to prove that for any $K \in Pol_k(\mathcal{C})$, the following implication holds:

$$u^{ph} \in K \quad \Rightarrow u^{ph_1} v u^{ph_2} \in K. \tag{4}$$

We prove that (4) holds for all $K \in Pol_{k-1}(\mathcal{C})$ and all languages $K = K_1 a K_2$ for $K_1, K_2 \in Pol_{k-1}(\mathcal{C})$ and $a \in A$. Since languages in $Pol_k(\mathcal{C})$ are built from these two kinds using finitely many unions and intersections, it will then be simple to verify that (4) actually holds for all $K \in Pol_k(\mathcal{C})$.

That (4) holds when $K \in Pol_{k-1}(\mathcal{C})$ is immediate by induction on k. Hence, we may concentrate on the case when $K = K_1 a K_2$ for $K_1, K_2 \in Pol_{k-1}(\mathcal{C})$ and $a \in A$. Assume that $u^{ph} \in K = K_1 a K_2$. We present a decomposition of u^{ph} witnessing this fact. In particular, we isolate the factor u^p of u^{ph} which contains the letter a in this decomposition. We have,

$$u^{ph} = u^{pn_1} w_1 a w_2 u^{pn_2}$$

with $h = n_1 + 1 + n_2$, $w_1 a w_2 = u^p$, $u^{pn_1} w_1 \in K_1$ and $w_2 u^{pn_2} \in K_2$.

By hypothesis, $h \geq 3^{k+1} - 1$. Hence, since $h = n_1 + 1 + n_2$, either $n_1 \geq 3^k - 1$ or $n_2 \geq 3^k - 1$ (possibly both). By symmetry, we assume that $n_1 \geq 3^k - 1$. The argument is based on the following claim which we use to build a decomposition of $u^{ph_1} u^p u^{ph_2}$ witnessing its membership in $K_1 a K_2$.

Claim. *There exist $\ell_1, \ell_2 \in \mathbb{N}$ such that $h_2 = \ell_1 + 1 + \ell_2$, $u^{pn_1} \leqslant_{k-1} u^{ph_1} v u^{p\ell_1}$ and $u^{pn_2} \leqslant_{k-1} u^{p\ell_2}$*

Before proving the claim, let us finish the argument. Since $\leqslant_{k-1}$ is compatible with concatenation, we have

$u^{pn_1}w_1 \leqslant_{k-1} u^{ph_1}vu^{p\ell_1}w_1$ and $w_2u^{pn_2} \leqslant_{k-1} w_2u^{p\ell_2}$. Hence, since $K_1, K_2 \in Pol_{k-1}(\mathcal{C})$, $u^{pn_1}w_1 \in K_1$ and $w_2u^{pn_2} \in K_2$, we get that $u^{ph_1}vu^{p\ell_1}w_1 \in K_1$ and $w_2u^{p\ell_2} \in K_2$. We may now conclude that

$$\begin{array}{rcll} u^{ph_1}vu^{ph_2} & = & u^{ph_1}vu^{p\ell_1}u^pu^{p\ell_2} & \\ & = & u^{ph_1}vu^{p\ell_1}w_1aw_2u^{p\ell_2} & \in \quad K_1aK_2 = K. \end{array}$$

This proves that (4) holds and terminates the proof of Lemma 59. It remains to prove the claim. There are two cases depending on whether $n_2 \geq 3^k - 1$ or not.

Assume first that $n_2 \geq 3^k - 1$. Since $h_2 \geq 3^{k+1} - 1$, there are $j, j', j'' \geq 3^k - 1$ such that $h_2 = j + j' + 1 + j''$. We let $\ell_1 = j$ and $\ell_2 = j' + 1 + j''$. We now obtain from induction that the following holds:

$$\begin{array}{lll} u^{pn_1} & \leqslant_{k-1} & u^{ph_1}vu^{p\ell_1} \\ u^{pn_2} & \leqslant_{k-1} & u^{pj'}u^pu^{pj''} = u^{p\ell_2} \end{array}$$

This proves the claim for the case $n_2 \geq 3^{k+1} - 1$. Finally, assume that $n_2 < 3^k - 1$. We let $\ell_2 = n_2$ and $\ell_1 = h_2 - 1 - \ell_2$. Clearly, $\ell_1 \geq 3^k - 1$. Hence, we get $u^{pn_1} \leqslant_{k-1} u^{ph_1}vu^{p\ell_1}$ from induction. Furthermore, $u^{pn_2} \leqslant_{k-1} u^{p\ell_2}$ is immediate since $n_2 = \ell_2$ by definition. □

B. Soundness proof

We now prove that $\mathcal{J}^n[\mathbf{L}]$ is closed under Operation 3. Let $(E, \mathbf{E}) \in \mathcal{J}^n[\mathbf{L}]$ be an idempotent. We have to show that,

$$(E, \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}} \odot \mathbf{E}) \in \mathcal{J}^n[\mathbf{L}]$$

By Lemma 56 this amount to proving that for all $k \in \mathbb{N}$, there exists $w \in E$, such that for any

$$(S_1, \ldots, S_n) \in \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}} \odot \mathbf{E},$$

one may find words $w_1, \ldots, w_n$ such that $w_i \in S_i$ for all i and $w \leqslant_k w_1 \leqslant_k \cdots \leqslant_k w_n$. We fix $k \in \mathbb{N}$ for the remainder of the proof. The argument is based on Lemma 59.

Let us first choose $w \in E$. Since $(E, \mathbf{E}) \in \mathcal{J}^n[\mathbf{L}]$ by hypothesis, Lemma 56 yields $u \in E$ such that for any $(E_1, \ldots, E_n) \in \mathbf{E}$, there exist $u_1 \in E_1, \ldots, u_n \in E_n$ such that $u \leqslant_k u_1 \leqslant_k \cdots \leqslant_k u_n$. We let $p \in \mathbb{N}$ as the constant defined in Fact 58 and we choose $w = u^{p3^{k+1}}$. It remains to prove this choice satisfies the above property.

First, we verify that $w \in E$. Since E is idempotent, the tame multiplication $E^{p3^{k+1}}$ is equal to E. Hence, since $u \in E$, we get $w = u^{p3^{k+1}} \in E$. We now turn to the second property.

Let $(S_1, \ldots, S_n) \in \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}} \odot \mathbf{E}$. We have to find $w_1, \in S_1, \ldots, w_n \in S_n$ such that $w \leqslant_k w_1 \leqslant_k \cdots \leqslant_k w_n$. Since $\mathbf{E}$ is idempotent, we have,

$$\mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}} \odot \mathbf{E} = \mathbf{E}^{p3^{k+1}} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}} \odot \mathbf{E}^{p3^{k+1}}$$

Therefore, by definition of tame multiplications, $(S_1, \ldots, S_n)$ is the multiplication of $2p3^{k+1} + 1$ elements:

$$(S_1, \ldots, S_n) = \overline{R_1} \odot \cdots \odot \overline{R_{p3^{k+1}}} \odot \overline{H} \odot \overline{T_1} \odot \cdots \odot \overline{T_{p3^{k+1}}}$$

Such that $\overline{R_j}, \overline{T_j} \in \mathbf{E}$ for all j and $\overline{H} \in \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}}$. Let us name the elements in these n-tuples. Let

$$\begin{array}{rcll} \overline{H} & = & (H_1, \ldots, H_n) & \\ \overline{R_j} & = & (R_{j,1}, \ldots, R_{j,n}) & \text{for all } j \leq p3^{k+1} \\ \overline{T_j} & = & (T_{j,1}, \ldots, T_{j,n}) & \text{for all } j \leq p3^{k+1} \end{array}$$

Since $(H_1, \ldots, H_n) \in \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}}$, we may use Lemma 53 to obtain $v_1 \in H_1, \ldots, v_n \in H_n$ such that $v_1 \leqslant_k \cdots \leqslant_k v_n$. Moreover, since $H_i \subseteq [E]_{\mathcal{C}}$ for all i, we have $v_i \in [E]_{\mathcal{C}}$. Hence, since $u^p \in E \subseteq [E]_{\mathcal{C}}$, we have $u^p \sim_{\mathcal{C}} v_1$. Thus, we obtain from Lemma 59 that,

$$w = u^{p3^{k+1}} \leqslant_k u^{p3^{k+1}}v_1u^{p3^{k+1}}$$

Hence, it now suffices to exhibit $w_1, \in S_1, \ldots, w_n \in S_n$ such that,

$$u^{p3^{k+1}}v_1u^{p3^{k+1}} \leqslant_k w_1 \leqslant_k \cdots \leqslant_k w_n$$

By transitivity, this will imply $w \leqslant_k w_1 \leqslant_k \cdots \leqslant_k w_n$ as desired. We know that for all $j \leq p3^{k+1}$, $\overline{R_j}, \overline{T_j} \in \mathbf{E}$. Therefore, by definition of u, we know that for all $j \leq p3^{k+1}$,

- We have $x_{j,1} \in R_{j,1}, \ldots, x_{j,n} \in R_{j,n}$ such that,
$$u \leqslant_k x_{j,1} \leqslant_k \cdots \leqslant_k x_{j,n}$$
- We have $y_{j,1} \in T_{j,1}, \ldots, y_{j,n} \in T_{j,n}$ such that,
$$u \leqslant_k y_{j,1} \leqslant_k \cdots \leqslant_k y_{j,n}$$

For all $i \leq n$, define $w_i = x_{1,i} \cdots x_{p3^{k+1},i}v_iy_{1,i} \cdots y_{p3^{k+1},i}$. Clearly, we have $w_i \in S_i$ for all i by definition of tame multiplications. Moreover, since $\leqslant_k$ is compatible with concatenation, we obtain,

$$u^{p3^{k+1}}v_1u^{p3^{k+1}} \leqslant_k w_1 \leqslant_k \cdots \leqslant_k w_n$$

This finishes the soundness proof.

APPENDIX F
GENERALIZED FACTORIZATION FORESTS

In this appendix, we present a combinatorial result which will play a crucial part in the proofs of Appendices G and H. This result is a generalized variant of Simon's factorization forest theorem [32] (in fact, this classical theorem is used as an ingredient in our proof of this variant).

We introduce a generalized notion of "factorization forest". Let S be a *finite partial semigroup* and T be a *finite semigroup*. Furthermore, let $\gamma : S \to T$ be an arbitrary map.

A γ-factorization forest is an ordered unranked tree whose nodes are labeled by pairs $(s, t) \in S \times T$. There can only be three distinct kinds of nodes in the tree:

- *Leaves* which have no children. The label of this leaf is any $(s, t) \in S \times T$.
- *Binary nodes* which have exactly two children. Moreover, if (s_1, t_1) and (s_2, t_2) are the labels of these children, then the binary node has label (s_1s_2, t_1t_2) (in particular, note that s_1s_2 has to be defined).
- *Idempotent nodes* which have three or more children. Moreover, if $(s_1, t_1), \ldots, (s_k, t_k)$ are their labels, the following conditions must be satisfied:

1) $s_1 = \cdots = s_k$ and it is an idempotent e of S (ee is defined and equal to e).
2) $t_1 = t_k$ and it is an idempotent f of T.
3) The idempotent node has label $(e, f\gamma(e)f)$.

We associate a nonempty word $x \in (S \times T)^+$ to each γ-factorization forest. Let $(s_1, t_1), \ldots, (s_k, t_k) \in S \times T$ be the labels of all leaves in the forest from left to right. The associated word is $x = (s_1, t_1) \cdots (s_k, t_k)$.

Given an arbitrary word $w = (s_1, t_1) \cdots (s_n, t_n)$, one may verify that there exists a γ-factorization forest associated to w if and only if the multiplication $s_1 \cdots s_n$ is defined in S (we say that "the evaluation of w is defined"). Indeed, such a forest is easily built with binary nodes only.

Remark 60. *In the particular case where the internal nodes of a forest are all binary, the root label is $(s_1 \cdots s_n, t_1 \cdots t_n)$, where $(s_1, t_1), \ldots, (s_k, t_k) \in S \times T$ is the sequence of labels of all leaves read from left to right. This is not the case in general: in the root label (s, t) of a γ-factorization forest, s is determined by the associated word w only, but because of idempotent nodes, this is not the case for t.*

We may now state our generalized factorization forest theorem. We call *height* of a γ-factorization forest the largest number $h \in \mathbb{N}$ such that the forest contains a branch with h inner nodes (binary or idempotent).

Theorem 61. *Let S be a finite partial semigroup, T be a finite semigroup and $\gamma : S \to T$ be some map. Then, any $w \in (S \times T)^+$ whose evaluation is defined admits a γ-factorization forest whose height is smaller than $3(|S| + 1)(|T| + 2)$.*

The remainder of this appendix is devoted to proving Theorem 61. We begin with a special case and then use the classical factorization forest theorem of Simon to generalize. For the proof, we assume that S, T and γ are fixed.

Special case. Let $e \in S$ be an idempotent. We call e-word an element of $(\{e\} \times T)^+$. Our special case is as follows:

Lemma 62. *Let $e \in S$ be an idempotent. Then any e-word admits a γ-factorization forest of height $|T| + 2$.*

We fix an idempotent $e \in S$ throughout this proof. Moreover, we let k be the number of elements in T that are **not** idempotent. We use an induction on a parameter of e-words that we define now. Let $w = (e, t_1) \cdots (e, t_n)$ be an e-word of length $n \geq 1$. Given $i \leq n$ and an idempotent $f \in T$, we say that f *occurs at position i in w* when there exists $\ell \leq 2^k - 1$ such that,

$$t_i \cdots t_{i+\ell} = f.$$

We call *index of w* the number of distinct idempotents $f \in T$ which occur at some position i of w. Note that the index of any word is necessarily bounded by the number of idempotents in T, *i.e.*, by $|T| - k$. We prove that any e-word w admits a γ-factorization forest of height $k + d + 2$, where d is the index of w. Since $d \leq |T| - k$ by the above remark, this will entail that any e-word admits a forest of height smaller than $k + (|T| - k) + 2 = |T| + 2$, whence Lemma 62 will follow.

The proof that any e-word w admits a γ-factorization forest of height $k + d + 2$ goes by induction on d. Before we start the induction, let us present two facts that we use multiple times. The first one can be used to build γ-factorization forests for "small" e-words.

Fact 63. *Let $h \in \mathbb{N}$ and let $w = (e, t_1) \cdots (e, t_n)$ be an e-word of length $n \leq 2^h$. Then w admits a γ-factorization forest of height at most h and whose root label is $(e, t_1 \cdots t_n)$.*

Proof. It suffices to use only binary nodes in the construction and to proceed by dichotomy on w. □

The second fact is a corollary of a theorem presented in [13]. This theorem states that for any finite semigroup T, if k is the number of nonidempotent elements in T, then for any $\ell = 2^k$ elements $t_1, \ldots, t_\ell \in T$, there exist $i < j \leq \ell$ such that $t_i t_{i+1} \cdots t_j$ is idempotent. This yields the following statement.

Fact 64. *Let w be an e-word such that $|w| \geq 2^k$. There exists an idempotent $f \in T$ occurring at a position $i \leq 2^k$ in w.*

We may now proceed with the induction. Let $w = (e, t_1) \cdots (e, t_n)$ be an e-word and d be its index. Using induction on d, we build a γ-factorization forest for w whose height is at most $k + d + 2$.

If $d = 0$, then $n < 2^k$ by Fact 64. Hence, we may build a forest of height at most $k < k + 2$ for w using Fact 63. We now assume that $d \geq 1$. It follows from Fact 64 that an idempotent f occurs at a position $i \leq 2^k$ in w. Hence, there exists $\ell \leq 2^k - 1$ such that $t_i \cdots t_{i+\ell} = f$. We define $u = (e, t_1) \cdots (e, t_{i-1})$, $v = (e, t_i) \cdots (e, t_{i+\ell})$ and $w' = (e, t_{i+\ell+1}) \cdots (e, t_n)$. By definition, we have $w = uvw'$.

We distinguish two cases depending on whether f occurs at some position in w'. If not, then the index of w' is smaller than $d - 1$. Therefore, it admits a forest of height smaller than $k + d + 1$ by induction. Moreover, uv has length smaller than 2^{k+1} by definition. Hence, it admits a forest of height smaller than $k + 1$ by Fact 63. These two forests may be combined with a binary node into a single one for $w = uvw'$ whose height is bounded by $k + d + 2$.

Otherwise, f occurs at some position of w'. We let j be the *rightmost* such position in w. Hence, we have $\ell' \leq 2^k - 1$ such that $t_j \cdots t_{j+\ell'} = f$. We let $z = (e, t_{i+\ell+1}) \cdots (e, t_{j-1})$, $v' = (e, t_j) \cdots (e, t_{j+\ell'})$ and $w'' = (e, t_{j+\ell'+1}) \cdots (e, t_n)$. By definition, $w = uvzv'w''$ and the following holds.

1) Since the length of u is smaller than 2^k, it admits a forest of height smaller than k by Fact 63.
2) Since v and v' are of length smaller than 2^k, they admit forests of height smaller than k by Fact 63. Moreover, the idempotent (e, f) is the root label of both forests by definition of v, v'. Hence, we may combine them with an idempotent node into a forest of height smaller than $k + 1$ for vzv' (the leftmost and rightmost children are the forests of v and v', and the others are leaves for the letters in z). Its root label is $(e, f\gamma(e)f)$.

3) By choice of j, the index of w'' is at most $d-1$. Therefore, w'' admits a γ-factorization forest of height smaller than $k+d+1$ by induction.

One may now combine the forests for u and vzv' into a forest for $uvzv'$ of height at most $k+2$ with a binary node. This forest can then be combined with that of w'' with another binary node. This yields a forest for $w = uvzv'w''$ whose height is bounded by $k+d+2$, as desired.

General case. We now turn to the general case, which requires considering classical factorization forests. Since S is a partial semigroup, one may turn it into a true semigroup by adding a zero. We denote by $S^0 = S \cup \{0\}$ the semigroup obtained by adding a new element 0 to S and extending the multiplication of S as follows. For any $s \in S^0$, $s0 = 0s = 0$ and for any $s, s' \in S$, if ss' is undefined in S, we now set $ss' = 0$ in S^0. We call *S^0-factorization forest* an ordered unranked tree whose nodes are labeled by elements of S^0. There can only be three distinct kinds of nodes:

- *Leaves*, which have no children. The label is any $s \in S^0$.
- *Binary nodes*, which have two children. If s_1 and s_2 are their labels, then the binary node has label $s_1 s_2$.
- *Idempotent nodes*, which have three or more children. All children must have the same label $e \in S^0$, which must be idempotent and which is also the label of the node.

We associate a nonempty word $x \in (S^0)^+$ to each S^0-factorization forest. Let $s_1, \ldots, s_k \in S^0$ be the labels of all leaves in the forest from left to right. The associated word of length k is $x = s_1 \cdots s_k$. Finally, the *height* of an S^0-factorization forest is the largest number $h \in \mathbb{N}$ such that it contains a branch with h inner nodes (binary or idempotent).

Observe that for any word $w \in (S \times T)^+$, we may associate a word $\pi(w) \in S^+ \subseteq (S^0)^+$ by discarding the second component of each letter. We prove the following lemma.

Lemma 65. *Let $h \in \mathbb{N}$ and $w \in (S \times T)^+$ such that $\pi(w)$ admits an S^0-factorization forest of height smaller than h and whose root label belongs to S. Then w admits a γ-factorization forest of height smaller than $h(|T|+2)$.*

Theorem 61 is a simple consequence of Lemma 65 and the factorization forest theorem of Simon [32] (we use a bound that was proved later in [15], [7]). Indeed, let $w \in (S \times T)^+$ whose evaluation is defined. It follows from the factorization forest theorem of Simon that $\pi(w)$ admits an S^0-factorization forest of height smaller than $3|S^0| = 3(|S|+1)$. Moreover, since the evaluation of w is defined, the root of this forest belongs to S and it follows from Lemma 65 that w admits a γ-factorization forest of height at most $3(|S|+1)(|T|+2)$ as desired.

It remains to prove Lemma 65. Let $h \in \mathbb{N}$ and $w \in (S \times T)^+$ such that $\pi(w)$ admits an S^0-factorization forest of height at most h and whose root label belongs to S. We denote this forest by $\mathcal{F}$. We use induction on h. If $h = 0$, then $\pi(w)$ has a single letter. Hence, this is the same for w and it admits a γ-factorization forest of height 0. We now assume that $h \geq 1$. There are two cases depending on the root of $\mathcal{F}$.

If the root is a binary node, then w can be decomposed as $w = w_1 w_2$ such that $\pi(w_1)$ and $\pi(w_2)$ admit S^0-factorization forests of height at most $h-1$. Moreover, the root labels of these forests must belong to S since their product (the root label of $\mathcal{F}$) does. By induction, it follows that w_1 and w_2 admit γ-factorization forests of height at most $(h-1)(|T|+2)$. Combining them with a binary node yields a forest for w whose height is at most $(h-1)(|T|+2)+1 \leq h(|T|+2)$.

Finally, assume that the root of $\mathcal{F}$ is idempotent and let $e \in S$ be its idempotent label. In that case, w admits a decomposition $w = w_1 \cdots w_k$ such that for all i, $\pi(w_i)$ admits an S^0-factorization forest of height at most $h-1$ whose root label is $e \in S$. By induction, it follows that for all i, w_i admits a γ-factorization forest $\mathcal{F}_i$ of height at most $(h-1)(|T|+2)$. Moreover, the root label of $\mathcal{F}_i$ is (e, t_i) for some $t_i \in T$. Consider the e-word $w' = (e, t_1) \cdots (e, t_k)$. We know from Lemma 62 that w' admits a γ-factorization of height at most $|T|+2$. We now replace each leaf (e, t_i) in this forest by the forest $\mathcal{F}_i$. By definition, this yields a γ-factorization forest for w and its height is at most:

$$|T| + 2 + (h-1)(|T|+2) = h(|T|+2).$$

This terminates the proof of Lemma 65.

Appendix G
Completeness in Theorem 33

In this appendix, we prove the completeness of our algorithm for computing $Pol(\mathcal{C})$-tied n-joins. For the whole section, we assume fixed a finite quotienting Boolean algebra $\mathcal{C}$, a tame $\mathcal{C}$-compatible multiset $\mathbf{L}$, and a natural number $n \geq 1$. For the sake of improved readability, we write $\mathcal{T}^n[\mathbf{L}]$ for $\mathcal{T}^n_{Pol(\mathcal{C})}[\mathbf{L}]$ and $\mathcal{J}^n[\mathbf{L}]$ for $\mathcal{J}^n_{Pol(\mathcal{C})}[\mathbf{L}]$. Our goal is to prove,

$$\mathcal{J}^n[\mathbf{L}] \subseteq Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$$

To present the construction, we first need to introduce a new notion: filterings. We begin with this definition.

A. Filterings

Given any language K, we call *filtering* of $\mathbf{L}$ by K the set $\langle \mathbf{L}|K \rangle \in 2^{\mathbf{L}^n}$ of all n-tuples $(L_1, \ldots, L_n) \in \mathbf{L}^n$ such that $(L_1, \ldots, L_n) \cap K$ is $Pol(\mathcal{C})$-tied (*i.e.*, not $Pol(\mathcal{C})$-separable).

Remark 66. *While this is not apparent on the notation it is important to keep in mind that $\langle \mathbf{L}|K \rangle$ depends on both n and $Pol(\mathcal{C})$ (which are fixed throughout the whole section).*

We may now connect filterings to the result that we want to prove. We do it with the following property.

Proposition 67. *For each $L \in \mathbf{L}$, there exists a finite set of languages $\mathbf{K}_L \subseteq Pol(\mathcal{C})$ which satisfies the two following properties:*

1) $L \subseteq \bigcup_{K \in \mathbf{K}_L} K$.
2) For all $K \in \mathbf{K}_L$, $(L, \langle \mathbf{L}|K \rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$.

Before we prove Proposition 67, we use it to finish the proof of Theorem 33. Let $(L, \mathbf{S}) \in \mathcal{J}^n[\mathbf{L}]$, we prove that $(L, \mathbf{S}) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$. Since $(L, \mathbf{S})$ is $Pol(\mathcal{C})$-tied, the set

$\mathbf{K}_L \subseteq Pol(\mathcal{C})$ given in the proposition cannot be a separating cover. Since $L \subseteq \bigcup_{K \in \mathbf{K}_L} K$ by the first item, it follows that there exists $K \in \mathbf{K}_L$ such that $\overline{T} \cap K$ is $Pol(\mathcal{C})$-tied for all $\overline{T} \in \mathbf{S}$. By definition of filterings, this means that $\mathbf{S} \subseteq \langle \mathbf{L}|K\rangle$. Hence, since $(L, \langle \mathbf{L}|K\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$ by the second item in the lemma, we conclude using closure under downset in the definition of Sat^n that $(L, \mathbf{S}) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$ as desired. This terminates the proof of Theorem 33.

Remark 68. *The proof of Proposition 67 is constructive: we actually build the sets $\mathbf{K}_L \subseteq Pol(\mathcal{C})$. This is of particular interest. Indeed, since we now know that $Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}]) = \mathcal{J}^n[\mathbf{L}]$, it is simple to verify from the statement that for any $(L, \mathbf{S}) \in \mathbf{L} \times 2^{\mathbf{L}^n}$ which is $Pol(\mathcal{C})$-separable (*i.e.*, not $Pol(\mathcal{C})$-tied), the set $\mathbf{K}_L \subseteq Pol(\mathcal{C})$ is a separating cover.*

The remainder of this appendix is now devoted to proving Proposition 67: we build the finite sets $\mathbf{K}_L \subseteq Pol(\mathcal{C})$ that it describes. The construction is based on our generalized factorization forest theorem. We start by explaining how.

B. Using generalized factorization forests

Since $\mathbf{L}$ is tame, we know that it is a partial semigroup and that $2^{\mathbf{L}^n}$ is a semigroup. Moreover since $\mathbf{L}$ is $\mathcal{C}$-compatible, we know from Lemma 32 and Lemma 28 that any element $(S_1, \ldots, S_n)$ of $\mathcal{T}^n[\mathbf{L}]$ has homogeneous $\mathcal{C}$-type, that is, $[S_1]_{\mathcal{C}} = \cdots = [S_n]_{\mathcal{C}}$. Recall that $\mathcal{T}^n[\mathbf{L}]|_{[L]_{\mathcal{C}}}$ denotes the set of all $Pol(\mathcal{C})$-tied n-tuples whose $\mathcal{C}$-type is $[L]_{\mathcal{C}}$, that is:

$$\mathcal{T}^n[\mathbf{L}]|_{[L]_{\mathcal{C}}} \stackrel{\text{def}}{=} \{(S_1, \ldots, S_n) \in \mathcal{T}^n[\mathbf{L}] \mid \forall i,\ [S_i]_{\mathcal{C}} = [L]_{\mathcal{C}}\}.$$

Note that since any element of $\mathcal{T}^n[\mathbf{L}]$ has homogeneous $\mathcal{C}$-type, $\mathcal{T}^n[\mathbf{L}]$ is the union of all $\mathcal{T}^n[\mathbf{L}]|_{[L]_{\mathcal{C}}}$ when L ranges over $\mathbf{L}$.

Consider now the following map γ:

$$\begin{array}{rccc} \gamma: & \mathbf{L} & \to & 2^{\mathbf{L}^n} \\ & L & \mapsto & \mathcal{T}^n[\mathbf{L}]|_{[L]_{\mathcal{C}}} \end{array}$$

We work with γ-factorization forests. By definition, such forests are associated to nonempty words in $x \in (\mathbf{L} \times 2^{\mathbf{L}^n})^+$ and their nodes are labeled by pairs $(L, \mathbf{S}) \in \mathbf{L} \times 2^{\mathbf{L}^n}$.

The first issue we have to deal with is that we are interested in building languages over A, not $\mathbf{L} \times 2^{\mathbf{L}^n}$. Hence, we begin by explaining how we associate γ-factorization forests to words in A^*. Consider a word $w \in A^*$. There are two cases:

1) If $w = \varepsilon$, a γ-factorization forest for w is a γ-factorization forest for any single letter word of the form,
$$(S, \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle) \in (\mathbf{L} \times 2^{\mathbf{L}^n})^+ \quad \text{such that } \varepsilon \in S$$
2) If $w \in A^+$, let $w = a_1 \cdots a_k$, where the a_i's are letters. A γ-factorization forest for w is a γ-factorization forest for any word $x = x_1 \cdots x_k \in (\mathbf{L} \times 2^{\mathbf{L}^n})^k$ of length k such that for all i, the letter x_i is of the form,
$$x_i = (S_i, \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}} a_i [\varepsilon]_{\mathcal{C}}\rangle) \quad \text{with } a_i \in S_i$$
Note that $[\varepsilon]_{\mathcal{C}} a_i [\varepsilon]_{\mathcal{C}}$ is a concatenation of three languages in A^*, which is not to be confused with the multiplication of $\sim_{\mathcal{C}}$-classes (actually, $\{a_i\}$ may not be a $\sim_{\mathcal{C}}$-class).

Finally, we say that a pair $(S, \mathbf{S}) \in \mathbf{L} \times 2^{\mathbf{L}^n}$ is *valid* when there exists a γ-factorization forest which is associated to at least one word $w \in A^*$ and whose root label is $(S, \mathbf{S})$. We now prove the following propositions.

Proposition 69. *Let $h \in \mathbb{N}$ and $(S, \mathbf{S}) \in \mathbf{L} \times 2^{\mathbf{L}^n}$. There exists a language $K \in Pol(\mathcal{C})$ satisfying the following:*

1) *K contains all $w \in A^*$ associated to a γ-factorization forest of height at most h and root label $(S, \mathbf{S})$.*
2) *$\langle \mathbf{L}|K\rangle \subseteq \mathbf{S}$.*

Proposition 70. *Let $(S, \mathbf{S}) \in \mathbf{L} \times 2^{\mathbf{L}^n}$ be a valid pair. Then $(S, \mathbf{S}) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$.*

We begin by using these two results to build the sets $\mathbf{K}_L$ and prove Proposition 67. For the construction, we fix $h = 3(|\mathbf{L}| + 1)(|2^{\mathbf{L}^n}| + 2)$. Let $L \in \mathbf{L}$. For all $\mathbf{S} \in 2^{\mathbf{L}^n}$, we denote by $K_{\mathbf{S}} \in Pol(\mathcal{C})$ the language associated by Proposition 69 to h and $(L, \mathbf{S}) \in \mathbf{L} \times 2^{\mathbf{L}^n}$. Finally, we define,

$$\mathbf{K}_L = \{K_{\mathbf{S}} \mid (L, \mathbf{S}) \text{ is a valid pair}\}.$$

By definition $\mathbf{K}_L$ is finite and included in $Pol(\mathcal{C})$. It remains to prove that it satisfies the two conditions in Proposition 67.

We start with second one. Let $K \in \mathbf{K}_L$, we prove that $(L, \langle \mathbf{L}|K\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$. By definition, $K = K_{\mathbf{S}}$ where $(L, \mathbf{S})$ is valid. Hence, by construction of $K_{\mathbf{S}}$ in Proposition 69, we know that $\langle \mathbf{L}|K\rangle \subseteq \mathbf{S}$. Moreover, we know from Proposition 70 that $(L, \mathbf{S}) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$. Finally, we conclude using closure under downset in the definition of Sat^n that $(L, \langle \mathbf{L}|K\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$.

We now turn to the first item: $L \subseteq \bigcup_{K \in \mathbf{K}_L} K$. Given $L \in \mathbf{L}$ and $w \in L$, we have to find $K \in \mathbf{K}_L$ such that $w \in K$. Let us first consider the special case when $w = \varepsilon$. Let $\mathbf{S} = \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle$. Since $\varepsilon \in L$, a forest for ε is the leaf associated to $(L, \mathbf{S}) \in \mathbf{L} \times 2^{\mathbf{L}^n}$ of height $0 \leq h$. Hence, $(L, \mathbf{S})$ is valid and $K_{\mathbf{S}} \in \mathbf{K}_L$. Moreover, by definition of $K_{\mathbf{S}}$ in Proposition 69 we have $w \in K_{\mathbf{S}}$.

Assume now that $w = a_1 \cdots a_k \in A^+$. Since $w \in L$ and $\mathbf{L}$ is tame, we know that there exist $S_1, \cdots, S_k \in \mathbf{L}$ such that $S_1 \odot \cdots \odot S_k = L$ and $a_i \in S_i$ for all i. Moreover, for all $i \leq k$, we let x_i as the following letter in $\mathbf{L} \times 2^{\mathbf{L}^n}$:

$$x_i = (S_i, \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}} a_i [\varepsilon]_{\mathcal{C}}\rangle) \in \mathbf{L} \times 2^{\mathbf{L}^n}$$

Finally, we let $x = x_1 \cdots x_k \in (\mathbf{L} \times 2^{\mathbf{L}^n})^+$. Since $S_1 \odot \cdots \odot S_k = L$, the evaluation of x is defined and by choice of h, we then obtain a γ-factorization forest of height at most h for x from Theorem 61. By definition, this forest is associated to w and its labels is $(L, \mathbf{S})$ for some $\mathbf{S}$. Hence, $(L, \mathbf{S})$ is valid and $K_{\mathbf{S}} \in \mathbf{K}_L$. Moreover, by definition of $K_{\mathbf{S}}$ in Proposition 69 we have $w \in K_{\mathbf{S}}$.

It remains to prove Proposition 69 and Proposition 70. We devote a subsection to each proof.

C. Proof of Proposition 69

Before we start the proof, we begin with a technical lemma about filterings which we will use several times.

Lemma 71. *Let $K_1, K_2 \in Pol(\mathcal{C})$. Then,*

$$\begin{aligned}\langle \mathbf{L}|K_1\rangle \cup \langle \mathbf{L}|K_2\rangle &= \langle \mathbf{L}|K_1 \cup K_2\rangle \\ \langle \mathbf{L}|K_1\rangle \odot \langle \mathbf{L}|K_2\rangle &= \langle \mathbf{L}|K_1K_2\rangle\end{aligned}$$

Proof. Let $K_1, K_2 \in Pol(\mathcal{C})$. The two properties are based on similar arguments which rely on the stratification of $Pol(\mathcal{C})$ introduced in the previous appendix and the associated preorders $\leqslant_k$. We focus on $\langle \mathbf{L}|K_1\rangle \odot \langle \mathbf{L}|K_2\rangle = \langle \mathbf{L}|K_1K_2\rangle$, the other is left to the reader.

There exists $h \in \mathbb{N}$ such that K_1, K_2 and K_1K_2 all belong to $Pol_h(\mathcal{C})$. Furthermore, we may use Lemma 54 to obtain $\ell \geq 1$ such that for all $k \geq \ell$, if $\overline{L} = (L_1, \ldots, L_n) \in \mathbf{L}^n$, then for $K = K_1, K_2$ or K_1K_2, $\overline{L} \in \langle \mathbf{L}|K\rangle$ if and only if there exists $w_1 \in L_1 \cap K, \ldots, w_n \in L_n \cap K$ such that $w_1 \leqslant_k \cdots \leqslant_k w_n$. For the whole proof, we let $k = max(h, \ell) + 1$.

Assume first that $(L_1, \ldots, L_n) \in \langle \mathbf{L}|K_1\rangle \odot \langle \mathbf{L}|K_2\rangle$. We get $(S_1, \ldots, S_n) \in \langle \mathbf{L}|K_1\rangle$ and $(T_1, \ldots, T_n) \in \langle \mathbf{L}|K_2\rangle$ such that $L_i = S_i \odot T_i$ for all i. Hence, for all i, we have $u_i \in S_i \cap K_1$ and $v_i \in T_i \cap K_2$ such that, $u_1 \leqslant_k \cdots \leqslant_k u_n$ and $v_1 \leqslant_k \cdots \leqslant_k v_n$. For all i, since $L_i = S_i \odot T_i$, we have,

$$u_iv_i \in (S_i \cap K_1)(T_i \cap K_2) \subseteq S_iT_i \cap K_1K_2 \subseteq L_i \cap K_1K_2$$

Moreover, since $\leqslant_k$ is compatible with concatenation, we get $u_1v_1 \leqslant_k \cdots \leqslant_k u_nv_n$. Hence, we conclude by choice of k that $(L_1, \ldots, L_n) \in \langle \mathbf{L}|K_1K_2\rangle$.

Assume now that $(L_1, \ldots, L_n) \in \langle \mathbf{L}|K_1K_2\rangle$. For all i, we get $w_i \in L_i \cap K_1K_2$ such that $w_1 \leqslant_k \cdots \leqslant_k w_n$. Since $w_1 \in K_1K_2$, it admits a decomposition $w_1 = u_1v_1$ with $u_1 \in K_1$ and $v_1 \in K_2$. Let $G = \uparrow_{k-1} u_1$ and $H = \uparrow_{k-1} v_1$. Both are upper sets for $\leqslant_{k-1}$ and therefore belong to $Pol_{k-1}(\mathcal{C})$. Moreover, we have $GH \in Pol_k(\mathcal{C})$. Indeed,

$$GH = \begin{cases} G \cup \bigcup_{a\in A} Ga(a^{-1}H) & \text{if } \varepsilon \in H, \\ \bigcup_{a\in A} Ga(a^{-1}H) & \text{if } \varepsilon \notin H. \end{cases}$$

Hence, since $w_1 = u_1v_1 \in GH$ and $w_1 \leqslant_k w_2$, we have $w_2 = u_2v_2$ with $u_2 \in G$ (*i.e.*, $u_1 \leqslant_{k-1} u_2$) and $v_2 \in H$ (*i.e.*, $v_1 \leqslant_{k-1} v_2$). One may repeat the argument for all w_i, thus we obtain that for all $i \geq 2$, $w_i = u_iv_i$ with $u_1 \leqslant_{k-1} \cdots \leqslant_{k-1} u_n$ and $v_1 \leqslant_{k-1} \cdots \leqslant_{k-1} v_n$.

Since $u_iv_i = w_i \in L_i$, for all i, we may use the definition of tame multiplications to obtain $S_i, T_i \in \mathbf{L}$ such that $u_i \in S_i$, $v_i \in T_i$ and $S_i \odot T_i = L_i$. We prove that $(S_1, \ldots, S_n) \in \langle \mathbf{L}|K_1\rangle$ and $(T_1, \ldots, T_n) \in \langle \mathbf{L}|K_2\rangle$ which terminates the proof. By symmetry, we concentrate on $(S_1, \ldots, S_n) \in \langle \mathbf{L}|K_1\rangle$. We know that $u_1 \in K_1$. Hence, since $K_1 \in Pol_h(\mathcal{C})$ and $h \leq k-1$, we know that $u_i \in K_1$ for all i. Hence, for all i, $u_i \in L_i \cap K_1$ and $u_1 \leqslant_{k-1} \cdots \leqslant_{k-1} u_n$. Since $k - 1 \geq \ell$, we conclude that $(S_1, \ldots, S_n) \in \langle \mathbf{L}|K_1\rangle$ by choice of ℓ. □

We now prove Proposition 69. Let $h \in \mathbb{N}$ and $(S, \mathbf{S}) \in \mathbf{L} \times 2^{\mathbf{L}^n}$. Our goal is to build $K \in Pol(\mathcal{C})$ satisfying the two conditions in the proposition. We proceed by induction on h.

Induction base. Let us first assume that $h = 0$. We let $A_{\mathbf{S}} \subseteq A$ as the set of all letters $a \in A$ such that $\mathbf{S} = \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}a[\varepsilon]_{\mathcal{C}}\rangle$. There are now two cases depending on whether $\mathbf{S} = \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle$ or not. If this equality holds, we let,

$$K = [\varepsilon]_{\mathcal{C}} \cup \bigcup_{a\in A_{\mathbf{S}}} [\varepsilon]_{\mathcal{C}}a[\varepsilon]_{\mathcal{C}}$$

Otherwise, we define,

$$K = \bigcup_{a\in A_{\mathbf{S}}} [\varepsilon]_{\mathcal{C}}a[\varepsilon]_{\mathcal{C}}$$

Clearly $K \in Pol(\mathcal{C})$ (it the union of marked concatenations of $[\varepsilon]_{\mathcal{C}} \in \mathcal{C}$). Let us prove that this choice satisfies the conditions of the proposition. We start with the first item. Consider $w \in A^*$ associated to a γ-factorization forest of height at most $h = 0$ (*i.e.*, a leaf) and root label $(S, \mathbf{S})$. We have to prove that $w \in K$. By definition, one of the two following properties hold:

- $w = \varepsilon$, $\varepsilon \in S$ and $\mathbf{S} = \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle$. Hence, $w \in [\varepsilon]_{\mathcal{C}} \subseteq K$ by definition.
- $w = a \in A$, $a \in S$ and $\mathbf{S} = \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}a[\varepsilon]_{\mathcal{C}}\rangle$. Hence, $w \in [\varepsilon]_{\mathcal{C}}a[\varepsilon]_{\mathcal{C}} \subseteq K$ by definition.

We turn to the second item: $\langle \mathbf{L}|K\rangle \subseteq \mathbf{S}$. By definition K is the union of languages $K' \in Pol(\mathcal{C})$ such that $\langle \mathbf{L}|K'\rangle = \mathbf{S}$. It is therefore immediate from Lemma 71 that $\langle \mathbf{L}|K\rangle = \mathbf{S}$ which terminates the proof for the case $h = 0$.

Inductive case. Assume now that $h \geq 1$. We first build the language K. Given any $(T, \mathbf{T}) \in \mathbf{L} \times 2^{\mathbf{L}^n}$, we may use induction to construct a language $K_{T,\mathbf{T}} \in Pol(\mathcal{C})$ such that,

1) $K_{T,\mathbf{T}}$ contains all $w \in A^*$ associated to a γ-factorization forest of height at most $h - 1$ and root label $(T, \mathbf{T})$.
2) $\langle \mathbf{L}|K_{T,\mathbf{T}}\rangle \subseteq \mathbf{T}$.

We are now ready to define our language $K \in Pol(\mathcal{C})$. It is the union of three distinct kinds of languages:

1) The single language $K_{S,\mathbf{S}}$.
2) For all $(T_1, \mathbf{T}_1), (T_2, \mathbf{T}_2) \in \mathbf{L} \times 2^{\mathbf{L}^n}$ satisfying $(T_1, \mathbf{T}_1) \odot (T_2, \mathbf{T}_2) = (S, \mathbf{S})$, the following language is in the union,
$$K_{T_1,\mathbf{T}_1}K_{T_2,\mathbf{T}_2}$$
3) For any idempotent $(E, \mathbf{E}) \in \mathbf{L} \times 2^{\mathbf{L}^n}$ which satisfies $(E, \mathbf{E} \odot \mathfrak{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}} \odot \mathbf{E}) = (S, \mathbf{S})$, the following language is in the union,
$$K_{E,\mathbf{E}}[E]_{\mathcal{C}}K_{E,\mathbf{E}}$$

By definition, K is a union of concatenations of languages belonging to $Pol(\mathcal{C})$. Since $Pol(\mathcal{C})$ is closed under concatenation (see Fact 5), it follows that $K \in Pol(\mathcal{C})$. It remains to prove that K satisfies the conditions of the proposition.

We start with the first item. Let $w \in A^*$ which is associated to some γ-factorization forest of height at most h and of root label $(S, \mathbf{S})$. We have to prove that $w \in K$. If this forest has height smaller than $h - 1$, then $w \in K_{S,\mathbf{S}} \subseteq K$. Otherwise, it has height h and we distinguish two cases depending on whether its root is binary or idempotent. When it is binary, it follows by definition that $w = w_1w_2$ where w_1 and w_2 are associated to γ-factorization forests of height $h-1$ whose labels $(T_1, \mathbf{T}_1)$ and $(T_2, \mathbf{T}_2)$ satisfy $(T_1, \mathbf{T}_1) \odot (T_2, \mathbf{T}_2) = (S, \mathbf{S})$. It follows that $w = w_1w_2 \in K_{T_1,\mathbf{T}_1}K_{T_2,\mathbf{T}_2} \subseteq K$.

Finally, if the root is idempotent, w admits a decomposition $w = w_1 \cdots w_k$ such that for all i, w_i is associated to a γ-factorization forests of height $h-1$ whose label is $(E, \mathbf{T}_i)$ for some idempotent $E \in \mathbf{L}$. Moreover $\mathbf{T}_1 = \mathbf{T}_k$ is an idempotent $\mathbf{E}$ of $2^{\mathbf{L}^n}$ and $(S, \mathbf{S}) = (E, \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}} \odot \mathbf{E})$ (by definition of γ). Hence, $K_{E,\mathbf{E}}[E]_{\mathcal{C}}K_{E,\mathbf{E}} \subseteq K$. By definition, we have $w_i \in K_{E,\mathbf{E}}$ for all i. Moreover, since the w_i are associated to γ-factorization forests whose root is $(E, \mathbf{T}_i)$, it follows from the definition that $w_i \in E$ for all i. Hence, since E is idempotent, $w_2 \cdots w_{k-1} \in E \subseteq [E]_{\mathcal{C}}$. We conclude that,

$$w = w_1 w_2 \cdots w_{k-1} w_k \in K_{E,\mathbf{E}}[E]_{\mathcal{C}}K_{E,\mathbf{E}} \subseteq K$$

This terminates the proof of the first item in the proposition.

We finish with the proof of the second item: $\langle \mathbf{L}|K\rangle \subseteq \mathbf{S}$. Since K is defined as the union of several languages, by the first equality in Lemma 71, it suffices that for any language K' in the union, we have $\langle \mathbf{L}|K'\rangle \subseteq \mathbf{S}$. Let us consider all three cases. If $K' = K_{S,\mathbf{S}}$, this is by definition. If $K' = K_{T_1,\mathbf{T}_1}K_{T_2,\mathbf{T}_2}$ for $(T_1, \mathbf{T}_1) \odot (T_2, \mathbf{T}_2) = (S, \mathbf{S})$, it follows from the second equality in Lemma 71 that,

$$\langle \mathbf{L}|K'\rangle = \langle \mathbf{L}|K_{T_1,\mathbf{T}_1}\rangle \odot \langle \mathbf{L}|K_{T_2,\mathbf{T}_2}\rangle$$

Hence, by hypothesis on $K_{T_1,\mathbf{T}_1}$ and $K_{T_2,\mathbf{T}_2}$, $\langle \mathbf{L}|K'\rangle \subseteq \mathbf{T}_1 \odot \mathbf{T}_2 = \mathbf{S}$.

Finally, assume that $K' = K_{E,\mathbf{E}}[E]_{\mathcal{C}}K_{E,\mathbf{E}}$ with $(E, \mathbf{E})$ and idempotent such that $(E, \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}} \odot \mathbf{E}) = (S, \mathbf{S})$. Using Lemma 71 again, we obtain,

$$\langle \mathbf{L}|K'\rangle = \langle \mathbf{L}|K_{E,\mathbf{E}}\rangle \odot \langle \mathbf{L}|[E]_{\mathcal{C}}\rangle \odot \langle \mathbf{L}|K_{E,\mathbf{E}}\rangle$$

Hence, by hypothesis on $K_{E,\mathbf{E}}$, we obtain $\langle \mathbf{L}|K'\rangle \subseteq \mathbf{E} \odot \langle \mathbf{L}|[E]_{\mathcal{C}}\rangle \odot \mathbf{E}$. Since $\mathbf{S} = \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}} \odot \mathbf{E}$, that $\langle \mathbf{L}|K'\rangle \subseteq \mathbf{S}$ is now immediate from the following fact.

Fact 72. *For any $H \in \mathbf{L}$, $\mathcal{T}^n[\mathbf{L}]|_{[H]_{\mathcal{C}}} = \langle \mathbf{L}|[H]_{\mathcal{C}}\rangle$.*

Proof. By $\mathcal{C}$-compatibility of $\mathbf{L}$: for any $L \in \mathbf{L}$ either $L \subseteq [H]_{\mathcal{C}}$ or $L \cap [H]_{\mathcal{C}} = \emptyset$. Hence, $\langle \mathbf{L}|[H]_{\mathcal{C}}\rangle$ is the set of all n-tuples $(S_1, \ldots, S_n) \in \mathcal{T}^n[\mathbf{L}]$ such that $S_i \subseteq [H]_{\mathcal{C}}$ for all i. This is exactly the definition of $\mathcal{T}^n[\mathbf{L}]|_{[H]_{\mathcal{C}}}$. □

D. Proof of Proposition 70

Let $(S, \mathbf{S}) \in \mathbf{L} \times 2^{\mathbf{L}}$ be a valid pair. We have to prove that $(S, \mathbf{S}) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$. By definition there exists $w \in A^*$ associated to a γ-factorization forest $\mathcal{F}$ whose root label is $(S, \mathbf{S})$. We proceed by induction on the height of $\mathcal{F}$.

Leaves. Assume first that $\mathcal{F}$ is a leaf. In that case, we know by definition that there are two possibilities. Either $w = \varepsilon$, therefore $\varepsilon \in S$ and $\mathbf{S} = \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle$. Or $w = a \in A$, therefore $a \in S$ and $\mathbf{S} = \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}a[\varepsilon]_{\mathcal{C}}\rangle$. Hence, this case amounts to proving the following lemma.

Lemma 73. *Let $T \in \mathbf{L}$ and $a \in A$ the following hold:*

1) *If $\varepsilon \in T$, then $(T, \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$.*
2) *If $a \in T$, then $(T, \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}a[\varepsilon]_{\mathcal{C}}\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$.*

The proof of Lemma 73 is rather technical and we postpone it to the end of the section. Let us first finish our induction.

Binary nodes. We now assume that $\mathcal{F}$ is not a leaf and that its root is a binary node. Hence, it has two children. We let $\mathcal{F}_1, \mathcal{F}_2$ as the forests which are rooted in these children. By definition they are associated to $w_1, w_2 \in A^+$ and $(S, \mathbf{S}) = (S_1, \mathbf{S}_1) \odot (S_2, \mathbf{S}_2)$ where $(S_1, \mathbf{S}_1)$ and $(S_2, \mathbf{S}_2)$ are the root labels of $\mathcal{F}_1$ and $\mathcal{F}_2$. Since $\mathcal{F}_1$ and $\mathcal{F}_2$ have smaller height than $\mathcal{F}$, it is immediate by induction that $(S_1, \mathbf{S}_1), (S_2, \mathbf{S}_2) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$. Hence, by closure under multiplication in the definition of Sat^n, we obtain $(S, \mathbf{S}) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$.

Idempotent nodes. Finally, we assume that $\mathcal{F}$ is not a leaf and that its root is an idempotent node. It has an arbitrary number of children $\mathcal{F}_1, \ldots, \mathcal{F}_k$ which have labels $(E, \mathbf{S}_1), \ldots, (E, \mathbf{S}_k)$ where $E \in \mathbf{L}$ is idempotent and $\mathbf{S}_1 = \mathbf{S}_k$ is an idempotent $\mathbf{E}$. Moreover, $(S, \mathbf{S}) = (E, \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}} \odot \mathbf{E})$ by definition of γ. Finally, it follows from the definition that w may be decomposed as $w = w_1 \cdots w_k$ where w_i is associated to $\mathcal{F}_i$ for all i. Hence, since $\mathcal{F}_1$ has smaller height than $\mathcal{F}$, it follows by induction that $(E, \mathbf{E}) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$. Therefore, we obtain from Operation 3 in the definition of Sat^n that,

$$(S, \mathbf{S}) = (E, \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}} \odot \mathbf{E}) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$$

Proof of Lemma 73. It remains to prove Lemma 73. We will need the following simple fact.

Fact 74. *Let $T \in \mathbf{L}$ and $w \in T$, then $(T, \langle \mathbf{L}|w\rangle) \in \mathcal{J}^n_{triv}[\mathbf{L}]$ and therefore $(T, \langle \mathbf{L}|w\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$.*

Proof. Since $w \in T$, by definition of $\mathcal{J}^n_{triv}[\mathbf{L}]$, it suffices to prove that for all $(T_1, \ldots, T_n) \in \langle \mathbf{L}|w\rangle$, $w \in T_1 \cap \cdots \cap T_n$. By hypothesis, $(T_1, \ldots, T_n) \cap \{w\}$ is $Pol(\mathcal{C})$-tied. It follows that for all i, $T_i \cap \{w\}$ is non-empty, *i.e.*, $w \in T_i$. □

We may now start the proof. Let $T \in \mathbf{T}$. We begin with the case when $\varepsilon \in T$. We prove that $(T, \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$. We first use the fact that $\varepsilon \in T$ to prove that there exists $T_1, T_2, E \in \mathbf{L}$ such that E is idempotent, $\varepsilon \in T_1$, $\varepsilon \in T_2$, $\varepsilon \in E$ and $T = T_1 \odot E \odot T_2$.

Let $k = |\mathbf{L}|+1$. By hypothesis, we have $\varepsilon^k = \varepsilon \in T$. Hence, using the definition of tame sets, we obtain $R_1, \ldots, R_k \in \mathbf{L}$ such that $\varepsilon \in R_i$ for all i and $T = R_1 \odot \cdots \odot R_k$. By choice of k, it follows from a pigeon-hole principle argument that there exist $i < j$ such that $R_1 \odot \cdots \odot R_i = R_1 \odot \cdots \odot R_j$. A little algebra then yields that,

$$T = R_1 \odot \cdots \odot R_i \odot (R_{i+1} \odot \cdots \odot R_j)^\omega \odot R_{j+1} \odot \cdots \odot R_{k+1}$$

We may now choose $T_1 = R_1 \odot \cdots \odot R_i$, $E = (R_{i+1} \odot \cdots \odot R_j)^\omega$ and $T_2 = R_{j+1} \odot \cdots \odot R_{k+1}$.

We may now prove that $(T, \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$. It follows from Fact 74 that $(E, \langle \mathbf{L}|\varepsilon\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$. Moreover, one may verify the following fact (whose proof is left to the reader):

Fact 75. *For any $K \subseteq A^*$,*

$$\langle \mathbf{L}|K\rangle \odot \langle \mathbf{L}|\varepsilon\rangle = \langle \mathbf{L}|\varepsilon\rangle \odot \langle \mathbf{L}|K\rangle = \langle \mathbf{L}|K\rangle$$

Hence, $(E, \langle \mathbf{L}|\varepsilon\rangle)$ is idempotent. Altogether, we obtain from Operation 3 in the definition of Sat^n that,

$$(E, \langle \mathbf{L}|\varepsilon\rangle \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}} \odot \langle \mathbf{L}|\varepsilon\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$$

Moreover, it follows from Fact 72 that $\mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}} = \langle \mathbf{L}|[E]_{\mathcal{C}}\rangle$ which is itself equal to $\langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle$ since $\varepsilon \in E$. Using Fact 75, this yields,

$$(E, \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$$

Finally, since $\varepsilon \in T_1$ and $\varepsilon \in T_2$, we obtain from Fact 74 that $(T_1, \langle \mathbf{L}|\varepsilon\rangle)$ and $(T_2, \langle \mathbf{L}|\varepsilon\rangle)$ both belong to $Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$. Hence, by closure under multiplication in the definition of Sat^n and Fact 75 again, we obtain,

$$(T_1 \odot E \odot T_2, \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$$

Since $T = T_1 \odot E \odot T_2$, this yields $(T, \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$ as desired.

It now remains to treat the case $a \in T$ for some $a \in A$. We prove that $(T, \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}} a [\varepsilon]_{\mathcal{C}}\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$. Observe that a can be decomposed as $a = \varepsilon a \varepsilon$. Hence, since $a \in T$, it follows from the definition of tame multiplications that we have $T_1, T_a, T_2 \in \mathbf{L}$ such that $\varepsilon \in T_1$, $\varepsilon \in T_2$, $a \in T_a$ and $T_1 \odot T_a \odot T_2 = T$. Using the above argument for the case $\varepsilon \in T$, we have,

$$\begin{array}{rcl} (T_1, \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle) & \in & Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}]) \\ (T_2, \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle) & \in & Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}]) \end{array}$$

Moreover, since we know that $a \in T_a$, we have $(T_a, \langle \mathbf{L}|a\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$ by Fact 74. Using closure under multiplication in the definition of Sat^n and the fact that $T_1 \odot T_a \odot T_2 = T$ this yields the following,

$$(S, \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle \odot \langle \mathbf{L}|a\rangle \odot \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$$

Using an argument similar to the proof of Lemma 71 (which is left to the reader, one may use the fact that $[\varepsilon]_{\mathcal{C}} \in \mathcal{C} \subseteq Pol(\mathcal{C})$ to prove the following,

$$\langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle \odot \langle \mathbf{L}|a\rangle \odot \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}}\rangle = \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}} a [\varepsilon]_{\mathcal{C}}\rangle$$

Therefore, we obtain $(T, \langle \mathbf{L}|[\varepsilon]_{\mathcal{C}} a [\varepsilon]_{\mathcal{C}}\rangle) \in Sat^n(\mathbf{L}, \mathcal{T}^n[\mathbf{L}])$ which terminates the proof of Lemma 73.

APPENDIX H
OMITTED PROOFS IN SECTION VIII

This appendix contains the two missing proofs in Section VIII for Propositions 42 and 43. Recall that a finite quotienting Boolean algebra $\mathcal{C}$ and a tame and $\mathcal{C}$-compatible multiset $\mathbf{L}$ are fixed. Moreover, for the sake of improved readability, given $n \geq 1$, we write $\mathcal{A}[\mathbf{L}]$ for $\mathcal{A}_{Pol(\mathcal{C})}[\mathbf{L}]$, $\mathcal{T}^n[\mathbf{L}]$ for $\mathcal{T}^n_{Pol(\mathcal{C})}[\mathbf{L}]$ and $\mathcal{J}^n[\mathbf{L}]$ for $\mathcal{J}^n_{Pol(\mathcal{C})}[\mathbf{L}]$.

A. Proof of Proposition 42

Proposition 42. *For all $n \geq 1$ and all computation trees $\mathbb{T}$ of level n, there exists a computation tree $\mathbb{T}'$ with the same label and whose operational height is smaller than $|\mathcal{C}|$.*

For the proof, we fix $n \geq 1$ and assume that all our computation trees have level n. We call *operational size* of a computation tree $\mathbb{T}$ the total number of operation nodes in $\mathbb{T}$. Proposition 42 is a consequence of the following lemma.

Lemma 76. *Consider a computation tree $\mathbb{T}$ and assume that it contains a branch with two distinct operation nodes x and x' whose labels $(T, \mathbf{T})$ and $(T', \mathbf{T}')$ satisfy $[T]_{\mathcal{C}} = [T']_{\mathcal{C}}$. Then there exists a second tree $\mathbb{T}'$ with strictly smaller operational size than $\mathbb{T}$ and such that $lab(\mathbb{T}) = lab(\mathbb{T}')$.*

Starting from any computation tree $\mathbb{T}$, one may use Lemma 76 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 $(T, \mathbf{T})$ and $(T', \mathbf{T}')$ satisfy $[T]_{\mathcal{C}} \neq [T']_{\mathcal{C}}$. Since $[T]_{\mathcal{C}} \in \mathcal{C}$, for any $T \in \mathbf{L}$, it follows that any branch of $\mathbb{T}'$ contains at most $|\mathcal{C}|$ operation nodes. Thus, the operational height of $\mathbb{T}'$ is bounded by $|\mathcal{C}|$ which terminates the proof of Proposition 42.

We now concentrate on proving Lemma 76. We let $\mathbb{T}$ and $x \neq x'$ be as 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 $\mathbb{S}$ be the subtree of $\mathbb{T}$ which is rooted at x. By definition, $lab(\mathbb{S}) = lab(x) = (T, \mathbf{T})$. 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 $\mathbb{T}'$ 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 operation nodes which are between the two. We let $x_1, \ldots, x_k$ be 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 $(F_i, \mathbf{T}_i)$ as label of x_i. By definition of operation nodes, F_i must be an idempotent of $\mathbf{L}$. Moreover, $(F_k, \mathbf{T}_k) = (T, \mathbf{T})$ 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 $(E, \mathbf{E})$ be the label of this child (which is idempotent since x_1 is an operation node). Note that $E = F_1$ and $\mathbf{T}_1 = \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}} \odot \mathbf{E}$ by definition.

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

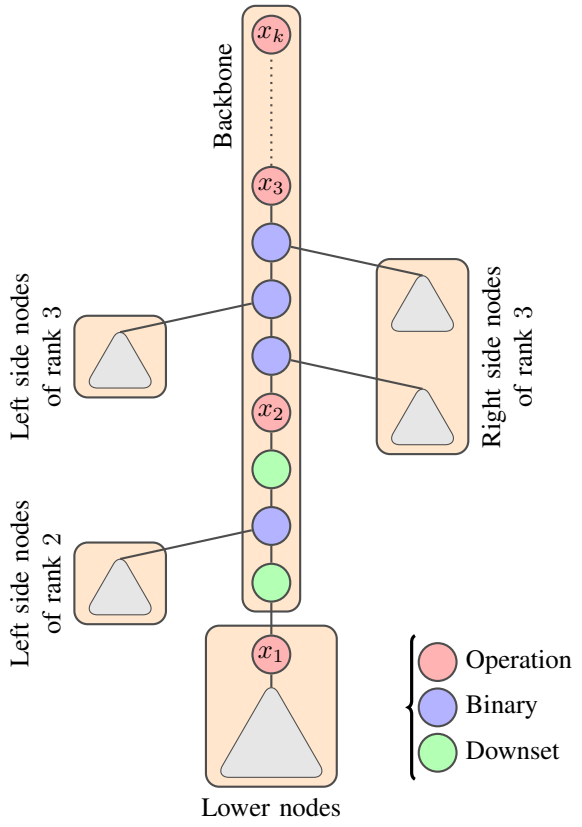

Fig. 3. Classification of the nodes in $\mathbb{S}$. In this example, there are no right side nodes of rank 2.

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 (note that $\ell_1 = r_1 = 0$). We illustrate these definitions in Figure 3.

Since backbone, lower and side nodes account for all nodes in the tree, we have the following fact.

Fact 77. *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 only downset and binary nodes. Thus, we obtain a new 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 $k - 1 \geq 1$. Before we present this construction, we introduce a last notation whose purpose is to simplify the presentation. We add artificial neutral elements to the partial semigroup $\mathbf{L}$ and to the semigroup $2^{\mathbf{L}^n}$ which we denote by $1_{\mathbf{L}}$ and $1_{2^{\mathbf{L}^n}}$. Having them will allow us to factorize some arguments. We may now start building $\mathbb{S}'$.

Lemma 78. *For all $i \leq k$, there exist $U_i, V_i, U_i', V_i' \in \mathbf{L} \cup \{1_{\mathbf{L}}\}$ and $\mathbf{U}_i, \mathbf{V}_i \in 2^{\mathbf{L}^n} \cup \{1_{2^{\mathbf{L}^n}}\}$ such that the following conditions hold:*

1) *For $X \in \{U_i, U_i'\}$ and $Y \in \{V_i, V_i'\}$, $F_i = X \odot E \odot Y$.*
2) *$\mathbf{T}_i \subseteq \mathbf{U}_i \odot \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E \odot V_i' \odot F_i \odot U_i' \odot E]_{\mathfrak{C}}} \odot \mathbf{E} \odot \mathbf{V}_i$.*
3) *If $(U_i, \mathbf{U}_i) \neq (1_{\mathbf{L}}, 1_{2^{\mathbf{L}^n}})$, it is the label of a computation tree whose operational size is bounded by $\ell_1 + \cdots + \ell_i$.*
4) *If $(V_i, \mathbf{V}_i) \neq (1_{\mathbf{L}}, 1_{2^{\mathbf{L}^n}})$, it is the label of a computation tree whose operational size is bounded by $r_1 + \cdots + r_i$.*

Let us first use Lemma 78, to finish the proof of Lemma 76. We construct a new tree $\mathbb{S}'$ whose operational size is bounded by $m + \ell_1 + \cdots \ell_k + r_1 + \cdots + r_k$ and whose label is $lab(\mathbb{S}) = (T, \mathbf{T}) = (F_k, \mathbf{T}_k)$.

We apply Lemma 78 in the special case when $i = k$. This yields $U_k, V_k, U_k', V_k' \in \mathbf{L} \cup \{1_{\mathbf{L}}\}$ and $\mathbf{U}_k, \mathbf{V}_k \in 2^{\mathbf{L}^n} \cup \{1_{2^{\mathbf{L}^n}}\}$ as in the lemma. Recall that by hypothesis, we have $[F_k]_{\mathfrak{C}} = [F_1]_{\mathfrak{C}}$ and $F_1 = E$. Thus, we have $[F_k]_{\mathfrak{C}} = [E]_{\mathfrak{C}}$ which yields the following fact.

Fact 79. $[E]_{\mathfrak{C}} = [E \odot V_k' \odot F_k \odot U_k' \odot E]_{\mathfrak{C}}$.

Proof. By Lemma 48, the quotient set $A^*/\sim_{\mathfrak{C}}$ is a finite monoid whose multiplication is denoted by "$\bullet$" and we have

$$[E \odot V_k' \odot F_k \odot U_k' \odot E]_{\mathfrak{C}} = [E]_{\mathfrak{C}} \bullet [V_k']_{\mathfrak{C}} \bullet [F_k]_{\mathfrak{C}} \bullet [U_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}}.$$

Therefore, since $[F_k]_{\mathfrak{C}} = [E]_{\mathfrak{C}}$, it suffices to prove that, $[E]_{\mathfrak{C}} = [E]_{\mathfrak{C}} \bullet [V_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}} \bullet [U_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}}$.

By the first item in the Lemma 78, we have $E = F_k = U_k' \odot E \odot V_k'$. Hence, $[E]_{\mathfrak{C}} = [U_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}} \bullet [V_k']_{\mathfrak{C}}$. Moreover, since E is idempotent in $\mathbf{L}$, $[E]_{\mathfrak{C}} = [E \odot E]_{\mathfrak{C}} = [E]_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}}$ is an idempotent of $A^*/\sim_{\mathfrak{C}}$. We therefore obtain:

$$\begin{aligned}
[E]_{\mathfrak{C}} &= [E]_{\mathfrak{C}} \bullet [U_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}} \bullet [V_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}} \\
&= ([E]_{\mathfrak{C}} \bullet [U_k']_{\mathfrak{C}})^{\omega} \bullet [E]_{\mathfrak{C}} \bullet ([V_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}})^{\omega} \\
&= [E]_{\mathfrak{C}} \bullet ([V_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}})^{\omega} \\
&= [E]_{\mathfrak{C}} \bullet [V_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}} \bullet ([V_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}})^{\omega - 1}
\end{aligned}$$

We may now replace the second copy of $[E]_{\mathfrak{C}}$ in the above with $[E]_{\mathfrak{C}} \bullet [U_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}} \bullet [V_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}}$ which yields,

$$[E]_{\mathfrak{C}} = [E]_{\mathfrak{C}} \bullet [V_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}} \bullet [U_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}} \bullet ([V_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}})^{\omega}$$

Finally, since $[E]_{\mathfrak{C}} = [E]_{\mathfrak{C}} \bullet ([V_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}})^{\omega}$, this yields $[E]_{\mathfrak{C}} = [E]_{\mathfrak{C}} \bullet [V_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}} \bullet [U_k']_{\mathfrak{C}} \bullet [E]_{\mathfrak{C}}$ as desired. □

We now combine Fact 79 with the second item in Lemma 78 which yields, $\mathbf{T}_k \subseteq \mathbf{U}_k \odot \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathfrak{C}}} \odot \mathbf{E} \odot \mathbf{V}_k$. Moreover, since we have $F_k = U_k \odot E \odot V_k$, by the first item, we obtain,

$$(F_k, \mathbf{T}_k) \subseteq (U_k \odot E \odot V_k, \mathbf{U}_k \odot \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathfrak{C}}} \odot \mathbf{E} \odot \mathbf{V}_k)$$

We may now construct $\mathbb{S}'$. By hypothesis, we already have a computation tree whose label is $(E, \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathfrak{C}}} \odot \mathbf{E})$: the subtree of $\mathbb{S}$ rooted at x_1 (whose operational size is m). Moreover, we know by Items 3 and 4 in Lemma 78 that $(U_k, \mathbf{U}_k)$ and $(V_k, \mathbf{V}_k)$ are the labels of computation trees whose operational sizes are bounded by $\ell_1 + \cdots + \ell_k$ and $r_1 + \cdots + r_k$ respectively. Because of the above inclusion, one may use two binary nodes and a downset node to combine

these three computation trees into a single one whose label is $(F_k, \mathbf{T}_k) = lab(\mathbb{S})$. This is our new tree $\mathbb{S}'$. By definition $\mathbb{S}'$ has operational size bounded by $m+\ell_1+\cdots+\ell_k+r_1+\cdots+r_k$. 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 77 and $k-1 \geq 1$). This terminates the proof of Lemma 76.

It now remains to prove Lemma 78. We proceed by induction on i. When $i=1$, since x_1 is an operation node whose unique child has label is $(E, \mathbf{E})$, we have:

$$(F_1, \mathbf{T}_1) = (E, \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathfrak{C}}} \odot \mathbf{E}).$$

Hence, it suffices to choose $U_1 = V_1 = U_1' = V_1' = 1_{\mathbf{L}}$ and $\mathbf{U}_1 = \mathbf{V}_1 = 1_{2^{\mathbf{L}^n}}$. We now assume that $i \geq 2$. By definition, x_i has a unique child and its label is an idempotent $(F_i, \mathbf{F}_i)$ such that

$$(F_i, \mathbf{T}_i) = (F_i, \mathbf{F}_i \odot \mathcal{T}^n[\mathbf{L}]|_{[F_i]_{\mathfrak{C}}} \odot \mathbf{F}_i).$$

We use the following fact to choose $U_i, V_i, U_i', V_i' \in \mathbf{L} \cup \{1_{\mathbf{L}}\}$ and $\mathbf{U}_i, \mathbf{V}_i \in 2^{\mathbf{L}^n} \cup \{1_{2^{\mathbf{L}^n}}\}$.

Fact 80. *There exist* $(S, \mathbf{S}), (R, \mathbf{R}) \in \mathbf{L} \times 2^{\mathbf{L}^n} \cup \{(1_{\mathbf{L}}, 1_{2^{\mathbf{L}^n}})\}$ *such that,*

- $(F_i, \mathbf{F}_i) \subseteq (S \odot F_{i-1} \odot R, \mathbf{S} \odot \mathbf{T}_{i-1} \odot \mathbf{R})$.
- *If* $(S, \mathbf{S}) \neq (1_{\mathbf{L}}, 1_{2^{\mathbf{L}^n}})$*, it is the label of a computation tree whose operational size is bounded by* ℓ_i.
- *If* $(R, \mathbf{R}) \neq (1_{\mathbf{L}}, 1_{2^{\mathbf{L}^n}})$*, it is the label of a computation tree whose operational size is bounded by* r_i.

Proof. Consider all binary nodes between x_i and x_{i-1}. 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 an 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 an all their descendants are right side nodes of rank i).

Note that these two sequences may be empty. We may now define $(S, \mathbf{S})$ and $(R, \mathbf{R})$. If the sequence $z_{h_1} < \cdots < z_1$ is empty, we let $(S, \mathbf{S}) = (1_{\mathbf{L}}, 1_{2^{\mathbf{L}^n}})$. Otherwise, we let,

$$(S, \mathbf{S}) = (S_{h_1}, \mathbf{S}_{h_1}) \odot \cdots \odot (S_1, \mathbf{S}_1)$$

where $(S_j, \mathbf{S}_j)$ is the label of the left child of z_j for all $j \leq h_1$. Similarly, if the sequence $z'_{h_2} < \cdots < z'_1$ is empty, we let $(R, \mathbf{R}) = (1_{\mathbf{L}}, 1_{2^{\mathbf{L}^n}})$. Otherwise, we let,

$$(R, \mathbf{R}) = (R_1, \mathbf{R}_1) \odot \cdots \odot (R_{h_2}, \mathbf{R}_{h_2})$$

where $(R_j, \mathbf{R}_j)$ is the label of the right child of z'_j for all $j \leq h_2$. One may now verify that this choice satisfies the conditions of the fact. □

We now define appropriate $U_i, V_i, U_i', V_i' \in \mathbf{L} \cup \{1_{\mathbf{L}}\}$ and $\mathbf{U}_i, \mathbf{V}_i \in 2^{\mathbf{L}^n} \cup \{1_{2^{\mathbf{L}^n}}\}$. Using our induction hypothesis, we obtain $U_{i-1}, V_{i-1}, U_{i-1}', V_{i-1}'$ and $\mathbf{U}_{i-1}, \mathbf{V}_{i-1}$ satisfying the four items in Lemma 78 for $i-1$. We define,

$$\begin{array}{rcl} (U_i, \mathbf{U}_i) & = & (S, \mathbf{S}) \odot (U_{i-1}, \mathbf{U}_{i-1}) \\ U_i' & = & S \odot F_{i-1} \odot U_{i-1}' \\ (V_i, \mathbf{V}_i) & = & (V_{i-1}, \mathbf{V}_{i-1}) \odot (R, \mathbf{R}) \\ V_i' & = & V_{i-1}' \odot F_{i-1} \odot R \end{array}$$

It now remains to verify that the four items in Lemma 78 hold. We begin with the third and fourth which are the simplest.

Items 3 and 4. Since both items are symmetrical, we concentrate on Item 3. Assume that $(U_i, \mathbf{U}_i) \neq (1_{\mathbf{L}}, 1_{2^{\mathbf{L}^n}})$, we want to build a computation tree with label $(U_i, \mathbf{U}_i)$ and operational size bounded by $\ell_1 + \cdots + \ell_i$. Let us assume that $(S, \mathbf{S}) \neq (1_{\mathbf{L}}, 1_{2^{\mathbf{L}^n}})$ or $(U_{i-1}, \mathbf{U}_{i-1}) \neq (1_{\mathbf{L}}, 1_{2^{\mathbf{L}^n}})$ (the cases when one of the two is equal to $(1_{\mathbf{L}}, 1_{2^{\mathbf{L}^n}})$ are similar). By construction, we already have computation trees whose labels are $(S, \mathbf{S})$ and $(U_{i-1}, \mathbf{U}_{i-1})$. Moreover, their operational sizes are bounded by ℓ_i and $\ell_1 + \cdots + \ell_{i-1}$ respectively. Hence, since $(U_i, \mathbf{U}_i) = (S, \mathbf{S}) \odot (U_{i-1}, \mathbf{U}_{i-1})$ by definition, one may use a binary node to combine these two trees into a single one of label $(U_i, \mathbf{U}_i)$. By definition, this tree has operational size bounded by $\ell_1 + \cdots + \ell_i$.

Item 1. We have four equalities to verify. Since the argument is similar for all four, we concentrate on $F_i = U_i \odot E \odot V_i$ and $F_i = U_i' \odot E \odot V_i'$. By Fact 80, $F_i = S \odot F_{i-1} \odot R$. Moreover, since $F_{i-1} = U_{i-1} \odot E \odot V_{i-1}$ by the inductive definition of U_{i-1} and V_{i-1}, we get,

$$F_i = S \odot U_{i-1} \odot E \odot V_{i-1} \odot R = U_i \odot E \odot V_i.$$

Furthermore, F_{i-1} is idempotent. Thus, $F_i = S \odot (F_{i-1})^3 \odot R$ and since by construction of U_{i-1}' and V_{i-1}', we have $F_{i-1} = U_{i-1}' \odot E \odot V_{i-1}'$, we obtain,

$$F_i = S \odot F_{i-1} \odot U_{i-1}' \odot E \odot V_{i-1}' \odot F_{i-1} \odot R = U_i' \odot E \odot V_i'.$$

Item 2. We finish with the second item which is the most involved. Our objective is to show that,

$$\mathbf{T}_i \subseteq \mathbf{U}_i \odot \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E \odot V_i' \odot F_i \odot U_i' \odot E]_{\mathfrak{C}}} \odot \mathbf{E} \odot \mathbf{V}_i$$

We start with two simple facts which we will need.

Fact 81. *For any* $X, Y \in \mathbf{L}$ *such that* $X \odot Y$ *is defined,* $\mathcal{T}^n[\mathbf{L}]|_{[X]_{\mathfrak{C}}} \odot \mathcal{T}^n[\mathbf{L}]|_{[Y]_{\mathfrak{C}}} \subseteq \mathcal{T}^n[\mathbf{L}]|_{[X \odot Y]_{\mathfrak{C}}}$.

Fact 82. *For* $(X, \mathbf{X})$ *which is the label of a computation tree, we have* $\mathbf{X} \subseteq \mathcal{T}^n[\mathbf{L}]|_{[X]_{\mathfrak{C}}}$.

We now start the proof. By definition, $(F_i, \mathbf{T}_i)$ is the label of the operation node x_i whose child has label $(F_i, \mathbf{F}_i)$. Hence, $\mathbf{T}_i = \mathbf{F}_i \odot \mathcal{T}^n[\mathbf{L}]|_{[F_i]_{\mathfrak{C}}} \odot \mathbf{F}_i$ and it follows from Fact 80 that,

$$\mathbf{T}_i \subseteq \mathbf{S} \odot \mathbf{T}_{i-1} \odot \mathbf{R} \odot \mathcal{T}^n[\mathbf{L}]|_{[F_i]_{\mathfrak{C}}} \odot \mathbf{S} \odot \mathbf{T}_{i-1} \odot \mathbf{R} \quad (5)$$

The result is a consequence of the two following inclusions:

$$\begin{array}{rcl} \mathbf{S} \odot \mathbf{T}_{i-1} \odot \mathbf{R} & \subseteq & \mathbf{U}_i \odot \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E \odot V_i']_{\mathfrak{C}}} \\ \mathbf{S} \odot \mathbf{T}_{i-1} \odot \mathbf{R} & \subseteq & \mathcal{T}^n[\mathbf{L}]|_{[U_i' \odot E]_{\mathfrak{C}}} \odot \mathbf{E} \odot \mathbf{V}_i \end{array}$$

Indeed, one may combine these two inclusions with (5) using Fact 81 which yields the desired result:

$$\mathbf{T}_i \subseteq \mathbf{U}_i \odot \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E \odot V'_i \odot F_i \odot U'_i \odot E]_{\mathrm{e}}} \odot \mathbf{E} \odot \mathbf{V}_i.$$

It remains to prove the two inclusions. Since they are based on symmetrical arguments, we concentrate on the first one and leave the other to the reader. Since we built $\mathbf{U}_{i-1}$ and $\mathbf{V}_{i-1}$ with induction, we have,

$$\mathbf{T}_{i-1} \subseteq \mathbf{U}_{i-1} \odot \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E \odot V'_{i-1} \odot F_{i-1} \odot U'_{i-1} \odot E]_{\mathrm{e}}} \odot \mathbf{E} \odot \mathbf{V}_{i-1}.$$

By Fact 82, $\mathbf{E} \subseteq \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathrm{e}}}$ and $\mathbf{V}_{i-1} \subseteq \mathcal{T}^n[\mathbf{L}]|_{[V_{i-1}]_{\mathrm{e}}}$. Hence, using Fact 81, we may simplify the above inclusion as follows:

$$\mathbf{T}_{i-1} \subseteq \mathbf{U}_{i-1} \odot \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E \odot V'_{i-1} \odot F_{i-1} \odot U'_{i-1} \odot E \odot V_{i-1}]_{\mathrm{e}}}.$$

Since U'_{i-1} and V_{i-1} were built by induction, we know that $U'_{i-1} \odot E \odot V_{i-1} = F_{i-1}$. Hence, since F_{i-1} is an idempotent,

$$\mathbf{T}_{i-1} \subseteq \mathbf{U}_{i-1} \odot \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E \odot V'_{i-1} \odot F_{i-1}]_{\mathrm{e}}}.$$

Using Fact 82 again, we have $\mathbf{R} \subseteq \mathcal{T}^n[\mathbf{L}]|_{[R]_{\mathrm{e}}}$. Thus, using Fact 81 together with the fact that $V'_i = V'_{i-1} \odot F_{i-1} \odot R$ by definition, this yields the following,

$$\begin{array}{rcl} \mathbf{T}_{i-1} \odot \mathbf{R} & \subseteq & \mathbf{U}_{i-1} \odot \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E \odot V'_{i-1} \odot F_{i-1} \odot R]_{\mathrm{e}}} \\ & \subseteq & \mathbf{U}_{i-1} \odot \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E \odot V'_i]_{\mathrm{e}}}. \end{array}$$

Finally, since $\mathbf{U}_i = \mathbf{S} \odot \mathbf{U}_{i-1}$ by definition, we have

$$\begin{array}{rcl} \mathbf{S} \odot \mathbf{T}_{i-1} \odot \mathbf{R} & \subseteq & \mathbf{S} \odot \mathbf{U}_{i-1} \odot \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E \odot V'_i]_{\mathrm{e}}} \\ & \subseteq & \mathbf{U}_i \odot \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E \odot V'_i]_{\mathrm{e}}}. \end{array}$$

This conclude the proof of Lemma 78.

B. Proof of Proposition 43

Proposition 43. *For all $g \in \mathbb{N}$, there exists $p_g \geq 1$ such that for any $n \geq 1$ and any computation tree $\mathbb{T}$ of level n and operational height at most g, if $(S, \mathbf{S}) = lab(\mathbb{T})$, then,*

$$(S, ex_{p_g}(\mathbf{S})) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}]).$$

We begin with a preliminary result about p-extractions which is central in the proof. We call it the *extraction lemma*.

Lemma 83 (Extraction lemma)**.** *Let $q, k \geq 1$, $p \geq q|\mathbf{L}|^{4(k-1)}$ and $\mathbf{S}_1, \ldots, \mathbf{S}_k \subseteq \mathbf{L}^n$ for some n. Then,*

$$ex_p(\mathbf{S}_1 \odot \cdots \odot \mathbf{S}_k) \subseteq ex_q(\mathbf{S}_1) \odot \cdots \odot ex_q(\mathbf{S}_k)$$

Proof. We use induction on k. When $k = 1$, it is immediate from the definition that $ex_p(\mathbf{S}_1) \subseteq ex_q(\mathbf{S}_1)$ since $p \geq q$.

Assume now that $k \geq 2$ and let $r = q|\mathbf{L}|^{4(k-2)}$. It is immediate from our induction hypothesis in the cases $k-1$ and 1 that:

$$\begin{array}{rcl} ex_r(\mathbf{S}_1 \odot \cdots \odot \mathbf{S}_{k-1}) & \subseteq & ex_q(\mathbf{S}_1) \odot \cdots \odot ex_q(\mathbf{S}_{k-1}). \\ ex_r(\mathbf{S}_k) & \subseteq & ex_q(\mathbf{S}_k). \end{array}$$

Hence, it now suffices to prove that,

$$ex_p(\mathbf{S}_1 \odot \cdots \odot \mathbf{S}_k) \subseteq ex_r(\mathbf{S}_1 \odot \cdots \odot \mathbf{S}_{k-1}) \odot ex_r(\mathbf{S}_k).$$

Let $(P_1, P_2) \in ex_p(\mathbf{S}_1 \odot \cdots \odot \mathbf{S}_k)$, we prove that $(P_1, P_2) \in ex_r(\mathbf{S}_1 \odot \cdots \odot \mathbf{S}_{k-1}) \odot ex_r(\mathbf{S}_k)$. By definition, $(P_1, P_2)^p$ is a subsequence of some n-tuple in $\mathbf{S}_1 \odot \cdots \odot \mathbf{S}_k$. Hence, there exist $(H_1, \ldots, H_{2p})$ and $(L_1, \ldots, L_{2p})$ which are respectively subsequences of an n-tuple in $\mathbf{S}_1 \odot \cdots \odot \mathbf{S}_{k-1}$ and an n-tuple in $\mathbf{S}_k$, and such that:

$$(P_1, P_2)^p = (H_1, \ldots, H_{2p}) \odot (L_1, \ldots, L_{2p}).$$

By choice of r and since $p \geq q|\mathbf{L}|^{4(k-1)}$ by hypothesis, we have $p \geq r|\mathbf{L}|^4$. By the pigeon-hole principle, there exist at least r identical 4-tuples $(H_{2k-1}, H_{2k}, L_{2k-1}, L_{2k}) \in \mathbf{L}^4$, which yields an index $i \leq p$ and at least r indices $j \leq p$ satisfying

$$\begin{array}{rcl} (H_{2i-1}, H_{2i}) & = & (H_{2j-1}, H_{2j}), \\ \text{and } (L_{2i-1}, L_{2i}) & = & (L_{2j-1}, L_{2j}). \end{array}$$

Hence, we have $(H_{2i-1}, H_{2i}) \in ex_r(\mathbf{S}_1 \odot \cdots \odot \mathbf{S}_{k-1})$ and $(L_{2i-1}, L_{2i}) \in ex_r(\mathbf{S}_k)$. Moreover, since $(P_1, P_2) = (H_{2i-1}, H_{2i}) \odot (L_{2i-1}, L_{2i})$, we conclude that $(P_1, P_2) \in ex_r(\mathbf{S}_1 \odot \cdots \odot \mathbf{S}_{k-1}) \odot ex_r(\mathbf{S}_k)$. This terminates the proof. □

We now start the proof of Proposition 43. We prove a slightly stronger statement. For any $g \in \mathbb{N}$, we exhibit $p_g \geq 1$ such that for any $p \geq p_g$ and any computation tree of operational height at most g, if $(S, \mathbf{S})$ is its label, then

$$(S, ex_p(\mathbf{S})) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}]).$$

The proof is an induction on g. We first treat the case $g = 0$. In this case, we prove that choosing $p_0 = 1$ suffices. Consider $p \geq p_0$ and let $\mathbb{T}$ be a computation tree whose operational height is 0. We let n be its level. By definition, it follows that $\mathbb{T}$ contains only leaves, downset nodes and multiplication nodes. Therefore, its label $(S, \mathbf{S})$ is the multiplication of elements belonging to $\mathcal{J}^n_{triv}[\mathbf{L}]$. One may verify from its definition that $\mathcal{J}^n_{triv}[\mathbf{L}]$ is closed under downset and multiplication. Hence, it follows that $(S, \mathbf{S})$ itself belongs to $\mathcal{J}^n_{triv}[\mathbf{L}]$. By definition, it then follows that $(S, ex_p(\mathbf{S})) \in \mathcal{J}^2_{triv}[\mathbf{L}]$. Finally, since $\mathcal{J}^2_{triv}[\mathbf{L}] \subseteq Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$ by definition, we conclude that $(S, ex_p(\mathbf{S})) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$.

This takes care of the base case, assume now that $g \geq 1$. We first choose the appropriate $p_g \geq 1$. This choice is based on two constants. The first one is $p_{g-1} \geq 1$ which we obtain by induction: for any $p \geq p_{g-1}$ and any computation tree of operational height at most $g-1$, if $(P, \mathbf{P})$ is its label, then we know that:

$$(P, ex_p(\mathbf{P})) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}]).$$

The second constant is given by the following lemma.

Lemma 84. *There exists $s \geq 1$ such that for any integers $q \geq s$ and $n \geq 1$ and any pair $(L_1, L_2) \in \mathbf{L}$, if $(L_1, L_2)^q \in \mathcal{T}^{2q}[\mathbf{L}]$, then $(L_1, L_2) \in \mathcal{A}[\mathbf{L}]$.*

Proof. By Lemma 12, $(L_1, L_2) \in \mathcal{A}[\mathbf{L}]$ if and only there exists arbitrarily large $q \in \mathbb{N}$ such that $(L_1, L_2)^q \in \mathcal{T}^{2q}[\mathbf{L}]$. Since $\mathbf{L}^2$ is finite, the existence of s follows. □

For the remainder of the proof, we let p_{g-1} and s be defined as above. Moreover, we let $m = max(s, p_{g-1})$. We now define p_g in two steps. Let

$$\begin{aligned} q &= m \times |\mathbf{L}|^{4\times 2\times |2^{\mathbf{L}^2}|}, \quad \text{and} \\ p_g &= q \times |\mathbf{L}|^{4\times 2\times 3(|\mathbf{L}|+1)(|2^{\mathbf{L}^2}|+2)}. \end{aligned}$$

It remains to prove that this choice of p_g satisfies the desired properties. Let $p \geq p_g$ and consider a computation tree $\mathbb{T}$ of operational height at most g and let n be its level. Finally, let $(S, \mathbf{S}) = lab(\mathbb{T})$. We have to prove that

$$(S, ex_p(\mathbf{S})) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}]).$$

Consider a node x of $\mathbb{T}$. We say that x is a *frontier node* when the two following properties hold:

1) The subtree $\mathbb{T}_x$ which is rooted in x has one of the two following properties:
   - $\mathbb{T}_x$ has operational height at most $g-1$, or,
   - $\mathbb{T}_x$ has operational height g and x is an operation node.
2) There exists no ancestor of x satisfying Item 1.

By definition, each branch of $\mathbb{T}$ contains a node satisfying Item 1, namely its leaf, whose operational height is $0 \leq g-1$. Hence, each branch contains a unique frontier node by Item 2. We let $x_1, \ldots, x_k$ as the list (from the left to right) of all frontier nodes in $\mathbb{T}$. Furthermore, for each $i \leq k$, we let $(S_i, \mathbf{S}_i) = lab(x_i)$. By definition, and since $\mathbb{T}$ has operational height at most g, we have the following fact.

Fact 85. $(S, \mathbf{S}) \subseteq (S_1, \mathbf{S}_1) \odot \cdots \odot (S_k, \mathbf{S}_k)$.

Proof. Since $\mathbb{T}$ has operational height smaller than g, all ancestors of a frontier node must be downset or binary nodes (an operation node would satisfy Item 1 above and this is not possible by Item 2). The fact is then immediate by definition of downset and binary nodes since $(S, \mathbf{S})$ is the label of $\mathbb{T}$. □

Hence, by closure under downset in the definition of $Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$, our new objective is to prove the following:

$$(S_1 \odot \cdots \odot S_k, ex_p(\mathbf{S}_1 \odot \cdots \odot \mathbf{S}_k)) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}]).$$

The argument is based on a sub-induction using our generalized factorization forest theorem. However, presenting it requires formalizing the property of $(S_1, \mathbf{S}_1), \ldots, (S_k, \mathbf{S}_k)$ which we use. Let us first introduce a name for it.

Consider a sequence $(R_1, \mathbf{R}_1), \ldots, (R_\ell, \mathbf{R}_\ell) \in \mathbf{L} \times 2^{\mathbf{L}^n}$. We say that this sequence is *good* when it satisfies the two following properties:

1) For all $i < j \leq \ell$,
$$ex_q(\mathbf{R}_{i+1} \odot \cdots \odot \mathbf{R}_j) \subseteq \mathcal{A}[\mathbf{L}]|_{[R_{i+1}\odot\cdots\odot R_j]_{\mathcal{C}}}$$
2) For all $i \leq \ell$, $(R_i, ex_q(\mathbf{R}_i)) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$.

Do note that we use the constant $q = m|\mathbf{L}|^{4\times 2\times |2^{\mathbf{L}^2}|}$ in the definition. In particular, q is strictly smaller than $p \geq p_g$. Let us prove that $(S_1, \mathbf{S}_1), \ldots, (S_k, \mathbf{S}_k)$ is indeed good.

Lemma 86. *The sequence $(S_1, \mathbf{S}_1), \ldots, (S_k, \mathbf{S}_k)$ is good.*

Proof. We have two properties to prove. We start with the first one. Let $i < j \leq k$, we prove that $ex_q(\mathbf{S}_{i+1} \odot \cdots \odot \mathbf{S}_j) \subseteq \mathcal{A}[\mathbf{L}]|_{[S_{i+1}\odot\cdots\odot S_j]_{\mathcal{C}}}$. By Proposition 41, $(S_i, \mathbf{S}_i) \in \mathcal{J}^n[\mathbf{L}]$ for all $i \leq k$ (it is the label of a computation tree of level n). Hence, since $\mathcal{J}^n[\mathbf{L}]$ is closed under multiplication by Lemma 31, for all $i < j \leq k$,

$$(S_{i+1} \odot \cdots \odot S_j, \mathbf{S}_{i+1} \odot \cdots \odot \mathbf{S}_j) \in \mathcal{J}^n[\mathbf{L}]$$

It now follows from Lemmas 28 and 32 that,

$$\mathbf{S}_{i+1} \odot \cdots \odot \mathbf{S}_j \subseteq \mathcal{T}^n[\mathbf{L}]|_{[S_{i+1}\odot\cdots\odot S_j]_{\mathcal{C}}}$$

Moreover, we have $q \geq s$ by definition, and by choice of s in Lemma 84, we have $ex_q(\mathcal{T}^n[\mathbf{L}]) \subseteq \mathcal{A}[\mathbf{L}]$. It then follows that,

$$ex_q(\mathbf{S}_{i+1} \odot \cdots \odot \mathbf{S}_j) \subseteq \mathcal{A}[\mathbf{L}]|_{[S_{i+1}\odot\cdots\odot S_j]_{\mathcal{C}}}$$

This terminates the proof of the first item. We turn to the second one. Let $i \leq k$, we have to prove that $(S_i, ex_q(\mathbf{S}_i)) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$.

By definition $(S_i, \mathbf{S}_i) = lab(x_i)$. Since x_i is a frontier node, we consider two cases depending on which property holds in Item 1 of the definition.

Let us first assume that the subtree rooted in x_i has operational height smaller than $g-1$. Observe that by definition, $q \geq m \geq p_{g-1}$. It is now immediate from our choice of p_{g-1} that $(S_i, ex_q(\mathbf{S}_i)) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$.

We now assume that the subtree rooted in x_i has operational height g and x_i is an operation node. The argument is more involved. By definition x_i has a single child y whose operational height is $g-1$ and whose label $(E, \mathbf{E}) \in \mathbf{L} \times 2^{\mathbf{L}^n}$ is an idempotent. Furthermore, we have,

$$(S_i, \mathbf{S}_i) = (E, \mathbf{E} \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}} \odot \mathbf{E})$$

In particular, we have $S = E$. Recall that $m = max(s, p_{g-1})$. This gives us the two following properties:

1) Since $m \geq p_{g-1}$ and $(E, \mathbf{E})$ is the label of a tree whose operational height $g-1$, $(E, ex_m(\mathbf{E})) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$ by definition of p_{g-1}.
2) Since $m \geq s$, we get by choice of s in Lemma 84 that, $ex_m(\mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}}) \subseteq \mathcal{A}[\mathbf{L}]|_{[E]_{\mathcal{C}}}$.

To simplify notations, we write $\mathbf{P} = ex_m(\mathcal{T}^n[\mathbf{L}]|_{[E]_{\mathcal{C}}})$ in the following.

Note that while $ex_m(\mathbf{E})$ is the m-extraction of an idempotent, it needs not be idempotent itself. However, since it is an element of $2^{\mathbf{L}^2}$, it follows from a standard semigroup theory argument that there exists $\ell \leq |2^{\mathbf{L}^2}|$ such that $(ex_m(\mathbf{E}))^\ell$ is an idempotent. We write $\mathbf{F} = (ex_m(\mathbf{E}))^\ell \in 2^{\mathbf{L}^2}$. Observe that by closure under multiplication in the definition of Sat^2 and since E is idempotent, we have $(E, \mathbf{F}) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$.

We may now use the fact that $\mathbf{P} \subseteq \mathcal{A}[\mathbf{L}]|_{[E]_{\mathcal{C}}}$ together with closure under downset and Operation 3 in the definition of Sat^2 to obtain,

$$(E, \mathbf{F} \odot \mathbf{P} \odot \mathbf{F}) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}]).$$

It remains to show that $(S_i, ex_q(\mathbf{S}_i)) \subseteq (E, \mathbf{F} \odot \mathbf{P} \odot \mathbf{F})$. By closure under downset, it will then follow that $(S_i, ex_q(\mathbf{S}_i)) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$ as desired.

We already know that $S_i = E$. Hence, it suffices to prove that $ex_q(\mathbf{S}_i) \subseteq \mathbf{F} \odot \mathbf{P} \odot \mathbf{F}$. Since $\mathbf{E}$ is idempotent, we have

$$\mathbf{S}_i = \mathbf{E}^\ell \odot \mathcal{T}^n[\mathbf{L}]|_{[E]_\mathrm{e}} \odot \mathbf{E}^\ell.$$

Moreover, since $\ell \leq |2^{\mathbf{L}^2}|$, this makes $\mathbf{S}_i$ the multiplication of at most $2 \times |2^{\mathbf{L}^2}| + 1$ elements. Recall that we chose

$$q = m \times |\mathbf{L}|^{4 \times 2 \times |2^{\mathbf{L}^2}|}$$

Hence, it is immediate from Lemma 83 that,

$$ex_q(\mathbf{S}_i) \subseteq (ex_m(\mathbf{E}))^\ell \odot ex_m(\mathcal{T}^n[\mathbf{L}])|_{[E]_\mathrm{e}} \odot (ex_m(\mathbf{E}))^\ell$$

This exactly says that $ex_q(\mathbf{S}_i) \subseteq \mathbf{F} \odot \mathbf{P} \odot \mathbf{F}$. □

We may now present the inductive argument. Consider the partial semigroup $\mathbf{L}$ and the semigroup $2^{\mathbf{L}^2}$. We work with γ-factorization forests for the following map γ.

$$\begin{array}{rccc} \gamma: & \mathbf{L} & \to & 2^{\mathbf{L}^2} \\ & H & \mapsto & \mathcal{A}[\mathbf{L}]|_{[H]_\mathrm{e}} \end{array}$$

That $(S_1 \odot \cdots \odot S_k, ex_p(\mathbf{S}_1 \odot \cdots \odot \mathbf{S}_k)) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$ is now a consequence of the following lemma.

Lemma 87. *Let $(R_1, \mathbf{R}_1), \ldots, (R_\ell, \mathbf{R}_\ell) \in \mathbf{L} \times 2^{\mathbf{L}^n}$ be a good sequence and let $w \in (\mathbf{L} \times 2^{\mathbf{L}^2})^+$ be as follows:*

$$w = (R_1, ex_q(\mathbf{R}_1)) \cdots (R_\ell, ex_q(\mathbf{R}_\ell)).$$

Let $h \in \mathbb{N}$ and assume that w admits a γ-factorization forest of height h and let $(P, \mathbf{P})$ be its root label. Then, for any $r \geq q \times |\mathbf{L}|^{4 \times 2 \times h}$, we have,

$$ex_r(\mathbf{R}_1 \odot \cdots \odot \mathbf{R}_\ell) \subseteq \mathbf{P}.$$

Before proving Lemma 87, we first finish the proof of Proposition 43. We know from Lemma 86 that $(S_1, \mathbf{S}_1), \ldots, (S_k, \mathbf{S}_k)$ is good. Consider the word,

$$w = (S_1, ex_q(\mathbf{S}_1)) \cdots (S_\ell, ex_q(\mathbf{S}_k)) \in (\mathbf{L} \times 2^{\mathbf{L}^2})^+$$

Clearly, the evaluation of w is defined: $S_1 \odot \cdots \odot S_\ell = S$. Hence, it follows from Theorem 61 that w admits a γ-factorization forest of height $h \leq 3(|\mathbf{L}| + 1)(|2^{\mathbf{L}^2}| + 2)$. Let $(P, \mathbf{P})$ be its root label. Since $(S_1, \mathbf{S}_1), \ldots, (S_k, \mathbf{S}_k)$ is good, we know that $(S_i, ex_q(\mathbf{S}_i)) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$ for all i. Therefore, by choice of γ it is simple to verify from the definition of $Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$ that $(P, \mathbf{P}) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$. Finally, by choice of p_g, we have $p \geq p_g \geq q \times |\mathbf{L}|^{4 \times 2 \times h}$. Hence, we get from Lemma 87 that,

$$(S_1 \odot \cdots \odot S_k, ex_p(\mathbf{S}_1 \odot \cdots \mathbf{S}_k)) \subseteq (P, \mathbf{P})$$

By closure under downset, we conclude that,

$$(S_1 \odot \cdots \odot S_k, ex_p(\mathbf{S}_1 \odot \cdots \mathbf{S}_k)) \in Sat^2(\mathbf{L}, \mathcal{A}[\mathbf{L}])$$

This terminates the proof of Proposition 43. It remains to prove Lemma 87. We finish the appendix with this proof.

Proof of Lemma 87. We use induction on the height h of the γ-factorization forest for w. When $h = 0$, the root node is a leaf which means that $\ell = 1$. Therefore, $(P, \mathbf{P}) = (R_1, ex_q(\mathbf{R}_1))$ by definition. Since $r \geq q$ by definition, we obtain from the case $k = 1$ in Lemma 83 that $(R_1, ex_r(\mathbf{R})) \subseteq (P, \mathbf{P})$.

We now assume that $h \geq 1$ and we let $t = q \times |\mathbf{L}|^{4 \times 2 \times (h-1)}$. We consider two cases depending on the nature of the root.

Assume first that the root is a binary node. Hence, it has two children of height at most $h-1$ which are associated to w_1 and w_2 such that $w = w_1 w_2$. Let $(P_1, \mathbf{P}_1)$ and $(P_2, \mathbf{P}_2)$ be the labels of these children, we have $(P, \mathbf{P}) = (P_1, \mathbf{P}_1) \odot (P_2, \mathbf{P}_2)$ by definition. Since $w = w_1 w_2$, there exists $j < \ell$ such that,

$$\begin{array}{rcl} w_1 & = & (R_1, ex_q(\mathbf{R}_1)) \cdots (R_j, ex_q(\mathbf{R}_j)) \\ w_2 & = & (R_{j+1}, ex_q(\mathbf{R}_{j+1})) \cdots (R_\ell, ex_q(\mathbf{R}_\ell)) \end{array}$$

We now use induction. Let $\mathbf{T}_1 = ex_t(\mathbf{R}_1 \odot \cdots \odot \mathbf{R}_j)$ and $\mathbf{T}_2 = ex_t(\mathbf{R}_{j+1} \odot \cdots \odot \mathbf{R}_\ell)$. By choice of t, we may apply our induction hypothesis to w_1 and w_2 which yields,

$$\mathbf{T}_1 \subseteq \mathbf{P}_1 \quad \text{and} \quad \mathbf{T}_2 \subseteq \mathbf{P}_2$$

Moreover, by definition, we have $r \geq t \times |\mathbf{L}|^4$. Hence, we may apply Lemma 83 (in the case $k = 2$) to obtain that,

$$ex_r(\mathbf{R}_1 \odot \cdots \odot \mathbf{R}_\ell) \subseteq \mathbf{T}_1 \odot \mathbf{T}_2 \subseteq \mathbf{P}_1 \odot \mathbf{P}_2 = \mathbf{P}$$

This terminates the binary case.

Let us now assume that the root is an idempotent node. Hence, it has an arbitrary number $c \geq 3$ of children of height at most $h - 1$ which are associated to $w_1, \ldots, w_c$ such that $w = w_1 \cdots w_c$. Let $(E, \mathbf{P}_1), \ldots, (E, \mathbf{P}_c)$ be their labels from left to right. By definition, we know that

1) E is an idempotent of $\mathbf{L}$.
2) $\mathbf{P}_1 = \mathbf{P}_c$ and it is an idempotent $\mathbf{E}$ of $2^{\mathbf{L}^2}$.
3) $(P, \mathbf{P}) = (E, \mathbf{E} \odot \gamma(E) \odot \mathbf{E})$.

There exist $i < j < \ell$ such that,

$$\begin{array}{rcl} w_1 & = & (R_1, ex_q(\mathbf{R}_i)) \cdots (R_j, ex_q(\mathbf{R}_i)) \\ w_c & = & (R_{j+1}, ex_q(\mathbf{R}_{j+1})) \cdots (R_\ell, ex_q(\mathbf{R}_\ell)) \end{array}$$

Let us first apply induction. Let $\mathbf{T}_1 = ex_t(\mathbf{R}_1 \odot \cdots \odot \mathbf{R}_i)$, and $\mathbf{T}_c = ex_t(\mathbf{R}_{j+1} \odot \cdots \odot \mathbf{R}_\ell)$. By choice of t, we may apply induction on w_1 and w_c to obtain that,

$$\mathbf{T}_1 \subseteq \mathbf{P}_1 = \mathbf{E} \quad \text{and} \quad \mathbf{T}_c \subseteq \mathbf{P}_c = \mathbf{E}$$

Furthermore, since $r \geq t \times |\mathbf{L}|^{4 \times 2}$ by definition, we may apply Lemma 83 (in the case $k = 3$) to obtain that,

$$ex_r(\mathbf{R}_1 \odot \cdots \odot \mathbf{R}_\ell) \subseteq \mathbf{T}_1 \odot ex_t(\mathbf{R}_{i+1} \odot \cdots \odot \mathbf{R}_j) \odot \mathbf{T}_c$$

We may now combine this with $\mathbf{T}_1, \mathbf{T}_2 \subseteq \mathbf{E}$ to obtain that,

$$ex_r(\mathbf{R}_1 \odot \cdots \odot \mathbf{R}_\ell) \subseteq \mathbf{E} \odot ex_t(\mathbf{R}_{i+1} \odot \cdots \odot \mathbf{R}_j) \odot \mathbf{E}$$

Therefore, since $\mathbf{P} = \mathbf{E} \odot \gamma(E) \odot \mathbf{E}$, it remains to prove that $ex_t(\mathbf{R}_{i+1} \odot \cdots \odot \mathbf{R}_j) \subseteq \gamma(E) = \mathcal{A}[\mathbf{L}]|_{[E]_\mathrm{e}}$. By hypothesis on our γ-factorization forest, we have $R_{i+1} \odot \cdots \odot R_j = E$. We use our hypothesis that $(R_1, \mathbf{R}_1), \ldots, (R_\ell, \mathbf{R}_\ell)$ is good:

$$ex_t(\mathbf{R}_{i+1} \odot \cdots \odot \mathbf{R}_j) \subseteq \mathcal{A}[\mathbf{L}]|_{[E]_\mathrm{e}} = \gamma(E)$$

This concludes the proof of Lemma 87. □

EXTRA REFERENCES FOR THE APPENDIX

[3] M. Bojańczyk, "The common fragment of ACTL and LTL," in *FoSSaCS'08*, R. Amadio, Ed. Springer, 2008.

[4] M. Bojańczyk, "Factorization forests," in *Proceedings of the 13th International Conference on Developments in Language Theory*, ser. DLT'09, Berlin, Heidelberg: Springer-Verlag, 2009.

[7] T. Colcombet, "Factorization forests for infinite words and applications to countable scattered linear orderings," *Theoret. Comp. Sci.*, vol. 411, no. 4-5, 2010.

[13] T. Hall and M. Sapir, "Idempotents, regular elements and sequences from finite semigroups," *Discrete Mathematics*, vol. 161, no. 1, pp. 151–160, 1996.

[15] M. Kufleitner, "The height of factorization forests," in *MFCS'08*, Berlin, Heidelberg: Springer-Verlag, 2008.

[32] I. Simon, "Factorization forests of finite height," *Theoret. Comp. Sci.*, vol. 72, no. 1, 1990.