The Covering Problem: a Unified Approach for Investigating the Expressive Power of Logics[*]

Thomas Place[1] and Marc Zeitoun[2]

1 **LaBRI, Bordeaux University, France**
thomas.place@labri.fr

2 **LaBRI, Bordeaux University, France**
marc.zeitoun@labri.fr

Abstract

An important endeavor in computer science is to precisely understand the expressive power of logical formalisms over discrete structures, such as words. Naturally, "understanding" is not a mathematical notion. Therefore, this investigation requires a concrete objective to capture such a notion. In the literature, the standard choice for this objective is the *membership problem*, whose aim is to find a procedure deciding whether an input regular language can be defined in the logic under study. This approach was cemented as the "right" one by the seminal work of Schützenberger, McNaughton and Papert on first-order logic and has been in use since then.

However, membership questions are hard: for several important fragments, researchers have failed in this endeavor despite decades of investigation. In view of recent results on one of the most famous open questions, namely the quantifier alternation hierarchy of first-order logic, an explanation may be that membership is too restrictive as a setting. These new results were indeed obtained by considering more general problems than membership, taking advantage of the increased flexibility of the enriched mathematical setting. This opens a promising avenue of research and efforts have been devoted at identifying and solving such problems for natural fragments. However, until now, these problems have been *ad hoc*, most fragments relying on a specific one. A unique new problem replacing membership as the right one is still missing.

The main contribution of this paper is a suitable candidate to play this role: the Covering Problem. We motivate this problem with three arguments. First, it admits an elementary set theoretic formulation, similar to membership. Second, we are able to reexplain or generalize all known results with this problem. Third, we develop a mathematical framework as well as a methodology tailored to the investigation of this problem.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases Membership Problem, Separation Problem, Covering Problem, Regular Languages, Logics, Decidable Characterizations

Digital Object Identifier 10.4230/LIPIcs.MFCS.2016.77

1 Introduction

One of the most successful applications of the notion of regularity in computer science is the investigation of logics on discrete structures such as words or trees. The story began in the 60s when Büchi [5], Elgot [10] and Trakhtenbrot [36] proved that the *regular languages* of

[*] This study has been carried out with financial support from the French State, managed by the French National Research Agency (ANR) in the frame of the "Investments for the future" Programme IdEx Bordeaux – CPU (ANR-10-IDEX-03-02).

finite words are those that can be defined in monadic second order logic (MSO). This result has since been exploited to study the expressive power of important fragments of MSO by relying on a decision problem: the *membership problem*. Given a regular language as input, this problem asks if it can be defined by a sentence of the fragment under investigation.

Getting membership algorithms is difficult. In fact, this is still open on finite trees for the most natural fragment of MSO, namely first-order logic (FO). On words however, this question was solved in the 70s by Schützenberger, McNaughton and Papert [30, 14]. This theorem was very influential and has often been revisited [38, 9, 7, 20]. It paved the way to a series of results of the same nature. A famous example is Simon's Theorem [31], which yields an algorithm for the first level of the quantifier alternation hierarchy of FO. Other examples include [4, 13, 39, 33] which consider fragments of FO where the linear order on positions is replaced by the successor relation or [34] which considers the 2-variable fragment of FO. The relevance of this approach is nowadays validated by a wealth of results.

The reason for this success is twofold. First, these results cemented membership as the "right" question: a solution conveys a deep intuition on the investigated logic. In particular, most results include a *generic method for building a canonical sentence* witnessing membership of an input language in the logic. Second, Schützenberger's solution established a suitable framework and a methodology to solve membership problems. This methodology is based on a canonical algebraic abstraction of a regular language which is finite and computable, the *syntactic monoid*. The core of the approach is to translate the semantic question (*is the language definable in the fragment?*) into a purely syntactical, easy question to be tested on the syntactic monoid (*does the syntactic monoid satisfy some equation?*).

Unfortunately, this methodology seems to have reached its limits for the hardest questions. An emblematic example is the *quantifier alternation hierarchy of first-order logic* which classifies sentences according to the number of alternations between $\exists$ and $\forall$ quantifiers in their prenex normal form. A sentence is Σ_i if its prenex normal form has $(i-1)$ alternations and starts with a block of existential quantifiers. A sentence is $\mathcal{B}\Sigma_i$ if it is a boolean combination of Σ_i sentences. Obtaining membership algorithms for all levels in this hierarchy is a major open question and has been given a lot of attention (see [37, 35, 15, 16, 17, 18, 27, 19] for details and a complete bibliography). However, progress on this question has been slow: until recently, only the lowest levels were solved: Σ_1 [3, 21], $\mathcal{B}\Sigma_1$ [31] and Σ_2 [3, 21].

It took years to solve higher levels. Recently, membership algorithms were obtained for Σ_3 [25], $\mathcal{B}\Sigma_2$ [25] and Σ_4 [22]. This was achieved by introducing new ingredients into Schützenberger's methodology: problems that are *more general than membership*. For each result, the strategy is the same: first, a well-chosen more general problem is solved for a lower level in the hierarchy, then, this result is transferred into a membership algorithm for the level under investigation. Let us illustrate what we mean by "more general problem" and present the simplest of them: the *separation problem*. It takes *two* regular languages as input and asks whether there exists a third one which is definable in the logic, contains the first, and is disjoint from the second. Being more general, such problems are also more difficult than membership. However, this generality also makes them more rewarding in the insight they give on the investigated logic. This motivated a series of papers on the separation problem [28, 8, 23, 24, 26] which culminated in the three results above [25, 22]. However, while this avenue of research is very promising, it presently suffers three important flaws:

1. The family of problems that have been considered up until now is a jungle: each particular result relies on a specific ad-hoc problem. For example, the results of [25, 22] rely on three different problems. In fact, even if one is only interested in separation, the actual solution often considers an even more general problem (see [28, 25, 22] for example).

2. Among the problems that have been investigated, separation is the only one that admits a simple and generic set-theoretic definition (which is why it is favored as an example). On the other hand, for all other problems, the definition requires to introduce additional concepts such as semigroups and Ehrenfeucht-Fraïssé games.
3. In contrast to membership solutions, the solutions that have been obtained for these more general problems are non-constructive. For example, most of the separation solutions do not include a generic method for building a separator language when it exists (the algorithms are built around the idea of proving that the two inputs are *not* separable).

Contributions. Our objective in this paper is to address these three issues. Our first contribution is the presentation of a single general problem, the "*covering problem*", which admits a purely set-theoretic definition and generalizes all problems that have already been considered. Furthermore, its definition is modular: the covering problem is designed so that it can easily be generalized to accommodate future needs. Its design is based on an analysis of the methods used to solve membership and separation. In both cases, the algorithms almost always exploit the fact that an input regular language L is *not isolated*: its recognizer defines a *set* of regular languages from which L is built. This set has a structure upon which the algorithms are based. The covering problem takes this observation into account: an input of the problem is directly *any finite set* of regular languages. Given such a set $\mathbf{L}$, the problem asks to compute the "best possible approximation" (called *optimal cover*, hence the name "covering") of this set of languages by languages belonging to the investigated fragment. In particular, the separation problem is just the special case when the input set is of size 2.

The main advantage of the covering problem is that it comes with a generic framework and a generic methodology designed for solving it. This framework is our second contribution. It generalizes the original framework of Schützenberger for membership in a natural way and lifts all its benefits to a more general setting. In particular, we recover *constructiveness*: a solution to the covering problem associated to a particular fragment yields a generic way for building an actual optimal cover of the input set.

Finally, the relevance of our new framework is supported by the fact that we are able to obtain covering algorithms for the fragments that were already known to enjoy a decidable separation problem. In contrast to the previous algorithms, these more general ones are presented within a single unified framework. This is our third contribution. We present actual covering algorithms for four particular logics: first-order logic (FO), two-variables FO (FO^2) and two logics within the quantifier alternation hierarchy of FO ($\mathcal{B}\Sigma_1$ and Σ_2). As explained, the payoff is that we obtain *effective* solutions to the covering problem. Hence, we obtain an effective method for building separators in the weaker separation problem.

Historical note. As observed by Almeida [1], separation is tied to a purely algebraic problem of Henckell and Rhodes (see [11, 12]): computing the "pointlike sets of a given finite semigroup with respect to a variety $\mathbf{V}$". This can probably be lifted to covering. However, there are two main advantages to our approach. First, it is more general: pointlike sets are restricted to classes and inputs that are both more specific than ours. Second, covering admits a simple set theoretic definition that pointlike sets obfuscate with heavy terminology.

Organization. We define the covering problem in Section 2 (for arbitrary input sets of languages, *i.e.*, not necessarily made of regular languages). We present our framework for the particular case of regular inputs in Sections 3 and 4. Four examples of covering algorithms are presented in Section 5. Due to lack of space, proofs are deferred to the journal version.

2 The Covering Problem

In this section, we define the covering problem. For the whole paper, we fix a finite alphabet A and work with finite words over A (*i.e.*, elements of A^*). A *language* is a subset of A^*. Note that we restrict ourselves to words for the sake of simplifying the presentation. However, the covering problem makes sense for *any structure* (such as infinite words or trees).

We focus on two kinds of classes of languages. We say that a class of languages $\mathscr{C}$ is a *lattice* when it contains the empty and universal languages ($\emptyset$ and A^*) and it is closed under finite union and finite intersection: $K, L \in \mathscr{C}$ implies $K \cup L, K \cap L \in \mathscr{C}$. Furthermore, $\mathscr{C}$ is a *boolean algebra* when $\mathscr{C}$ is a lattice that is closed under complement: $L \in \mathscr{C}$ implies $\{w \in A^* \mid w \notin L\} \in \mathscr{C}$. The covering problem then comes into two variants:

- a variant that can be associated to any class of languages that is a lattice. We call this variant the *pointed covering problem.*
- a weaker variant that can be associated to any class of languages that is a boolean algebra. We call it the *covering problem.* While weaker than the first one, this variant enjoys simpler terminology, which makes it our choice when working with boolean algebras.

We now define these two variants. In the definition, we use the separation problem as a foundation to motivate and explain our design choices. As we explained, given a class of languages $\mathscr{C}$, solutions to membership and separation exploit the fact that the recognizer of an input regular language L recognizes a *set* of regular languages from which L is built. The covering problem is based on this observation: its input is any finite set of languages $\mathbf{L}$.

▶ Remark. A "set of languages" is a purely mathematical object. An actual input is a set of recognizing devices for these languages. In particular, it may happen that two such devices recognize the same language. Therefore our inputs are actually finite sets of languages *names* (which may contain "several copies" of the same language). This is harmless: two sets of names for the same underlying set of languages are equivalent for both covering problems.

2.1 The Covering Problem for Boolean Algebras

We begin with the simpler covering problem. Let $\mathscr{C}$ be a boolean algebra[1]. Given a finite set of languages names $\mathbf{L} = \{L_1, \dots, L_n\}$, a *$\mathscr{C}$-cover* of $\mathbf{L}$ is a finite set of languages $\mathbf{K} = \{K_1, \dots, K_m\}$ such that $K_i \in \mathscr{C}$ for all $i \leq n$ and:

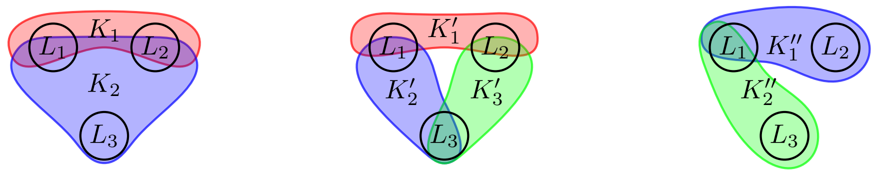
$$L_1 \cup \cdots \cup L_n \subseteq K_1 \cup \cdots \cup K_m.$$

Note that since $\mathscr{C}$ is a boolean algebra, there always exists a $\mathscr{C}$-cover of $\mathbf{L}$: the singleton $\{A^*\}$. When we have a $\mathscr{C}$-cover $\mathbf{K}$ of $\mathbf{L}$ in hand, our main interest will be to know how good $\mathbf{K}$ is at separating languages in $\mathbf{L}$: what languages in $\mathbf{L}$ are separated by unions of languages in $\mathbf{K}$? What are the "best $\mathscr{C}$-covers" of $\mathbf{L}$ (called optimal $\mathscr{C}$-covers)? This information is captured by a new object that we associate to any cover of $\mathbf{L}$, its imprint on $\mathbf{L}$.

Filterings and Imprints. Imprints are based on filterings. Given a finite set of names $\mathbf{L}$ and a language K, the *filtering of* $\mathbf{L}$ *by* K, measures the "interaction" between $\mathbf{L}$ and K. More precisely, the filtering of $\mathbf{L}$ by K, denoted by $\langle \mathbf{L} | K \rangle$, is defined as the following set:

$$\langle \mathbf{L} | K \rangle = \{L \in \mathbf{L} \mid L \cap K \neq \emptyset\} \subseteq \mathbf{L}\,.$$

[1] The problem actually makes sense for any class that contains the universal language and is closed under intersection. However, we need $\mathscr{C}$ to be a boolean algebra for the connection with separation.



Cover $\mathbf{K} = \{K_1, K_2\}$ $\mathcal{I}[\mathbf{L}](\mathbf{K}) =$ ${\downarrow}\{\{L_1, L_2, L_3\}\}$

Cover $\mathbf{K}' = \{K'_1, K'_2, K'_3\}$ $\mathcal{I}[\mathbf{L}](\mathbf{K}') =$ ${\downarrow}\{\{L_1, L_2\}, \{L_1, L_3\}, \{L_2, L_3\}\}$

Cover $\mathbf{K}'' = \{K''_1, K''_2\}$ $\mathcal{I}[\mathbf{L}](\mathbf{K}'') =$ ${\downarrow}\{\{L_1, L_2\}, \{L_1, L_3\}\}$

Figure 1 Some $\mathscr{C}$-covers of $\mathbf{L} = \{L_1, L_2, L_3\}$ and their imprint on $\mathbf{L}$.

▶ Remark. This notion is what makes the problem modular. It can be strengthened to define harder variants of the problem and accommodate future needs.

We may now define imprints. Given a subset E of $2^{\mathbf{L}}$, we write ${\downarrow}E$ to denote the *downset of* E, *i.e.*, the set ${\downarrow}E = \{\mathbf{H} \mid \exists \mathbf{H}' \in E \text{ such that } \mathbf{H} \subseteq \mathbf{H}'\}$. If $\mathbf{K}$ is a finite set of languages, the *imprint of* $\mathbf{K}$ *on* $\mathbf{L}$ is the set,

$$\mathcal{I}[\mathbf{L}](\mathbf{K}) \quad = \quad {\downarrow}\{\langle \mathbf{L}|K\rangle \mid K \in \mathbf{K}\} \subseteq 2^{\mathbf{L}} .$$

Note that we shall mainly use this definition when $\mathbf{K}$ is a $\mathscr{C}$-cover of $\mathbf{L}$. However, in some proofs, it will be convenient to have it for an arbitrary set of languages $\mathbf{K}$. We present examples of imprints when $\mathbf{K}$ is a $\mathscr{C}$-cover of $\mathbf{L}$ in Figure 1.

Let us make a few observations about imprints. An imprint on $\mathbf{L}$ is a subset of $2^{\mathbf{L}}$. Therefore, for a fixed *finite* set $\mathbf{L}$, there are *finitely many* possible imprints on $\mathbf{L}$, even though there are infinitely many finite sets $\mathbf{K}$ of languages. Another simple observation is that all imprints are closed under downset: $\mathcal{I}[\mathbf{L}](\mathbf{K}) = {\downarrow}\mathcal{I}[\mathbf{L}](\mathbf{K})$. Also notice that if $\mathbf{K}$ is a $\mathscr{C}$-cover of $\mathbf{L}$, its imprint captures separation-related information: if $\{L_1, L_2\} \notin \mathcal{I}[\mathbf{L}](\mathbf{K})$, then L_1 (resp. L_2) can be separated from L_2 (resp. L_1) by a union (in $\mathscr{C}$) of languages in $\mathbf{K}$.

▶ Remark. Imprints capture more than just separation-related information. From the separation point of view, the $\mathscr{C}$-covers $\mathbf{K}$ and $\mathbf{K}'$ of Figure 1 are equivalent: they cannot separate any pair of languages in $\mathbf{L}$. However, their imprints on $\mathbf{L}$ tell us that $\mathbf{K}'$ is "better" as it covers $\mathbf{L}$ without containing a language that intersects all languages in $\mathbf{L}$ at the same time.

Finally, observe that if $\mathbf{K}$ is a $\mathscr{C}$-cover of a finite set $\mathbf{L}$, then its imprint on $\mathbf{L}$ always contains some trivial elements. To any finite set of names $\mathbf{L}$, we associate the following set:

$$\mathcal{I}_{triv}[\mathbf{L}] = {\downarrow}\{\langle \mathbf{L}|\{w\}\rangle \mid w \in A^*\} = \{\mathbf{H} \subseteq \mathbf{L} \mid \cap_{H \in \mathbf{H}} H \neq \emptyset\} .$$

▶ **Fact 1.** *For any $\mathscr{C}$-cover $\mathbf{K}$ of $\mathbf{L}$, we have $\mathcal{I}_{triv}[\mathbf{L}] \subseteq \mathcal{I}[\mathbf{L}](\mathbf{K})$.*

Optimal $\mathscr{C}$-covers. We now use imprints to define our notion of "best" $\mathscr{C}$-cover of $\mathbf{L}$ which we call *optimal $\mathscr{C}$-covers.* A necessary (but not sufficient) property for a $\mathscr{C}$-cover of $\mathbf{L}$ to be optimal will be that $L_1, L_2 \in \mathbf{L}$ are $\mathscr{C}$-separable if and only if they can be separated by a union of languages in the $\mathscr{C}$-cover. Formally, we say that a $\mathscr{C}$-cover $\mathbf{K}$ of $\mathbf{L}$ is *optimal* when,

$$\mathcal{I}[\mathbf{L}](\mathbf{K}) \subseteq \mathcal{I}[\mathbf{L}](\mathbf{K}') \quad \text{for any } \mathscr{C}\text{-cover } \mathbf{K}' \text{ of } \mathbf{L}.$$

In general, there can be infinitely many optimal $\mathscr{C}$-covers of a given finite set of names $\mathbf{L}$. We now state that for any $\mathbf{L}$, there always exists an optimal $\mathscr{C}$-cover of $\mathbf{L}$. Note that the proof only requires $\mathscr{C}$ to be closed under finite intersection.

▶ **Lemma 2.** *For any finite set of languages names* $\mathbf{L}$*, there exists an optimal* $\mathscr{C}$*-cover of* $\mathbf{L}$*.*

Note that the proof of Lemma 2 is non-constructive. Given a finite set of names $\mathbf{L}$, computing an actual optimal $\mathscr{C}$-cover is a difficult problem in general. In fact, as seen in Theorem 4 below, this is more general than solving $\mathscr{C}$-separability for any pair of languages in $\mathbf{L}$. Before we present this theorem, let us make a key observation about optimal $\mathscr{C}$-covers.

By definition, given a boolean algebra $\mathscr{C}$ and a finite set of names $\mathbf{L}$, all optimal $\mathscr{C}$-covers of $\mathbf{L}$ have the same imprint on $\mathbf{L}$. Hence, this unique imprint on $\mathbf{L}$ is a *canonical* object for $\mathscr{C}$ and $\mathbf{L}$. We call it the *optimal imprint with respect to* $\mathscr{C}$ *on* $\mathbf{L}$ and we denote it by $\mathcal{I}_{\mathscr{C}}[\mathbf{L}]$:

$$\mathcal{I}_{\mathscr{C}}[\mathbf{L}] = \mathcal{I}[\mathbf{L}](\mathbf{K}) \quad \text{for any optimal } \mathscr{C}\text{-cover } \mathbf{K} \text{ of } \mathbf{L}.$$

We can now state the *covering problem.* We parametrize it by two classes of languages, a class $\mathscr{D}$ constraining the input, and a boolean algebra $\mathscr{C}$.

▶ **Definition 3.** The Covering problem for $\mathscr{C}$ inside $\mathscr{D}$ is as follows:

INPUT:	A finite set of languages $\mathbf{L} \subseteq \mathscr{D}$.
QUESTION:	Compute $\mathcal{I}_{\mathscr{C}}[\mathbf{L}]$.

As expected, we only consider the covering problem when the input class $\mathscr{D}$ is the class of regular languages (in particular we will often simply say "covering problem" for this particular variant). There are two stages when solving the covering problem.

1. *Stage One*: find an algorithm that, given a finite set of regular languages $\mathbf{L}$ as input, computes $\mathcal{I}_{\mathscr{C}}[\mathbf{L}]$ (we call such an algorithm a covering algorithm for $\mathscr{C}$). In Theorem 4 below, we prove that this generalizes separation as a *decision problem.*
2. *Stage Two*: find an algorithm that, given a finite set of regular languages $\mathbf{L}$ as input, computes an optimal $\mathscr{C}$-cover of $\mathbf{L}$ (*i.e.*, one whose imprint is $\mathcal{I}_{\mathscr{C}}[\mathbf{L}]$). We prove below that this generalizes separation as a *computational problem*: if one has an optimal $\mathscr{C}$-cover of $\mathbf{L}$, one may build a separator in $\mathscr{C}$ for any two separable languages in $\mathbf{L}$.

▶ **Theorem 4.** *Let* $\mathscr{C}$ *be a boolean algebra and let* $\mathbf{L}$ *be a finite set of languages names. Given any two language name' 's* $L_1, L_2 \in \mathbf{L}$*, the following properties are equivalent:*

1. L_1 *and* L_2 *are* $\mathscr{C}$*-separable.*
2. $\{L_1, L_2\} \notin \mathcal{I}_{\mathscr{C}}[\mathbf{L}]$.
3. *For any optimal* $\mathscr{C}$*-cover* $\mathbf{K}$ *of* $\mathbf{L}$*,* L_1 *and* L_2 *are* $\mathscr{C}$*-separable by a union of languages in* $\mathbf{K}$*.*

Theorem 4 will be proved in the journal version of this paper. It entails that 'covering' is a more general problem than 'separation'. It is actually *strictly* more general as $\mathcal{I}_{\mathscr{C}}[\mathbf{L}]$ captures more information than which pairs of languages in $\mathbf{L}$ are $\mathscr{C}$-separable.

2.2 The Pointed Covering Problem for Lattices

So far, we connected the separation problem to the more general *covering problem.* Unfortunately, while the definition of the covering problem makes sense for all lattices, the connection with separation stated in Theorem 4 requires the investigated class $\mathscr{C}$ to be a boolean algebra. When $\mathscr{C}$ is not closed under complement, the optimal imprint $\mathcal{I}_{\mathscr{C}}[\mathbf{L}]$ does not capture enough information to decide whether two languages in $\mathbf{L}$ are $\mathscr{C}$-separable.

▶ **Example 5.** Let $\mathscr{C}$ be the class of languages which are unions and intersections of languages of the form A^*aA^* for some $a \in A$. Observe that $L_1 = A^*aA^* \cap A^*bA^*$ is $\mathscr{C}$-separable from $L_2 = a^*$ (L_1 belongs to $\mathscr{C}$ and $L_1 \cap L_2 = \emptyset$). However, it can be verified that the optimal imprint with respect to $\mathscr{C}$ on $\{L_1, L_2\}$ is $\mathcal{I}_{\mathscr{C}}[\{L_1, L_2\}] = \{\emptyset, \{L_1\}, \{L_2\}, \{L_1, L_2\}\}$.

We solve this issue with a new problem generalizing separation for any lattice of languages $\mathscr{C}$: the *pointed $\mathscr{C}$-covering problem*. The main idea behind this new problem is to replace the notion of cover of a finite set of languages names $\mathbf{L}$ with a more general one: *pointed covers*. When a class of languages $\mathscr{C}$ is a lattice but not a boolean algebra (*i.e.*, $\mathscr{C}$ is *not closed under complement*), the associated separation problem is asymmetric: given $L_1, L_2 \subseteq A^*$, the two following problems are non-equivalent:

- finding $K_1 \in \mathscr{C}$ such that $L_1 \subseteq K_1$ and $K_1 \cap L_2 = \emptyset$.
- finding $K_2 \in \mathscr{C}$ such that $L_2 \subseteq K_2$ and $K_2 \cap L_1 = \emptyset$.

From the point of view of $\mathscr{C}$-covers, this means that we have to define a notion of "$\mathscr{C}$-cover of $\{L_1, L_2\}$" making a distinction between the languages used to cover L_1 and those used to cover L_2. This is what pointed $\mathscr{C}$-covers are designed for.

Pointed $\mathscr{C}$-covers. Let $\mathbf{L}$ be a finite set of names. An $\mathbf{L}$-*pointed set of languages* is a *finite* set $\mathbb{P} \subseteq \mathbf{L} \times 2^{A^*}$ (i.e., elements of $\mathbb{P}$ are pairs (L, K) where L is a name in $\mathbf{L}$ and K is an arbitrary language). Furthermore, we call *support* of $\mathbb{P}$ the set $\mathbf{K} = \{K \mid (L, K) \in \mathbb{P} \text{ for some } L \in \mathbf{L}\}$. In other words the support of $\mathbb{P}$ is the smallest set of languages such that $\mathbb{P} \subseteq \mathbf{L} \times \mathbf{K}$. Finally, when we have an $\mathbf{L}$-pointed set of languages $\mathbb{P}$ with support $\mathbf{K}$ in hand, for all $L \in \mathbf{L}$, we will denote by $\mathbb{P}(L) \subseteq \mathbf{K}$ the set of all $K \in \mathbf{K}$ such that $(L, K) \in \mathbb{P}$.

We may now define *pointed $\mathscr{C}$-covers*. Let $\mathscr{C}$ be a lattice. Given a finite set of languages names $\mathbf{L}$, a *pointed $\mathscr{C}$-cover* of $\mathbf{L}$ is an $\mathbf{L}$-pointed set of languages $\mathbb{P}$ such that all K in the support of $\mathbb{P}$ belong to $\mathscr{C}$ and for all $L \in \mathbf{L}$,

$$L \subseteq \bigcup_{K \in \mathbb{P}(L)} K \quad (\textit{i.e.}, \mathbb{P}(L) \text{ is a cover of } \{L\})$$

Note that since $\mathscr{C}$ is a lattice, we have $A^* \in \mathscr{C}$. Hence, for all finite sets $\mathbf{L}$, there always exists a pointed $\mathscr{C}$-cover of $\mathbf{L}$: the set $\{(L, A^*) \mid L \in \mathbf{L}\}$.

▶ Remark. Pointed $\mathscr{C}$-covers are more general than $\mathscr{C}$-covers: if $\mathbb{P}$ is a pointed $\mathscr{C}$-cover of $\mathbf{L}$, then the support $\mathbf{K}$ of $\mathbb{P}$ is a $\mathscr{C}$-cover of $\mathbf{L}$. Intuitively, pointed $\mathscr{C}$-covers capture more information: they record for each $L \in \mathbf{L}$ which languages in $\mathbf{K}$ are needed to cover L. We use this additional information to define a finer notion of optimality.

Pointed Imprints. We now generalize imprints to pointed covers with the notion of pointed imprint (also based on the notion of filtering which is unchanged). To define pointed imprints, we first have to generalize the notion of downset to our new setting. If $\mathbf{L}$ is a finite set of language names and $E \subseteq \mathbf{L} \times 2^{\mathbf{L}}$, we denote by $\downarrow E$ the set,

$$\downarrow E = \{(L, \mathbf{H}) \mid \text{there exists } (L, \mathbf{H}') \in E \text{ such that } \mathbf{H} \subseteq \mathbf{H}'\}$$

We may now define pointed imprints. Let $\mathbf{L}$ be a finite set of language names and let $\mathbb{P}$ be an $\mathbf{L}$-pointed set of languages. The *pointed imprint of* $\mathbb{P}$ *on* $\mathbf{L}$ is the set,

$$\mathcal{P}[\mathbf{L}](\mathbb{P}) \quad = \quad \downarrow\{(L, \langle \mathbf{L} | K \rangle) \mid (L, K) \in \mathbb{P}\} \subseteq \mathbf{L} \times 2^{\mathbf{L}}$$

This new notion of pointed imprint has similar properties to those of the original notion of imprint. For a fixed $\mathbf{L}$, any pointed imprint on $\mathbf{L}$ is a subset of $\mathbf{L} \times 2^{\mathbf{L}}$, so there are finitely many pointed imprints on $\mathbf{L}$. Furthermore, pointed imprints are closed under downset.

Moreover, as for imprints, pointed imprints contain some trivial elements. If $\mathbf{L}$ is a finite set of languages, we let

$$\mathcal{P}_{triv}[\mathbf{L}] = \downarrow\{(L, \langle \mathbf{L} | \{w\} \rangle) \mid L \in \mathbf{L} \text{ and } w \in L\} = \{(L, \mathbf{H}) \mid (\cap_{H \in \mathbf{H}} H) \cap L \neq \emptyset\}$$

▶ **Fact 6.** *For any pointed $\mathscr{C}$-cover $\mathbb{P}$ of $\mathbf{L}$, we have $\mathcal{P}_{triv}[\mathbf{L}] \subseteq \mathcal{P}[\mathbf{L}](\mathbb{P})$.*

Optimal Pointed $\mathscr{C}$-Covers. We can now define optimal pointed $\mathscr{C}$-covers. The definition is similar to that of optimal $\mathscr{C}$-covers. We say that a pointed $\mathscr{C}$-cover $\mathbb{P}$ of $\mathbf{L}$ is *optimal* when,

$$\mathcal{P}[\mathbf{L}](\mathbb{P}) \subseteq \mathcal{P}[\mathbf{L}](\mathbb{P}') \quad \text{for any pointed } \mathscr{C}\text{-cover } \mathbb{P}' \text{ of } \mathbf{L}\,.$$

▶ **Lemma 7.** *For any finite set of languages names $\mathbf{L}$, there exists an optimal pointed $\mathscr{C}$-cover of $\mathbf{L}$.*

As Lemma 2, Lemma 7 is based on closure under intersection. We now generalize the notion of optimal imprint. By definition, all optimal pointed $\mathscr{C}$-covers of $\mathbf{L}$ share the same pointed imprint on $\mathbf{L}$. Hence, this unique pointed imprint is a *canonical object* for $\mathscr{C}$ and $\mathbf{L}$. We call it the *optimal pointed imprint with respect to $\mathscr{C}$ on* $\mathbf{L}$ denoted by $\mathcal{P}_{\mathscr{C}}[\mathbf{L}]$:

$$\mathcal{P}_{\mathscr{C}}[\mathbf{L}] = \mathcal{P}[\mathbf{L}](\mathbf{K}) \quad \text{for any optimal pointed } \mathscr{C}\text{-cover } \mathbf{K} \text{ of } \mathbf{L}\,.$$

We are now ready to state the pointed covering problem. As before, it is parametrized by a class $\mathscr{D}$ constraining the input, and a lattice $\mathscr{C}$.

▶ **Definition 8.** The Pointed covering problem for $\mathscr{C}$ inside $\mathscr{D}$ is as follows:

INPUT:	A finite set of languages $\mathbf{L} \subseteq \mathscr{D}$.
QUESTION:	Compute $\mathcal{P}_{\mathscr{C}}[\mathbf{L}]$.

Similarly to the covering problem, there are two stages when solving the pointed covering problem for a given lattice $\mathscr{C}$. The first one is to find an algorithm that computes $\mathcal{P}_{\mathscr{C}}[\mathbf{L}]$ from $\mathbf{L}$ and the second one is to find a generic method for constructing optimal pointed $\mathscr{C}$-covers. We now make the connection with the $\mathscr{C}$-separation problem in the following theorem.

▶ **Theorem 9.** *Let $\mathscr{C}$ be a lattice and let $\mathbf{L}$ be a finite set of languages. Given any two languages $L_1, L_2 \in \mathbf{L}$, the following properties are equivalent:*
1. *L_1 is $\mathscr{C}$-separable from L_2.*
2. *$(L_1, \{L_2\}) \notin \mathcal{P}_{\mathscr{C}}[\mathbf{L}]$.*
3. *For any optimal pointed $\mathscr{C}$-cover $\mathbb{P}$ of $\mathbf{L}$, the language $\bigcup_{K \in \mathbb{P}(L_1)} K$ separates L_1 from L_2.*

Let us make two remarks. The first one is that for any lattice $\mathscr{C}$, pointed covering is more general than covering. The second is that while this relation can be strict (see Example 5), this only happens when the class $\mathscr{C}$ is not closed under complement: if $\mathscr{C}$ is a boolean algebra, then the two problems are equivalent. In other words, when $\mathscr{C}$ is a boolean algebra, there is no point in considering pointed covering: the covering problem (which relies on simpler terminology) suffices. We refer to the journal version of this paper for details.

Now that we have defined both covering problems, the remaining sections are devoted to presenting their benefits. In particular, we present a general methodology for regular inputs in Sections 3 and 4 and use it in Section 5 on specific examples. Note that in contrast to this section which was generic to all types of structures and inputs, the remainder of the paper is specific to words and regular languages: we will rely on the fact that our inputs are sets of regular languages of finite words in our methodology.

3 Tame Sets of Languages

We now present a special class of input sets for the covering problem that we call the class of *tame* sets of languages names. A tame set contains only regular languages and has a specific algebraic structure (which is connected to language concatenation). While not all finite sets of regular languages are tame, we will be able to restrict our algorithms to such inputs without loss of generality. This restriction is central: we rely heavily on the properties of tame inputs in all our algorithms. The typical example of a tame set is the following.

▶ **Example 10.** Given a nondeterministic finite automaton (NFA) $\mathscr{A} = (A, Q, I, F, \delta)$, the set $\{L_{q,r} \mid (q,r) \in Q^2\}$ is tame (where $L_{q,r}$ is a name for the language $\{w \mid q \xrightarrow{w} r\}$).

3.1 Definition

A finite set of languages names is said to be *tame* if it can be given a *partial semigroup structure*. Let us first define *partial semigroups*. A partial semigroup is a set S equipped with a partial multiplication (*i.e.*, st may not be defined for all $s, t \in S$) such that for all $r, s, t \in S$, if rs and st are both defined, then $(rs)t$ and $r(st)$ are defined and equal.

We may now define tame sets. Let $\mathbf{L}$ be a finite set of languages names. A *tame multiplication* for $\mathbf{L}$ is a partial semigroup multiplication "$\odot$" (we use this notation to avoid confusion with language concatenation) that satisfies 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 may be decomposed as $w = uu'$, then there exist $L, L' \in \mathbf{L}$ such that $u \in L$, $u' \in L'$ and $H = L \odot L'$.

We say that a finite set of languages names $\mathbf{L}$ is *tame* if it can be equipped with a *tame multiplication*. Note that when working with tame sets, we will implicitly assume that we have a tame multiplication "$\odot$" for this set. Furthermore, since $\mathbf{L}$ is a finite partial semigroup, it is known that there exists an integer $\omega(\mathbf{L})$ (denoted by ω when $\mathbf{L}$ is understood) such that if $L \odot L$ is defined, then L^ω is defined and idempotent (*i.e.*, $L^\omega \odot L^\omega = L^\omega$).

An important observation is that tame sets of languages names may only contain *regular languages*, as stated in the following lemma (proved in the journal version).

▶ **Lemma 11.** *Any language in a tame set of languages is regular.*

Unfortunately, the converse of Lemma 11 is not true: there are finite sets of regular languages that are not tame. For example, the set $\mathbf{L} = \{\{ab\}\}$ fails Condition 2. However, this issue is easily solved with the following proposition.

▶ **Proposition 12.** *Let* $\mathbf{H} = \{H_1, \dots, H_n\}$ *be a finite set of languages given by* n NFAs $\mathscr{A}_1, \dots, \mathscr{A}_n$. *There exists a tame set of languages names* $\mathbf{L}$ *such that for any lattice* $\mathscr{C}$,

- $\mathcal{I}_\mathscr{C}[\mathbf{H}]$ *(resp.* $\mathcal{P}_\mathscr{C}[\mathbf{H}]$*) can be computed from* $\mathcal{I}_\mathscr{C}[\mathbf{L}]$ *(resp.* $\mathcal{P}_\mathscr{C}[\mathbf{L}]$*).*
- *any optimal (pointed)* $\mathscr{C}$*-cover of* $\mathbf{L}$ *is an optimal (pointed)* $\mathscr{C}$*-cover of* $\mathbf{H}$.
- $\mathbf{L}$ *and its tame multiplication can be computed from* $\mathscr{A}_1, \dots, \mathscr{A}_n$ *in polynomial time and has size* $|\mathscr{A}_1|^2 + \dots + |\mathscr{A}_n|^2$ *(where* $|\mathscr{A}_i|$ *stands for the number of states of* $\mathscr{A}_i$*).*

Proposition 12 is proved in the journal version (the construction is based on Example 10). From now on, we will assume that our inputs are tame. We finish the section by explaining the benefits of considering tame inputs in the covering and pointed covering problems.

3.2 Tame Sets of Languages and the Covering Problems

As explained, we will restrict our inputs to tame sets. We now have to explain the benefits of such a restriction. In order to get these benefits, we need the investigated class $\mathscr{C}$ to satisfy a new property in addition to being a boolean algebra or a lattice. The *left quotient* of a language L by a word w is the language $w^{-1}L = \{u \in A^* \mid wu \in L\}$. The *right quotient* Lw^{-1} is defined symmetrically. A class of languages is a *quotienting boolean algebra* if it is a boolean algebra of regular languages closed under left and right quotient. A *quotienting lattice* is a lattice of regular languages closed under left and right quotients.

When $\mathbf{L}$ is tame, the partial semigroup multiplication $\odot$ over $\mathbf{L}$ can be extended as a semigroup multiplication over $2^{\mathbf{L}}$: $\mathbf{S} \odot \mathbf{R} = \{S \odot R \mid S \in \mathbf{S},\ R \in \mathbf{R} \text{ and } S \odot R \text{ is defined}\}$. Hence, $2^{\mathbf{L}}$ is a semigroup and $\mathbf{L} \times 2^{\mathbf{L}}$ a partial semigroup. It turns out that when $\mathscr{C}$ is a quotienting lattice these structures are transferred to $\mathcal{I}_{\mathscr{C}}[\mathbf{L}] \subseteq 2^{\mathbf{L}}$ and $\mathcal{P}_{\mathscr{C}}[\mathbf{L}] \subseteq \mathbf{L} \times 2^{\mathbf{L}}$.

▶ **Lemma 13.** *Let $\mathscr{C}$ be a quotienting lattice and let $\mathbf{L}$ be a tame set of languages. Then the two following properties holds:*
1. *$\mathcal{P}_{\mathscr{C}}[\mathbf{L}]$ is closed under multiplication: for all $(L_1, \mathbf{L}_1)$, $(L_2, \mathbf{L}_2)$ in $\mathcal{P}_{\mathscr{C}}[\mathbf{L}]$, if $L_1 \odot L_2$ is defined, then $(L_1 \odot L_2, \mathbf{L}_1 \odot \mathbf{L}_2) \in \mathcal{P}_{\mathscr{C}}[\mathbf{L}]$.*
2. *$\mathcal{I}_{\mathscr{C}}[\mathbf{L}]$ is closed under multiplication: for all $\mathbf{L}_1$ and $\mathbf{L}_2$ in $\mathcal{I}_{\mathscr{C}}[\mathbf{L}]$, $\mathbf{L}_1 \odot \mathbf{L}_2 \in \mathcal{I}_{\mathscr{C}}[\mathbf{L}]$.*

Lemma 13 will be proved in the full version. Let us explain why it is crucial. We do it in the setting of the covering problem, which is simpler. We start with the following statement.

▶ **Lemma 14.** *Let $\mathbf{L}$ be a tame set of languages and let K_1, K_2 be two languages, then $\langle \mathbf{L} | K_1 \rangle \odot \langle \mathbf{L} | K_2 \rangle = \langle \mathbf{L} | K_1 K_2 \rangle$.*

Let $\mathscr{C}$ be a boolean algebra and $\mathbf{L}$ be a finite set of names. A natural method for building an optimal $\mathscr{C}$-cover $\mathbf{K}$ of $\mathbf{L}$ is to start from $\mathbf{K} = \mathcal{I}_{triv}[\mathbf{L}]$ and to add new languages K in $\mathscr{C}$ to $\mathbf{K}$ until $\mathbf{K}$ covers $\mathbf{L}$. By definition of imprints, for $\mathbf{K}$ to be optimal, we need all such candidate languages K to satisfy $\langle \mathbf{L} | K \rangle \in \mathcal{I}_{\mathscr{C}}[\mathbf{L}]$. It follows from Lemma 13 and Lemma 14 that when $\mathscr{C}$ is a quotienting boolean algebra and $\mathbf{L}$ is tame, these K may be built with concatenation: if we already have K_1 and K_2 such that $\langle \mathbf{L} | K_1 \rangle, \langle \mathbf{L} | K_2 \rangle \in \mathcal{I}_{\mathscr{C}}[\mathbf{L}]$, then we may add $K_1 K_2$ as well since by Lemmas 13 and 14, $\langle \mathbf{L} | K_1 K_2 \rangle = \langle \mathbf{L} | K_1 \rangle \odot \langle \mathbf{L} | K_2 \rangle \in \mathcal{I}_{\mathscr{C}}[\mathbf{L}]$.

This is central for classes of languages defined through logic (such as first-order logic). Indeed, concatenation is a fundamental process for building new languages in such classes.

4 General Approach

In this section, we present a natural methodology for attempting to solve the covering or pointed covering problem for a particular input class $\mathscr{C}$. This is the methodology that we use for all examples of Section 5.

Let $\mathscr{C}$ be a quotienting boolean algebra or a quotienting lattice. Recall that since we restrict ourselves to tame sets, the two objectives of the covering (resp. pointed covering) problem are as follows. Given as input a tame set $\mathbf{L}$,
1. we want an algorithm that computes $\mathcal{I}_{\mathscr{C}}[\mathbf{L}]$ (resp. $\mathcal{P}_{\mathscr{C}}[\mathbf{L}]$).
2. we want an algorithm that computes optimal $\mathscr{C}$-covers (resp. pointed $\mathscr{C}$-covers).

We now detail our methodology for the pointed covering problem (the case of the weaker covering problem is similar, see Section 5). This methodology consists in three steps.

Step 1: Presentation of the Pointed Covering Algorithm. The first step presents a solution to stage one: an algorithm that takes as input a tame set $\mathbf{L}$ and computes $\mathcal{P}_{\mathscr{C}}[\mathbf{L}]$. This step only presents the algorithm: the second and third steps are devoted to its proof.

A key point is that pointed covering algorithms are designed as *lowest fixpoint algorithms*. Since $\mathcal{P}_{\mathscr{C}}[\mathbf{L}]$ is a pointed imprint on $\mathbf{L}$, we have $\mathcal{P}_{triv}[\mathbf{L}] \subseteq \mathcal{P}_{\mathscr{C}}[\mathbf{L}]$ (Fact 6). All our algorithms start from $\mathcal{P}_{triv}[\mathbf{L}]$, and then add new elements using finitely many operations until a fixpoint is reached. Among these operations, some are specific to the particular quotienting lattice $\mathscr{C}$ that we consider, and some are *generic* to all quotienting lattices. In particular, the set of operations that we use will always include downset and multiplication (see Lemma 13). To sum up, our algorithms compute $\mathcal{P}_{\mathscr{C}}[\mathbf{L}]$ as a the smallest set $Sat_{\mathscr{C}}(\mathbf{L}) \subseteq \mathbf{L} \times 2^{\mathbf{L}}$ (*Sat* means 'saturation'), containing $\mathcal{P}_{triv}[\mathbf{L}]$ and closed under the following operations:

1. Downset: $Sat_{\mathscr{C}}(\mathbf{L}) = \downarrow Sat_{\mathscr{C}}(\mathbf{L})$.
2. Multiplication: if $(L, \mathbf{H}), (L', \mathbf{H}') \in Sat_{\mathscr{C}}(\mathbf{L})$, then $(L \odot L', \mathbf{H} \odot \mathbf{H}') \in Sat_{\mathscr{C}}(\mathbf{L})$ (if defined).
3. $\cdots$ (additional operation(s) specific to $\mathscr{C}$).

Step 2: Soundness. The second step is devoted to proving that the covering algorithm of Step 1 is sound, *i.e.*, that $Sat_{\mathscr{C}}(\mathbf{L}) \subseteq \mathcal{P}_{\mathscr{C}}[\mathbf{L}]$: for any pointed $\mathscr{C}$-cover $\mathbb{P}$ of $\mathbf{L}$, $Sat_{\mathscr{C}}(\mathbf{L}) \subseteq \mathcal{P}[\mathbf{L}](\mathbb{P})$. This is the "easy" direction and it involves Ehrenfeucht-Fraïssé arguments.

Step 3: Completeness. The third step is devoted to proving that the covering algorithm of Step 1 is complete, *i.e.*, that $\mathcal{P}_{\mathscr{C}}[\mathbf{L}] \subseteq Sat_{\mathscr{C}}(\mathbf{L})$. While usually difficult, this proof is of particular interest as it yields a solution to second stage of the pointed covering problem as a byproduct: an algorithm that computes optimal pointed $\mathscr{C}$-covers.

The proof of this step should be presented as a generic construction for building an actual pointed $\mathscr{C}$-cover $\mathbb{P}$ of $\mathbf{L}$ whose imprint on $\mathbf{L}$ is included in $Sat_{\mathscr{C}}(\mathbf{L})$. This proves that $\mathcal{P}_{\mathscr{C}}[\mathbf{L}] \subseteq \mathcal{P}[\mathbf{L}](\mathbb{P}) \subseteq Sat_{\mathscr{C}}(\mathbf{L})$, and therefore completeness. However, by combining this with the knowledge that the algorithm is also sound (this is proved in Step 2), we obtain that $\mathcal{P}_{\mathscr{C}}[\mathbf{L}] = \mathcal{P}[\mathbf{L}](\mathbb{P})$. In other words the proof builds an *optimal pointed $\mathscr{C}$-cover* $\mathbb{P}$ of $\mathbf{L}$.

5 Examples of Covering Algorithms

We now present examples of covering algorithms for several classical logical fragments, all based on first-order logic on words. Let us first briefly recall the definition of first-order logic over finite words. A word is viewed as logical structure made of a sequence of positions labeled over A. In first-order logic over words (FO), for each $a \in A$, one is allowed to use a unary predicate "$a(x)$" which selects positions x labeled with an a, as well as a binary predicate "$<$" for the linear order. A language L is said to be *first-order definable* if there is an FO sentence φ such that $L = \{w \mid w \models \varphi\}$. Also denote by FO the class of all first-order definable languages. We present algorithms for FO itself and its fragments $\mathcal{B}\Sigma_1$, FO^2, Σ_2.

Note that we only present Step 1 of our methodology in the main text, *i.e.*, algorithms without their proofs. An important remark is that these proofs are all difficult: while we have a generic template, proving a covering algorithm always requires arguments specific to the investigated class. We present proofs for $\mathcal{B}\Sigma_1$, FO^2 and Σ_2 in the full version of this paper. The proof for FO is omitted as it is close to proof of [28] (which is based on a prototype of the present framework). On the other hand, the algorithms and proofs for $\mathcal{B}\Sigma_1$, FO^2 and Σ_2 are new.

First-Order Logic: FO. The first algorithm that we present is for FO itself, which is among the most famous classes of regular languages in the literature. The decidability of the membership problem for FO was proved by Schützenberger, McNaughton and Papert [30, 14] and the result is among those that started this line of research. Separation was later proved to be decidable as well [11, 12, 28]. As explained the covering algorithm is a generalization of that of [28] (which is based on a prototype of this framework). As FO is known to be a quotienting boolean algebra, we use the covering problem.

▶ **Theorem 15.** *Let $\mathbf{L}$ be a tame set of languages. Then $\mathcal{I}_{\mathrm{FO}}[\mathbf{L}]$ is the smallest subset of $2^{\mathbf{L}}$ containing $\mathcal{I}_{triv}[\mathbf{L}]$, closed under downset, multiplication and such that for all $\mathbf{S} \in \mathcal{I}_{\mathrm{FO}}[\mathbf{L}]$, we have $\mathbf{S}^{\omega} \cup \mathbf{S}^{\omega+1} \in \mathcal{I}_{\mathrm{FO}}[\mathbf{L}]$.*

Boolean Combinations of Σ_1: $\mathcal{B}\Sigma_1$. The next class that we use as an example is $\mathcal{B}\Sigma_1$, which is the restriction of FO to sentences that are boolean combinations of Σ_1 sentences. A sentence is Σ_1 if its prenex normal form uses only existential quantifiers. The class $\mathcal{B}\Sigma_1$ is famous in the literature. the decidability of $\mathcal{B}\Sigma_1$-membership was proved by Simon [31]. $\mathcal{B}\Sigma_1$-separation is also known to be decidable [8, 23]. As $\mathcal{B}\Sigma_1$ is known to be a quotienting boolean algebra, we use the covering problem. Given a word $w \in A^*$, we denote by $\mathbf{alph}(w)$ the set of letters occurring in w, i.e. the smallest subset of B of A such that $w \in B^*$.

▶ **Theorem 16.** *Let $\mathbf{L}$ be a tame set of languages. $\mathcal{I}_{\mathcal{B}\Sigma_1}[\mathbf{L}]$ is the smallest subset of $2^{\mathbf{L}}$ containing $\mathcal{I}_{triv}[\mathbf{L}]$, closed under downset, multiplication and such that for all $B \subseteq A$, if $\mathbf{H} = \{L \in \mathbf{L} \mid \exists w \in L,\ s.t.\ \mathbf{alph}(w) = B\}$, then $\mathbf{H}^{\omega} \in \mathcal{I}_{\mathcal{B}\Sigma_1}[\mathbf{L}]$.*

Two-variable First-Order Logic: FO^2. The logic FO^2 is the restriction of FO to sentences that use at most two distinct variables (which may be reused). That the associated membership problem is decidable is due to Thérien and Wilke [34]. The separation problem was proved to be decidable in [23]. As FO^2 is known to be a quotienting boolean algebra, we use the covering problem. Our algorithm requires the input to satisfy a new condition in addition to being tame: *alphabet compatibility* (this may be assumed without loss of generality, as will be shown in the full version). A set $\mathbf{L}$ is said to be *alphabet compatible* if for all languages $L \in \mathbf{L}$, there exists a unique $B \subseteq A$ such that for any $w \in L$, $\mathbf{alph}(w) = B$. Note that when $\mathbf{L}$ is alphabet compatible, then $\mathbf{alph}(L)$ is well-defined for all $L \in \mathbf{L}$ as this unique alphabet.

▶ **Theorem 17.** *Let $\mathbf{L}$ be a tame and alphabet compatible set of languages. $\mathcal{I}_{\mathrm{FO}^2}[\mathbf{L}]$ is the smallest subset of $2^{\mathbf{L}}$ containing $\mathcal{I}_{triv}[\mathbf{L}]$, closed under downset, multiplication and such that for all $B \subseteq A$ and $\mathbf{S}, \mathbf{T} \in \mathcal{I}_{\mathrm{FO}^2}[\mathbf{L}]$ containing S, T with $\mathbf{alph}(S) = \mathbf{alph}(T) = B$,*

$$\mathbf{S}^{\omega} \odot \langle \mathbf{L} | B^* \rangle \odot \mathbf{T}^{\omega} \in \mathcal{I}_{\mathrm{FO}^2}[\mathbf{L}].$$

One Quantifier Alternation: Σ_2. Our third example is Σ_2, which is the restriction of FO to sentences whose prenex normal form have a quantifier prefix of the form '$\exists^*\forall^*$'. It was proved that Σ_2-membership is decidable in [3, 21] and the same was proved for separation in [25]. As Σ_2 is a quotienting lattice but not a boolean algebra, we use the pointed covering problem. Our algorithm requires the input to be tame and alphabet compatible.

▶ **Theorem 18.** *Let $\mathbf{L}$ be a tame and alphabet compatible set of languages $\mathcal{P}_{\Sigma_2}[\mathbf{L}]$ is the smallest subset of $\mathbf{L} \times 2^{\mathbf{L}}$ containing $\mathcal{P}_{triv}[\mathbf{L}]$, closed under downset, multiplication and such that for any $B \subseteq A$, and $(S, \mathbf{S}) \in \mathcal{P}_{\Sigma_2}[\mathbf{L}]$ satisfying $\mathbf{alph}(S) = B$ and $S \odot S$ is defined,*

$$(S^{\omega}, \mathbf{S}^{\omega} \odot \langle \mathbf{L} | B^* \rangle \odot \mathbf{S}^{\omega}) \in \mathcal{P}_{\Sigma_2}[\mathbf{L}].$$

6 Conclusion

We introduced the covering and pointed covering problems which are designed to investigate quotienting boolean algebras and quotienting lattices respectively. We also presented a methodology outlining how these problems should be approached. Furthermore, we presented four examples of algorithms for the instances associated to FO, $\mathcal{B}\Sigma_1$, FO^2 and Σ_2.

It is worth noting that while our examples include the most significant logics for which separation is known to be decidable, an important one is missing: Σ_3. This is not surprising as the algorithm of [22] considers an *ad hoc* problem which is associated to two logics at the same time: Σ_2 and Σ_3. However, it is possible to generalize this result as well within our framework: this is where the modularity of our problems comes into play. Using a stronger notion of filtering, one can reformulate and generalize the problem of [22] as an instance of the pointed covering problem (we leave the presentation of this instance for further work).

Our results raise several questions. The most natural is to apply our framework to classes for which no membership or separation algorithm is known yet. Another one is related to the classical membership algorithms. These algorithms are usually stated as equations on the syntactic monoid of the language which share similarities with fixpoint operations of our (pointed) covering algorithms. An interesting question would be to find a criterion under which membership equations can be lifted as a fixpoint operation for the covering problem.

References

1 Jorge Almeida. Some algorithmic problems for pseudovarieties. *Publicationes Mathematicae Debrecen*, 54:531–552, 1999.

2 Jorge Almeida, José C. Costa, and Marc Zeitoun. Closures of regular languages for profinite topologies. *Semigroup Forum*, 89(1):20–40, 2014.

3 Mustapha Arfi. Polynomial operations on rational languages. In *Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science (STACS'87)*, pages 198–206, 1987.

4 Janusz A. Brzozowski and Imre Simon. Characterizations of locally testable events. *Discrete Mathematics*, 4(3):243–271, 1973.

5 Julius R. Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960.

6 Julius R. Büchi. Symposium on decision problems: On a decision method in restricted second order arithmetic. In *Logic, Methodology and Philosophy of ScienceProceeding of the 1960 International Congress*, volume 44, pages 1–11. Elsevier, 1966.

7 Thomas Colcombet. Green's relations and their use in automata theory. In *Proceedings of Language and Automata Theory and Applications, 5th International Conference (LATA'11)*, pages 1–21, 2011.

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

9 Volker Diekert and Paul Gastin. First-order definable languages. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives*, volume 2 of *Texts in Logic and Games*, pages 261–306. Amsterdam University Press, 2008.

10 Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the AMS*, 98(1):21–51, 1961.

11 Karsten Henckell. Pointlike sets: the finest aperiodic cover of a finite semigroup. *J. Pure Appl. Algebra*, 55:85–126, 1988.

12 Karsten Henckell, John Rhodes, and Benjamin Steinberg. Aperiodic pointlikes and beyond. *Internat. J. Algebra Comput.*, 20(2):287–305, 2010.

13 Robert McNaughton. Algebraic decision procedures for local testability. *Mathematical Systems Theory*, 8(1):60–76, 1974.

14 Robert McNaughton and Seymour A. Papert. *Counter-Free Automata*. The MIT Press, 1971.

15 Jean-Éric Pin. Finite semigroups and recognizable languages: An introduction. In *Semigroups, Formal Languages and Groups*, pages 1–32. Springer-Verlag, 1995.

16 Jean-Éric Pin. Syntactic semigroups. In *Handbook of Formal Languages*, pages 679–746. Springer-Verlag, 1997.

17 Jean-Éric Pin. Bridges for concatenation hierarchies. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming, ICALP'98*, Lecture Notes in Computer Science, pages 431–442, Berlin, Heidelberg, 1998. Springer-Verlag.

18 Jean-Éric Pin. Theme and variations on the concatenation product. In *Proceedings of the 4th International Conference on Algebraic Informatics, CAI'11*, Lecture Notes in Computer Science, pages 44–64, Berlin, Heidelberg, 2011. Springer-Verlag.

19 Jean-Éric Pin. The dot-depth hierarchy, 45 years later. In *WSPC Proceedings*, 2016. To appear.

20 Jean-Éric Pin. Mathematical foundations of automata theory. In preparation, 2016. URL: `https://www.irif.univ-paris-diderot.fr/~jep/MPRI/MPRI.html`.

21 Jean-Éric Pin and Pascal Weil. Polynomial closure and unambiguous product. *Theory of Computing Systems*, 30(4):383–422, 1997.

22 Thomas Place. Separating regular languages with two quantifiers alternations. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'15)*, pages 202–213, 2015.

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

24 Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. Separating regular languages by locally testable and locally threshold testable languages. *Logical Methods in Computer Science*, 10(3), 2014.

25 Thomas Place and Marc Zeitoun. Going higher in the first-order quantifier alternation hierarchy on words. In *Automata, Languages, and Programming - 41st International Colloquium (ICALP'14)*, pages 342–353, 2014.

26 Thomas Place and Marc Zeitoun. Separation and the successor relation. In *32nd International Symposium on Theoretical Aspects of Computer Science (STACS'15)*, pages 662–675, 2015.

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

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

29 Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Bull. Amer. Math. Soc.*, 74(5):1025–1029, 09 1968.

30 Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.

31 Imre Simon. Piecewise testable events. In *Proc. of the 2nd GI Conf. on Automata Theory and Formal Languages*, pages 214–222, 1975.

32 James W. Thatcher and Jesse B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, 1968.

33 Denis Thérien and Alex Weiss. Graph congruences and wreath products. *J. Pure Appl. Algebra*, 36:205–215, 1985.

34 Denis Thérien and Thomas Wilke. Over words, two variables are as powerful as one quantifier alternation. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC'98)*, pages 234–240, 1998.

35 Wolfgang Thomas. Languages, automata, and logic. In *Handbook of formal languages*. Springer, 1997.

36 Boris A. Trakhtenbrot. Finite automata and logic of monadic predicates. *Doklady Akademii Nauk SSSR*, 149:326–329, 1961. In Russian.

37 Pascal Weil. Concatenation product: a survey. In *Formal Properties of Finite Automata and Applications*, volume 386 of *Lecture Notes in Computer Science*, pages 120–137. Springer-Verlag, Berlin, Heidelberg, 1989.

38 Thomas Wilke. Classifying discrete temporal properties. In *Proceedings of the 16th Annual Conference on Theoretical Aspects of Computer Science, STACS'99*, Lecture Notes in Computer Science, pages 32–46, Berlin, Heidelberg, 1999. Springer-Verlag.

39 Yechezkel Zalcstein. Locally testable languages. *Journal of Computer and System Sciences*, 6(2):151–167, 1972.