INDUCTIVE INFERENCE OF THE UNIONS OF THE PATTERN LANGUAGES FROM POSITIVE DATA BY THE MOST FITTING HYPOTHESES

NG YEN KAOW

Abstract

A *pattern* is a finite string over a set of constants and a set of variables. The language of a pattern, called a *pattern language*, is the set of all the strings obtainable by substituting every variable in the pattern with constant strings. *Inductive inference* refers to the act of guessing a language inductively from examples taken from the language. The case of inductive inference where no negative examples are given is called *inductive inference from positive examples*.

Due to the lack of negative examples in inductive inference from positive examples, there is a possibility that an inference algorithm may produce intermediate hypotheses that contain exceedingly many elements. For this reason, a preference for smaller hypotheses seems unavoidable. However, placing requirements on the hypotheses allowed, depending on the requirement, may impose restrictions on the classes of languages that can be inferred. Nevertheless, the restrictions on the use of hypotheses that are minimal with respect to inclusion, called *minl*s, are less severe, and their use is known to allow the inference of many classes, including the unions of the pattern languages. In order to compute such *minl*s efficiently, others have studied approximate hypotheses that are minimal in terms of the *subsumption* relation (called *mmg*).

There are situations when numerous *minl*s are applicable to a given set of examples. To further evaluate among these *minl*s, we consider another notion of minimality related to their cardinalities. The minimal hypothesis we aim at is one which, among all the hypotheses that include the given examples, contains the least number of elements of up to an appropriately specified length. We call such a hypothesis a *fittest cover*.

We first show how the use of these fittest covers as hypotheses affects inference. We define two general methods for inference that base their conjectures entirely on a search for such fittest covers, and characterize the classes of languages that can be inferred by these methods. While the conditions are restrictive, we show that there are very rich classes of languages that are nonetheless inferable, including the unions of the pattern languages. We then study how hypotheses that are both fittest cover and *minl* at the same time can be obtained. We show that such hypotheses can be found using the concept of *characteristic sets*. Motivated by this result we investigate the characteristic sets for several subclasses of the pattern languages (and the classes of their unions), as well as for the unions of the *tree pattern languages*. We also propose an algorithm for finding such fittest covers, and show that the algorthm efficiently finds every fittest covers of the unions of the *erasing substring pattern languages*.

We then investigate the empirical value of the minimality of the fittest covers, through two experiments based on the problem of finding *consensus patterns*. Here, a consensus pattern is a pattern which characterizes a set of biological sequences. In the first experiment, we use only a portion of the positive examples to obtain an *mmg* of a few patterns, and then measure its accuracy in classifying the (remaining) positive examples and a set of generated negative examples. In the second experiment, we obtain an *mmg* for a sample with a known consensus pattern and compare the patterns within the *mmg* with the known pattern. In both cases, we found that in some situations, the better patterns tend to demonstrate the minimality of the fittest covers.

Acknowledgements

I owe the most to my advisor, Prof. Takeshi Shinohara for the realization of this thesis. I need to thank Dr. Hirotaka Ono from Kyushu University for friendship, and for his ideas in our collaborations. I owe much to Prof. Kouichi Hirata, Prof. Masateru Harao and Prof. Hiroki Ishizaka for advices that helped improve this thesis; as well as Prof. Hiroki Arimura from Hokkaido University and Prof. Akihiro Yamamoto from Kyoto University for valuable comments. I am also grateful to the anonymous referees who reviewed the papers which contained the results in this thesis. I need to thank Prof. Sanjay Jain, Prof. Frank Stephan from the National University of Singapore, as well as Prof. John Case from the University of Delaware for advices on related ideas. I am very indebted to Prof. Hon Wai Leong from the University of Singapore for his encouragement and support.

Other than the people mentioned, this thesis also owes many of its results to the works of Masako Sato, Yasuhito Mukouchi, Dana Angluin, Keith Wright, Satoshi Kobayashi, Takashi Yokomori, Uemura Jin, and many other excellent researchers, to whom I am grateful for.

Personally I wish to thank Swee Loong Ong, Seng Hooi Low, Jenn Chuan Ling, Francis Ng, Harold Tay, Shuai Cheng Li, Kenichiro Hata, Takasumi Iwamoto, Yoshikazu Nakanishi, Daishi Shiraishi, and others (beer on me if your name is not mentioned here) for excellent company. I thank the Ministry of Education, Science, Sports, Culture and Technology of Japan for a very generous scholarship that more than supported me for this study. I also thank the Meisenkai of Kyushu Institute of Technology for twice supporting my travel expenditures.

This thesis is dedicated to my parents.

Contents

1 Introduction **1**

1.1 Learning unions of languages . . . 1

1.2 Minimal hypotheses . . . 2

1.3 Strings and patterns . . . 3

1.4 Overview of this thesis . . . 4

2 Preliminaries **7**

2.1 Inductive inference . . . 7

2.2 Learning by languages minimal with respect to inclusion . . . 9

2.3 Languages generated from patterns . . . 12

2.3.1 The pattern languages . . . 13

2.3.2 Subclasses of the pattern languages . . . 14

2.3.3 The tree pattern languages . . . 15

3 Fittest Covers **17**

3.1 Fittest Cover Problem . . . 18

3.2 Learning with fittest covers . . . 18

3.3 Comparing fittest covers to *minl*s and *mmg*s . . . 20

3.4 Fittest covers that are also *minl*s . . . 23

4 Fittest Covers that are *minl*s **25**

4.1 Substring patterns and subsequence patterns . . . 25

4.2 Patterns with at most 1 variable occurrence . . . 27

4.3 Tree pattern languages . . . 30

4.3.1 Existence of characteristic sets for unions of languages in $\mathcal{L}(TP)$ . . . 31

4.3.2 Characteristic sets within unions of less than $card(\Gamma)$ languages . . . 31

4.3.3 Unions of up to a bound independent of $card(\Gamma)$ . . . 34
4.4 Conclusions and open problems . . . 40

5 Finding Fittest Covers **43**
5.1 Counting strings . . . 44
5.1.1 Finite automata . . . 44
5.1.2 Dynamic programming . . . 44
5.2 Comparing string counts . . . 47
5.3 Finding the fittest covers $\mathcal{FC}(\mathcal{L}_\epsilon(\mathbf{Psub}), \cdots)$ . . . 50
5.3.1 The case of $k = 1$ and $\lfloor S \rfloor < card(\Sigma) - 2$ . . . 50
5.3.2 The case of $k \in N^+$ . . . 51
5.4 Counting trees . . . 52
5.5 Discussions and open problems . . . 53

6 Validating the Value of the Fittest Covers **55**
6.1 Signal peptide prediction . . . 57
6.1.1 Target pattern classes and refinement operators . . . 58
6.1.2 Signal peptide samples . . . 59
6.1.3 Measures for evaluating *mmg*s . . . 59
6.1.4 Reproduced earlier results using updated samples . . . 60
6.1.5 Correlation between accuracies and coverages . . . 60
6.2 Comparison with known consensus patterns . . . 64
6.2.1 Target pattern classes and their refinement operators . . . 64
6.2.2 Samples and method for evaluation . . . 65
6.2.3 Comparison results . . . 66
6.3 Conclusions . . . 67

7 Conclusions and Future Work **71**

Chapter 1

Introduction

In this thesis we study how to inductively infer, from positive examples, unions of the pattern languages using hypotheses minimal in a notion related to their cardinalities. This chapter gives an introduction to these concepts.

1.1 Learning unions of languages

This thesis is to do with the learning of *languages*. Language learning can be considered an act where students are given sentences in a language (in other words, the elements of the language), and in the end, hypothesizes a grammar which describes the language correctly. Gold devised a mathematical model called *inductive inference from positive examples* for this [24]. In this model, a learner is given as examples, one by one, with or without repetition, ad infinitum, all the elements in a language. The learner is said to *identify the language in the limit* if and only if regardless of the order and frequency these examples are presented, at some unspecified time the learner outputs a grammar for that language and then never changes its mind. The learner is said to *identify a class of languages* just in case it identifies every language in that class. Partly due to problems of dealing with infinitely many languages of infinite cardinality in a class, such identifications are not always possible. Nevertheless, results have shown that many interesting and useful classes of languages can be identified in the limit from positive examples [3, 68, 69, 83, 73].

Gold's model had been studied under various modifications. One form of modifications is in the criteria that says when a learner has succeeded in the identification task — for example, the learner may be allowed some amount of errors in the language it outputs [54]. Similarly, one may also consider modifying the way examples are presented, for instance, by introducing slight inaccuracies in the examples presented [23, 77]. Refinements and reinterpretations of Gold's model have become so pervasive and so extensively studied, that this approach is today almost synonymous with the notion of inductive inference in the philosophy of science. As a result of these works, many interesting (and sometimes quite surprising) facts regarding the act of learning have been discovered. A compilation of these results can be found in the textbook by

Jain *et al.* [32].

We study a modification that allows unions of languages. More precisely, the learning situation we consider in this thesis is that where a learner is shown examples from not only one language, but from many different languages at the same time, as a mixture. The learner is then expected to provide a grammar for the language of the mixture. That is, we are putting the learner to the task of learning the *unions* of languages drawn from a language class, a situation which has previously been studied by others [83, 73, 25, 31, 30]. Such a requirement for learning is reasonable since data from many natural processes are often available only as mixtures. Also, from a theoretical point of view, for a hypothesis space that is not closed under union, a hypothesis speficied using the union of a few languages can sometimes fit the given data better than that of any single language in the hypothesis space.

For practical reasons, we also restrict ourselves to consider only hypothesis spaces that are *indexed families* [3], language classes where membership in all languages in the class can be uniformly decided.

1.2 Minimal hypotheses

In inductive inference from only positive examples, due to the lack of negative examples, there is a danger for a learner to conjecture intermediate hypotheses that contain outrageously many elements. This would not be a problem if one is able to tell if a final, correct hypothesis has been obtained, and hence would have less incentives to look for good intermediate hypotheses. However, it is in general difficult to tell when the correct hypothesis has been obtained in the course of inductive inference [22, 47]. To ensure intermediate hypotheses to be not exceedingly large, a general preference for hypotheses that are in some sense minimal in terms of the elements they contain, seems unavoidable.

However, in many cases, requirements on the kinds of hypotheses a learner may conjecture have been found to restrict the classes of languages that can be inductively inferred from positive examples. For example, learning using hypotheses with minimal program-size complexity [75, 76] has been found by many to be restrictive [21, 19, 32]. Similar, any learner that fulfills some specific subset relations between subsequent conjectures will be unable to inductively infer some classes of languages that can otherwise be inferred [33, 86].

On the other hand, it is known that any indexed family that has the property called *finite elasticity* [83], can be learned by simply conjecturing a language that is minimal with respect to *inclusion*, among all the languages in the class that contains the observed examples [9]. A language that fulfills this property has been called a *descriptive pattern* [2] (referring to a hypothesis that is a *pattern language*, see Section 1.3 for definition), or more generally, a *minimal language* (*minl*) [68].

Most methods to compute such *minl*s for the pattern languages rely on demonstrating inclusion using another relation between grammars of the languages [2, 68, 69]. Such a relation coincides with the idea of *subsumption* that had been studied in logical inference [55]. A grammar that is minimal with respect to this relation is known as an *lgg* (*least general generalization*). It is often easier to compute for such *lgg*s, and there are occasions when an *lgg* is also a *minl*. Arimura *et al.* [8, 10, 11] extended the concept of *lgg*s to the case of grammars definable from up to k grammars in a class, resulting in the k-*mmg*s (k-minimal multiple generalizations), and studied how a k-*mmg* can be computed efficiently. Note that an *lgg* is a special case of the k-*mmg* where k is unity. When k is implicit, we sometimes refer to a k-*mmg* as simply an *mmg*. Note that the concept of *minl*, unlike *lgg*, applies to unions immediately.

In this thesis we consider another notion of minimality related to the elements contained in a language, that is not necessarily fulfilled by every *minl*. A hypothesis that fulfills this minimality is one that, among all the hypotheses that include the given examples, contains the least number of elements of up to an appropriately specified length. (Here we assume that each language has only finitely many elements of any length. This is true for any language of strings over a finite alphabet.) We refer to such a hypothesis as a *fittest cover*. Note that the hypothesis space from which a fittest cover is drawn may be a class of unions. That is, like *minl*, the concept of fittest covers applies to unions immediately.

1.3 Strings and patterns

The languages we deal mainly with in this thesis are those that describe strings. That is, we assume that learners are given strings that are generated from some patterns, and are asked to identify these patterns from the given strings. We study learning in this context because string processing is considered very important today in various applications, from networked computers to databases, and more recently for the analysis of biosequences.

We consider strings to be generated from the kinds of patterns introduced in [2]. In this context, a *pattern* is a string over a finite alphabet Σ and an infinite set V of variables. The language generated by a pattern p (called the *pattern language* of p) are the strings obtainable by replacing variables in p with strings over Σ. As an example let $\Sigma = \{a, b\}$, $V = \{x_1, x_2, \ldots\}$ and let pattern p be ax_1x_1b. The pattern language of p would then be all the strings that begin with a, followed by a string over Σ repeated twice, and then ending in b, for instance, $ababab$. The seemingly simple languages of these patterns (and their subclasses) possess very rich properties [70, 48, 34, 35, 45, 58, 59, 61, 60], and are very widely studied in learning theory [68, 73, 64, 44, 17, 67]. Besides these, the class of the unions of these languages is also the basis for the elementary formal systems, a kind of logic programs over strings [5].

In this thesis we will consider variants and subclasses of the pattern languages that are indexed families (e.g. the full class of the pattern languages is an indexed family), and we will also introduce elements from the so-called *PROSITE patterns* [74] into these patterns. Even though our treatment of results herein is directed towards the pattern languages, many of the results are actually applicable to the more general classes of indexed families.

We will also study, to some extent, the *tree pattern languages*, which are first-order terms in formal logic (hence their unions form the basis of logic programs). The strings generated by these languages can also be considered as tree structures, and hence they can be used to represent hierarchical documents such as XML documents.

1.4 Overview of this thesis

We now give a summary of the content of this thesis.

We begin with a formal introduction of Gold's model of learning, and other necessary definitions in Chapter 2. We introduce *minl*, *mmg* and earlier results on learning using these as hypotheses. We also introduce the pattern languages, their subclasses, and the tree pattern languages.

In Chapter 3, we introduce a minimality in terms of the elements contained in a hypothesis, that is not guaranteed by every *minl*. As example we show two distinct *minl*s (of the pattern languages) for a set of input strings, where one of the *minl*s likely contains less elements of any length than the other. This minimality is the basis for the *fittest covers*, which are hypotheses that contain the least number of elements of up to some specified length. We define a general learning strategy that obtains its hypothesis through only a computation of such fittest covers, and show that such a strategy inductively infers an indexed family if and only if for every language L in the class, there exists a finite set $S \subseteq L$ such that every superset of S witnesses the minimality of L in the "fittest cover" sense for any sufficiently long length. While this may seem exceedingly restrictive, it turns out that any indexed family where each language in it has a *characteristic set within the class* [4] fulfills this requirement, and this includes all the classes which were previously known to be inferable through *minl* hypotheses.

A question then arises: Is it possible to obtain a hypothesis that is both a fittest cover and a *minl* at the same time? Such a hypothesis is interesting because it is more "minimal" than hypotheses that are only fittest cover or *minl*, and it would also possess the theoretical benefits of *minl*s (a learner that conjectures *minl*s is *conservative* and *set-driven* [3], making it possible to analyze its convergence), as well as the practical benefits of the fittest covers [18, 49, 52].

To answer this question, we first clarify the relationship between the fittest covers and the *minl*s (as well as *mmg*s). We then observe that given a class $\mathcal{L}$, for any set S, if there is a way to decide a length n for which every language $L \in \mathcal{L}$ where $S \cap L \neq \emptyset$ has a characteristic set consisting only of elements no longer than n, then a fittest cover, minimal in the number of elements of up to length n, is also a *minl*. Note that this actually gives us an alternative way to compute *minl*s, in addition to the method using *mmg*s. Using previous results by others [66, 80], we show that for the subclass called *regular pattern languages*, in the case that the number of unions allowed is less than a number given by the alphabet size, the fittest covers that are minimal in their constituents of up to some length n (determined from S), are also *minl*s.

In Chapter 4 we continue this search for sufficient lengths to the elements of characteristic sets for several subclasses of the pattern languages. We first show that for any union of finitely many languages of the *substring* or *subsequence patterns* (subclasses of the pattern languages): (1) any fittest cover that is minimal in their elements of up to the longest length of the examples is also a *minl*, and (2) an *mmg* of these pattern classes is also a *minl*, and vice versa. We also performed the same study on the class of bounded unions of the languages generated by patterns with at most one variable occurrence. Such a class, while not very useful, is nonetheless interesting because an *mmg* of the class is not necessarily a *minl*, and hence we cannot find a *minl* by finding an *mmg*. However, we can show a sufficient length n for the elements of the characteristic sets for the class, and hence any fittest cover that is minimal in their constituents of up to the length n is also a *minl*. (Note that the substring patterns, the subsequence patterns, and the one variable occurrence patterns are all incomparable.) Finally, we show a more complete result for such a sufficient size for the elements of the characteristic sets, for the bounded unions of the tree pattern languages.

In Chapter 5 we briefly discuss the computation that might be involved in finding the fittest covers. We propose a simple brute-force computation for the fittest covers, and then consider the time-complexity for this computation. The algorithm looks at every possible set of up to k pattern languages (each which contains some of the observed examples), and then verify if the union of the languages (1) includes all the examples, and (2) contains the least number of elements of up to a given length. In the case of general pattern languages, due to earlier NP-hardness results on the computation for (1), such an algorithm cannot be efficient. However, the same NP-hardness result does not apply to some subclasses of the patterns languages. In fact, we show that for the erasing substring pattern languages, given any k, all the fittest covers of up to k substring patterns can be computed in time polynomial to the input size. Here, we also suggest how the computation may be improved. More discussions on these can be found at the end of Chapter 5.

Finally, in Chapter 6, we give experimental justifications for the use of our notion of minimality. We show results from two experiments on the search for so-called *consensus patterns*, a common practice in molecular biology. A consensus pattern is a pattern that characterizes a set of biological sequences. The first experiment is concerned with the use of such consensus patterns in the prediction of gene functions. More precisely, we use only a portion of the positive examples to obtain an *mmg*, and then measure the *mmg*'s accuracy in classifying the (remaining) positive examples and a set of generated negative examples. We show that the accuracy in such predictions is dependent on the number of elements in the *mmg* of up to the longest length of the examples. Furthermore, when sufficiently many examples are used in obtaining the *mmg*s, the *mmg*s that contain less elements (of up to a specified length) almost always give better predictions. In the second experiment, we obtain an *mmg* for a sample with a well-established consensus pattern, and compare the patterns in the *mmg* with the known consensus pattern. In this case as well, the minimality that the fittest covers aim for has some effects on this resemblance.

The results in Chapter 3 and parts of Chapter 4 appeared in [50], while the remaining results in Chapter 4 (for the tree pattern languages) are published in [51]. Results in Chapter 5 are extracted from [53]. All the results in Chapter 6 are contained in the papers [49] and [52].

Chapter 2

Preliminaries

In this chapter we give formal definitions and necessary background to facilitate subsequent discussions. The symbol N denotes the set of natural numbers. The symbol N^+ denotes the set of positive natural numbers. $\emptyset$, $\in$, $\subset$, $\subseteq$ respectively denote empty set, element of, proper subset, and subset. Sets and sequences are treated similarly when the context is clear. For example, for a sequence $A = a_0, a_1, \ldots$, we write $B \subseteq A$ just in case B is a set or a sequence of elements from A. The symbols max(.), min(.) denote maximum and minimum of a set, where by convention $\max(\emptyset) = 0$ and $\min(\emptyset) = \infty$. Cardinality of a set S is written $card(S)$. For any sets A and B, $A - B$ denotes the set $\{x \mid x \in A \text{ and } x \notin B\}$. For a set A, let A^* be the free monoid over A, and let $A^+ = A^* \setminus \{\varepsilon\}$, where ε denotes the empty string. For a set S, $S^{=n}$ and $S^{\leq n}$ denote the strings of length n and strings of length up to n in S.

For the purpose of this thesis, a *language* is a subset of a generalized infinite set $\mathcal{S}$. Here, $\mathcal{S}$ should be considered general unless when used in a more specific context. For example, in the context of the *pattern languages* in Section 2.3.1, $\mathcal{S} = \Sigma^*$; and in the context of the *tree pattern languages* in Section 2.3.3, $\mathcal{S} = TP_\Gamma$. A set of languages is called a *class*.

2.1 Inductive inference

We present concepts in inductive inference [24] in this section. We first introduce concepts in the presentation of a language to a learner.

A *sequence* is an ordered presentation of elements in $\mathcal{S}$ where repetitions are allowed. For each sequence σ, $content(\sigma)$ denotes the set of elements that appeared in σ and $|\sigma|$ denotes the length of σ. SEQ denotes the set of all finite sequences. A *text* of a language L is an infinite sequence w_0, w_1, w_2, ... of elements from $\mathcal{S}$ such that $\{w_i \mid i \in N\} = L$. The initial segment of length n of a text T is denoted $T[n]$.

An *inductive inference machine* (**IIM**) is an algorithmic device which computes a mapping from SEQ into N. We say that an **IIM** M *converges* on a text T to i just in case there exists $n \in N$ such that for all $n' \geq n$, M outputs i on input $T[n']$.

We now introduce Gold's model for *inductive inference from positive examples.* For simplicity we adapt the definition to apply to only the classes of languages known as *indexed families of uniformly recursive languages.* We first give definition for these classes.

Definition 2.1 (Indexed families [3]) *A sequence of non-empty languages* $\mathcal{L} = L_0, L_1, L_2, \ldots$ *is* an indexed family of uniformly recursively languages*, or simply an* indexed family*, just in case there exists a computable function* f *such that for each* $i \in N$ *and for each* $w \in \mathcal{S}$*,* $f(i, w) = 1$ *if* $w \in L_i$ *and* $f(i, w) = 0$ *otherwise.*

The following, then, is Gold's model for identification in the limit, applied to the indexed families.

Definition 2.2 ([24]) *(a) An indexed family* $L_0, L_1, L_2, \ldots$ *is said to be* **TxtEx**-identified by an **IIM** M *just in case for each* $i \in N$*, for each text* T *for* L_i*,* M *converges on* T *to* $j^T \in N$*, and* $L_{j^T} = L_i$*.*
(b) An indexed family is said to be learnable *just in case some* ***IIM*** M ***TxtEx****-identifies it.*

We now present some known concepts and results in inductive inference that are relevant to our study. These will be used to show learnability results in this section (through the *minl* strategy, to be introduced next), and in the next chapter.

Definition 2.3 (Finite tell-tale [3]) *A* finite tell-tale *for a language* L *within a class* $\mathcal{L}$ *is a finite set* $S \subseteq L$ *such that there exists no language* $L' \in \mathcal{L}$ *where* $S \subseteq L' \subset L$*.*

Theorem 2.1 (Angluin [3]) *An indexed family* $L_0, L_1, L_2, \ldots$ *is learnable just in case there exists a procedure, effective in each* i*, that enumerates a finite tell-tale for* L_i *within the class.*

Definition 2.4 (Finite thickness [2]) *A class* $\mathcal{L}$ *of languages has* finite thickness *just in case for each* $w \in \mathcal{S}$*,* $\{L \in \mathcal{L} \mid w \in L\}$ *is finite.*

Definition 2.5 (Finite elasticity [83, 46]) *A class* $\mathcal{L}$ *of languages has* infinite elasticity *just in case there exists an infinite sequence of pairwise distinct elements* $w_0, w_1, w_2, \ldots$ *from* $\mathcal{S}$ *and an infinite sequence of pairwise distinct languages* $A_1, A_2, \ldots$ *such that for each* $k \in N^+$*,* $\{w_i \mid i < k\} \subseteq A_k$*, but* $w_k \notin A_k$*.* $\mathcal{L}$ *is said to have* finite elasticity *just in case* $\mathcal{L}$ *does not have infinite elasticity.*

Theorem 2.2 (Wright [83]) *An indexed family with finite elasticity is learnable.*

Note that any class of indexed family with finite thickness has finite elasticity.

For $k \in N^+$, $D_0^k, D_1^k, \ldots$ stand for a computable sequence of all finite non-empty sets of up to k natural numbers. $D_0, D_1, \ldots$ stand for a computable sequence of all finite non-empty sets. Details on how to generate such sequences can be found in [63].

Definition 2.6 (Unions of languages [83, 73, 31]) *For an indexed family $\mathcal{L} = (L_i)_{i \in N}$,*
(1) $\mathcal{L}^k = (\bigcup_{j \in D_i^k} L_j)_{i \in N}$. That is, $\mathcal{L}^k$ is an indexed family of the unions of up to k languages from $\mathcal{L}$,
(2) $\mathcal{L}^* = (\bigcup_{j \in D_i} L_j)_{i \in N}$. That is, $\mathcal{L}^*$ is an indexed family of the unions of finite but unboundedly many languages from $\mathcal{L}$.

$\mathcal{L}^k$ and $\mathcal{L}^*$ for finite $\mathcal{L}$ are defined similarly using sequences D_0^k, D_1^k, ... and D_0, D_1, ... which contain only indices of languages in $\mathcal{L}$.

Theorem 2.3 (Wright [83]) *If an indexed family $\mathcal{L}$ has finite elasticity, then for any $k \in N^+$, the indexed family $\mathcal{L}^k$ has finite elasticity.*

The following concept of characteristic sets is central to this thesis.

Definition 2.7 (Characteristic set [4]) *A* characteristic set for a language L within a class $\mathcal{L}$ *is a finite set S_L such that for each $L' \in \mathcal{L}$, $S_L \subseteq L' \Rightarrow L \subseteq L'$.*

Theorem 2.4 (Sato [65], Kobayashi *et al.* **[39])** *A class $\mathcal{L}$ of languages has finite elasticity if and only if every language $L \subseteq \mathcal{S}$ has a characteristic set within $\mathcal{L}$.*

Note that for a class with finite elasticity, *every* language (that is, including languages outside of $\mathcal{L}$) has a characteristic set within $\mathcal{L}$.

2.2 Learning by languages minimal with respect to inclusion

We now introduce the concept of a language that is minimal with respect to inclusion. The idea originated from consideration for the following general purpose learner M, which on any given class $\mathcal{L} = L_0, L_1, L_2, \ldots$ and each input sequence σ, conjectures hypothesis $H_{|\sigma|}$ (we let $H_0 = \emptyset$):

$M(\sigma)$:
If $content(\sigma) \subseteq H_{|\sigma|-1}$, let $H_{|\sigma|} = H_{|\sigma|-1}$.
Otherwise find i where
  (C1) $content(\sigma) \subseteq L_i$, and
  (C2) no other $L_j \in \mathcal{L}$ has $content(\sigma) \subseteq L_j \subset L_i$.
  If such i is found, let $H_{|\sigma|} = L_i$.
  (Note that we may additonally bound the search, to consider only values of i and j that are less than $|\sigma|$ or some other bounds, to remove concerns for computability in unbounded searches.)
Conjecture $H_{|\sigma|}$ (by its index i).

Intuitive, M outputs a language which is minimal with respect to inclusion among all the languages in $\mathcal{L}$. Note that for inductive inference, it suffices that a correct hypothesis can be found in the limit. That is, we may allow M to not produce any hypothesis, or even produce hypotheses that do not fulfill the conditions (C1) and (C2), prior to the point when its hypotheses converge. However, since it is in general difficult to tell when a final hypothesis has been obtained [22, 47], it is desirable for M to produce a hypothesis for every σ that fulfills (C1) and (C2), for use as intermediate hypothesis.

We hence look at the computation of M for arbitrary σ. Clearly, for any indexed family and all σ, (C1) is computable. On the other hand, the computation of (C2) is possible if we are able to find a finite search space where

1. A hypothesis fulfilling (C1) and (C2) exist, and

2. Inclusion is decidable between every two languages in the set.

For example, in the case of the (non-erasing) pattern languages (see Section 2.3.1), a hypothesis that fulfills (C1) and (C2) is guaranteed to exist among all the languages of the patterns of length $\lfloor \mathit{content}(\sigma) \rfloor$ (see Theorem 2.9), and furthermore, inclusion between any two languages of this length is decidable (see Theorem 2.10). Hence M can be computed for the class [69]. Nevertheless, such conditions are not always available, and hence the computability of M has to be considered for each individual case. In general, the computation of M is complicated by the problems of (A) having a possibly infinite search space, and (B) deciding inclusions between languages. Note that a hypothesis space with finite thickness will have no problem with (A), as long as the search space can be enumerated recursively.

Alternatively, to remove concerns for (A) suppose that we bound the search for i and j in M with $i \leq |\sigma|$ and $j \leq |\sigma|$ (as suggested in the listing for M), and denote the resultant learner M'. Intuitively, such a learner outputs a language which is minimal with respect to inclusion, among all the languages in $\mathcal{L}$ of indices less than $|\sigma|$ that contains $\mathit{content}(\sigma)$. This is less ideal as an intermediate hypothesis, but at least we no longer have the problem of having an infinite search space. For such an M', if inclusion can be decided between every two languages in the hypothesis space, then the following Theorem applies.

Theorem 2.5 *If inclusion is decidable for an indexed family $\mathcal{L}$ where each language in $\mathcal{L}$ has a characteristic set within $\mathcal{L}$, then M'* ***TxtEx****-identifies $\mathcal{L}$.*

Proof. For any language $L_i \in \mathcal{L}$, let $S_i \subseteq L_i$ be a characteristic set for L_i. Let T be any text for L_i. Let $m \geq i$ be such that $S_i \subseteq \mathit{content}(T[m])$. By the property of a characteristic set, and that M' cannot output a proper superset of L_i, the output of $M'(T[m])$ must be i, or some $i' \leq m$ where $L_{i'} = L_i$. Denote this output i''. Now, for all $m' \geq m$, since $\mathit{content}(T[m']) \subseteq L_{i''}$, $M'(T[m'])$ must continue to output i''. Since this argument holds for every language L_i and every text for L_i, Theorem follows. ∎

The following result follows from Theorem 2.4 and 2.5. It first appeared in [9, 10] in a slightly different form.

Corollary 1 *If inclusion is decidable for an indexed family $\mathcal{L}$ with finite elasticity, then M' **TxtEx**-identifies $\mathcal{L}$.*

We now generalize the above computation of a minimal language to the unions of languages.

Definition 2.8 *Given a class of languages $\mathcal{L}$, a finite non-empty set $S \subseteq \mathcal{S}$ and $k \in N^+$, the problem **MINL**$(\mathcal{L}, S, k)$ is to find a set of up to k languages $\mathcal{L}' \subseteq \mathcal{L}$ such that $S \subseteq \bigcup_{L \in \mathcal{L}'} L$ and there are no other set of up to k languages $\mathcal{L}'' \subseteq \mathcal{L}$ where $S \subseteq \bigcup_{L' \in \mathcal{L}''} L' \subset \bigcup_{L \in \mathcal{L}'} L$.*

Solutions of **MINL** are called *k-minl*s, or simply *minl*s. M can now be generalized to the following learner M^k, which uses the computation of **MINL**. M is simply M^1.

$M^k(\sigma)$:
If $content(\sigma) \subseteq H_{|\sigma|-1}$, let $H_{|\sigma|} = H_{|\sigma|-1}$.
Otherwise let $H_{|\sigma|}$ be a solution of **MINL**$((L_i)_{i\in N}, content(\sigma), k)$, if such a solution exists.
Conjecture $H_{|\sigma|}$.

Due to concerns of having an infinite search space, similar to what we did previously with M, we may restrict the search space of M^k, by finding **MINL**$((L_i)_{i\in N \wedge i \leq |\sigma|}, content(\sigma), k)$ instead of **MINL**$((L_i)_{i\in N}, content(\sigma), k)$. Let such a learner be denoted M'^k, then Theorem 2.5, with $\mathcal{L}$ replaced with $\mathcal{L}^k$ for any $k \in N^+$, holds for M'^k.

Even if we are able to compute M^k, there is still a concern that its computation may be very costly. In order to compute **MINL** efficiently, Arimura *et al.* considered reducing the problem (of finding languages minimal with respect to inclusion) to one of finding languages minimal with respect to a relation called *subsumption*, a concept from logic programming [55].

In logic programming, subsumption defines a relation between logical clauses. To make such a relation relevant to languages in a class $\mathcal{L}$, we define a sequence of descriptors $\mathcal{D}_{\mathcal{L}} = d_0, d_1, \ldots$ such that each d_i is associated with the language L_i in $\mathcal{L}$. We then define a subsumption relation $\preceq$ over the descriptors $\mathcal{D}_{\mathcal{L}}$. To see how such descriptors may help us in the computation of **MINL**, consider the case where for any $d_i, d_j \in \mathcal{D}_{\mathcal{L}}$, $L_i \subseteq L_j \Rightarrow d_i \preceq d_j$. That is, it suffices that we find a descriptor d_i minimal with respect to the relation $\preceq$ (called a *least general generalization*) to find a language L_i minimal with respect to inclusion (also see Proposition 3.3).

Note that the concept of descriptors comes naturally to languages used in Computer Science, which are typically specified using a set of expressions in a grammar system, in which case, each set of these expressions can be taken to be the descriptor for the language it specify. For example, Prolog programs [56, 57] and elementary formal systems [5, 71, 72] are two such grammar systems commonly studied in the context of learning.

We now show the extension of $\preceq$ to the case of unions given in [10]. For any two sets of descriptors $D, D' \subseteq \mathcal{D}_{\mathcal{L}}$, we write $D \sqsubseteq D'$ just in case for each $d \in D$, there exists some $d' \in D'$ such that $d \preceq d'$. We write $D \sqsubset D'$ just in case $D \sqsubseteq D'$ but not $D' \sqsubseteq D$. In such a case, if for any two sets of descriptors $D, D' \subseteq \mathcal{D}_{\mathcal{L}}$, $\bigcup_{d_i \in D} L_i \subseteq \bigcup_{d_j \in D'} L_j \Rightarrow D \sqsubseteq D'$, then a set of descriptors D that is minimal with respect to the relation $\sqsubseteq$ (such a D is called a *minimal multiple generalization*) would give us a language $\bigcup_{d_i \in D} L_i$ that is minimal with respect to inclusion. This is restated as Proposition 2.6.

Definition 2.9 *Given a class $\mathcal{L} = L_0, L_1, \ldots$, its descriptors $\mathcal{D}_{\mathcal{L}} = d_0, d_1, \ldots$, a finite non-empty set $S \subseteq \mathcal{S}$ and $k \in N^+$, the problem **MMG**$(\mathcal{D}_{\mathcal{L}}, S, k)$ is to find a set of up to k descriptors $D \subseteq \mathcal{D}_{\mathcal{L}}$ such that $S \subseteq \bigcup_{d_i \in D} L_i$ and there are no other set of up to k descriptors $D' \subseteq \mathcal{D}_{\mathcal{L}}$ where $S \subseteq \bigcup_{d_j \in D'} L_j$ such that $D' \sqsubset D$.*

Solutions of **MMG** are called *k-mmg*s, or simply *mmg*s. The following Proposition can be easily verified.

Proposition 2.6 *Given a class $\mathcal{L} = L_0, L_1, \ldots$, its descriptors $\mathcal{D}_{\mathcal{L}} = d_0, d_1, \ldots$, a finite non-empty set $S \subseteq \mathcal{S}$ and $k \in N^+$. If for any two sets of up to k descriptors $D, D' \subseteq \mathcal{D}_{\mathcal{L}}$, $\bigcup_{d_i \in D} L_i \subseteq \bigcup_{d'_j \in D'} L_j \Rightarrow D \sqsubseteq D'$, then D is a solution for **MMG**$(\mathcal{D}_{\mathcal{L}}, S, k) \Rightarrow \bigcup_{d_i \in D} L_i$ is a solution for **MINL**$(\mathcal{L}, S, k)$.*

Let M_2^k be a learner modified from M^k such that **MINL** is replaced with **MMG** in its computation, as follows.

$M_2^k(\sigma)$:
If *content*$(\sigma) \subseteq H_{|\sigma|-1}$, let $H_{|\sigma|} = H_{|\sigma|-1}$.
Otherwise let $H_{|\sigma|} = \bigcup_{d \in D} L(d)$, where D is a solution for **MMG**$(\mathcal{D}_{\mathcal{L}}, \mathit{content}(\sigma), k)$.
Conjecture $H_{|\sigma|}$.

Then the following result for M_2^k follows by Proposition 2.6.

Theorem 2.7 *Let $k \in N^+$, indexed family $\mathcal{L} = L_0, L_1, \ldots$ where each $L \in \mathcal{L}^k$ has a characteristic set within $\mathcal{L}^k$, and corresponding descriptors $\mathcal{D}_{\mathcal{L}} = d_0, d_1, \ldots$ be given. If for any two sets of up to k descriptors $D, D' \subseteq \mathcal{D}_{\mathcal{L}}$, $\bigcup_{d_i \in D} L_i \subseteq \bigcup_{d_j \in D'} L_j \Rightarrow D \sqsubseteq D'$, then M_2^k **TxtEx**-identifies $\mathcal{L}^k$.*

2.3 Languages generated from patterns

We now introduce two main classes of languages: namely, the *pattern languages* and the *tree pattern languages*. The pattern languages and tree pattern languages are generated from some finite strings called *patterns* and *tree patterns*, respectively. We consider these patterns as the descriptors ($\mathcal{D}_{\mathcal{L}}$ in the previous section) for the languages they generate. The relations $\preceq$ and $\sqsubseteq$ for these languages will be made clear in this section.

2.3.1 The pattern languages

We now introduce the pattern languages. Let Σ be a finite alphabet with $card(\Sigma) \geq 2$, and $V = \{x_1, x_2, \ldots\}$ be a countable alphabet disjoint from Σ. Elements in Σ are called *constants* and elements in V are called *variables*.

An element in $(V \cup \Sigma)^*$ is called a *pattern*, and the length of a pattern p is denoted $|p|$. Each constant or variable has unit length. For a set of patterns P, $\lceil P \rceil = \max(\{|p| \mid p \in P\})$ and $\lfloor P \rfloor = \min(\{|p| \mid p \in P\})$. $\|P\|$ denotes the sum $\sum_{p \in P} |p|$.

Pat denotes an enumeration of all patterns.

A *substitution* is a homomorphism from patterns to patterns that maps every constant to itself. The image of a pattern p under a substitution θ is written $p\theta$. A substitution is *erasing* if it maps some or all variables to the empty string, and it is *non-erasing* otherwise. Unless stated otherwise, substitutions are non-erasing in this thesis. We write $p \preceq q$ iff $p = q\theta$ for some substitution θ, and we write $p \preceq_\epsilon q$ iff $p = q\theta$ for some erasing substitution θ. We write $p \prec q$ just in case $p \preceq q$ but not $q \preceq p$. Given two sets of patterns P and Q, we write $P \sqsubseteq Q$ iff for each $p \in P$, $p \preceq q$ for some $q \in Q$. $P \sqsubset Q$ just in case $P \sqsubseteq Q$ but not $Q \sqsubseteq P$. Similarly we define from $\preceq_\epsilon$ the relations $\prec_\epsilon$, $\sqsubseteq_\epsilon$ and $\sqsubset_\epsilon$. To improve readability we will often abbreviate the erasing version of some notations and results when they can be straightforwardly derived from the non-erasing case.

Every subset of Σ^* is said to be a *language*. The *pattern language* generated by a pattern p, written $L(p)$, is the set $\{w \in \Sigma^* \mid w \preceq p\}$. For a set P of patterns, $L(P)$ denotes the language $\bigcup_{p \in P} L(p)$. The symbols $L_\epsilon(p)$ and $L_\epsilon(P)$ are defined similarly. For an enumeration of patterns $\mathcal{P} = p_0, p_1, p_2, \ldots$, $\mathcal{L}(\mathcal{P})$ denotes the indexed family $L(p_0), L(p_1), L(p_2), \ldots$, and $\mathcal{L}_\epsilon(P) = L_\epsilon(p_0), L_\epsilon(p_1), L_\epsilon(p_2), \ldots$.

Theorem 2.8 (Angluin [2]) *$\mathcal{L}(Pat)$ has finite thickness.*

It is easy to see that a class of languages with finite thickness has finite elasticity. Applying Theorem 2.2 and 2.3, we get the following Corollary.

Corollary 2 *For any $k \in N^+$, $\mathcal{L}(Pat)^k$ is learnable.*

More precisely, $\mathcal{L}(Pat)^k$ is **TxtEx**-identified by the learner in [3]. We now show the results on *minl* computations for these classes. While inclusion is undecidable for $\mathcal{L}(Pat)$ [35], the following two properties of the class allows us to always find a *minl* from $\mathcal{L}(Pat)$, as discussed earlier.

Theorem 2.9 (Based on Shinohara [69]) *Given finite non-empty $S \subseteq \Sigma^+$. There exists a minl in $\{L(p) \mid p \in Pat \wedge |p| = \lfloor S \rfloor\}$.*

PROOF. Let $m = \lfloor S \rfloor$ and let $\mathcal{P}_m = \{p \in Pat \mid |p| = m\}$. Let $p \in \mathcal{P}_m$ be such that $S \subseteq L(p)$ and no $q \in \mathcal{P}_m$ has $L(q) \subset L(p)$. The existence of p can be deduced from that $\mathcal{P}_m$ is finite and that $S \subseteq L(x_1x_2 \ldots x_m) \in \mathcal{P}_m$. Now observe that (1) no $q \in Pat$ where $|q| > |p|$ generates a language that include the strings of length m in S, and (2) no $q \in Pat$ where $|q| < |p|$ generates a proper subset of $L(p)$, since $L(q) - L(p) \neq \emptyset$ (because there must exist some strings of length $|q|$ in $L(q)$, but no string in $L(p)$ is of length less than $|p| > |q|$). Hence $L(p)$ is a *minl*. ∎

Theorem 2.10 (Angluin [2]) *Given $p, q \in Pat$ where $|p| = |q|$. Then $L(p) \subseteq L(q) \Leftrightarrow p \preceq q$.*

In the case of unions, the computability of *minl* for $\mathcal{L}(\mathbf{Preg})^k$ where $k \geq 2$ is partially solved in [10, 66], via the subsumption relation.

We will use the following definition to define sets which will be shown to be characteristic sets in the next two chapters.

Definition 2.10 ([66, 80]) *For each pattern p and $n \in N$, we denote by*
(a) $S_n(p)$ the set of all the patterns obtainable from p by substituting each variable in p with a non-empty element from $(\Sigma^*)^{\leq n}$.
(b) $S_n^\epsilon(p)$ the set of all the patterns obtainable from p by substituting each variable in p with an element from $(\Sigma^*)^{=n}$.
(c) For a set of patterns P, $S_n(P) = \bigcup_{p \in P} S_n(p)$ and $S_n^\epsilon(P) = \bigcup_{p \in P} S_n^\epsilon(p)$.

2.3.2 Subclasses of the pattern languages

A *class* (or *subclass*) *of patterns* is an enumeration of patterns selected from Pat. We now define some important subclasses of the pattern languages.

2.3.2.1 Regular pattern

A *regular pattern* is a pattern where each variable appears at most once [68]. For a regular pattern, sometimes for simplicity we write $*$ in the place of each variable. Since it is understood that every variable is independent this creates no ambiguity.

We let **Preg** denote an enumeration of all the regular patterns.

2.3.2.2 Substring pattern

A pattern is a *substring pattern* just in case it is of the form x_1sx_2 and $s \in \Sigma^+$. We let **Psub** denote an enumeration of all the substring patterns. Note that **Psub** $\subset$ **Preg**.

2.3.2.3 Subsequence pattern

A pattern is a *subsequence pattern* just in case it is of the form $x_1a_1x_2a_2 \ldots x_na_nx_{n+1}$ where $a_i \in \Sigma$ for $i = 1, \ldots, n$ [28]. We let **Psubseq** denote an enumeration of all the subsequence patterns. Note that **Psubseq** $\subset$ **Preg**.

2.3.2.4 Sort regular patterns

Takae [78] *et al.* considered using a feature from the patterns used in the PROSITE database [74] called *sort symbols*. A *sort symbol* is a letter associated with a non-empty set $S \subseteq \Sigma$, and is written $[S]$, or $[A_1 A_2 \ldots]$ where $A_1, A_2, \ldots$ is an enumeration of the elements of S. Π denotes the set $\{[S] | S \neq \emptyset \wedge S \subseteq \Sigma\}$. For any $A \in \Sigma$, $[A]$ is treated as equivalent to A, and hence Σ is treated as a subset of Π. An element in $(\Pi \cup V)^+$ is called a *sort pattern*. In a substitution for a sort pattern, a sort symbol $[S]$ may be replaced only with another sort symbol $[S']$ where $S' \subseteq S$. A *sort regular pattern* is a sort pattern where each variable appears at most once.

2.3.2.5 Range specifiers

We introduce another feature from the patterns in the PROSITE database called *range specifiers*. A *range specifier* is a pair (m, n) of numbers m, $n \in N$ where $m \leq n$, that is attached to a variable (x say, the resultant expression is then $x(m, n)$) to indicate that the variable, when substituted with constant strings, can only be replaced with strings of lengths at least m and at most n.

2.3.3 The tree pattern languages

We now define the tree patterns and their languages. A tree pattern is a first-order term in formal logic. The following definitions follow mainly those in [7].

The symbol Γ denotes a finite alphabet, where each element of Γ is called a *function symbol* and associated with a non-negative integer called an *arity*. A function with arity 0 is also called a *constant*. We assume that Γ contains at least one constant. Let $V = \{x_1, x_2, \ldots\}$ be a countable set of symbols disjoint from Γ. Each element in V is called a variable.

A *tree pattern* p over Γ is an expression defined recursively as follows: (1) a constant $f \in \Gamma$, or a variable $x \in V$ is a tree pattern; (2) For a function f with arity n, where $n \geq 1$, and tree patterns $t_1, \ldots, t_n$, $f(t_1, \ldots, t_n)$ is a tree pattern. The function f or variable x in (1) and (2) is the *root symbol* of p. We consider tree patterns to be defined from left to right, that is, in $f(t_1, \ldots, t_n)$ of (2), t_i is defined before t_j if $i < j$. A tree pattern t *occurs* in a tree pattern p just in case t is used in the recursive definition of p. We call each such occurrence of t a *node of p* (or simply *a node*). We say that a node u is the i-th *child* (or simply *a child*) of a node v of the term $f(t_1, \ldots, t_n)$, just in case u occurs as t_i $(1 \leq i \leq n)$ in v.

A tree pattern is said to be *ground* if it contains no variable. Let TP denote an enumeration of the tree patterns and let TP_Γ denote the set of all ground tree patterns. The *size of a tree pattern p*, denoted by $\|p\|$, is the number of symbol occurrences in p (Note: this definition differs from that in [7]). A tree pattern p is said to be *canonical* just in case the first variable defined in a tree pattern p is x_1, the second distinct variable x_2, and so on.

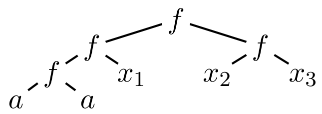
Given a tree pattern p and a node v in p, the *position of v*, written pos(v), is a string of natural numbers recursively defined as follows:

$$\mathrm{pos}(v) = \begin{cases} 1 & \text{if the node } v \text{ is } p \text{ itself.} \\ \mathrm{pos}(v')i & \text{if } v \text{ is the } i\text{-th child of the node } v' \text{ in } p. \\ \text{undefined} & \text{otherwise.} \end{cases}$$

We say that a symbol c *occurs at a position* ε in p just in case $\mathrm{pos}(v) = \varepsilon$ for some node v in p and c is the root symbol of v. Likewise a tree pattern t *occurs at a position* ε in p just in case $\mathrm{pos}(v) = \varepsilon$ and v is an occurrence of t. The height of a tree pattern t, $|t|$, is $\max(\{|\mathrm{pos}(v)| - 1 \mid v$ is a node in $t\})$, where $|\mathrm{pos}(v)|$ is the length of the string $\mathrm{pos}(v)$. We say that a node v is at depth h just in case $|\mathrm{pos}(v)| - 1 = h$.

Finally, we let $\langle n \rangle_\Gamma$ denote the smallest size s such that there are n elements of the same height in $\{t \in TP_\Gamma \mid \|t\| \leq s\}$, if such elements exist.

Example 1 *Let $\Gamma = \{a, f\}$ where a is a constant and f is of arity 2. Consider $\langle 7 \rangle_\Gamma$. Let $p = f(f(f(a, a), x_1), f(x_2, x_3))$ (see below).*



By replacing either a or $f(a, a)$ in each of the variables in p (hence the maximum size of these tree patterns is 15), we can verify that there are at least 7 tree patterns (8 for this case) of equal height (that is, 3) in TP_Γ. Hence $\langle 7 \rangle_\Gamma \leq 15$.

The following Proposition, which can be easily verified, states that such n ground tree patterns of equal height exist under very trivial conditions.

Proposition 2.11 *If Γ consists of at least one function of arity ≥ 2, then for all n, there exists n elements of the same height in TP_Γ.*

A *substitution* is a finite set of the form $\{y_1/t_1, \ldots, y_n/t_n\}$, where each y_i is a distinct variable in V, and each t_i is a tree pattern. An *instance* of a tree pattern p by a substitution $\theta = \{y_1/t_1, \ldots, y_n/t_n\}$, written $p\theta$, is the tree pattern obtained by simultaneously replacing each occurrence of the variable y_i with the term t_i $(1 \leq i \leq n)$. We write $p \preceq q$ iff $p = q\theta$ for some substitution θ. We write $p \prec q$ just in case $p \preceq q$ but not $q \preceq p$. Given two sets of tree patterns P and Q, we write $P \sqsubseteq Q$ iff for each $p \in P$, $p \preceq q$ for some $q \in Q$. $P \sqsubset Q$ just in case $P \sqsubseteq Q$ but not $Q \sqsubseteq P$.

Every subset of TP_Γ is said to be a *language*. The *tree pattern language* of $p \in TP$, written $L(p)$, is the set $\{w \in TP_\Gamma \mid w \preceq p\}$. For a set $P \subseteq TP$, $L(P) = \bigcup_{p \in P} L(p)$. For a sequence of tree patterns $\mathcal{P} = p_0, p_1, \ldots$, $\mathcal{L}(\mathcal{P})$ denotes the indexed family $L(p_0), L(p_1), \ldots$.

For any $p \in TP$, we denote by $S_n(p)$ and $H_n(p)$ the set of all elements of $L(p)$ generated by replacing variables in p with ground tree patterns of *sizes up to* n, and *height up to* n respectively.

Chapter 3

Fittest Covers

We begin this chapter with an example, using the regular pattern languages, of a *minl* which is in a sense too general. Let $S = \{cabcc,\ cbcac\}$, and consider the language generated by the pattern $*a*$. Since no other language that includes S is a proper subset of it, $*a*$ is minimal with respect to inclusion and hence $L(*a*)$ is a *minl*. However, another *minl*, $L(*bc*)$, likely has less number of strings of any given length than $L(*a*)$, and may be preferred in some applications (see [18] and results in Chapter 6). Considerations for this minimality give us the idea for the *fittest covers*.

A fittest cover is a hypothesis that is consistent with the given examples, and contains the least number of elements of up to a specified length. An instance of the problem to find a fittest cover involves the following specifications: (1) a class $\mathcal{L}$ of languages from which an output set of languages $\mathcal{L}'$ is drawn, (2) k, the number of languages allowed in $\mathcal{L}'$, (3) a set S of examples that we want $\bigcup_{L\in\mathcal{L}'} L$ to contain, and (4) a length ℓ, where we want $\bigcup_{L\in\mathcal{L}'} L$ to contain the least number of elements of up to length ℓ, among all the other sets of languages that fulfill (1)–(3). We denote an instance of the problem specified in such a way $\mathbf{FC}(\mathcal{L}, S, k, \ell)$.

We first look at how we may use an algorithm that computes $\mathbf{FC}(\mathcal{L}, S, k, \ell)$ for learning. First of all we want to base our conjectures solely on the output from a computation of $\mathbf{FC}(\mathcal{L}, S, k, \ell)$, without using knowledge obtainable via other means (e.g. information possibly encoded in the indexing of the languages). In this sense, all the solutions in $\mathbf{FC}(\mathcal{L}, S, k, \ell)$ are treated as equals. We then consider suitable values for ℓ. We can, perhaps, set ℓ to $\lceil S \rceil$. This however creates a problem: Suppose there are two languages $L,\ L' \in \mathcal{L}$ where L is finite, $L \subseteq L'$, and the two languages contains exactly the same elements of up to $\lceil L \rceil$. In this case, given any complete representation of the elements in L, both L and L' will be in $\mathbf{FC}(\mathcal{L}, L, k, \lceil L \rceil)$, and hence the output of $\mathbf{FC}(\mathcal{L}, L, k, \lceil L \rceil)$ alone is insufficient for us to decide on the correct conjecture L. This problem, however, does not arise for a class that consists only of infinite languages, since there is no bound to the length of the elements of any language. Of course, the problem is also avoided if we simply allow ℓ to be extended indefinitely. Hence we consider the following two cases: (Case 1) allow only infinite languages in the class, and compute $\mathbf{FC}(\mathcal{L}, S, k, \lceil S \rceil)$; and (Case 2) allow both finite and infinite languages in the class, but allow ℓ to extend indefinitely beyond the lengths of the examples.

We now proceed formally.

3.1 Fittest Cover Problem

Definition 3.1 *Given a class $\mathcal{L} = L_0, L_1, \ldots$ of languages, k, $\ell \in N^+$, and a finite non-empty set of examples $S \subseteq \bigcup_{L\in\mathcal{L}} L$, the* Fittest Cover Problem *(written as $\mathbf{FC}(\mathcal{L}, S, k, \ell)$) is to find, among every set $\mathcal{L}'$ of up to k languages in $\{L \in \mathcal{L} \mid L \cap S \neq \emptyset\}$ where $S \subseteq \bigcup_{L\in\mathcal{L}'} L$, for one that minimizes card$(\bigcup_{L\in\mathcal{L}'} L^{\leq\ell})$.*

Solutions from $\mathbf{FC}(\mathcal{L}, S, k, \ell)$ are called *fittest covers*.

3.2 Learning with fittest covers

We want to consider the use of fittest covers as conjectures in inductive inference, that is, we let a learner conjecture one of the solutions obtained in an instance of the problem. Besides the minimal property of the hypotheses, doing so is also interesting because in a sense, this reduces learning into a task of simply counting the number of elements of up to some length in each language given.

An immediate concern regarding the use of such a strategy is that its minimality is partial. That is, while the minimality of a *minl* is with respect to Σ^*, the minimality of a fittest cover is only with respect to $(\Sigma^*)^{\leq\ell}$ for some finite length ℓ. In this section we show how the requirement of such a minimality restricts learning. To simplify the discussion in this section we use only $\mathbf{FC}(\mathcal{L}, S, 1, \ell)$, that is, the case without union. This does not make the discussion any less relevant to the study of unions since the finite unions of an indexed family is also an indexed family.

As discussed earlier, we study the following two learners, which on any input $\sigma \in$ SEQ:

1. ***FCLnrConLen***(σ) conjectures L_i where $i \in N$ is the least index such that $L_i \in \mathbf{FC}(\mathcal{L}, \mathit{content}(\sigma), 1, \lceil \mathit{content}(\sigma) \rceil)$.

2. ***FCLnrTxtLen***(σ) conjectures L_i where $i \in N$ is the least index such that $L_i \in \mathbf{FC}(\mathcal{L}, \mathit{content}(\sigma), 1, |\sigma|)$.

Note that the least index is used to simplify convergence and its use does not add a dependency on the numberings for the cases we consider (where ℓ extends indefinitely), as the results will show. In the case that $\mathbf{FC}(\mathcal{L}, \cdots)$ is not computable due to that $\mathcal{L}$ is infinite, we may alternatively compute for $\mathbf{FC}(\{L_1, L_2, \ldots, L_{|\sigma|}\}, \cdots)$. Since the correct language is included in the hypothesis space in the limit, this does not weaken the strength of the learner in terms of **TxtEx**-identification (although one may find a fittest cover found from a bounded search space to be less desirable as an intermediate hypothesis).

Definition 3.2 *Given a class $\mathcal{L}$ of languages and a language L, we say that a finite set $S \subseteq L$ is a*
(1) witness of minimality for L within $\mathcal{L}$ *just in case there exists $\ell \in N$ such that for all $L' \in \mathcal{L}$, $S \subseteq L' \Rightarrow$ for all $\ell' \geq \ell$, $card(L^{\leq \ell'}) \leq card(L'^{\leq \ell'})$,*
(2) witness of neighborhood minimality for L within $\mathcal{L}$ *just in case for every S' where $S \subseteq S' \subseteq L$ and for every $L' \in \mathcal{L}$, $S' \subseteq L' \Rightarrow card(L^{\leq \lceil S' \rceil}) \leq card(L'^{\leq \lceil S' \rceil})$.*

As mentioned in the introduction, ***FCLnrConLen*** does not identify a class of languages where there exists L_1 and L_2 such that $L_2^{\leq \lceil L_1 \rceil} = L_1$. To avoid this we consider classes that consists only of infinite languages in the following characterization of ***FCLnrConLen***.

Theorem 3.1 ***FCLnrConLen TxtEx****-identifies an indexed family $\mathcal{L} = L_0, L_1, L_2, \ldots$ of infinite languages if and only if for each $i \in N$ there exists a witness of neighborhood minimality for L_i within $\mathcal{L}$.*

Proof. ($\Leftarrow$) For $L_i \in \mathcal{L}$, let T be any text for L_i, and let S_i be a witness of neighborhood minimality for L within $\mathcal{L}$. Let $\ell \in N$ and finite $A_i \subseteq \Sigma^*$ be such that for each language $L_j \neq L_i$ where $j < i$, (1) if $L_i \subset L_j$, then $card(L_i^{\leq \ell}) < card(L_j^{\leq \ell})$, (2) otherwise A_i contains an element in $L_i - L_j$. Let x_i be an element longer than ℓ in L_i. Let $n \in N$ (where we furthermore require that $n \geq i$ if ***FCLnrConLen*** is made to compute **FC** from the class $\{L_1, \ldots, L_{|\sigma|}\}$ instead of $\mathcal{L}$ as mentioned) be such that $A_i \cup S_i \cup \{x_i\} \subseteq content(T[n])$. (Note that we do not require such n to be recursively determined.) Then, for all $T[n']$ where $n' \geq n$, in the case that $j < i$, either (1) $L_j = L_i$, (2) $content(T[n]) \not\subseteq L_j$, or (3) $card(L_i^{\leq n}) < card(L_j^{\leq n})$; and in the case that $j > i$, $card(L_i^{\leq n}) \leq card(L_j^{\leq n})$. From this it is not difficult to see that ***FCLnrConLen*** will converge to some $i' \leq i$ where $L_{i'} = L_i$ after reading $T[n]$.

($\Rightarrow$) Given a language L_i where no such witness of neighborhood minimality exists, we show how to construct a text T for L_i for which ***FCLnrConLen*** fails to converge on to an index for a language for L_i. At each stage l of the construction we look for a finite set $S^l \subseteq L_i$ and sequence σ^l where every S^{l+1} includes $L_i^{\leq \lceil S^l \rceil}$ and every sequence σ^{l+1} begins with σ^l. At stage 0, let σ^0 be the empty sequence and let $S^0 = \emptyset$. At every stage $l+1$, we find a language $L_j \neq L_i$ and S^{l+1} where $L_i^{\leq \lceil S^l \rceil} \subseteq S^{l+1} \subseteq L_j \cap L_i$ and $card(L_i^{\leq \lceil S^{l+1} \rceil}) > card(L_j^{\leq \lceil S^{l+1} \rceil})$. This search will always succeed, because otherwise $L_i^{\leq \lceil S^l \rceil}$ is a witness of neighborhood minimality for L_i within $\mathcal{L}$. Let σ^{l+1} be a sequence (of length $\geq j$ if ***FCLnrConLen*** is made to compute **FC** from the class $\{L_1, \ldots, L_{|\sigma|}\}$ instead of $\mathcal{L}$ as mentioned) extending σ^l where $content(\sigma^{l+1}) = S^{l+1}$. Let T be a text where each initial portion begins with a sequence σ^l for some l. Since at every stage $l+1$, σ^{l+1} contains all the elements in L_i of up to length $\lceil S^l \rceil$, T is a text for L_i. However, ***FCLnrConLen*** fails to converge on T to an index for L_i. ∎

Note that the existence of such witnesses of neighborhood minimality suffices for their effective enumeration in the limit.

Corollary 3 *There exists a learnable indexed family $\mathcal{L}$ of infinite languages not **TxtEx**-identified* *by **FCLnrConLen**.*

PROOF. Let $a \in \Sigma$ and let a^n denote the elements in $\{a\}^*$ of length n. For each $i, j \in N^+$, $j \geq 2$, let $L_{ij} = \{a^n \mid n \leq i \ \vee \ n \equiv i \pmod{j}\}$. Let $\mathcal{L}$ be an indexed family for the class of languages $\{L_{ij} \mid i,j \in N^+, j \geq 2\}$. For each i, $j \in N^+$, let S_{ij} be a finite set including $\{a^n \mid n \leq i\}$ and $\{a^{i+j}\}$. It is easy to verify that S_{ij} is a finite tell-tale for L_{ij} and hence $\mathcal{L}$ is learnable. We now show that there is no witness of neighborhood minimality for any language in the class. For any i, j, let S be any finite subset of L_{ij}. Let i', $j' \in N^+$ be such that: (1) $i' \geq \max(\{i, \lceil S \rceil\})$, (2) $i' + j' \equiv i \pmod{j}$, and (3) $i + (i' + j' - i)/j > i' + 1$. Let $S' = S \cup \{a^{i'+j'}\}$. By (1), $S' \subseteq L_{i'j'}$; and by (2), $S' \subseteq L_{ij}$. However, there are at least $i + (i' + j' - i)/j$ $(> i' + 1)$ elements in $L_{ij}^{\leq i'+j'}$, but only $i' + 1$ elements in $L_{i'j'}^{\leq i'+j'}$. ∎

The following can be shown using essentially the same argument as that in proving Theorem 3.1. We skip the details.

Theorem 3.2 ***FCLnrTxtLen*** ***TxtEx****-identifies an indexed family $\mathcal{L} = L_0, L_1, L_2, \ldots$ if and only if for each $i \in N$ there exists a witness of minimality for L_i within $\mathcal{L}$.*

Corollary 4 *If every language in an indexed family $\mathcal{L}$ has a characteristic set within $\mathcal{L}$, then* ***FCLnrTxtLen*** ***TxtEx****-identifies $\mathcal{L}$.*

Note that each language in the class of all finite languages has itself as a characteristic set, and hence the class can be learned via ***FCLnrTxtLen***. For an example of a class learnable by ***FCLnrTxtLen*** where no language in it has a characteristic set within the class, let $w_0, w_1, w_2, \ldots$ be an enumeration of Σ^* and consider the class $\mathcal{L} = (\Sigma^* - \{w_i\})_{i \in N}$.

Corollary 5 *Given an indexed family $\mathcal{L}$ of recursive languages with finite elasticity and $k \in N^+$,* ***FCLnrTxtLen*** ***TxtEx****-identifies the class of unions of up to k languages in $\mathcal{L}$.*

Hence by Theorem 2.3, for each $k \in N^+$, $\mathcal{L}(Pat)^k$ is learnable by ***FCLnrTxtLen***. For erasing patterns, even though the full class of the erasing pattern languages does not have finite elasticity, some of its interesting subclasses do, for example the *erasing regular pattern languages* [68].

There is also a class of pattern languages with infinite elasticity but nonetheless learnable by ***FCLnrTxtLen***, namely, the class of unbounded unions of the erasing languages of substring patterns. The class has infinite elasticity, but each language in it has a characteristic set within the class [73]. We show more results on this class in Section 4.1.

3.3 Comparing fittest covers to *minl*s and *mmg*s

We now turn to the relationship among the fittest covers, *minl* and *mmg*. To do so we compare between the sets of all the solutions obtainable from **MINL**$(\mathcal{L}, S, k)$, **MMG**$(\mathcal{L}, S, k)$ and

$\mathbf{FC}(\mathcal{L}, S, k, \ell)$, for a class $\mathcal{L}$, a finite non-empty $S \subseteq \Sigma^*$, $k \in N^+$, and $\ell \in N^+$. We define these as $\mathcal{MINL}$, $\mathcal{MMG}$ and $\mathcal{FC}$ respectively below. The following definitions extend straightforwardly to the erasing pattern languages by replacing the symbols L with L_ϵ, and $\mathcal{L}$ with $\mathcal{L}_\epsilon$.

Definition 3.3 *Given a class of patterns $\mathcal{P} \subseteq Pat$, $\ell \in N^+$ and $S \subseteq \Sigma^*$,*
(1) $\mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k) = \{P \subseteq \mathcal{P} \mid L(P)$ is a solution for $\boldsymbol{MINL}(\mathcal{L}(\mathcal{P}), S, k)\}$,
(2) $\mathcal{MMG}(\mathcal{P}, S, k) = \{P \subseteq \mathcal{P} \mid P$ is a solution for $\boldsymbol{MMG}(\mathcal{P}, S, k)\}$,
(3) $\mathcal{FC}(\mathcal{L}(\mathcal{P}),\ S, k, \ell) = \{P \subseteq \mathcal{P} \mid L(P)$ is a solution for $\boldsymbol{FC}(\mathcal{L}(\mathcal{P}), S, k, \ell)\}$.

Note that $\mathcal{FC}$ and $\mathcal{MINL}$ are defined to be classes of patterns rather than classes of languages solely to simplify the comparison with $\mathcal{MMG}$. The results in this thesis involving only $\mathcal{FC}$ and $\mathcal{MINL}$ can be easily adapted to remove the dependency on a pattern class $\mathcal{P}$.

For a class of patterns $\mathcal{P} \subseteq Pat$, since (from [2]) $(\forall P,\ Q \subseteq \mathcal{P})\ [P \sqsubseteq Q \Rightarrow L(P) \subseteq L(Q)]$, a solution $P \in \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$ is either in $\mathcal{MMG}(\mathcal{P}, S, k)$, or there exists some solution Q for $\mathcal{MMG}(\mathcal{P}, S, k)$ where $L(P) = L(Q)$ (i.e. $P \sqsubset Q \Rightarrow L(P) \subset L(Q) \vee L(P) = L(Q)$). On the other hand, $\mathcal{MMG}(\mathcal{P}, S, k) \subseteq \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$ if for all $P,\ Q \in \mathcal{P}$, $L(P) \subseteq L(Q) \Rightarrow P \sqsubseteq Q$, as noted in [10]. The following states this condition more precisely.

Proposition 3.3 *For any class of patterns $\mathcal{P} \subseteq Pat$, if for every sets of up to k patterns P, $Q \subseteq \mathcal{P}$ we have $L(P) \subseteq L(Q) \Rightarrow P \sqsubseteq Q$, then $\mathcal{MMG}(\mathcal{P}, S, k) = \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$.*

Proof. To show the result it suffices that we show $L(P) \subset L(Q) \Leftrightarrow P \sqsubset Q$. $(\Rightarrow)$ is given by $L(P) \subset L(Q)$ (by the hypothesis)$\Rightarrow P \sqsubseteq Q$, and $P \sqsubseteq Q \wedge Q \sqsubseteq P \Rightarrow L(P) = L(Q)$. We now show $P \sqsubset Q \Rightarrow L(P) \subset L(Q)$. Let P and Q be sets of patterns from $\mathcal{P}$ where $P \sqsubset Q$. We argue that $L(Q) - L(P) \neq \emptyset$. Suppose otherwise, then $P \sqsubset Q \Rightarrow L(P) \subseteq L(Q)$ (since $L(Q) - L(P) = \emptyset$)$\Rightarrow$ $L(Q) = L(P)$ (by the hypothesis)$\Rightarrow Q \sqsubseteq P$, a contradiction. ∎

The following states necessary and sufficient conditions for the solutions of $\mathcal{FC}$ to be included in $\mathcal{MMG}$ and $\mathcal{MINL}$.

Proposition 3.4 *Given $k,\ \ell \in N^+$, class of patterns $\mathcal{P} \subseteq Pat$ and a finite non-empty $S \subseteq \Sigma^*$. Let $\mathcal{P}' = \{p \in \mathcal{P} \mid L(p) \cap S \neq \emptyset\}$. Then,*
***1**. $\mathcal{FC}(\mathcal{L}(\mathcal{P}), S, k, \ell) \subseteq \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$ if and only if, for each solution $P \in \mathcal{FC}(\mathcal{L}(\mathcal{P}), S, k, \ell)$, every set of up to k patterns $Q \subseteq \mathcal{P}'$ where $S \subseteq L(Q)$ has $L(P) \subset L(Q) \Rightarrow L(Q)^{\leq \ell} - L(P)^{\leq \ell} \neq \emptyset$.*
***2**. $\mathcal{FC}(\mathcal{L}(\mathcal{P}), S, k, \ell) \subseteq \mathcal{MMG}(\mathcal{P}, S, k)$ if and only if, for each solution $P \in \mathcal{FC}(\mathcal{L}(\mathcal{P}),\ S, k, \ell)$, every set of up to k patterns $Q \subseteq \mathcal{P}'$ where $S \subseteq L(Q)$ has $P \sqsubset Q \Rightarrow L(Q)^{\leq \ell} - L(P)^{\leq \ell} \neq \emptyset$.*

Proof. Part **1**. $(\Rightarrow)$ Let $P \in \mathcal{FC}(\mathcal{L}(\mathcal{P}), S, k, \ell)$. Suppose there exists a set of up to k patterns $Q \subseteq \mathcal{P}'$ where $S \subseteq L(Q)$, $L(P) \subset L(Q)$, and $L(Q)^{\leq \ell} - L(P)^{\leq \ell} = \emptyset$, then clearly, $Q \in \mathcal{FC}(\mathcal{L}(\mathcal{P}), S, k, \ell)$ but $Q \notin \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$ (as witnessed by P). Then, $\mathcal{FC}(\mathcal{L}(\mathcal{P}), S, k, \ell)$ $\not\subseteq \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$.

($\Leftarrow$) Towards a contradiction let $Q \in \mathcal{FC}(\mathcal{L}(\mathcal{P}), S, k, \ell) - \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$. We then have:

(A) $Q \subseteq \mathcal{P}'$ (since $Q \in \mathcal{FC}(\mathcal{L}(\mathcal{P}), S, k, \ell)$), and

(B) there exists $L(P) \subset L(Q)$ where $S \subseteq L(P)$, that is, P witnesses that $Q \notin \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$. We assume $P \subseteq \mathcal{P}'$, since otherwise we may remove the patterns in P that are not in $\mathcal{P}'$. Since P contains strictly less elements than Q, $P \in \mathcal{FC}(\mathcal{L}(\mathcal{P}), S, k, \ell)$.

Then, by the Proposition, $L(Q)^{\leq \ell} - L(P)^{\leq \ell} \neq \emptyset$, and hence $Q \notin \mathcal{FC}(\mathcal{L}(\mathcal{P}), S, k, \ell)$, a contradiction.

Part **2** is proven similarly. ∎

Propositions 3.3 and 3.4 extend staightforwardly, with the same arguments, to the erasing pattern languages. The following result relates the classes $\mathcal{FC}$ and $\mathcal{MINL}$ using the characteristic sets, and extends immediately to the erasing case.

Theorem 3.5 *Given a class of patterns $\mathcal{P} \subseteq Pat$, a finite non-empty $S \subseteq \Sigma^*$ and $k \in N^+$. Let $\mathcal{P}'$ be an enumeration of all the patterns $p \in \mathcal{P}$ where $L(p) \cap S \neq \emptyset$. If there exists $\ell \in N$ such that every language $L(P) \in \mathcal{L}(\mathcal{P}')^k$ has a characteristic set $S_P \subseteq L(P)^{\leq \ell}$ within $\mathcal{L}(\mathcal{P}')^k$, then $\mathcal{FC}(\mathcal{L}(\mathcal{P}), S, k, \ell) \subseteq \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$.*

Proof. Let $P \in \mathcal{FC}(\mathcal{L}(\mathcal{P}), S, k, \ell)$. Suppose there exists $L(Q) \in \mathcal{L}(\mathcal{P}')^k$ where $L(P) \subset L(Q)$ and $L(Q)^{\leq \ell} - L(P)^{\leq \ell} = \emptyset$. Let $S_Q \subseteq L(Q)^{\leq \ell}$ be a characteristic set of $L(Q)$ within $\mathcal{L}(\mathcal{P}')^k$. Since $L(Q)^{\leq \ell} - L(P)^{\leq \ell} = \emptyset$, we would have $S_Q \subseteq L(P)$ and hence $L(Q) \subseteq L(P)$, a contradiction. Hence no such $L(Q)$ exists, and therefore the Theorem holds by Part 1 of Proposition 3.4. ∎

Hence it suffices that we specify a sufficiently large ℓ, depending only on the maximum length of the elements in the characteristic sets, in the **FC** problem to obtain **MINL** solutions. A caveat here is that even though a characteristic set is known to exist for every non-erasing pattern language within the class (since the class has finite elasticity), such maximum length of the elements in a characteristic set cannot be recursively determined for every non-erasing pattern language. The reason is that, if such lengths can be decided, then inclusion for any two patterns can be decided — by recursively generating their constituent elements of up to that length. However, Jiang *et al.* has shown that the problem of inclusion between any two given non-erasing pattern languages is undecidable [35].

Nevertheless, the same problem does not arise for the regular patterns nor its bounded unions. It has been shown that under specific conditions, such sufficient lengths for the elements of characteristic sets can be computed for these regular pattern languages. We present these results next.

3.4 Fittest covers that are also *minl*s

To demonstrate how Theorem 3.5 may be applied, in this section we use earlier results in [66] and [80] to show how we may obtain hypotheses that are at the same time fittest covers and *minl*s.

In [66], Sato *et al.* showed that it is possible to use $S_1(.)$ and $S_2(.)$ as characteristic sets for the bounded unions of the regular pattern languages, when the alphabet size is sufficiently larger than the number of unions allowed, as follows.

Theorem 3.6 (Theorem 11 and 20 in [66]) *Let $k \in N^+$. For every set of patterns $P \subseteq$ **Preg** and up to k patterns $Q \subseteq$ **Preg**,*
(1) if $card(\Sigma) \geq 2k+1$, then the following three statements are equivalent:
(i) $S_1(P) \subseteq L(Q)$, (ii) $P \sqsubseteq Q$, (iii) $L(P) \subseteq L(Q)$.
(2) if $k \geq 3$ and $card(\Sigma) \geq 2k-1$, then the following three statements are equivalent:
(i) $S_2(P) \subseteq L(Q)$, (ii) $P \sqsubseteq Q$, (iii) $L(P) \subseteq L(Q)$.

The equivalence of (i) and (iii) in Theorem 3.6 shows that $S_1(P)$ and $S_2(P)$ are characteristic sets for any set of languages from $\mathcal{L}(\mathbf{Preg})$ within the class $\mathcal{L}(\mathbf{Preg})^k$, under the respective conditions. The following Corollary follows.

Corollary 6 *For every finite non-empty $S \subseteq \Sigma^*$ and $k \in N^+$.*
(1) if $card(\Sigma) \geq 2k+1$, $\mathcal{FC}(\mathcal{L}(\boldsymbol{Preg}), S, k, \lceil S \rceil) \subseteq \mathcal{MINL}(\mathcal{L}(\boldsymbol{Preg}), S, k) = \mathcal{MMG}(\mathcal{L}(\boldsymbol{Preg}), S, k)$.
(2) if $card(\Sigma) \geq 2k-1$ and $k \geq 3$, then $\mathcal{FC}(\mathcal{L}(\boldsymbol{Preg}), S, k, 2\lceil S \rceil) \subseteq \mathcal{MINL}(\mathcal{L}(\boldsymbol{Preg}), S, k) = \mathcal{MMG}(\mathcal{L}(\boldsymbol{Preg}), S, k)$.

Proof. (1) We first note that given any S, the class $\mathcal{P}$ of patterns $p \in \mathbf{Preg}$ where $L(p) \cap S \neq \emptyset$ consists only of patterns of length up to $\lceil S \rceil$. $S_1(P)$ where $P \subseteq \mathcal{P}$ hence contains only elements of up to length $\lceil S \rceil$. Theorem 3.5 and 3.6 then give us $\mathcal{FC}(\mathcal{L}(\mathbf{Preg}), S, k, \lceil S \rceil) \subseteq \mathcal{MINL}(\mathcal{L}(\mathbf{Preg}), S, k)$. That $\mathcal{MINL}(\mathcal{L}(\mathbf{Preg}), S, k) = \mathcal{MMG}(\mathcal{L}(\mathbf{Preg}), S, k)$ is from Proposition 3.3 and the equivalence of (ii) and (iii) in Theorem 3.6. Part (2) is shown similarly. The equivalence between $\mathcal{MINL}$ and $\mathcal{MMG}$ is due to the equivalence of (ii) and (iii) in Theorem 3.6. ∎

For the erasing regular pattern languages, Uemura and Sato [80] obtained the following.

Theorem 3.7 (Theorem 2 and 3 in [80]) *Let $k \in N^+$. For every set of up to k patterns $P, Q \subseteq$ **Preg**,*
(1) if $card(\Sigma) \geq 2k+1$, then the following three statements are equivalent:
(i) $S_1^\epsilon(P) \subseteq L_\epsilon(Q)$, (ii) $P \sqsubseteq_\epsilon Q$, (iii) $L_\epsilon(P) \subseteq L_\epsilon(Q)$.
(2) if $card(\Sigma) \geq k+2$, then the following three statements are equivalent:
(i) $S_2^\epsilon(P) \subseteq L_\epsilon(Q)$, (ii) $P \sqsubseteq_\epsilon Q$, (iii) $L_\epsilon(P) \subseteq L_\epsilon(Q)$.

Corollary 7 *For every finite non-empty $S \subseteq \Sigma^*$ and $k \in N^+$,*
(1) if $card(\Sigma) \geq 2k+1$, then $\mathcal{FC}(\mathcal{L}_\epsilon(\boldsymbol{Preg}), S, k, 2\lceil S \rceil + 1) \subseteq \mathcal{MINL}(\mathcal{L}_\epsilon(\boldsymbol{Preg}), S, k)$;
(2) if $card(\Sigma) \geq k+2$, then $\mathcal{FC}(\mathcal{L}_\epsilon(\boldsymbol{Preg}), S, k, 3\lceil S \rceil + 2) \subseteq \mathcal{MINL}(\mathcal{L}_\epsilon(\boldsymbol{Preg}), S, k) = \mathcal{MMG}(\mathcal{L}_\epsilon(\boldsymbol{Preg}), S, k)$.

Proof. Without loss of generality we consider only patterns where no variables appear consecutively, because replacing any such consecutive variables with a single variable does not alter the corresponding erasing pattern language. Denote this class of patterns by $\mathbf{Preg}'$. Now given any S, the class $\mathcal{P}$ of patterns $p \in \mathbf{Preg}'$ where $L(p) \cap S \neq \emptyset$ consists only of patterns of length up to $2\lceil S \rceil + 1$, each containing at most $\lceil S \rceil + 1$ variables. Hence for any $P \subseteq \mathcal{P}$, strings in $S_1^\epsilon(P)$ are of length at most $2\lceil S \rceil + 1$, while strings in $S_2^\epsilon(P)$ are of length at most $3\lceil S \rceil + 2$. Theorem 3.5 and 3.7 then give us the subset relations between $\mathcal{FC}$ and $\mathcal{MINL}$. The equivalence between $\mathcal{MINL}$ and $\mathcal{MMG}$ is due to the equivalence of *(ii)* and *(iii)* in Theorem 3.7. ∎

In the next chapter, we shall continue this search for length bounds of the characteristic sets for other subclasses of the pattern languages, and for the tree pattern languages.

Chapter 4

Fittest Covers that are *minl*s

Similar to what we did in Section 3.4, in this chapter we want to apply Theorem 3.5 on subclasses of the pattern languages to show how we can find from these subclasses, languages containing a set of examples that fulfill both the FC and MINL notions of minimality. This requires us to first find the characteristic sets as required in Theorem 3.5. We first look at the substring and subsequence patterns.

4.1 Substring patterns and subsequence patterns

Very simple characteristic sets exist for the substring and subsequence patterns, allowing for stronger results relating FC to MINL and MMG for these patterns. We first demonstrate these characteristic sets.

Lemma 4.1 *Suppose* $card(\Sigma) \geq 2$ *and* $\mathcal{P} \in \{\boldsymbol{Psub}, \boldsymbol{Psubseq}\}$*. For any* $p \in \mathcal{P}$*,*
(a) there exists $w \in L(p)$ *of length* $\lfloor L(p) \rfloor$ *such that for all* $q \in \mathcal{P}$ *where* $p \not\preceq q$*,* w *is not in* $L(q)$*.*
(b) there exists $w \in L_\epsilon(p)$ *of length* $\lfloor L_\epsilon(p) \rfloor$ *such that for all* $q \in \mathcal{P}$ *where* $p \not\preceq_\epsilon q$*,* w *is not in* $L_\epsilon(q)$*.*

Proof. We first show Part (a) for **Psub**. Let $p = x_1 u x_2$ where $x_1,\ x_2 \in V$ and $u \in \Sigma^+$. We claim that any string $w = aub$ for some (possibly equal) $a,\ b \in \Sigma$ is not in $L(q)$ if $p \not\preceq q$. Towards a contradiction let $q = y_1 v y_2$ where $y_1,\ y_2 \in V$, $v \in \Sigma^+$, and suppose $w \in L(q)$. Since $w \in L(q)$, $w = a w_1 v w_2 b$ for some $w_1,\ w_2 \in \Sigma^*$. However, the substitution θ where $y_1\theta = x_1 w_1$ and $y_2\theta = w_2 x_2$ then witnesses that $p \preceq q$, a contradiction. Part (b) for **Psub** can be shown similarly by letting $w = u$.

We now show Part (a) for **Psubseq**. Let $p = x_1 a_1 x_2 a_2 \ldots a_n x_{n+1}$ where $x_1,\ \ldots,\ x_{n+1} \in V$ and $a_1,\ \ldots,\ a_n \in \Sigma$. Let $w = p\theta$ where θ is a substitution transforming each x_i to a_i for $1 \leq i \leq n$, and transforming x_{n+1} to a_n. Now for any $q \in$ **Psubseq**, if $w \preceq q$, it is not difficult to see that q must be of the form $y_1 a_{i_1} y_2 a_{i_2} \ldots a_{i_k} y_{k+1}$ where $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ and y_1,

..., $y_{k+1} \in V$. However, then $p \preceq q$. Part (b) for **Psubseq** can be shown with $w = p\theta'$ where θ' is a substitution which erases all the variables. ∎

Note that Lemma 4.1 implies $L(P) \subseteq L(Q)$ or $L_\epsilon(P) \subseteq L_\epsilon(Q) \Rightarrow P \sqsubseteq Q$. In the following Lemma, for a set of patterns P, $c(P)$ denotes the maximum of $\{\lfloor L(p) \rfloor \mid p \in P\}$.

Lemma 4.2 *Suppose card*$(\Sigma) \geq 2$ *and* $\mathcal{P} \in \{\boldsymbol{Psub}, \boldsymbol{Psubseq}\}$. *For any two sets of finite* P, $Q \subseteq \mathcal{P}$,
(a) $P \sqsubset Q \Rightarrow L(Q)^{\leq c(Q)} - L(P)^{\leq c(Q)} \neq \emptyset$,
(b) $L(P) \subset L(Q) \Rightarrow L(Q)^{\leq c(Q)} - L(P)^{\leq c(Q)} \neq \emptyset$, *and*
(c) $P \sqsubset Q \Leftrightarrow L(P) \subset L(Q)$.

Proof. We first proof Part (a). Suppose there exists P, $Q \subseteq \mathcal{P}$ where $P \sqsubset Q$ and $L(Q)^{\leq c(Q)} - L(P)^{\leq c(Q)} = \emptyset$. Since $L(Q)^{\leq c(Q)} - L(P)^{\leq c(Q)} = \emptyset$, by Lemma 4.1, $q \in Q$ implies $q \preceq p$ for some $p \in P$, and hence $Q \sqsubseteq P$, a contradiction. For Part (c), that $P \sqsubset Q \Rightarrow L(P) \subset L(Q)$ is from Part (a) and the fact that $P \sqsubset Q \Rightarrow L(P) \subseteq L(Q)$. We now show $L(P) \subset L(Q) \Rightarrow P \sqsubset Q$. By Lemma 4.1, $L(P) \subset L(Q)$ implies that $P \sqsubseteq Q$. If furthermore $Q \sqsubseteq P$ then $L(P) = L(Q)$, contradicting $L(P) \subset L(Q)$. Hence we have $P \sqsubset Q$. Part (b) follows from (a) and (c). ∎

Note also that $L(P) \subseteq L(Q) \Rightarrow P \sqsubseteq Q$ is sufficient to show Lemma 4.2(c). Due to the identity between the non-erasing and erasing case in Lemma 4.1, Lemma 4.2 extends immediately to the erasing case, with L replaced by L_ϵ, $\mathcal{L}$ replaced by $\mathcal{L}_\epsilon$, and $c(P)$ redefined as $\{\lfloor L_\epsilon(p) \rfloor \mid p \in P\}$.

Theorem 4.1 *Suppose card*$(\Sigma) \geq 2$. *Given* $\mathcal{P} \in \{\boldsymbol{Psub}, \boldsymbol{Psubseq}\}$, *for every finite non-empty* $S \subseteq \Sigma^*$ *and* $k \in N^+$,
(1) $\mathcal{FC}(\mathcal{L}(\mathcal{P}), S, k, \lceil S \rceil) \subseteq \mathcal{MMG}(\mathcal{L}(\mathcal{P}), S, k) = \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$;
(2) $\mathcal{FC}(\mathcal{L}_\epsilon(\mathcal{P}), S, k, \lceil S \rceil) \subseteq \mathcal{MMG}(\mathcal{L}_\epsilon(\mathcal{P}), S, k) = \mathcal{MINL}(\mathcal{L}_\epsilon(\mathcal{P}), S, k)$.

Proof. We first note that given any S, every pattern in a class $\mathcal{P}'$ of patterns $p \in \mathcal{P}$ where $L(p) \cap S \neq \emptyset$ must generate a language with every element longer than $\lceil S \rceil$. Hence for any sets P, $Q \subseteq \mathcal{P}'$, $P \sqsubset Q$ (by Lemma 4.2(a))$\Rightarrow L(Q)^{\leq c(Q)} - L(P)^{\leq c(Q)} \neq \emptyset \Rightarrow L(Q)^{\leq \lceil S \rceil} - L(P)^{\leq \lceil S \rceil} \neq \emptyset$. Then by Proposition 3.4, $\mathcal{FC}(\mathcal{L}(\mathcal{P}), S, k, \lceil S \rceil) \subseteq \mathcal{MMG}(\mathcal{L}(\mathcal{P}), S, k)$. That $\mathcal{MMG}(\mathcal{L}(\mathcal{P}), S, k) = \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S, k)$ is from Lemma 4.2(c).

The erasing case is shown identically. ∎

Note that Theorem 4.1 continues to hold in the case of unbounded unions.

In general, $\mathcal{MINL}(\mathcal{L}, S, k) - \mathcal{FC}(\mathcal{L}, S, k, \lceil S \rceil)$ is non-empty. Examples such as the following can be easily obtained.

Example 2 *Let* $\Sigma = \{a, b\}$. *(For* ***Psub****) Let* $P = \{x_1aaax_2, x_1bbbx_2\}$, $Q = \{x_1abax_2, x_1bbax_2\}$, *and* $S = L(P) \cap L(Q) \cap (\Sigma^*)^{\leq 9}$. *It can be verified that both* P, $Q \in \mathcal{MINL}(\mathcal{L}(\boldsymbol{Psub}), S, 2)$, *but* $P \notin \mathcal{FC}(\mathcal{L}(\boldsymbol{Psub}), S, 2, \lceil S \rceil)$.

(For $\boldsymbol{Psubseq}$*) Let* $P = \{x_1ax_2ax_3ax_4, x_1bx_2bx_3bx_4\}$*,* $Q = \{x_1ax_2bx_3ax_4, x_1bx_2bx_3ax_4\}$*, and* $S = L(P) \cap L(Q) \cap (\Sigma^*)^{\leq 8}$*. It can be verified that both* P*,* $Q \in \mathcal{MINL}(\mathcal{L}(\boldsymbol{Psubseq}), S, 2)$*, but* $P \notin \mathcal{FC}(\mathcal{L}(\boldsymbol{Psubseq}), S, 2, \lceil S \rceil)$*.*

Nevertheless, on some very specific examples, for **Psub** and **Psubseq**, $\mathcal{FC}$ is no different from the $\mathcal{MINL}$ or $\mathcal{MMG}$, as the following result shows. (Note that the following holds similarly for the erasing case.)

Proposition 4.2 *Suppose* $card(\Sigma) \geq 2$ *and* $\mathcal{P} \in \{\boldsymbol{Psub}, \boldsymbol{Psubseq}\}$*. For each finite set* $P \subseteq \mathcal{P}$*, there exists a finite non-empty set* $S_P \subseteq \Sigma^*$ *such that for all* $k \geq card(P)$ *and all finite* S' *where* $S_P \subseteq S' \subseteq L(P)$*,* $\mathcal{FC}(\mathcal{L}(\mathcal{P}), S', k) = \mathcal{MINL}(\mathcal{L}(\mathcal{P}), S', k) = \mathcal{MMG}(\mathcal{L}(\mathcal{P}), S', k)$*.*

Proof. Let θ be an arbitrary substitution that transforms each variable to a letter in Σ. Given set $P \subseteq \mathcal{P}$, we first note from the proof of Lemma 4.1 that for each $p \in P$, $p\theta \in L(Q)$ implies that $p \preceq q$ for some $q \in Q$. Hence, let $S_P = \{p\theta \mid p \in P\}$. Now for any set $Q \subseteq \mathcal{P}$, $S_P \subseteq L(Q)$ implies that for each $p \in P$, there exists some $q \in Q$ such that $p \preceq q$. Hence for any superset S of S_P, $S \subseteq L(Q) \Rightarrow P \sqsubseteq Q$. Now if $P \sqsubset Q$, then $Q \notin \mathcal{MMG}(\mathcal{L}(\mathcal{P}), S, k)$. Hence $\mathcal{MMG}(\mathcal{L}(\mathcal{P}), S, k) = \{P' \mid P' \sqsubseteq P \ \wedge P \sqsubseteq P'\}$. That each $P' \in \mathcal{MMG}(\mathcal{L}(\mathcal{P}), S, k)$ generates the same language then gives us the result through Theorem 4.1. ∎

4.2 Patterns with at most 1 variable occurrence

In the classes discussed so far, for any class $\mathcal{L}$, set S of examples and $k \in N^+$ any solution for $\mathbf{MMG}(\mathcal{L}, S, k)$ is at the same time a solution for $\mathbf{MINL}(\mathcal{L}, S, k)$. We now show a class of pattern languages where it is possible to specify a length ℓ such that $\mathbf{FC}(\mathcal{L}, S, k, \ell)$ solutions are also solutions for $\mathbf{MINL}(\mathcal{L}, S, k)$, but solutions from $\mathbf{MMG}(\mathcal{L}, S, k)$ are not necessarily solutions of $\mathbf{MINL}(\mathcal{L}, S, k)$.

Let $\mathbf{P}_{\leq 1}$ be the class of patterns where only one variable occurrence is allowed in each pattern. It is easy to see that the condition $(\forall P, Q \in \mathbf{P}_{\leq 1})[L(P) \subseteq L(Q) \Rightarrow P \sqsubseteq Q]$ (for Proposition 3.3) does not hold when $k \geq card(\Sigma)$. The following example shows a set $P \subseteq \mathbf{P}_{\leq 1}$ such that $P \in \mathbf{MMG}(\mathbf{P}_{\leq 1}, S, k) - \mathbf{MINL}(\mathbf{P}_{\leq 1}, S, k)$.

Example 3 *Let* $\{a, b, c\} \subseteq \Sigma$*. Let* $k = 3$ *and let* $S = \{aa, ba, ab, bb, ac, bc\}$*. Let* $P = \{x_1a, x_1b, x_1c\}$ *and* $Q = \{ax_1, bx_1\}$*. It is easy to verify that no* $P' \sqsubset P$ *of up to* k *patterns from* $\boldsymbol{P}_{\leq 1}$ *can have* $S \subseteq L(P')$*, and hence* P *is minimal with respect to the* $\sqsubseteq$ *relation. However,* $L(Q) \subset L(P)$*.*

In the following, for any rational number r, $\lfloor r \rfloor$ denotes the largest natural number smaller than r.

Theorem 4.3 *For any finite* $P \subseteq \boldsymbol{P}_{\leq 1}$, $S_t(P)$ *where* $t = 2\lfloor \frac{k-1}{card(\Sigma)-1} \rfloor + 2$ *is a characteristic set for* $L(P)$ *within* $\mathcal{L}(\boldsymbol{P}_{\leq 1})^k$.

PROOF. Let variables $V = \{x\}$ and constants $\Sigma = \{a_1, \ldots, a_n\}$. We define two sets of substitutions,

(1) $\Theta_1 = \{\theta \mid x\theta = a_i x a_j, 1 \leq i, j \leq n\}$, and

(2) $\Theta_2 = \{\theta \mid x\theta \in (\Sigma^*)^{\leq 2}\}$.

For any set of patterns P and for any set of substitutions Θ, we let $\Theta \cdot P$ denote the set of patterns $\{p\theta \mid p \in P \wedge \theta \in \Theta\}$. Furthermore, for $l \in N$, we let $(\Theta)^l \cdot P$ denote the set of patterns obtainable from l applications of substitutions from Θ on patterns in P, that is, for example, $(\Theta)^2 = \Theta \cdot (\Theta \cdot P)$. Note that $S_{2l}(P) = \bigcup_{0 \leq i \leq l-1} \Theta_2 \cdot ((\Theta_1)^i \cdot P)$.

Claim 4.3 *For any* $P \subseteq \boldsymbol{P}_{\leq 1}$, *and finite* $Q \subseteq \boldsymbol{P}_{\leq 1}$, $L(P) \subseteq L(Q)$ *if and only if there exists* $l \in N$ *such that (I)* $(\Theta_1)^l \cdot P \sqsubseteq Q$, *and (II)* $\bigcup_{0 \leq i \leq l-1} \Theta_2 \cdot ((\Theta_1)^i \cdot P) \subseteq L(Q)$.

Proof of Claim. ($\Leftarrow$) is not difficult to see. We show ($\Rightarrow$). We claim that (I) must hold for some $l \in N$. Let $Q = \{u_i x v_i \mid 1 \leq i \leq card(Q) \wedge u_i, v_i \in \Sigma^*\}$ be, and let m be the longest length of such u_i and v_i. Now consider $P' = (\Theta_1)^{m+1} \cdot P$. Clearly, every pattern in P' is of the form uxv where $u, v \in \Sigma^*$, $|u| > m$ and $|v| > m$. Suppose $P' \not\sqsubseteq Q$, then there exists $uxv \in P'$, u, $v \in \Sigma^*$ such that for every $u_i x v_i \in Q$, u does not begin with u_i or v does not end in v_i, and hence $ua_1v \notin L(Q) \Rightarrow L(P) \not\subseteq L(Q)$, a contradiction. Now note that if (II) fails for any $l \in N$ then $L(P) \not\subseteq L(Q)$ also. Claim follows from these observations. □

The following computation uses the principle in the Claim to decide, given $p \in \mathbf{P}_{\leq 1}$ and finite $Q \subseteq \mathbf{P}_{\leq 1}$, if $L(p) \subseteq L(Q)$. At each stage $l \in N$, the computation is as follows:

stage 0:
  If $p \preceq q$ for some $q \in Q$, output true.
  Otherwise let $P_1 = \{p\}$ and $Q_1 = Q$.
  Enter stage 1.

stage l:
  If $\Theta_2 \cdot P_l \not\subseteq L(Q_l)$, output false.
  If $\Theta_1 \cdot P_l \sqsubseteq Q_l$, output true.
  Otherwise let
    $P_{l+1} \subseteq \Theta_1 \cdot P_l$ be the set of patterns not refinable from any q in Q_l,
    $Q_{l+1} \subseteq Q_l$ be the set of patterns $q \in Q$ where either
      1. q contains a variable, but refines to nothing in $\Theta_1 \cdot P_l$, or
      2. q contains no variable, and $|q| \geq |p| + 2l$.
  Enter stage $l + 1$.

Note that:

(A) At any stage l, patterns in $Q_l - Q_{l+1}$ do not refine to any pattern in $\Theta_1 \cdot P_{l'}$ and $\Theta_2 \cdot P_{l'}$ for any $l' > l$.

Hence they are excluded from evaluation at subsequent stages. From the computation, we also see that at any stage l, if $Q_l = \emptyset$, then either

(1) $P_l = \emptyset$, in which case $(\Theta_1)^{l-1} \cdot \{p\} \sqsubseteq Q$ and (by Claim 4.3) $L(p) \subseteq L(Q)$, or

(2) $\Theta_2 \cdot P_l \not\subseteq L(Q)$, hence $S_{2l}(p) \not\subseteq L(Q)$.

Hence we have:

(B) $Q_l = \emptyset \Rightarrow [S_{2l}(p) \subseteq L(Q) \Leftrightarrow L(p) \subseteq L(Q)]$.

We will show that for any $l > \lfloor (k-1)/(n-1) \rfloor$, either $Q_l = \emptyset$ or $S_{2l}(p) \not\subseteq L(Q)$, which together with (B) completes the proof.

For each stage l, let $Q'_l = Q_l - Q_{l+1}$, that is, Q'_l are the patterns removed from Q_l at stage l. We assume each $q \in Q$ which contains a variable refine to some $p' \in \Theta_1 \cdot P_m$ for some $m \in N$, since otherwise $L(q) \cap L(p) = \emptyset$, in which case q can be discarded from consideration. That is,

(C) Each $q \in Q$, with or without a variable occurrence, is in Q'_m for some $m \in N$.

We also note the following:

(D) $card(Q) \leq k$.

(E) $card(\Theta_1 \cdot P_l) = n^2 card(P_l)$.

(F) Each $q \in Q'_m$ refines to at most n patterns in $\Theta_1 \cdot P_m$, and at most $n+1$ patterns in $\Theta_2 \cdot P_m$.

By (C)–(F), we can see that there is an upper bound to the number of stages that can be reached for any set Q before $card(Q_l) \leq n$. Let Q be such that this number of stages reached is maximized, and let ℓ be the stage when $card(Q_\ell) \leq n$. We note that such ℓ is maximized when $card(Q) = k$ and $card(P_m)$ is minimized for each $m < \ell$. Hence for each m, $card(P_m) = 1$ and $card(\Theta \cdot P_m) = n^2$. By (F), this gives us that $card(\Theta_1 \cdot P_m) = n^2 \leq n \cdot card(Q'_m) + 1 \Rightarrow card(Q'_m) \geq n - 1$. Hence at least $(\ell - 1)(n-1)$ patterns from Q are in some Q'_m in the stages $m < \ell$, and so there are $k - (\ell-1)(n-1)$ patterns in Q_ℓ. We consider the following two cases:

(Case I) $card(Q_\ell) = n$. Note that $k - (\ell-1)(n-1) = n \Rightarrow \ell = (k-1)/(n-1)$. In this case $Q_{\ell+1} = \emptyset$.

(Case II) $card(Q_\ell) < n$. Note that $k - (\ell-1)(n-1) = m$ for some $m < n \Rightarrow \ell = (k+n-1-m)/(n-1) \Rightarrow (k-1)/(n-1) \leq \ell \leq (k-1)/(n-1)+1$. Since $card(\Theta_2 \cdot P_\ell) = n^2+n$, by Condition (F), this case has $\Theta_2 \cdot P_\ell \not\subseteq L(Q_\ell)$.

Hence as claimed, for $l > \lfloor (k-1)/(n-1) \rfloor$, either $Q_l = \emptyset$ or $S_{2l}(p) \not\subseteq L(Q)$. In the case that $Q_l = \emptyset$, as argued, $S_{2l}(p) \subseteq L(Q) \Leftrightarrow L(p) \subseteq L(Q)$. Hence $S_{2l}(p) \subseteq L(Q) \Leftrightarrow L(p) \subseteq L(Q)$ holds in general when $l > \lfloor (k-1)/(n-1) \rfloor$, and we are done. The case for any set of patterns $P \subseteq \mathbf{P}_{\leq 1}$ follows immediately. ∎

The erasing case has an identical result.

Theorem 4.4 *For any finite* $P \subseteq \boldsymbol{P}_{\leq 1}$*,* $S_t(P)$ *where* $t = 2\lfloor \frac{k-1}{card(\Sigma)-1} \rfloor + 2$ *is a characteristic set for* $L_\epsilon(P)$ *within* $\mathcal{L}_\epsilon(\boldsymbol{P}_{\leq 1})^k$*.*

Proof. The proof is the same as Theorem 4.3, and it can be verified that each of the conditions remains valid, with the following minor differences.

1. For the computation at stage l, in constructing Q_{l+1}, the condition for the patterns without variables is now $|q| \geq |p| + 2l - 1$.

2. Condition (F) becomes: *Each* $q \in Q'_m$ *refines to at most* n *patterns in* $\Theta_1 \cdot P_m$*, and at most* $n+2$ *patterns in* $\Theta_2 \cdot P_m$. This does not change the result in (Case II). (Note that $card(\Theta_2 \cdot P_m) = n^2 + n + 1$ for the present case.) ∎

Noting that Theorem 3.5 requires only the characteristic sets to be within a subset of the target class which intersect non-emptily with the examples, we obtain the following result.

Theorem 4.5 *Given any finite non-empty set* $S \subseteq \Sigma^*$*. Let* $\boldsymbol{Q}_{\leq 1}$ *be the class of patterns* $p \in \boldsymbol{P}_{\leq 1}$ *where* $L(p) \cap S \neq \emptyset$*. For any finite* $P \subseteq \boldsymbol{Q}_{\leq 1}$*,*
(1) $S_t(P)$ *where* $t = 2\min(\{\lceil S \rceil, \lfloor \frac{k-1}{card(\Sigma)-1} \rfloor\}) + 2$ *is a characteristic set for* $L(P)$ *within* $\mathcal{L}(\boldsymbol{Q}_{\leq 1})^k$*.*
(2) $S_t(P)$ *where* $t = 2\min(\{\lceil S \rceil + 1, \lfloor \frac{k-1}{card(\Sigma)-1} \rfloor\}) + 2$ *is a characteristic set for* $L_\epsilon(P)$ *within* $\mathcal{L}_\epsilon(\boldsymbol{Q}_{\leq 1})^k$*.*

Proof. We show Part (1) first. We follow the same proof in Theorem 4.3. Recall that each $q \in Q$ is in Q'_m for some $m \in N$, otherwise $L(q) \cap L(p) = \emptyset$. We furthermore note that, $q \in Q'_m \Rightarrow |q| \geq m$. That is, any pattern q of length up to n which is not in some Q'_m where $m \leq n$ has $L(q) \cap L(p) = \emptyset$. Since every pattern in $\mathbf{Q}_{\leq 1}$ is of length at most $\lceil S \rceil$, each $q \in \mathbf{Q}_{\leq 1}$ which intersects non-emptily with $L(p)$ must be in some Q'_m where $m \leq \lceil S \rceil$. Discarding the patterns in Q which does not intersect $L(p)$, we have that when $l > \lceil S \rceil$, $Q_l = \emptyset$.

For Part (2), the same argument applies, with the difference that every pattern in $\mathbf{Q}_{\leq 1}$ is of length at most $\lceil S \rceil + 1$ instead of $\lceil S \rceil$. ∎

Corollary 8 *For every finite non-empty* $S \subseteq \Sigma^*$ *and* $k \in N^+$*,*
(1) $\mathcal{FC}(\mathcal{L}(\boldsymbol{P}_{\leq 1}), S, k, \lceil S \rceil + 2\min(\{\lceil S \rceil, \lfloor \frac{k-1}{card(\Sigma)-1} \rfloor\}) + 2) \subseteq \mathcal{MINL}(\mathcal{L}(\boldsymbol{P}_{\leq 1}), S, k)$*.*
(2) $\mathcal{FC}(\mathcal{L}_\epsilon(\boldsymbol{P}_{\leq 1}), S, k, \lceil S \rceil + 2\min(\{\lceil S \rceil + 1, \lfloor \frac{k-1}{card(\Sigma)-1} \rfloor\}) + 2) \subseteq \mathcal{MINL}(\mathcal{L}_\epsilon(\boldsymbol{P}_{\leq 1}), S, k)$*.*

4.3 Tree pattern languages

We now turn our attention to the tree pattern languages. We study the *size* of a tree pattern rather than its *length*, since it is more natural to speak of the sizes of trees rather than lengths.

4.3.1 Existence of characteristic sets for unions of languages in $\mathcal{L}(TP)$

The following result, that shows that a characteristic set exists for every $L \in \mathcal{L}(TP)^k$ within $\mathcal{L}(TP)^k$, follows from Theorem 2.3 and Theorem 2.4.

Theorem 4.6 ([83] and [65, 39]) *For every $k \in N^+$, every language $L \in \mathcal{L}(TP)^k$ has a characteristic set within $\mathcal{L}(TP)^k$.*

By Theorem 4.6 and Theorem 3.2, the class $\mathcal{L}(TP)^k$ can be inferred by a learner that bases its conjecture completely on a computation of **FC**.

Regrettably, Theorem 4.6 does not hold in the case of unbounded unions. To see this, let p be a tree pattern with at least one variable, and let S be any finite subset of $L(p)$. Now S itself is a union of $card(S)$ tree pattern languages (each of a single ground tree pattern), and hence not a characteristic set for $L(p)$. One might ask if the same problem would persist if we remove the ground tree pattern languages from the class, that is, consider instead the class $\mathcal{L}(TP - TP_\Gamma)$. However, a simple analysis shows that if Γ contains any function of arity ≥ 2, then the case is no different from that which allows ground tree pattern languages. For example, let f be a function of arity 2, let $p = f(x_1, x_2)$ and let $TP' \subseteq TP$ be the class of all trees patterns of the form $f(x_1, t)$ where $t \in TP_\Gamma$. Then every language $L \in \mathcal{L}(TP')^*$ has $L \subset L(p)$, and for every $S \subseteq L(p)$, there is a language in $\mathcal{L}(TP')^*$ which includes S. (In fact, this shows that $\mathcal{L}(TP - TP_\Gamma)^*$ is not inferable from positive examples.) Conversely, if every function in Γ is of arity ≤ 1, then trivial characteristic sets exist for languages in $\mathcal{L}(TP - TP_\Gamma)^*$.

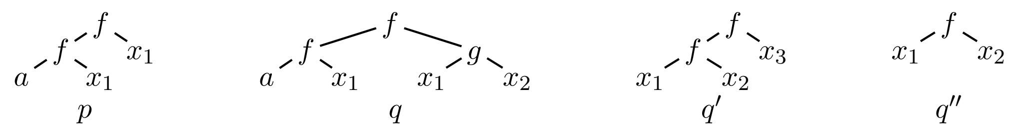
Lemma 4.4 *If Γ consists only of functions of arities ≤ 1, then for any finite set $P \subseteq TP - TP_\Gamma$, $S_1(P)$ is a characteristic set for $L(P)$ within $\mathcal{L}(TP - TP_\Gamma)^*$.*

Proposition 4.7 *A characteristic set exists for every language in $\mathcal{L}(TP - TP_\Gamma)^*$ if and only if every function in Γ is of arity ≤ 1.*

We now consider characteristic sets for the bounded unions of languages in $\mathcal{L}(TP)$.

4.3.2 Characteristic sets within unions of less than $card(\Gamma)$ languages

We make some preliminary definitions. For any $p, q \in TP$, q is said to *dutifully extend* p just in case if we take each of the variable occurrences in p to be mutually independent, then p refines to q. More intuitively, q dutifully extends p just in case if a function c occurs at position ε in p, c also occurs at position ε in q. In the example below, q dutifully extends p but q' and q'' do not. Note that $q \preceq p \Rightarrow q$ dutifully extends p.



We now attempt to find a most general refinement of a tree pattern q that dutifully extends a tree pattern p. Consider the following method, the output of which we denote by $\text{mge}\{q \supseteq p\}$.

dutifully_extend_p

1. Set $\text{mge}\{q \supseteq p\}$ to q.
2. If there is any function in $\text{mge}\{q \supseteq p\}$ that occurs at a position where some other function occurs in p, output "failure".
3. While there exists a function node, v say, in p where no corresponding function occurs at $\text{pos}(v)$ in $\text{mge}\{q \supseteq p\}$.
   3.1 Find a variable node u in $\text{mge}\{q \supseteq p\}$ at some prefixes of $\text{pos}(v)$. If no such variable node exists in $\text{mge}\{q \supseteq p\}$, output "failure".
   3.2 Let t be the tree pattern at $\text{pos}(u)$ in p. Replace each variable occurrence in t with a distinct variable not in $\text{mge}\{q \supseteq p\}$ and let t' denote the resultant tree pattern.
   3.3 Let x be the root symbol of u. Substitute each occurrence of x in $\text{mge}\{q \supseteq p\}$ with t'. If this results in any function to occur in $\text{mge}\{q \supseteq p\}$ at a position where some other function occurs in p, output "failure".
4. Output $\text{mge}\{q \supseteq p\}$.

It is clear that (1) $\text{mge}\{q \supseteq p\} \preceq q$, and (2) $\text{mge}\{q \supseteq p\}$ dutifully extends p. Intuitively, the method preserves the generality of $\text{mge}\{q \supseteq p\}$, by making only the necessary refinements to q to dutifully extend p (in steps 3.2$\sim$3.3). From this, we can verify that $L(p) \cap L(q) \subseteq \text{mge}\{q \supseteq p\}$. It can also be shown that (1) there exists exactly one such $\text{mge}\{q \supseteq p\}$, unique to renaming of variables, and (2) if the method outputs "failure", then no refinement of q dutifully extends p. We skip the details.

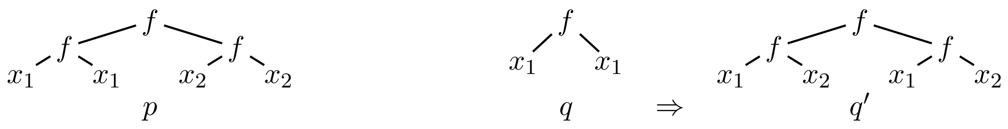
We now find a sufficient size to the elements of the characteristic sets for the unions of the tree pattern languages, when the number of languages allowed in a union is less than the alphabet size. This restriction is the same as that in [8].

Lemma 4.5 *Let $p \in TP$ and set $Q \subseteq TP$ be where $card(Q) < card(\Gamma)$. If $\{p\} \not\sqsubseteq Q$ then there exists a ground tree pattern $t \in L(p) - L(Q)$ of size $(1+\alpha)\|p\|$, where α is the largest arity of the functions in Γ.*

Proof. Without loss of generality we assume that for each $q \in Q$, either q dutifully extends p, or there exists some $q' \preceq q$ that dutifully extends p, since otherwise $L(p) \cap L(q) = \emptyset$. Let $Q' = \{\text{mge}\{q \supseteq p\} \mid q \in Q\}$. Note that $card(Q') < k$ and $L(p) \cap L(Q) = L(p) \cap L(Q')$. We now show that any such $q' \in Q'$ must be of one of two possible forms. Since q' dutifully extends p, $\|q'\| \geq \|p\|$. Now since for every $q \in Q$, $p \not\preceq q$, one of the following cases must apply for $q' \in Q'$:

(C1) A variable x occurs in p at a position where a function c occurs in q'. In this case we say that x and c witnesses that q' fulfills (C1).

(C2) A variable x occurs in p at a position where a variable y occurs in q', however y also occurs in q' at a position where a variable $x' \neq x$ occurs in p. In this case we say that x witnesses that q' fulfills (C2). In the following example, the left-most occurrence of x_1 in p witnesses that q' fulfills (C2).



This is how we construct a ground tree pattern that cannot be refined from the tree patterns that fulfill (C1). Let variable x in p (with some $c \in \Gamma$) witness that the tree patterns $Q^x \subseteq Q'$ fulfill (C1), and let $\Gamma^x \subseteq \Gamma$ be such that for each $q' \in Q^x$, there exists $c \in \Gamma^x$ that witnesses that q' fulfills (C1) as witnessed by x and c. Clearly, no $q' \in Q^x$ refines to the tree pattern obtained by replacing every occurrence of x in p with a symbol in $\Gamma - \Gamma^x$. For the tree patterns fulfilling (C2) we do the following. Let X be the set of the variables occurring in p such that for each q' that fulfills (C2), there is a variable in X that witnesses it. If we replace each of the variables in X with a distinct function, and all other variables that occur in p but is not in X with yet another distinct function, then all the q' fulfilling (C2) cannot refine to the resultant tree pattern. The following algorithm $find\_t$ does exactly these.

We first construct a basic set of ground tree patterns of at most height 1, for use in $find\_t$.

Definition 4.1 **(Λ_a)** *Let a be an arbitrary constant in Γ. For each function $f \in \Gamma$, let $f_a = f$ if f is a constant, otherwise f_a is tree pattern with root symbol f and the constant a as children. We let $\Lambda_a = \{f_a \mid f \in \Gamma\}$.*

It is clear that $card(\Lambda_a) = card(\Gamma)$.

$find\_t$
1. Let $C = \Gamma$.
2. For each $q' \in Q'$, find a variable x in p that witnesses that q' fulfills (C1).
   Let X be the set of all such x found.
3. While Q' is not empty,
   3.1 For each $x \in X$, let Q^x be the set of all $q' \in Q'$ with a function occurring at some position where x occurs at in p.
   3.2 For each $q' \in Q^x$ find one function c which occurs at a position where x occurs at in p. Let C' be the set of all such c found.
   3.3 Replace x with an element $f_a \in \Lambda_a$ where $f \in C - C'$ (this is possible because $card(C') \leq card(Q') < card(C)$), and remove f from C.
   3.4 Remove Q^x from Q'.
4. Let Q'' be the set of all q' that fulfills (C2) but not (C1).
5. For each $q' \in Q''$, find a variable x in p that witnesses that q' fulfills (C2).

Let X be the set of all such x found.
5.1 If $x \in X$ has not already been replaced, replace x with an element $f_a \in \Lambda_a$ where $f \in C$, and remove f from C.
6. Replace all remaining variables in p with an element $f_a \in \Lambda_a$ where $f \in C$. (Note that at most $card(Q') < card(\Gamma)$ elements have been removed from C prior to this step.)
7. Let t be the resultant pattern. Output t.

Since there are at most $\|p\|$ variable occurrences in p, the total number of terms added to p in $find\_t$ is at most $\alpha\|p\|$. Since $t \not\preceq q'$ for any $q' \in Q'$, $t \notin L(q')$ for each $q' \in Q'$. Clearly, $t \in L(p)$ by the substitutions done in $find\_t$, but since $t \notin L(Q')$, $t \notin L(Q)$. ∎

Theorem 4.8 *Let α be the largest arity of the functions in Γ and let $k < card(\Gamma)$. For every sets of less than k tree patterns $P, Q \subseteq TP$, the following three statements are equivalent: (i) $S_\alpha(P) \subseteq L(Q)$, (ii) $P \sqsubseteq Q$, (iii) $L(P) \subseteq L(Q)$.*

PROOF. (ii)⇒(iii) by definition. (iii)⇒(i) is immediate. (i)⇒(ii) by Lemma 4.5. ∎

The equivalence of (i) and (iii) says that for any set of less than k tree patterns $P \subseteq TP$, $S_\alpha(P)$ is a characteristic set of $L(P)$ within $\mathcal{L}(TP)^k$.

Theorem 4.9 *Let α be the largest arity of the functions in Γ. For every finite, non-empty set $S \subseteq TP_\Gamma$ and $k < card(\Gamma)$, $\mathcal{FC}(\mathcal{L}(TP), S, k, \max(\{(1+\alpha)\|t\| \mid t \in S\})) \subseteq \mathcal{MINL}(\mathcal{L}(TP), S, k)$.*

4.3.3 Unions of up to a bound independent of $card(\Gamma)$

We now remove the alphabet size restriction placed on the number of unions allowed. We obtain, for any tree pattern language L, a sufficient size ℓ for constructing a characteristic set of L using only elements of up to ℓ, for within one of the following two classes: (1) $\mathcal{L}(TP)^k$, and (2) $\mathcal{L}(TP(S))^k$ where for any set $S \subseteq TP_\Gamma$, $TP(S)$ is the class of tree patterns $p \in TP$ where $L(p) \cap S \neq \emptyset$. For (2), we add the requirement that there is at least one function of arity ≥ 2 in Γ.

4.3.3.1 Characteristic sets within $\mathcal{L}(TP)^k$

We first obtain some preliminary results. Two tree patterns p and q are said to be *structurally equivalent* just in case p dutifully extends q and q dutifully extends p.

A tree pattern is *simple* just in case it is either a constant, or has only variables as children (hence the height of a simple tree is at most one). For $p \in TP$, let $\Theta(p)$ denote the set of all the canonical tree patterns obtainable by replacing each variable in p with a simple tree pattern such that

- (A) every distinct variable appearing in p is substituted with a simple tree of a different root symbol (condition needed for Lemma 4.7), and
- (B) at most $card(\Gamma)$ distinct variables appear in each tree pattern.

For a set of tree patterns P, let $\Theta(P) = \bigcup_{p \in P} \Theta(p)$. For $n \in N^+$, let $\Theta^n(p) = \Theta^{n-1}(\Theta(p))$. Note that due to condition (A), every tree pattern $p' \in \Theta^{n-1}(p)$ has $\Theta(p') \neq \emptyset$, and hence:

- (C) For each $p' \in \Theta(p)$, either $\Theta(p') = \emptyset$ (when $p' \in TP_\Gamma$), or $card(\Theta(p')) \geq 2$.
- (D) For each $p' \in \Theta(p) - TP_\Gamma$, there is an element in $\Theta(p')$ obtained by substituting variables in p' with only elements in Λ_a. (Λ_a is as defined in the proof of Theorem 4.5).

Finally, let $Lang(p) = TP_\Gamma \cap \bigcup_{n \in N^+} \Theta^n(p)$.

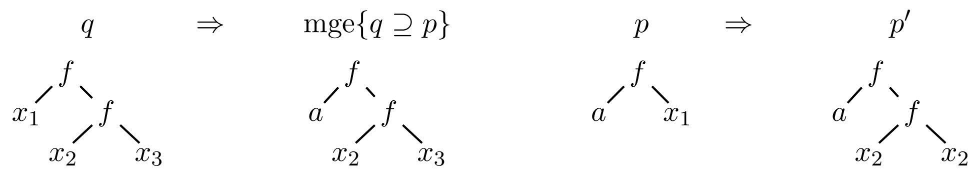
Lemma 4.6 *For all $p \in TP$, $L(p) = Lang(p)$.*

Lemma 4.7 *For any two structurally equivalent tree patterns $p, q \in TP$, $p \not\preceq q \Rightarrow Lang(p) \cap L(q) = \emptyset$.*

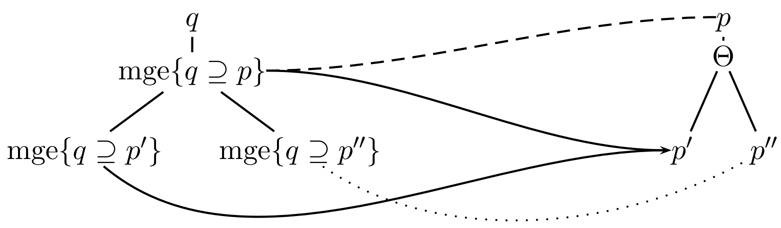
PROOF. Since p and q are structurally equivalent, that $p \not\preceq q$ implies that there is a variable x in p that occurs at a position a variable x' occurs at in q, however x' also occurs in q where a variable $x'' \neq x$ occurs in p. From this it is clear that $Lang(p) \cap L(q) = \emptyset$. ∎

Lemma 4.8 *For any $p, q \in TP$, if $p' \preceq q$ for some $p' \in \Theta(p)$, then for all $p'' \in \Theta(p)$, if $mge\{q \supseteq p''\}$ exists, then it is structurally equivalent to p''.*

PROOF. Note first that for any $p'' \in \Theta(p)$, $\mathrm{mge}\{q \supseteq p''\} \preceq \mathrm{mge}\{q \supseteq p\}$. Since $p' \preceq \mathrm{mge}\{q \supseteq p\}$, we can see that all the nodes in p and $\mathrm{mge}\{q \supseteq p\}$ are almost identical, except that some variables might occur in p at the positions where some simple tree patterns occur in $\mathrm{mge}\{q \supseteq p\}$. For example,



Now for any $p'' \in \Theta(p)$, if $\mathrm{mge}\{q \supseteq p''\}$ exists, it must also be obtained from $\mathrm{mge}\{q \supseteq p\}$ via a substitution which substitutes variables in $\mathrm{mge}\{q \supseteq p\}$ to their corresponding simple tree patterns in p''.



Hence a function is in p'' if and only if it is a function in $\text{mge}\{q \supseteq p''\}$. That is, p'' and $\text{mge}\{q \supseteq p''\}$ are structurally equivalent. ∎

Lemma 4.9 *For any $p, q \in TP$, if $p' \preceq q$ for some $p' \in \Theta(p)$ but $p'' \not\preceq q$ where $p'' \in \Theta(p)$, then $Lang(p'') \cap L(q) = \emptyset$.*

PROOF. By Lemma 4.7 and 4.8 we have $Lang(p'') \cap L(\text{mge}\{q \supseteq p''\}) = \emptyset$. Since every $t \in Lang(p'') \cap L(q)$ can be refined from $\text{mge}\{q \supseteq p''\}$ it follows that $Lang(p'') \cap L(q) = \emptyset$. ∎

Theorem 4.10 *Given $p \in TP$ and a set of up to k tree patterns $Q \subseteq TP$. If $\{p\} \not\sqsubseteq Q$ then there exists a ground tree pattern $t \in H_{k+1}(p) - L(Q)$.*

PROOF. Let $\Xi(p)$ be the set of all tree patterns $p' \preceq p$ where:

1. variables in p are substituted only with variables, and
2. at most $card(\Gamma)$ distinct variables appear in each tree pattern in $\Xi(p)$.

It is clear that $L(p) = \bigcup_{p' \in \Xi(p)} Lang(p')$. Now consider the following computation which, given a tree pattern p and a set of tree patterns Q, decides if $L(p) \subseteq L(Q)$.

stage 0:
  Let $G^0_{0,0} = \Xi(p)$.
  If $G^0_{0,0} \sqsubseteq Q$, output true.
  Goto stage 1.

stage l:
  Let $n = 0$.
  For each $G^{l-1}_{i,j}$ defined at stage $l-1$,
    For each $p' \in G^{l-1}_{i,j}$,
      Let $G^l_{n,i} = \Theta(p')$. (The index i identifies the tree pattern from
        which p' is obtained, while the index n is a new index, unique
        for each p', assigned to $\Theta(p')$.)
      If $TP_\Gamma \cap G^l_{n,i} \not\sqsubseteq Q$, output false.

Increment n by 1.
If every $G^l_{n,i}$ defined in this stage has $G^l_{n,i} \sqsubseteq Q$, output true.
If there exists $t \in TP_\Gamma$ in any $G^l_{n,i}$ defined where $\{t\} \not\sqsubseteq Q$, output false.
Goto stage $l+1$.

It is easy to see that if at any stage l, the computation outputs true, then $L(p) \subseteq L(Q)$. We now show that if at the stage k, there is a set $G^k_{i,j}$ defined where $G^k_{i,j} \not\sqsubseteq Q$, then $L(p) \not\subseteq L(Q)$. Let $P_0 = G^0_{0,0}, P_1 = G^1_{i_1,0}, \ldots, P_k = G^k_{i_k,i_{k-1}} = G^k_{i,j}$ be the sequence of sets of patterns at the stages $0, 1, \ldots, k$ leading to $G^k_{i,j}$. Also, let p_0 be the tree pattern in P_0 from which P_1 is obtained, p_1 the tree pattern in P_1 from which P_2 is obtained, and so on. Now for each $i \in N$, $1 \leq i \leq k$,

(E) $card(P_i) \geq 2$. (by (C))

(F) $TP_\Gamma \cap P_i \neq \emptyset$. (by (D))

(G) $\{p_i\} \not\sqsubseteq Q$.

(H) $Lang(p_k) \subseteq Lang(p_{k-1}) \subseteq \ldots \subseteq Lang(p_1)$.

By (E) and (F), at each stage $i \leq k$, there exists a tree pattern $q_i \in Q$ such that $t \in TP_\Gamma \cap P_i$ is refinable from q_i, but $p_i \not\preceq q_i$, and hence by Lemma 4.9, $Lang(p_i) \cap L(q_i) = \emptyset$. Hence each $q \in Q$ refines to some $t \in TP_\Gamma \cap P_i$ at some stage $i \leq k$, and so by (G) and (H), $Lang(p_k) \cap L(Q) = \emptyset$.

Now p_k is obtained by substituting tree patterns of height up to k in p. Let t be the ground tree pattern obtained by substituting each variable in p_k with an element in Λ_a where a is of 0 arity (this is possible by (D)), and we are done. ∎

Corollary 9 *For any $p \in TP$, $H_{k+1}(p)$ is a characteristic set for $L(p)$ within $\mathcal{L}(TP)^k$.*

The following Theorem follows.

Theorem 4.11 *Given finite non-empty set $S \subseteq TP_\Gamma$ and $k \in N^+$. Let $s = \max(\{\|t\| \mid t \in S\})$ and let α be the largest arity of the functions in Γ. Then $\mathcal{FC}(\mathcal{L}(TP), S, k, s(1+\alpha^{1+k})) \subseteq \mathcal{MINL}(\mathcal{L}(TP), S, k)$.*

4.3.3.2 Characteristic sets within $\mathcal{L}(TP(S))^k$

We now find, for any tree pattern language L, a sufficient size for constructing a characteristic set of L of only elements of up to that size, within $\mathcal{L}(TP(S))^k$ for some finite set $S \subseteq TP_\Gamma$, with the condition that there there is at least a function of arity ≥ 2 in Γ. By doing so (that is, instead of finding characteristic sets within $\mathcal{L}(TP)^k$) we exploit the weaker condition required of the characteristic sets as stated in Theorem 3.5.

A node u in q *exceeds* a node v in p just in case pos(v) is a prefix of pos(u). u properly exceeds v just in case u exceeds v but v does not exceed u. A tree pattern p *exceeds* a tree pattern q just in case no leaf node in p is properly exceeded by some node in q.

Lemma 4.10 *Let $p, q \in TP$ be such that no node in q properly exceeds any node of a variable in p, then $L(q) \cap L(p) \neq \emptyset \Rightarrow p$ exceeds q.*

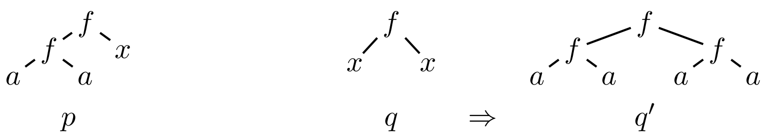
Proof. If any node in q properly exceeds any node of a constant, then clearly $L(q) \cap L(p) = \emptyset$. Hence $L(q) \cap L(p) \neq \emptyset \Rightarrow$ no node q exceeds node of constants or variables in $p \Rightarrow p$ exceeds q. ∎

Lemma 4.11 *Suppose Γ contains at least one function of arity ≥ 2. Let $p \in TP$ and finite set $Q \subseteq TP$ be such that (1) no node in any $q \in Q$ properly exceeds any node of a variable in p, and (2) $\{p\} \not\sqsubseteq Q$, then there exists $w \in S_d(p) - L(Q)$, where $d = \langle \|p\| + card(Q) \rangle_\Gamma$.*

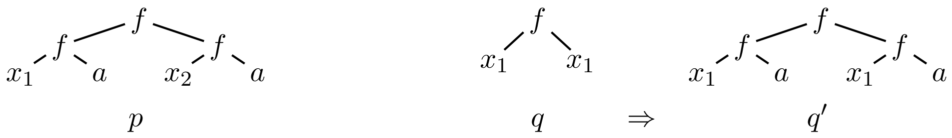
Proof. By Lemma 4.10, without loss of generality we assume p exceeds each $q \in Q$. Let $Q' = \{\text{mge}\{q \supseteq p\} \mid q \in Q\}$. It is clear that $L(p) \cap L(Q') = \emptyset \Rightarrow L(p) \cap L(Q) = \emptyset$.

We will now show that any $q' \in Q'$ must be of one of a few forms, which will allow us to construct a $w \in L(p) - \bigcup_{q' \in Q'} L(q')$. Since p covers q and $p \not\preceq q$, one of the following cases must apply for q':

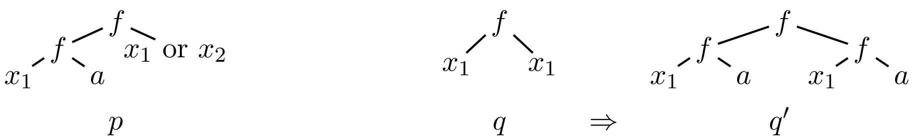
(C1) A variable x in p occurs at a position where a ground tree pattern t is found in q'. The following example shows how such a q' comes to be. Let set $T_Q \subseteq TP_\Gamma$ be such that for each $q' \in Q'$ where (C1) is true, there is a $t \in T_Q$ which witnesses this.



(C2) A variable x in p occurs at a position where at the same position in q' we find a variable which has another occurrence at a position which is not occupied by x in p, as demonstrated in the following example with the variable x_2 in p.



(C3) A variable x in p occurs at a position where in q' we find a variable which has another occurrence at a node extending a variable leaf in p, as in the following example with the variable x_1 in p.



Suppose there are n distinct variables in p. Let set $T \subseteq TP_\Gamma$ be a set of n tree patterns not in T_Q, each of the same height (this is possible by Proposition 2.11). Hence such T consists of trees of up to size $\langle n + card(Q)\rangle_\Gamma$. Since there are at most $\|p\|$ variables in p, each $t \in T$ is of size at most $\langle \|p\| + card(Q)\rangle_\Gamma$.

Let w be a tree obtained by replacing each distinct variable in p with a different tree in T. We claim that $w \not\preceq q'$. The case of (C1) is evident from that the variable corresponding to some $t \in T_Q$ is replaced with a $t' \notin T_Q$. The case of (C2) is evident from that every variable in p has been replaced with a different tree pattern. The case of (C3) is from that w is structurally impossible for q' to refine to, as observed by the height of the tree patterns required in variable substitutions. Hence no q' refines to w. Since $w \notin L(q')$ for all $q' \in Q'$, we have $w \notin L(Q') \Rightarrow w \notin L(Q)$. ∎

The next Theorem uses Lemma 4.11 to show how to find, given any $p \in TP$ and set $Q \subseteq TP$ where $L(p) - L(Q) \neq \emptyset$, a finite set $S \subseteq L(p)$ where $S \not\subseteq L(Q)$.

Theorem 4.12 *Suppose Γ contains at least one function of arity ≥ 2, and let α be the largest arity of the functions in Γ. Given finite, non-empty set $S \subseteq TP_\Gamma$, $p \in TP$ and finite $Q \subseteq \{t \in TP \mid L(t) \cap S \neq \emptyset\}$. Let $h = \max(\{|s| \mid s \in S\})$. Then either $L(p) \subseteq L(Q)$, or there exists $w \in S_{\langle \|p\|(1+\alpha^h)+card(Q)\rangle_\Gamma}(p) - L(Q)$.*

PROOF. Let Λ be the set of all constants, and tree patterns of height 1 with only variables as children. For any pattern p, let $\Theta(p)$ denote the set of all the unique tree patterns $\{p\theta \mid \theta$ is a substitution that maps each variable in p with a $t \in \Lambda\}$.

One can show that:

- (A) $L(p) = L(\Theta(p))$.

and verify that:

- (B) For any tree pattern t let $\mathrm{minvardepth}(t) = \min(\{|\mathrm{pos}(x)| \mid x$ is a variable node in $t\})$, then for all tree pattern $p' \in \Theta(p)$, $\mathrm{minvardepth}(p') = \mathrm{minvardepth}(p) + 1$. That is, intuitively, the depth of each variable node increments by 1.

Consider the following computation which, given a tree pattern p and a set of tree patterns Q, decides if $L(p) \subseteq L(Q)$.

stage 0:
  Let $G_0 = \{p\}$. Go to stage 1.

stage l:
  If there exists some ground tree pattern in G_{l-1} not refinable from some
    $q \in Q$, output false.

Remove from G_{l-1} the patterns that can be refined from some $q \in Q$.
If $G_{l-1} = \emptyset$, output true.
Let $G_l = \bigcup_{p' \in G_{l-1}} \Theta(p')$, and go to stage $l + 1$.

It is clear that if the computation halts at any stage and outputs false, $L(p) \not\subseteq L(Q)$. By Condition (A), if the computation halts at any stage and outputs true, $L(p) \subseteq L(Q)$. Note that:

- (C) At each level l, each variable in every tree pattern $p' \in G_l$ is at depth at least l (by Condition (B)).

Assume that the computation continues pass stage h. By Condition (C), at stage h, every pattern in G_h has variables at depth $\geq h$ (that is, exactly h in the case that minvardepth$(p) = 0$). On the other hand, for each $q \in Q$, $L(q) \cap S \neq \emptyset \Rightarrow$ no node in q can be at a depth larger than h. Hence every variable node for any $p' \in G_h$ must exceed its corresponding node in any $q \in Q$ (if such corresponding nodes exist), and fulfills condition (1) of Lemma 4.11.

Let $p' \in G_h$ such that $\{p'\} \not\sqsubseteq Q$ (such p' exists by our assumption that the computation passes stage h). By Lemma 4.11, there exists $w \in S_d(p') - L(q)$ where $d = \langle \|p'\| + card(Q) \rangle_\Gamma$. Since there are at most $\|p\|$ variables in p, $\|p'\|$ is at most $\|p\|(1 + \alpha^h)$. ∎

Corollary 10 *Given finite non-empty set $S \subseteq TP_\Gamma$, $k \in N^+$ and $p \in TP$. Let $h = \max(\{|s| \mid s \in S\})$. Then $S_{\langle \|p\|(1+\alpha^h)+k \rangle_\Gamma}(p)$ is a characteristic set for $L(p)$ within $\mathcal{L}(TP(S))^k$.*

This gives us the following result, in addition to Theorem 4.11.

Theorem 4.13 *Suppose Γ contains at least one function of arity ≥ 2. Given finite non-empty set $S \subseteq TP_\Gamma$ and $k \in N^+$. Let $h = \max(\{|t| \mid t \in S\})$, $s = \max(\{\|t\| \mid t \in S\})$, and let α be the largest arity of the functions in Γ. Then $\mathcal{FC}(\mathcal{L}(TP), S, k, s(1+d)) \subseteq \mathcal{MINL}(\mathcal{L}(TP), S, k)$ where $d = \langle s(1 + \alpha^h) + k \rangle_\Gamma$.*

4.4 Conclusions and open problems

In summary, the results mentioned in this and the previous chapter are for the following classes:

(A) the unions of the (erasing and non-erasing) regular pattern languages where the number of unions allowed is bounded by a factor of the alphabet size.

(B) the unbounded unions of the (erasing and non-erasing) substring and subsequence patterns.

(C) the bounded unions of the (erasing and non-erasing) languages generated by patterns which contain at most one variable occurrence.

(D) the bounded unions of the tree pattern languages: one for the case where the number of unions allowed is bounded by the alphabet size, and one for the case without such a restriction.

Below we list some classes which we hope to see similar sufficient lengths for the elements of their characteristic sets, if such lengths may be obtained.

(I) bounded unions of the (erasing and non-erasing) regular pattern languages where the bound is independent of the alphabet size. (That is, same as class (A) except that the alphabet size restriction on the number of unions is removed.)

(II) the unbounded unions of the (erasing and non-erasing) regular pattern languages of bounded segment length [73].

Chapter 5

Finding Fittest Covers

In this chapter we consider the problem of solving an instance of the **FC** problem. Given a class $\mathcal{L}$, a finite non-empty set $S \subseteq \Sigma^*$, k and $\ell \in N^+$, $\mathbf{FC}(\mathcal{L}, S, k, \ell)$ can be solved by finding, among every set of up to k languages $\mathcal{L}' \subseteq \{L \in \mathcal{L} \mid L \cap S \neq \emptyset\}$, for one where:

(1) $S \subseteq \bigcup_{L \in \mathcal{L}'} L$, and
(2) $card(\bigcup_{L \in \mathcal{L}'} L^{\leq \ell})$ is minimized.

Here, note that if the collection of all candidate sets $\mathcal{L}'$ is finite and can be enumerated recursively, then the fittest covers can be computed. A similar computation for *minl*s, with Condition (2) replaced with the minimality of inclusion, would still be difficult, due to the fact that deciding inclusions between any two (erasing or non-erasing) pattern languages is uncomputable [34, 35].

Condition (1) cannot be computed efficiently for the full class of pattern languages, since it is known that the membership problem is NP-complete for the class [2]. However, for any string $w \in \Sigma^*$ and any $p \in \mathbf{Preg}$, the question of whether $w \in L(p)$ or $w \in L_\epsilon(p)$, can be computed in time $O(|w| + |p|)$ [68]. This gives us the following.

Proposition 5.1 *For any set of strings S and any $P \subseteq$ **Preg**, whether $S \subseteq L(P)$, or $S \subseteq L_\epsilon(P)$, can be computed in time $O(card(P) \cdot \|S\| + card(S) \cdot \|P\|)$.*

We turn our attention to solving Condition (2). The condition can be checked by counting the number of elements of up to length ℓ in each $L \in \mathcal{L}'^k$, and keeping all the languages L with the smallest such counts. It is clear that such countings can be done by enumerating elements in $(\Sigma^*)^{\leq \ell}$ and testing for the membership of each of these elements in every language. In Section 5.1 we discuss more efficient counting methods for the unions of the erasing substring pattern languages.

In Section 5.2 we discuss a strategy for solving Condition (2) that does not involve counting. That is, in the situations that we can compare the number of elements of up to some length between any two languages, then such counting would not be actually necessary. We show how this is possible for the case of the erasing substring pattern languages, when the alphabet size is sufficiently large.

Results from Section 5.1 and 5.2 are finally used in Section 5.3 to show how we can efficiently find all the fittest covers of the substring pattern languages, that is, $\mathcal{FC}(\mathcal{L}_\epsilon(\mathbf{Psub}), \cdots)$.

In Section 5.4, we consider the counting of the trees in any given tree pattern languages.

5.1 Counting strings

There are many approaches to string counting. We show two methods for counting strings in this section. The first method can be used to count the strings generated by any kind of languages that has an accepting *deterministic finite automata* (DFA) [29]. The second method is based on a recurrence relation in the substring patterns.

5.1.1 Finite automata

There is a well-known method, in the field of formal languages, to count the number of strings of exactly length ℓ accepted by a DFA [37, 18]. This is done by counting the number of distinct paths of length ℓ from the initial state of the DFA to an accepting state of the DFA. For a DFA of size M and a length ℓ, this counting can be done in time $O(\ell \cdot |M| \min(\{|\Sigma|, |M|\}))$. Since by the method, the strings of the lengths $1, 2, \ldots, \ell - 1$ are also counted in the counting of the strings of length ℓ, a simple adaptation of the method allows us to compute the number of elements of up to any length ℓ in the same order of time.

Because there is a DFA that accepts the (erasing or non-erasing) language generated by any set P of regular pattern languages, we can use this method to count the strings in any language of $\mathcal{L}(\mathbf{Preg})^*$. Furthermore, for any set P of substring patterns, there is a DFA of size at most $\|P\|$ that accepts the erasing language of P (for example, by the Aho-Corasick algorithm [1]). Hence, counting the elements of up to any length in the unions of the erasing substring pattern languages can be done efficiently, as stated in the following Theorem.

Theorem 5.2 *Given a set of substring patterns $P \subseteq$ **Psub** and $m \in N^+$, there exists an algorithm which computes the total number of strings in $L_\epsilon(P)$ of length up to m in time $O(m \cdot \|P\| \cdot \min(\{card(\Sigma), \|P\|\}))$.*

5.1.2 Dynamic programming

For any set of substring patterns $P \subseteq \mathbf{Psub}$, there is a recurrence relation which allows us to count $card(L_\epsilon(P)^{\leq m})$ where $m \in N^+$ using a dynamic programming approach.

To simplify notation we work with constant strings. That is, we associate any constant string t with the substring pattern $x_1 t x_2$. For any set of constant strings T we let $P(T)$ denote the set $\{x_1 t x_2 \mid t \in T\}$.

The following algorithm ($CountStrings$) computes, given as input a set of constant strings T and $m \in N^+$, the total number of unique strings of length up to m within $L_\epsilon(P(T))$. We assume that no string in T is a substring of some other string in T. Otherwise, the strings that are substring of some other string in T can be removed in time $O(\mathit{card}(T)^2\lceil T \rceil)$, and their removal will not affect the computation of $\mathit{card}(L_\epsilon(P(T))^{\leq m})$.

$CountStrings(T, m)$

For each r from 1 to m,
    For each string $t \in T$ where $|t| \leq r$,
        For each r' from $r - |t| + 1$ to $r - 1$, calculate $S(T, t, r, r')$.
    Calculate $\Phi(T, r)$.
Output $\sum_{r=1}^{m} \Phi(T, r)$.

For each $r \in N^+$, $\Phi(T, r)$ is the total number of strings of length r in $L_\epsilon(P(T))$. For $t \in T$, r, $r' \in N^+$ where $r > r'$, $S(T, t, r, r')$ is the number of strings in $L_\epsilon(x_1 t)$ of length r which does not begin with some string of length r' in $L_\epsilon(P(T))$. It is easy to see that $S(T, t, r + |t|, r) = \mathit{card}(\Sigma)^{r-|t|} - \Phi(T, r)$. The following is the computation of $S(T, t, r, r')$. For a string t and $n \in N$, prefix(t, n) denotes the prefix of t of length n, while suffix(t, n) denotes the suffix of t of length $|t| - n$.

$S(T, t, r, r')$

If $|t| > r$, output 0 and stop.
Else if $|t| = r$, output 1 and stop.
Otherwise, for each $t' \in T$, we try to find all $0 < n < |t'|$ such that
    (a) suffix(t', n) matches some prefix of t,
    (b) $n + |t| \leq r$,
    (c) $|t| - (|t'| - n) \geq r - r'$,
    and collect all such prefix(t', n) into a set $A_{t,r,r'}$.
We then remove from $A_{t,r,r'}$ all the strings for which some of its suffixes
    are also in $A_{t,r,r'}$. (Hence $A_{t,r,r'}$ is a set of shortest prefix strings of T
    where the suffix part is consistent with t.)
Let $B_{t,r,r'} = \{(t', \text{suffix}(t', n)) \mid \text{prefix}(t', n) \in A_{t,r,r'}\}$.
Output $\mathit{card}(\Sigma)^{r-|t|} - \Phi(T, r - |t|) - \sum_{(t',t'') \in B_{t,r,r'}} S(T, t', r - |t| + |t''|, r - |t|)$.

For each r, the following is the computation of $\Phi(T, r)$.

$\Phi(T, r)$

Initialize ϕ to $\mathit{card}(\Sigma) \cdot \Phi(T, r - 1)$ if $r \geq 1$, 0 otherwise.
    For each $t \in T$ where $|t| \geq r$, add $S(T, t, r, r - 1)$ to ϕ.
Output ϕ.

Theorem 5.3 *Given a set T of constant strings and $m \in N^+$, $m \gg \lceil T \rceil$, $CountStrings$ computes the number of strings in $L_\epsilon(P(T))$ of length up to m in time $O(m \cdot \|T\|^2)$.*

Proof. To see the correctness of *CountStrings*, first consider any string s of length m in $L_\epsilon(x_1 t x_2)$ for some string t shorter than m. Now s must be of one of the following forms:

$$\begin{array}{ll} t?^{m-|t|} & \text{(F.0)} \\ ?t?^{m-|t|-1} & \text{(F.1)} \\ \ldots & \\ ?^{m-|t|}t & (\text{F.}m-|t|) \end{array}$$

Now for each n, $0 \le n \le m - |t|$, if we know the total number of strings, $g(n)$ say, of the form (F.n) but not of the forms (F.0), (F.1), …, (F.$n-1$), the sum of $g(0)$, $g(1)$, …, $g(m-|t|)$ would then give us the number of strings of length m in $L_\epsilon(x_1 t x_2)$. The same method is used in *CountStrings*, where for $t \in T$ and $r \in N$, $\lfloor T \rfloor \le r \le m$, we compute the total number, $f(t, r)$ say, of strings of length r of the form $w_1 t$ (that is, ending in t), but not of any of the form $w_2 t' w_3$ for some $t' \in T$, $w_1, w_2 \in \Sigma^*$, $w_3 \in \Sigma^+$. (Here, the case where $|w_3| = 0$ is not considered since no string in T is a substring of another.) It is easy to verify that the total number of strings of length m in $L_\epsilon(P(T))$, $\Phi(T, m) = \sum_{r=\lfloor T \rfloor}^{m} \sum_{t \in T \wedge |t| \ge r} \mathit{card}(\Sigma)^{m-r} f(t, r) = \mathit{card}(\Sigma) \cdot \Phi(T, m-1) + \sum_{t \in T \wedge |t| \ge m} f(t, m)$.

To calculate $f(t, r)$, we first see that there are $\mathit{card}(\Sigma)^{r-|t|}$ strings of length r of the form $x_1 t$, and $f(t, r)$ is given by this number deducted by the number of strings of length r that are simultaneously of the forms $w_1 t$ and $w_2 t' w_3$ for some $t' \in T$ and $w_1, w_2 \in \Sigma^*$, $w_3 \in \Sigma^+$. The strings of length r of the form $wt'w't$ where w, $w' \in \Sigma^*$ trivially matches these two forms, and it is easy to see that their number is given by $\Phi(T, r - |t|)$.

The remaining strings simultaneously having the forms $w_1 t$ and $w_2 t' w_3$ must have: (C1) $\max(\{0, |t| - |t'|\}) < |w_3| < |t|$, (C2) some suffix of t' matching some prefix of t, and (C3) not of the form $wt''w't$ for some $t'' \in T$, w, $w' \in \Sigma^*$. As an example, string (A) below fulfills (C1), (C2) but fails (C3); while string (B) fulfills all three conditions for $T = \{bab\}$.

$$\begin{array}{ll} & b\ a\ b\ ?\ ?\ \overbrace{b\ a\ b}^{t} \\ & ?\ b\ a\ b\ ?\ b\ a\ b \\ & ?\ ?\ b\ a\ b\ b\ a\ b \\ \text{(A)} & a\ b\ a\ \underbrace{b\ a\ b}\ a\ b \\ \text{(B)} & a\ b\ b\ \overbrace{b\ a\ b}^{t}\ a\ b \end{array}$$

In general, for any t, $t' \in T$, $r \in N$ and $w_3 \in \Sigma^+$ where $\max(\{0, |t| - |t'|\}) < |w_3| < |t|$, the number of strings of length r ending in $t'w_3$ which fulfill (C3) is the same as the number of strings of length $r - |w_3|$ ending in t' and with the initial $r - |t|$ long segment not in $L_\epsilon(P(T))$. For this reason we let $S(T, t', r', r'')$ denote the number of strings of length r' that ends in t' and does not begin with a string in $L_\epsilon(P(T))^{=r''}$. Hence, it is sufficient to compute $f(t, r)$ with only $\Phi(T, r - |t|)$ and the values $S(T, t', r', r - |t|)$ for all $t' \in T$ and for all r' where $r - |t| < r' < r - \max(\{0, |t| - |t'|\})$.

Noting that $f(t,r)$ is simply $S(T,t,r,r-1)$, it is easy to verify that using the same strategy for computing $f(t,r)$, it is sufficient for us to compute $S(T,t,r,r')$ for any $t \in T$, r, $r' \in N$ with only $\Phi(T, r-|t|)$ and $S(T,t',r'',r-|t|)$ for all $t' \in T$ and all r'' where $r-|t| < r'' < r$. The listing for $S(T,t,r,r')$ does exactly this. Note that lines-8∼10 remove the prefix strings where their suffixes are also included, since they need not be considered because a string which ends in one of these prefix strings also ends in their suffixes.

We now look at the runtime complexity for *CountStrings*. $S(T,t,r,r')$ is called $m \cdot \|T\|$ times, while $\Phi(T,r)$ is called m times. The computation for $\Phi(T,r)$ takes time $O(card(T))$ for each r, costing $O(m \cdot card(T))$ in total. In $S(T,t,r,r')$, the computation at line-12 takes time $O(\|T\|)$, i.e. the maximum value of $|B_{t,r,r'}|$; while the computation for $B_{t,r,r'}$ takes time $O(\sum_{t \in T} |t|^2)$. However, we note that for all $r \geq 2\lceil T \rceil - 1$, $B_{t,r,r'} = B_{t,r+1,r'+1}$, and hence some trivial modification to $S(T,t,r,r')$ would make it run in time $O(\|T\|)$ for $r \geq 2\lceil T \rceil - 1$. Hence the algorithm runs in time $O(m\|T\|^2)$ for $m \gg \lceil T \rceil$. ∎

Note that its runtime is comparable to the algorithm in Theorem 5.2 for the cases where $card(\Sigma)$ is large.

5.2 Comparing string counts

In this section we use the algorithm *CountStrings* in Section 5.1.2 to show how, given any two substring patterns p, p' of a length shorter than $card(\Sigma) - 2$ and $m \in N^+$, it is possible to decide if $card(L_\epsilon(p)^{\leq m}) \geq card(L_\epsilon(p')^{\leq m})$, without actually counting $card(L_\epsilon(p)^{\leq m})$ or $card(L_\epsilon(p')^{\leq m})$.

We first note that for all $i > 0$,

(A) $S(\{t\}, t, x+i, x) \leq card(\Sigma)^{x+i-|t|}$

(By the recurrence relation), and

(B) $\Phi(\{t\}, x+i) \leq card(\Sigma)^{x+i-|t|}$

(Because there are at most $card(\Sigma)^{x+i-|t|}$ strings of length $x+i$ in $L_\epsilon(x_1 t x_2)$).

Lemma 5.1 *Let $x \in N$ and $d \in N^+$. For all t, $t' \in \Sigma^*$ where $|t'| = |t| > d$,*

$$S(\{t\}, t, x+d, x) \geq card(\Sigma)^{x+d-|t|} - \sum_{i=x+d-2|t|}^{x-|t|} card(\Sigma)^i.$$

Proof. From the recurrence relation,

$$S(\{t\}, t, x+d, x) \geq card(\Sigma)^{x+d-|t|} - \Phi(\{t\}, x+d-|t|) - \sum_{i=0}^{|t|-d-1} S(\{t\}, t, x-i, x+d-|t|)$$

Then by (A) and (B) we have

$$S(\{t\}, t, x+d, x) \geq card(\Sigma)^{x+d-|t|} - card(\Sigma)^{x+d-2|t|} - \sum_{i=0}^{|t|-d-1} card(\Sigma)^{x-i-|t|}$$

Lemma follows. ∎

Lemma 5.2 *Let $x \in N$ and $d \in N^+$. For all t, $t' \in \Sigma^*$ where $|t| = |t'| > d$,*

$$|S(\{t\}, t, x+d, x) - S(\{t'\}, t', x+d, x)| \leq \textstyle\sum_{i=x+d-2|t|}^{x-|t|} card(\Sigma)^i$$

Proof. From the recurrence relation,

$$\begin{aligned}
&|S(\{t\}, t, x+d, x) - S(\{t'\}, t', x+d, x)| \\
&\quad \leq \max(\{\Phi(\{t'\}, x+d-|t|), \Phi(\{t\}, x+d-|t|)\}) \\
&\qquad + \max(\{\textstyle\sum_{i=0}^{|t|-d-1} S(\{t\}, t, x-i, x+d-|t|), \textstyle\sum_{i=0}^{|t|-d-1} S(\{t'\}, t', x-i, x+d-|t|)\})
\end{aligned}$$

By (A) and (B) we have

$$|S(\{t\}, t, x+d, x) - S(\{t'\}, t', x+d, x)| \leq card(\Sigma)^{x+d-2|t|} + \textstyle\sum_{i=0}^{|t|-d-1} card(\Sigma)^{x-i-|t|}$$

Lemma follows. ∎

Since $card(T) = 1$, for all t, r and r', for all (t, t_1), $(t, t_2) \in B_{t,r,r'}$, $|t_1| = |t_2|$ implies $t_1 = t_2$. That is, only the length of the overlap is needed. Let $\mathbf{B}(t, r) = \{|t''| \mid (t, t'') \in B_{t,r,r-1}\}$. It is clear that $\max(\mathbf{B}(t, r)) < |t|$. Note that for any string t and $r \geq 2|t| - 1$, $\mathbf{B}(t, r+1) = \mathbf{B}(t, r)$. For this reason we also let $\mathbf{B}(t)$ denote $\mathbf{B}(t, 2|t| - 1)$.

Lemma 5.3 *Given t, $t' \in \Sigma^*$ where $|t| = |t'|$ and $\max(\mathbf{B}(t) - \mathbf{B}(t')) < \max(\mathbf{B}(t') - \mathbf{B}(t))$. Let $j = \max(\mathbf{B}(t') - \mathbf{B}(t))$ and let $k = card(\{d \in \mathbf{B}(t') \cap \mathbf{B}(t) \mid d > j\})$. Then if $card(\Sigma) > 2 + k$,*

$$\textstyle\sum_{i \in \mathbf{B}(t,r)} S(\{t\}, t, x+i+1, x) < S(\{t'\}, t', x'+j+1, x)$$

Proof. In the case of t', by Lemma 5.1,

$$S(\{t'\}, t', x+j+1, x) \geq card(\Sigma)^{x+j+1-|t|} - \textstyle\sum_{i=x+j+1-2|t|}^{x-|t|} card(\Sigma)^i.$$

On the other hand for t we have the following inequality,

$$\textstyle\sum_{i \in \mathbf{B}(t,r) \wedge d<j} S(\{t\}, t, x+i+1, x) \leq \textstyle\sum_{1 \leq i < j} card(\Sigma)^{x+i+1-|t|}$$

and by Lemma 5.2,

$$\textstyle\sum_{i \in \mathbf{B}(t,r) \wedge i>j} |S(\{t\}, t, x+i+1, x) - S(\{t'\}, t', x+i+1, x)| \leq k \cdot \textstyle\sum_{i=x+j+1-2|t|}^{x-|t|} card(\Sigma)^i$$

Hence,

$$\begin{aligned}
&\textstyle\sum_{i' \in \mathbf{B}(t',r)} S(\{t'\}, t', x'+i'+1, x) - \textstyle\sum_{i \in \mathbf{B}(t,r)} S(\{t\}, t, x+i+1, x) \\
&\qquad \geq card(\Sigma)^{x+j+1-|t|} - (k+1) \textstyle\sum_{i=x+j+1-2|t|}^{x-|t|} card(\Sigma)^i - \textstyle\sum_{1 \leq i < j} card(\Sigma)^{x+i+1-|t|}
\end{aligned}$$

Since $\Sigma > k + 2$, Lemma follows. ∎

Theorem 5.4 *For all t, $t' \in \Sigma^*$ where $|t| = |t'| < card(\Sigma) - 2$, if $\mathbf{B}(t) = \mathbf{B}(t')$, then for all $r \in N$, $\Phi(\{t\}, r) = \Phi(\{t'\}, r)$. Otherwise without loss of generality let $\max(\mathbf{B}(t) \setminus \mathbf{B}(t')) < \max(\mathbf{B}(t') \setminus \mathbf{B}(t))$, in which case*
(1) $\Phi(\{t\}, r) = \Phi(\{t'\}, r)$ for $r < 2|t| - \max(\mathbf{B}(t') \setminus \mathbf{B}(t))$, and
(2) $\Phi(\{t\}, r) > \Phi(\{t'\}, r)$ for $r \geq 2|t| - \max(\mathbf{B}(t') \setminus \mathbf{B}(t))$.

Proof. Let Σ, t, t' be as in Claim. From the recurrence relation in *CountStrings* we have the following equation. For convenience we denote the first term of the left hand side of the equation (A), the second term (B), and the two remaining terms (C).

$$\begin{aligned}
&\Phi(\{t\}, r) - \Phi(\{t'\}, r) \\
&\quad = card(\Sigma) \cdot (\Phi(\{t\}, r-1) - \Phi(\{t'\}, r-1)) && \text{(A)} \\
&\qquad -(\Phi(\{t\}, r-|t|) - \Phi(\{t'\}, r-|t|)) && \text{(B)} \\
&\qquad\quad - \left(\textstyle\sum_{x \in \mathbf{B}(t,r)} S(\{t\}, t, r-|t|+x, r-1-|t|)\right. \\
&\qquad\qquad \left. - \textstyle\sum_{x' \in \mathbf{B}(t',r)} S(\{t'\}, t', r-|t|+x', r-1-|t|)\right) && \text{(C)}
\end{aligned}$$

From the recurrence relation we further know that for any t and for any $x \in \mathbf{B}(t)$, $x \in \mathbf{B}(t, r)$ if and only if $r \geq 2|t| - x$. Intuitively, if we consider each $x \in \mathbf{B}(t)$ as being added to $\mathbf{B}(t, r)$ as r increases, then the value x is added when $r = 2|t| - x$.

Hence if $\mathbf{B}(t) = \mathbf{B}(t')$, then for all $r \in N$, $\Phi(\{t\}, r) = \Phi(\{t'\}, r)$. We now consider the case where $\mathbf{B}(t) \neq \mathbf{B}(t')$. As in the hypothesis, let $\max(\mathbf{B}(t) - \mathbf{B}(t')) < \max(\mathbf{B}(t') - \mathbf{B}(t))$. For $r < 2|t| - \max(\mathbf{B}(t') - \mathbf{B}(t))$, since $\mathbf{B}(t, r) = \mathbf{B}(t', r)$ for $\Phi(\{t\}, r) = \Phi(\{t'\}, r)$. This concludes Part (1) of the Theorem.

For all $r \geq 2|t| - \max(\mathbf{B}(t') - \mathbf{B}(t))$, by Lemma 5.3, the following holds.

$$\sum_{x \in \mathbf{B}(t,r)} S(\{t\}, t, ., .) < \sum_{x' \in \mathbf{B}(t',r)} S(\{t'\}, t', ., .).$$

That is, part (C) is non-zero and negative, and since negated, it takes non-zero positive values.

Since for all $r \leq 2|t| - \max(\mathbf{B}(t') - \mathbf{B}(t))$, $\Phi(\{t\}, r) = \Phi(\{t'\}, r)$, (A) and (B) are both zero for these values of r. (C) part then gives us that

$$\Phi(\{t\}, 2|t| - \max(\mathbf{B}(t') - \mathbf{B}(t))) > \Phi(\{t'\}, 2|t| - \max(\mathbf{B}(t') - \mathbf{B}(t))) \tag{5.1}$$

We claim that for all r,

$$\Phi(\{t\}, r) - \Phi(\{t'\}, r) \geq \Phi(\{t\}, r-1) - \Phi(\{t'\}, r-1). \tag{5.2}$$

We have seen that this is true for all r where $|t| \leq r \leq 2|t| - \max(\mathbf{B}(t') - \mathbf{B}(t))$. We now consider $r = 2|t| - \max(\mathbf{B}(t') - \mathbf{B}(t)) + 1$. Considering only (A) and (B) part of $\Phi(\{t\}, r) - \Phi(\{t'\}, r)$, since $card(\Sigma) \geq 2$,

$$\begin{aligned}
&card(\Sigma) \cdot (\Phi(\{t\}, r-1) - \Phi(\{t'\}, r-1)) - (\Phi(\{t\}, r-|t|) - \Phi(\{t'\}, r-|t|)) \\
&\quad \geq \quad (\Phi(\{t\}, r-1) - \Phi(\{t'\}, r-1)) + && \text{(AB1)} \\
&\quad\quad\quad \big(\Phi(\{t\}, r-1) - \Phi(\{t'\}, r-1)) - (\Phi(\{t\}, r-|t|) - \Phi(\{t'\}, r-|t|))\big) && \text{(AB2)}
\end{aligned}$$

Since (5.2) holds for all $r' \leq \max(\mathbf{B}(t') - \mathbf{B}(t)) + 1$, (AB2) is positive. Hence $\Phi(\{t\}, r) - \Phi(\{t'\}, r)$ is of at least (AB1)$= \Phi(\{t\}, r-1) - \Phi(\{t'\}, r-1)$ plus part (C), which is positive. Hence (5.2) holds for $r = 2|t| - \max(\mathbf{B}(t') - \mathbf{B}(t)) + 1$. The same argument applies to all subsequent values of r. (5.1) and (5.2) then concludes Part (2) of the Theorem. ∎

From Theorem 5.4, it is clear that for any strings t, t' where $|t| = |t'| < card(\Sigma) - 2$, it suffices that we compute the sets $\mathbf{B}(t)$ and $\mathbf{B}(t')$ to decide if $CountStrings(\{t\}, m) > CountStrings(\{t'\}, m)$ for any $m \in N^+$.

5.3 Finding the fittest covers $\mathcal{FC}(\mathcal{L}_\epsilon(\mathbf{Psub}), \cdots)$

We now study how, given a finite non-empty $S \subseteq \Sigma^*$ and k, $\ell \in N^+$, to compute $\mathcal{FC}(\mathcal{L}_\epsilon(\mathbf{Psub}), S, k, \ell)$. We first look at the following case, where no union is allowed, and the substring pattern lengths are less than $card(\Sigma)$.

5.3.1 The case of $k = 1$ and $\lfloor S \rfloor < card(\Sigma) - 2$

For the case of $k = 1$, for any finite non-empty set $S \subseteq \Sigma^*$, k and $\ell \in N^+$, $\mathcal{FC}(\mathcal{L}_\epsilon(\mathbf{Psub}), S, 1, \ell)$ can be computed in polynomial time, as follows.

Theorem 5.5 *Given finite non-empty set $S \subseteq \Sigma^*$ where $\lfloor S \rfloor < card(\Sigma) - 2$ and $\ell \in N^+$. There is an algorithm that computes $\mathcal{FC}(\mathcal{L}_\epsilon(\boldsymbol{Psub}), S, 1, \ell)$ in time*

$$O(\lfloor S \rfloor \cdot (\lfloor S \rfloor^2 + \|S\|) \cdot \min(\{\|S\|, card(\Sigma)^{\lfloor S \rfloor}\}) + \ell \cdot \lfloor S \rfloor^2 \cdot \min(\{card(\Sigma), \lfloor S \rfloor\}))$$

Proof. We first consider how we may solve the problem for strings of the same length m. That is, we consider solving $\mathcal{FC}(\mathcal{L}_\epsilon(\mathbf{Psub}_m), S, 1, \ell)$ where $\mathbf{Psub}_m = \{x_1 t x_2 \mid t \in \Sigma^* \wedge |t| = m\}$.

Since each pattern in the solution must cover some strings in S, we choose only substrings of strings in S for candidates. There are possibly $\|S\|$ such substrings, of which at most $card(\Sigma)^m$ are unique. This gives us a total of at most $\min(\{\|S\|, card(\Sigma)^m\})$ candidates. We denote this candidate set by C. It is easy to verify, using Theorem 5.4, that the following algorithm *FittestCovers* finds $\mathcal{FC}(\mathcal{L}_\epsilon(\mathbf{Psub}_m), S, 1, \ell)$.

FittestCovers(S, m)
1. Let C be the set of all substrings of some $s \in S$ of length m.

2. Let $maxSet = \{-1\}$.
3. Let $\mathcal{FC}_m = \emptyset$.
4. For each candidate t in C where $S \subseteq L_\epsilon(x_1 t x_2)$
   4.1. Compute $\mathbf{B}(t)$.
   4.2. If (1) $\max(\mathbf{B}(t) \setminus maxSet) = \max(maxSet \setminus \mathbf{B}(t))$, or
   (2) $\ell < 2|t| - \max(\mathbf{B}(t) \setminus maxSet)$,
   Then set $\mathcal{FC}_m$ to $\mathcal{FC}_m \cup \{L_\epsilon(x_1 t x_2)\}$.
   4.3. Otherwise if $\max(\mathbf{B}(t) \setminus maxSet) > \max(maxSet \setminus \mathbf{B}(t))$,
   Then set $maxSet$ to $\mathbf{B}(t)$ and let $\mathcal{FC}_m = \{L_\epsilon(x_1 t x_2)\}$.
5. Output $\mathcal{FC}_m$.

As discussed in the beginning of this chapter, membership checks for each candidate at line-4 take $O(\|S\| + \lfloor S \rfloor |S|) = O(\|S\|)$ time and each computation of line-4.1 takes $O(m^2) = O(\lfloor S \rfloor^2)$ time, that is, the time needed to find $\mathbf{B}(t)$. Each of these is performed at most once for each candidate in C. Now to find the fittest covers for $\mathcal{L}_\epsilon(\mathbf{Psub})$, we compute $FittestCovers(S, m)$ for each $m \leq \lfloor S \rfloor$ (this gives us the first part of the stated runtime), and then compare the number of elements of up to ℓ for one language from each non-empty $\mathcal{FC}_m$. There are a total of $\lfloor S \rfloor$ such $\mathcal{FC}_m$ sets, and the counting for a language from $\mathcal{FC}_m$ takes $O(\ell \cdot m \cdot \min(\{card(\Sigma), m\}))$ time by Theorem 5.2. Hence this counting takes $O(\ell \cdot \lfloor S \rfloor^2 \cdot \min(\{card(\Sigma), \lfloor S \rfloor\}))$ time in total. ▮

It is not surprising, but worth noting, that the runtime in this case is not dependent on $\lceil S \rceil$. We next look at the general case where unions are allowed, that is, where $k \in N^+$.

5.3.2 The case of $k \in N^+$

Theorem 5.6 *Let $card(\Sigma) \geq 2$. Given a finite non-empty set $S \subseteq \Sigma^*$ and $k, \ell \in N^+$, let $m = \lceil S \rceil$, then there is an algorithm that finds $\mathcal{FC}(\mathcal{L}_\epsilon(\boldsymbol{Psub}), S, k, \ell)$ in time*

$$O\Big(\big(\sum_{s \in S} |s|^2\big)^k \cdot (\ell \cdot mk \cdot \min(\{card(\Sigma), mk\}) + mk \cdot card(S))\Big)$$

Proof. On any finite non-empty set $S \subseteq \Sigma^*$ and $k, \ell \in N^+$, the following algorithm $FittestCovers$ computes $\mathcal{FC}(\mathcal{L}_\epsilon(\mathbf{Psub}), S, k, \ell)$ in a brute-force manner.

$FittestCovers(S, k, \ell)$
1. Let $\mathcal{FC} = \emptyset$ and let $count = \infty$.
2. Let C be the set of all substrings of some $s \in S$ of length up to m.
3. For each $T \subseteq C$ where $card(T) \leq k$ and $S \subseteq L_\epsilon(P(T))$
   3.1. Compute $count_T = L_\epsilon(P(T))^{\leq \ell}$.
   3.2. If $count_T < count$ then let $count = count_T$ and let $\mathcal{FC} = \{P(T)\}$.
   3.3. Else if $count_T$ is the same as $count$ then enumerate $P(T)$ into $\mathcal{FC}$.
4. Output $\mathcal{FC}$.

We consider only substring pattern languages that intersect non-emptily with S, hence we consider only substrings of some string in S (there are at most $\sum_{s\in S}(1+2+\ldots+|s|)$ such substrings), or strings of up to length m (there are at most $card(\Sigma)+card(\Sigma)^2+\ldots+card(\Sigma)^m \leq card(\Sigma)^{m+1}$ such strings). Hence the number of candidates T to check at line-3 is of the order of $\min(\{\sum_{s\in S}|s|^2,\ card(\Sigma)^{m+1}\})^k$.

As discussed in the beginning of this chapter, membership checks for each candidate set at line-3 take $O(k\|S\|+\ \|T\|\cdot card(S))$ time. Each check at line-3.1, using the algorithm in Theorem 5.2, is done in time of the order of $\ell\cdot\|T\|\cdot\min(\{card(\Sigma),\|T\|\})$. Since $\|T\|\leq mk$ and $\|S\|\leq m\cdot card(S)$, the total runtime is hence of the order as given in the Theorem. ∎

5.4 Counting trees

We now show how to count, given any tree pattern p, all the instances of p generated by replacing variables in p with ground tree patterns of height up to n, that is, $card(H_n(p))$.

$Count\_H_n(p)$
1. Let i be the number of independent variables occurring in p.
2. Output $CountTreesUpToHeight(n)^i$.

where for any $n\in N^+$, $CountTreesUpToHeight(n)$ gives us the number of the trees in TP_Γ of height up to at most n (that is, $card(\{t\in TP_\Gamma \mid |t|\leq n\})$). We show an efficient computation for $CountTreesUpToHeight(n)$ later.

It is also easy to count the number of trees of up to any height n generated by a tree pattern p, by the following.

$CountUpToHeight(n,p)$
1. Let X' be the set of independent variables occurring in p.
2. For each x in X', let $m_x = n-\max(\{|\mathrm{pos}(v)| \mid x \text{ occurs in node } v\})$.
3. Output $\prod_{m\in\{m_x|x\in X'\}} CountTreesUpToHeight(m)$.

To compute $CountTreesUpToHeight(height)$ we can use the following recursion relation between $CountTreesUpToHeight(n)$ and $CountTreesUpToHeight(n-1)$.

$CountTreesUpToHeight(n)$
1. Set *count* to the number of functions in Γ of arity 0.
2. If $n>0$ then for each function f of a non-zero arity, i say, in Γ,
3. Add $CountTreesUpToHeight(n-1)^i$ to *count*.
4. Output *count*.

The time complexity of $CountTreesUpToHeight(n)$ is $O(n\cdot card(\Gamma'))$, where Γ' is the set of functions in Γ of non-zero arity. Since $CountTreesUpToHeight(m)$ for all $m\leq n$ is computed in a computation of $CountTreesUpToHeight(n)$, it is easy to see that $CountUpToHeight(n,p)$ has similar time complexity as $CountTreesUpToHeight(n)$.

5.5 Discussions and open problems

We end this chapter leaving unsolved embarassingly many questions on how to solve the **FC** problem efficiently for the pattern languages and its subclasses. For example, whether the **FC** problem is NP-hard for the full class of pattern languages is yet unsolved. Even though the membership problem is NP-complete for the class, this does not immediately mean that the **FC** problem is hard for the class also. To illustrate the complications involved, consider the class of the unbounded unions of the pattern languages. For any finite set of input strings S, S itself is a fittest cover from the class for S, and hence there is no need for any computation in finding a fittest cover. On the other hand, the **FC** problem for a proper subclass is not neccessarily easier. For example, let $\mathbf{Psub}_l^*$ be the class of unbounded unions of the substring patterns of length l. Hence $\mathcal{L}_\epsilon(\mathbf{Psub}_l)^*$ is a proper subclass of the unbounded unions of the pattern languages, $\mathcal{L}_\epsilon(Pat)^*$. Finding a fittest cover from $\mathcal{L}_\epsilon(Pat)^*$ is trivial, since for any finite set S, $P = S$ is a fittest cover for S. However, it has been shown that finding a fittest cover from $\mathcal{L}_\epsilon(\mathbf{Psub}_l)^*$ is NP-complete [53].

One of the more hopeful classes where efficient computations for **FC** might exist is the bounded unions of the regular pattern languages, since membership for the class is decidable in time polynomial to the input. For the same reason, the bounded unions of the tree pattern languages is another class that is worth investigating.

As demonstrated in Section 5.2, in some cases, using the brute-force method to find fittest covers does not require an actual counting of elements in the languages, since it suffices that we compare between such counts of any two sets of languages given. For this purpose, bounds such as the following may be useful towards solving such problems.

Proposition 5.7 *For any set of k substring patterns P of length $m+2$, and positive integer $n \geq m$,*

$$k\frac{card(\Sigma)^{n-m+1}-1}{card(\Sigma)-1} \leq card(L_\epsilon(P)^{\leq n}) \leq k\frac{(2\cdot card(\Sigma))^{n-m+1}-1}{2\cdot card(\Sigma)-1}.$$

Proof. Consider the number of strings of length $m+1$ that can be generated from a pattern of length $m+2$: (1) a total of $2 \cdot card(\Sigma)$ strings are possible, and (2) each of these strings can be generated from at most 2 of these patterns, or twice from the same pattern. That is, a maximum of half of these strings can be redundant. This gives us that a set of k patterns of length $m+2$ must generate at least $k \cdot card(\Sigma)$ distinct strings of length m.

Consider the generated $k \cdot card(\Sigma)$ strings as a new set of patterns of length $m+3$. These will in turn generate $(k \cdot card(\Sigma)) \cdot card(\Sigma)$ number of distinct strings of length $m+1$. Recursively we must have at least $k + k \cdot card(\Sigma) + k \cdot card(\Sigma)^2 + \cdots + k \cdot card(\Sigma)^{n-m} = k\frac{card(\Sigma)^{n-m+1}-1}{card(\Sigma)-1}$ distinct strings in $(\Sigma^*)^{\leq n}$.

A maximum of $k\frac{(2card(\Sigma))^{n-m+1}-1}{2card(\Sigma)-1}$ strings can be similarly deduced from the observation that each pattern of length $m+1$ can generate at most $2 \cdot card(\Sigma)$ strings of length m. ∎

We end this chapter with the following remark. Throughout this chapter we have assumed an exhaustive search space, of all the unions of the languages that intersect non-emptily with the input set S. In many situations, there are better ways to enumerate the search space. For instance, we may consider only the *minl*s as candidates, since, for any input set S and length l, there must be some fittest covers that are also *minl*s. Similarly, for any class of languages where the subsumption relation implies the inclusion relation, one may instead consider only the *mmg*s as search space. In fact, such an approach has some similarities to our experimental method in the next chapter.

Chapter 6

Validating the Value of the Fittest Covers

Recall that a fittest cover is an explanation for a set of examples with the least number of elements of up to a certain length. Intuitively, a language which contains less elements may be considered more *general* and a language with less elements more *specific*. In this chapter we show, experimentally, evidence that the number of elements of up to some length contained in a hypothesis —which gives us an estimate of how general or specific the hypothesis is— can be a very important measurement when searching for so-called *consensus patterns*.

Consensus patterns are patterns that characterize a set of biological sequences. Finding such patterns has been a common task in molecular biology [15]. In such a task we have as sample a set of genes that are known to share certain traits, while the discovered patterns are to be used for the study of the genes, or for predicting if an unknown gene will demonstrate similar traits.

Many computational means to find such consensus patterns are based on statistical methods. Nevertheless, many non-statistical criteria have also been proposed. For example, Jonassen *et al.* suggested several criteria for evaluating pattern fitness [36]. Brāzma *et al.* [16] suggested achieving so-called minimal description length (MDL) for the set of patterns plus the sample biosequences; while Li *et al.* considered minimizing the sum of Hamming distances between the consensus pattern and the biosequences [41]. The kinds of patterns used in the various methods may also differ very significantly [15]. For example, the patterns used in the PROSITE [74] database is quite different from that used in the TEIRESIAS algorithm [62].

The classes we consider here are subclasses of the erasing regular pattern languages. Now to show that fittest covers are favorable for such consensus patterns, ideally, we want to solve the **FC** problem for our target classes on the given sample of biological sequences. However, due to the runtime complexity involved in a computation of **FC** for these classes, we consider another approach.

This is our approach: we first find several *mmg*s using the polynomial time algorithm **mmg** by Arimura *et al.* [10], which finds an *mmg* for any given instance of **MMG**. We then compare, between the *mmg*s obtained, the following measure of *coverage*, defined for any *mmg* P and $n \in N$, to be

$$coverage(P, n) = card(L_\epsilon(P)^{\leq n})$$

It is clear how this notion of coverage is related to the fittest covers, and we expect it to give us an indication of how general or specific an *mmg* is, at least with respect to the space $(\Sigma^*)^{\leq n}$. In our experiments we will let n be determined by the longest length of the input sequences. To count $coverage(P, n)$, we first construct a DFA that accepts the strings in $L_\epsilon(P)$, and then count the number of distinct paths of up to length n which reaches an accepting state (see Section 5.1).

In this chapter we will be showing that such coverages have very significant effects on:

1. the use of *mmg*s as signal peptide prediction, and
2. the *mmg*s' resemblance with well-established consensus patterns.

To enable our discussion we first introduce the **mmg** algorithm by Arimura *et al.* [10]. Given a class $\mathcal{P}$ of patterns, a finite non-empty set $S \subseteq \Sigma^*$ and $k \in N^+$, the following lists the algorithm **mmg**, that outputs a solution for $\mathbf{MMG}(\mathcal{L}_\epsilon(\mathcal{P}), S, k)$. All patterns in the following are implicitly elements of $\mathcal{P}$. The algorithm assumes that there is a pattern $\perp$ in the class $\mathcal{P}$ such that every pattern $p \in \mathcal{P}$ can be obtained from $\perp$ (i.e. $p \preceq \perp$).

mmg($\mathcal{P}$, S, k)
(1) Let $P \leftarrow \{\perp\}$.
(2) Let $\Delta k \leftarrow k$.
(3) While $\Delta k \geq 2$ and there exists $p \in P$, and a set
$Q \sqsubseteq \{p\}$ of more than one and up to Δk patterns where
(i) $S - L_\epsilon(P - \{p\}) \subseteq L_\epsilon(Q)$, but
(ii) no $Q' \subset Q$ has $S - L_\epsilon(P - \{p\}) \subseteq L_\epsilon(Q')$ (i.e. no pattern in Q is redundant),
(3.1) Replace each $q \in Q$ with a more specific $q' \prec q$ until any further such replacement will result in $S \not\subseteq L_\epsilon(P) - L_\epsilon(\{p\}) \cup L_\epsilon(Q)$.
(3.2) Let $P \leftarrow P - \{p\} \cup Q$.
(3.3) Let $\Delta k \leftarrow \Delta k - |Q| + 1$.
(4) Output P.

Correctness and runtime analysis of the algorithm can be found in [10], here we only explain the parts that are needed in our discussion. Intuitively, the **mmg** algorithm starts with a most general pattern, and continues making the patterns more specific until no further refinement can be performed.

To make the search for Q at step (3) efficient, the **mmg** algorithm looks for candidates for Q from only a subset $\rho(p)$ of refinements of p of cardinality polynomial in the length of p. Such a ρ, called a *refinement operator*, is said to be *complete with respect to the class $\mathcal{P}$* just in case for all $p, p' \in \mathcal{P}$, $p' \preceq p \Leftrightarrow p' \in \rho^+(p)$, where ρ^+ is the transitive closure of ρ. It has been shown that for a complete refinement operator ρ, for any p, such Q that fulfill the requirement at step (3) exists if and only if some $Q'' \subseteq \rho(p)$ also fulfills the requirement — hence limiting the search to $\rho(p)$ does not make the search less complete.

In short, **mmg** solves **MMG** for a class $\mathcal{P}$ when:

1. There is a most general pattern $\perp$ for $\mathcal{P}$, and
2. There is a complete refinement operator ρ with respect to the class $\mathcal{P}$.

Note that whether ρ is efficient or not affects only the runtime of the algorithm. In our experiments we will use different classes for $\mathcal{P}$ to obtain *mmg*s of different forms. The operators we use for these classes $\mathcal{P}$ will be efficient, but regrattably, not complete in every case. In such cases, we consider the outputs to be heuristically obtained patterns rather than actual *mmg*s, even though for convenience we call these outputs *mmg*s nonetheless in this thesis. We are not too concerned with this shortcoming since it does not directly affect our comparisons of the coverages of the outputs.

Our implementation of the algorithm is in C, and compiles on GCC 3.3 and 3.4 (on both `gcc` and `g++`). The program supports most of the modifications studied in the earlier publications [6, 78], and provides other useful features such as the automatic compilation of alphabet from the input sample (that is, the alphabet for the amino acids is not coded into the program). We also enabled the program to produce more than one *mmg*s on a given sample set, by letting **mmg** continue on a few different branches at step (3.1), each branch with a different order on the patterns in Q to refine. This feature is used in Section 6.1.5.1.

6.1 Signal peptide prediction

Arimura *et al.* first considered using *mmg*s obtained over the class **Preg** as gene function predictors [6]. The samples they used are the genes known as *signal peptides*. In their experiments, *mmg*s were first obtained using only small portions of the signal peptides. The obtained *mmg*s were then used to match the remaining signal peptides as well as samples of non-signal peptides. The accuracies of these matches were taken and evaluated. The results from their experiments were in general positive about the use of *mmg*s as predictors for the signal peptides.

Their experiments were subsequently repeated by Yamaguchi *et al.* [84]. Then, in an experiment by Takae *et al.* [78], it was observed that for any number k of languages allowed in a union, the k-*mmg*s found from the class of sort regular patterns (i.e. regular patterns with sort symbols, refer Section 2.3.2.4) achieved very much higher accuracies than those from the class of regular patterns.

The higher accuracies of these *mmg*s of the sort regular pattern languages are due to the lesser generalization they achieved. To see this, first note that any sort regular pattern can be expressed as the union of a few regular patterns. For example, $x_1[AC]x_2$ is the union of x_1Ax_2 and x_1Cx_2. Hence allowing the use of sort symbols achieves similar effects as allowing more languages to be in a union, and as a result, the obtained *mmg*s are more specific. To study this effect, we can perhaps compare the k-*mmg*s of sort regular patterns to the k'-*mmg*s of regular patterns for some k' that is "suitably" larger than k. However, such k' and k cannot be straightforwardly decided from the alphabet size, or the number of occurrences of sort symbols that appeared in the k-*mmg*. This is because allowing for more languages in a union provides more flexibility than simply allowing the use of sort symbols, since patterns in a union can bear no similarity at all to each other. In a more delicate example, we see that the union of the languages of Ax_1C and Cx_1A cannot be expressed as the language of a single sort regular pattern.

To see this difficulty in comparisons, note an experiment (in Section 6.1.5.3) where we show that with respect to accuracies in signal peptide prediction, a 4-*mmg* of sort regular pattern languages with at most 3 occurrences of sort symbols allowed in each pattern is somewhat comparable to an 8-*mmg* of regular pattern languages. These accuracies seem to be difficult to predict for any given number of unions allowed. Nevertheless, our results will show that these accuracies are indeed due to the reduction in generality achieved by the *mmg*s of sort regular pattern languages.

6.1.1 Target pattern classes and refinement operators

We use similar pattern classes as in [78], where only up to a number of occurrences of the letters in $\Pi - \Sigma$ and $\{*\}$ are allowed in each pattern. In this sense, the regular patterns are simply patterns where 0 occurrence of the letters in $\Pi - \Sigma$ are allowed. Let $\mathcal{P}(m, s)$ denote the class of patterns with at most m occurrences of variables and at most s occurrences of the letters in $\Pi - \Sigma$ in each pattern. It is clear that for each m, $s \in N$, $\mathcal{P}(m, s) \subset \mathcal{P}(m + 1, s)$ and $\mathcal{P}(m, s) \subset \mathcal{P}(m, s + 1)$. For each class $\mathcal{P}(m, s)$, we use the following as refinement operator.

$\rho(p)$
(1) Let $P \leftarrow \emptyset$.
(2) For each variable occurrence '$*$' in p and
  for each q in $\{*[\Sigma]*, *[\Sigma], [\Sigma]*, \varepsilon\}$,
  (2.1) Let p' be the pattern obtained from p by replacing the $*$ with q,
  (2.2) if $p' \in \mathcal{P}(m, s)$, add p' to P.
(3) For each sort symbol occurrence $[S]$ in p, for each $S' \in \{S - \{X\} \mid X \in S\}$,
  (3.1) Add the pattern obtained by replacing $[S]$ with $[S']$ to P.
(4) Output P.

Hence in this case, $card(\rho(p))$ is linear in the length of p, $|p|$ say, and furthermore, $\rho(p)$ can be computed in time $O(|p|)$. Note that this operator is not complete with respect to any class $\mathcal{P}(m, s)$ where $1 \leq m \leq 2$ and $s \geq 1$. As an example, let A, $C \in \Sigma$, $p = *A[AC]A*$ and $q = *$. Clearly, p, $q \in \mathcal{P}(2, 1)$ and $p \preceq q$, but it can be shown that $p \notin \rho^+(q)$. We let $m = 4$ for our experiments, that is, we consider only the classes $\mathcal{P}(4, s)$ for some $s \in N$. For this reason we also write $\mathcal{P}(4, s)$ simply $\mathcal{P}(s)$.

6.1.2 Signal peptide samples

We obtain signal peptide samples from the same source as in the earlier studies [6, 84, 78], that is, GenBank. Due to accumulation in the past years, the number of available signal peptide sequences has multiplied since the last study by Kasai *et al.*.

Our samples are biosequences selected from the flat files for bacterial (release 137.0), plant (release 138.0) and rodent (release 138.0) sequences from NCBI GenBank [13]. All the signal peptide entries (that is, entries with the primary tag `sig_peptide`) were extracted, and from among these, we discarded the entries where the corresponding coding sequence (CDS)

1. has more than one translation table specified, or
2. translates into a sequence different from that given in the file.

We also removed the signal peptide entries which

1. translates into a sequence containing the '$*$' symbol, or
2. does not begin with a Methionine, or where
3. the location specified is fuzzy, or references an external sequence.

Finally, we remove the prefixing Methionine from each sequence obtained. We denote this set of sequences by $\texttt{POS}_T$, where T is a sequence type of either bacterial, plant, or rodent. The following table lists the number of signal peptide sequences obtained using this method. Entries in brackets are numbers for the sequences used in [78] (corrected to count only distinct sequences) and the increase comes from the growth in the available sequences from GenBank.

Sequence type T	$card(\texttt{POS}_T)$	$\texttt{POS}_T$ min/max/ave length	$card(\texttt{NEG}_T)$
bacterial	1869 (422)	10 / 31 / 23	18690
plant	1694 (324)	10 / 30 / 22	16940
rodent	1803 (745)	10 / 30 / 21	18030

Table 6.1: sequences used in finding *mmg*s ($\texttt{POS}_T$) and in evaluation ($\texttt{POS}_T$, $\texttt{NEG}_T$)

The negative examples $\texttt{NEG}_T$ for each type T are (1) randomly chosen substrings of (randomly selected) type T CDSs, of a randomly decided length between 21 to 35, which are (2) not a substring of any sequence in $\texttt{POS}_T$. While it is not guaranteed that sequences obtained this way will not turn out to be signal peptides, we trust that this odds is small, and by using a large number of sequences we can reduce its effects.

6.1.3 Measures for evaluating *mmg*s

We now formalize our measures for evaluating an *mmg*.

To measure the accuracy of an *mmg* in signal peptide prediction, for a sequence type T, let $S \subset \texttt{POS}_T$ be the sample used in deriving an *mmg* P, then for each such P and S we define:
1. The *positive accuracy*, $p(P,S) = 100 \times \frac{card((\texttt{POS}_T - S) \cap L_\epsilon(P))}{card(\texttt{POS}_T - S)}$

2. The *negative accuracy*, $n(P) = 100 \times \frac{card(\texttt{NEG}_T - L_\epsilon(P))}{card(\texttt{NEG}_T)}$
3. The *overall accuracy*, $acc(P, S) = \sqrt{p(P, S) \cdot n(P)}$.

For our measurement of coverage we let $n = 30$ (to simplify notation we let $cov(P) = coverage(P, 30)$), since that is roughly the longest length of our positive examples.

6.1.4 Reproduced earlier results using updated samples

We first show that the results obtained in [78] can be reproduced using our new samples. For each T and for each n in $\{50, 100, 150, \ldots, 800\}$, we randomly choose 100 sets of samples $S_1^n, S_2^n, \ldots, S_{100}^n$ each of size n from $\texttt{POS}_T$. From each such S_i^n we obtain a 5-*mmg* P_i^n from the class $\mathcal{P}(9)$. We then compute the average accuracy $\sum_i acc(P_i^n, S_i^n)/100$, of the 100 5-*mmg*s obtained from samples of the size n, and we plot these accuracy values against n to see how this accuracy depends on the input size. These dependencies are similar to those observed in [78]. In our plots (Fig. 6.1) we also show the averages of $n(P_i^n)$ and $p(P_i^n, S_i^n)$ for a better picture of what leads to the trend observed in $acc(P_i^n, S_i^n)$. We see that the positive accuracy increases but negative accuracy decreases as we increase the sample size. This tells us that the *mmg*s produced becomes more general as we add more examples to **mmg**.

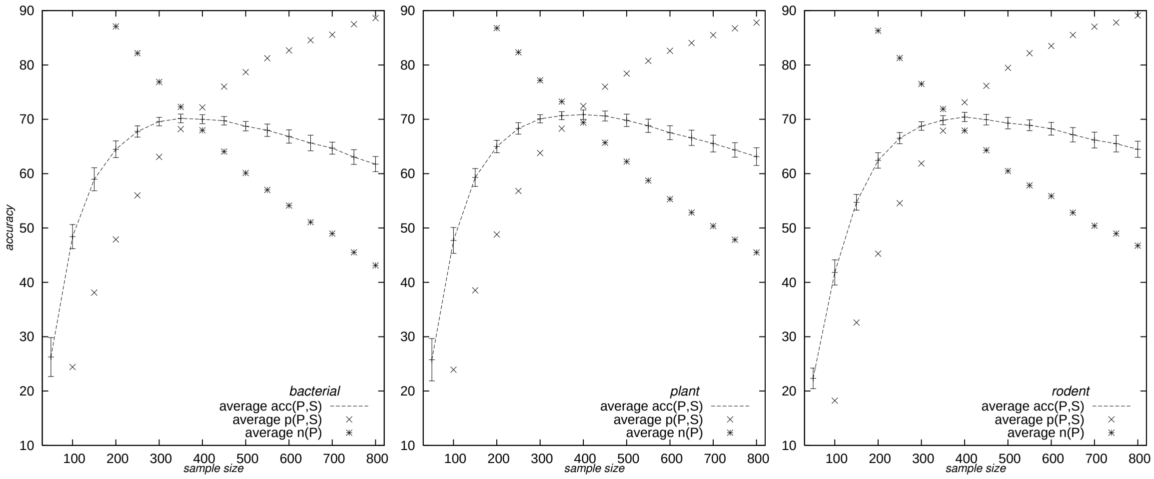


Figure 6.1: accuracies ($acc(P, S)$) vs input sample sizes ($card(S)$) of *mmg*s (as in [78])

6.1.5 Correlation between accuracies and coverages

We now show a strong correlation between the accuracies of the *mmg*s with their coverages. Since results obtained for the plant and rodent samples are exactly identical to those for bacterial samples, we only show the plots for the bacterial samples here.

6.1.5.1 Coverages of *mmg*s from the same k, $\mathcal{P}(s)$ and sample S

We first compare among the *mmg*s obtained from the same sample.

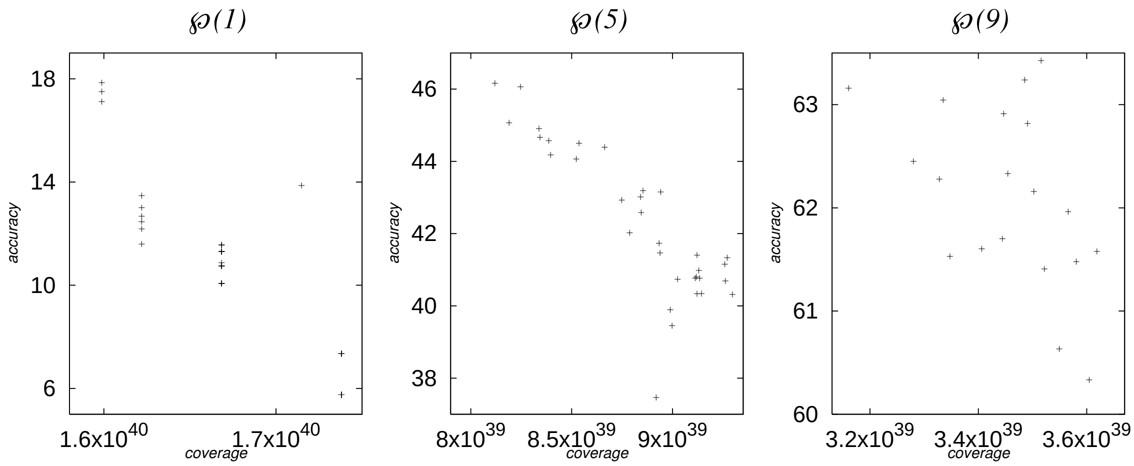


Figure 6.2: $acc(P, S)$ vs $cov(P)$ of 5-*mmg*s obtained from the same sample.

Fig. 6.2 shows accuracies ($acc(P, S)$) against coverage ($cov(P)$) for the 5-*mmg*s obtained from respectively $\mathcal{P}(1)$, $\mathcal{P}(5)$ and $\mathcal{P}(9)$, all from the same sample (of size 800). We see a very close correlation for the case of $\mathcal{P}(1)$ and the correlation becomes less obvious for the case of $\mathcal{P}(9)$, since the accuracies then conglomerate around similar values.

6.1.5.2 Coverages of *mmg*s for various k, $\mathcal{P}(s)$ at different sample sizes

The correlation of accuracies with coverages can be observed across *mmg*s obtained from different samples as well as classes. Fig. 6.3 shows these values for *mmg*s obtained from the classes

(1) 3-*mmg*s from $\mathcal{P}(0)$ (legend $\boxdot$)
(2) 3-*mmg*s from $\mathcal{P}(3)$ (legend $\times$)
(3) 5-*mmg*s from $\mathcal{P}(0)$ (legend $\triangle$)
(4) 5-*mmg*s from $\mathcal{P}(5)$ (legend $\circ$)
(5) 5-*mmg*s from $\mathcal{P}(9)$ (legend $+$)

In each graph in Fig. 6.3 we fix a sample size n, and for each of the classes we obtain 50 *mmg*s, each from a different sample of size n. The values obtained are in consistency with those in Section 6.1.5.1. Interestingly, *mmg*s of similar coverages almost always score about the same accuracies, regardless of k, or the class where the patterns are drawn. We also notice, from the plots for samples of the sizes 50$\sim$150, that *mmg*s with coverages below a certain threshold score lower accuracies as their coverages decrease. We look at this region of the plots more closely in Section 6.1.5.4. When only slightly more examples are available, the *mmg*s of lower coverages almost always perform better than *mmg*s of higher coverages. This suggests that it may be beneficial as a rule to look for hypotheses with lower coverages, i.e. the fittest covers.

One may relate this observation to the fact that when all the elements of a characteristic set of the target hypothesis is represented, it becomes difficult for a hypothesis that contains the examples to be overly specific, and hence a hypothesis with a lower coverage would be closer to the target hypothesis. Ideally, if a characteristic set is represented in the examples, only

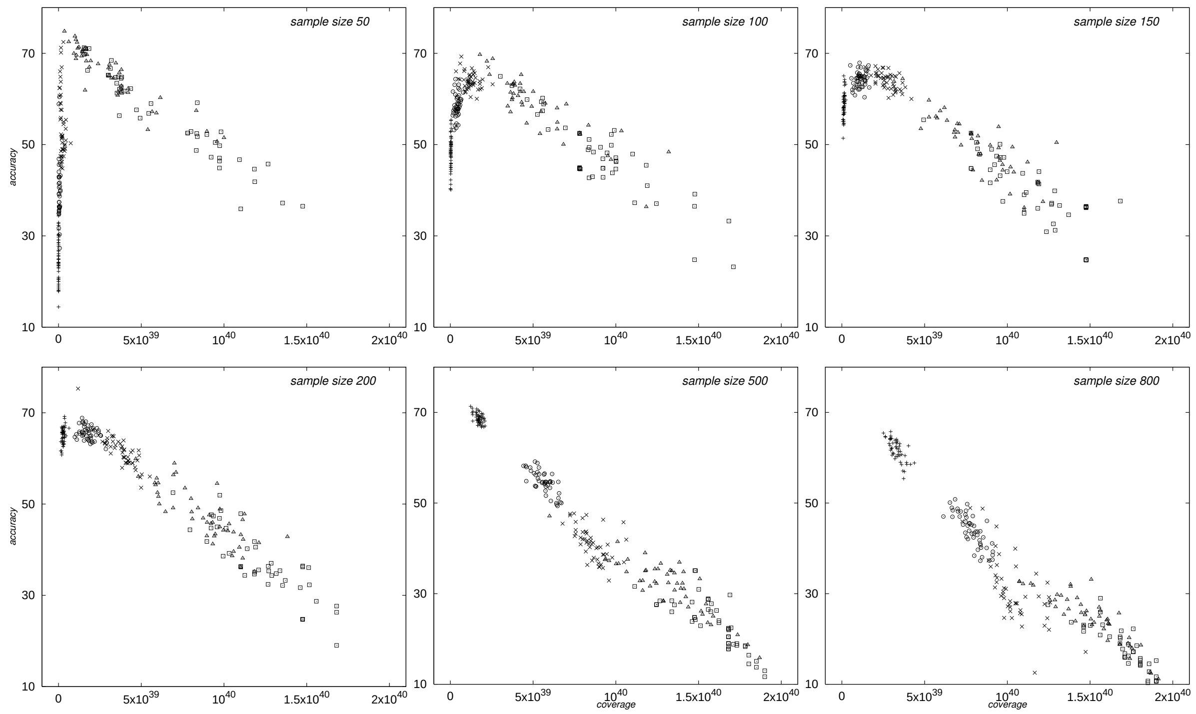


Figure 6.3: $acc(P, S)$ vs $cov(P)$ of *mmg*s obtained from different classes and samples.

$\boxdot$ = 3-*mmg*s from $\mathcal{P}(0)$ $\quad$ $\times$ = 3-*mmg*s from $\mathcal{P}(3)$ $\quad$ $\triangle$ = 5-*mmg*s from $\mathcal{P}(0)$
$\circ$= 5-*mmg*s from $\mathcal{P}(5)$ $\quad$ $+$ = 5-*mmg*s from $\mathcal{P}(9)$.

one *minl*, with the lowest of coverages, should exist in the hypothesis space. For larger sample sizes, since it becomes more probable for a characteristic set to appear, one might expect such a *minl* more likely to be found. However, our conditions are not ideal, with (1) the possibility of having used incorrect examples, (2) an assumed hypothesis space, and (3) our method does not guarantee the finding of such a *minl*, and its behaviour may also be affected by (1) and (2).

6.1.5.3 Similarly accurate *mmg*s of different k, $\mathcal{P}(s)$ at different sample sizes

As discussed, allowing letters in $\Pi - \Sigma$ to occur in patterns achieves similar effects as increasing the number k of patterns allowed in a k-*mmg*. However, given any k-*mmg* from $\mathcal{P}(n)$ for some k and n, it is difficult to decide if some k'-*mmg*s from $\mathcal{P}(n')$ for some $k' \neq k$ and $n' \neq n$ will be similarly accurate for signal peptide prediction. Here we show a specific case where such accuracies are comparable for all the sample sizes used in our tests: the case for the 4-*mmg*s obtained from $\mathcal{P}(3)$ (legend $\boxdot$) and 8-*mmg*s from $\mathcal{P}(0)$ (legend $\times$). This is shown in Fig. 6.4.

Interestingly, while the two different classes produce *mmg*s of similar accuracies, it is very much faster for **mmg** to find 4-*mmg*s in $\mathcal{P}(3)$ than 8-*mmg*s in $\mathcal{P}(0)$, since the runtime of the algorithm increases more in k than in the size of the refinement operator.

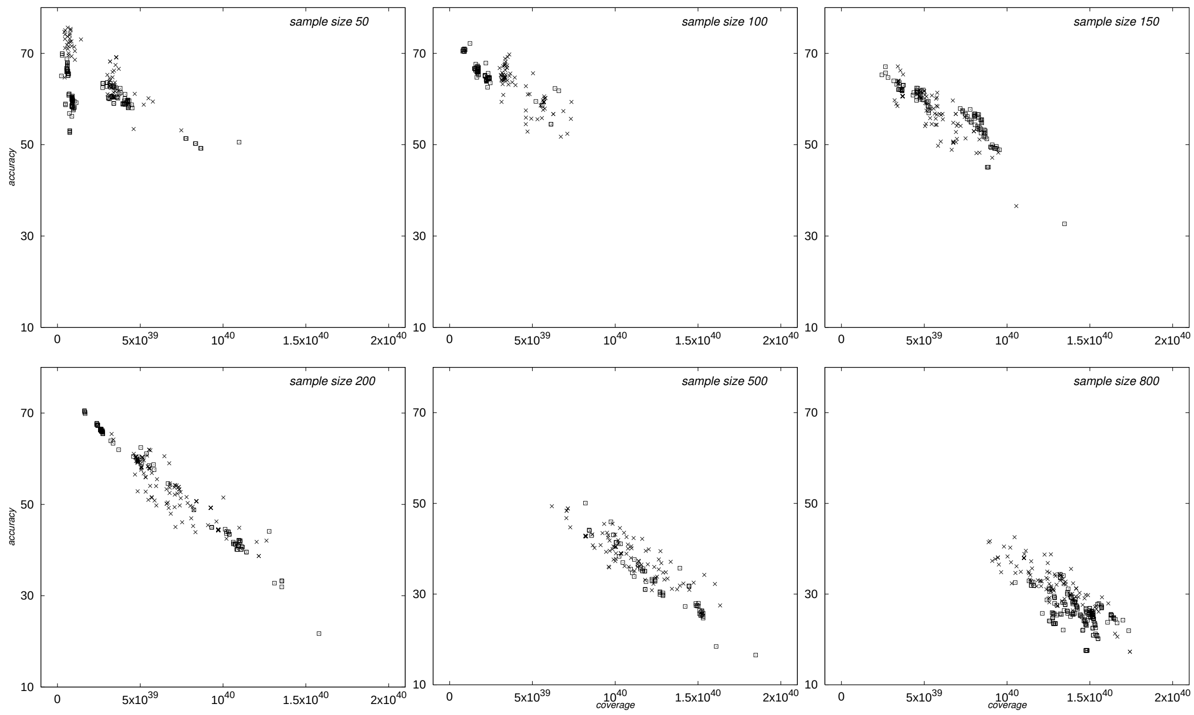


Figure 6.4: $acc(P, S)$ vs $cov(P)$ of 4-*mmg*s from $\mathcal{P}(3)$ and 8-*mmg*s from $\mathcal{P}(0)$.

$\boxdot$ = 4-*mmg*s from $\mathcal{P}(3)$ $\qquad$ $\times$ = 8-*mmg*s from $\mathcal{P}(0)$

6.1.5.4 Overly specific *mmg*s

We now show the details on the positive correlation between accuracies and coverage in the low coverage region in Fig. 6.3, by showing the x-axis in log scale.

In Fig. 6.5 we show respectively, the accuracies, the positive accuracies, and the negative accuracies of

(1) 5-*mmg*s from $\mathcal{P}(4)$ (legend $\boxdot$) (3) 5-*mmg*s from $\mathcal{P}(9)$ (legend $\circ$)
(2) 5-*mmg*s from $\mathcal{P}(5)$ (legend $\times$) (4) 7-*mmg*s from $\mathcal{P}(20)$ (legend $\blacksquare$)

obtained from samples of size 50.

We see a rapid decline of the positive accuracies around the coverage range of $10^{30}{\sim}10^{38}$, and for this region the trend of the accuracies follows closely that of the positive accuracies.

These results show that while in general the *mmg*s obtained in our tests are overly general, and hence as a rule it is preferable to find *mmg*s with lower coverages, we should also be on the guard so that they do not become overly specific, for which we see that the accuracies will go into a rapid decline very suddenly.

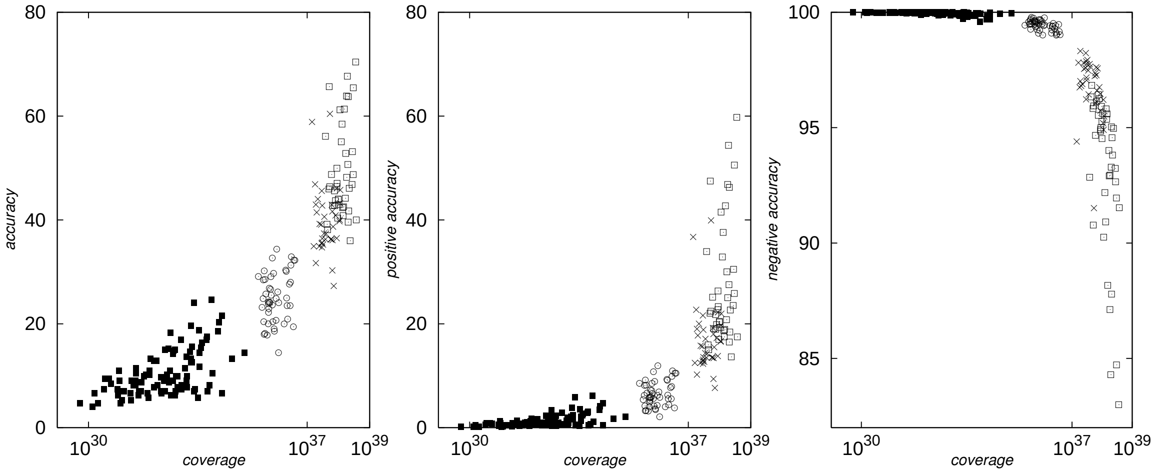


Figure 6.5: details of $acc(P, S)$, $p(P, S)$ and $n(P)$, against $cov(P)$ for samples of size 50

$\boxdot$ = 5-*mmg*s from $\mathcal{P}(4)$ $\times$ = 5-*mmg*s from $\mathcal{P}(5)$
$\circ$ = 5-*mmg*s from $\mathcal{P}(9)$ $\blacksquare$ = 7-*mmg*s from $\mathcal{P}(20)$

6.2 Comparison with known consensus patterns

In this section we see how **mmg** obtained patterns actually compare to already known consensus patterns, and we also see how the coverage measure may help us in finding suitable patterns. As samples, we use the sigma factor dependent promoter sequences in *Bacillus Subtilis*. These biosequences can be easily obtained and their consensus patterns are well established [27].

The difficult part in using these samples is that they are not very abundantly available. This presents a problem for pattern discovery since it becomes difficult for candidate patterns to achieve sufficient levels of statistical significance. To overcome this we constrain our search to only patterns of slightly more specific forms.

6.2.1 Target pattern classes and their refinement operators

For our target pattern class $\mathcal{P}$ we first consider subclasses of the regular patterns and sort regular patterns with restricted use of variables and sort symbols. We also want to consider patterns of the following form, the rational of which will be made clear in its description:

Assume that each of the target patterns consists of exactly two segments. That is, they are of the form $x_1 v x_2 w x_3$, where v, w are elements in Σ^+ and x_1, x_2, $x_3 \in V$. We further assume that v is not too close to the end of the sequences, while w is not too close to the beginning of the sequences. For this reason, we place upper bounds on the maximum length of the strings that can be used to replace x_1 and x_3. In this study we use the same length n as the upper bound for both x_1 and x_2, and we set n to 0.5 times the longest length of the biosequences in the sample. Hence, this class of patterns is "$*(0, n)\ v\ *(0, \infty)\ w\ *(0, n)$". We also consider the class "$*(0,\infty)\ v *(0,\infty)\ w *(0,\infty)$" for a comparison with the above class, to observe the effects brought about by the range specifiers.

This results in the following classes of patterns we use for $\mathcal{P}$:

(A) "$*(0,n)\ v *(0,\infty)\ w *(0,n)$" where v, $w \in (\Sigma \cup \{*\})^*$, regular patterns of a fixed form with range specifiers, and

(B) "$*(0,\infty)\ v *(0,\infty)\ w *(0,\infty)$" where v, $w \in (\Sigma \cup \{*\})^*$, regular patterns of a fixed form without range specifiers,

(C) $\{(\Sigma \cup \{*\})^* \mid$ '$*$' occurs at most 3 times$\}$, that is, regular patterns with at most 3 variable occurrences ($\mathcal{P}(3,0)$ by the definition in Section 6.1.1).

(D) $\{(\Pi \cup \{*\})^* \mid$ '$*$' occurs at most 3 times and symbols in $\Pi - \Sigma$ occur at most 5 times in total$\}$, that is, sort regular patterns with at most 5 uses of sort symbols and with at most 3 variable occurrences ($\mathcal{P}(3,5)$ by the definition in Section 6.1.1).

(E) "$*(0,n)\ v *(0,\infty)\ w *(0,n)$", v, $w \in (\Pi \cup \{*\})^*$, sort regular patterns of a fixed form with range specifiers. We also restrict the occurrences of symbols in $\Pi - \Sigma$ for this class as in pattern class (D).

For classes (C) and (D), we use the refinement operator in Section 6.1.1. For each of the classes (A), (B) and (E) (that is, the patterns of fixed form), we use the following as refinement operator.

$\rho(p)$

(1) Let $P \leftarrow \emptyset$.
(2) Let $\Pi' = \{*A, A* \mid A \in \Sigma\}$.
    (For class (E) only) If there are less than 5 occurrences of symbols from $\Pi - \Sigma$ in p, let $\Pi' = \{*[\Sigma], [\Sigma]*\}$.
(3) For each variable occurrence $*$ in p and each q in Π',
    (3.1) Let p' be the pattern obtained from p by replacing $*$ with q.
    (3.2) Add p' to P.
(4) For each sort symbol occurrence $[S]$ in p, for each $S' \in \{S - \{X\} \mid X \in S\}$,
    add the pattern obtained by replacing $[S]$ with $[S']$ to P.
(5) Output P.

It is clear that $card(\rho(p))$ is linear in the length of p, and ρ can be computed in time polynomial to this length.

6.2.2 Samples and method for evaluation

In our tests, the program is given as input a set of biosequences S, a maximum number k of patterns allowed in an *mmg*, and finally, a class of patterns $\mathcal{P}$. Patterns in the output *mmg* are then compared with known consensus patterns. This is repeated for $k = 5$ and $k = 10$, over every combination of the pattern classes $\mathcal{P}$ of (A) to (E) in Section 6.2.1, and all of the candidates for sample S, as follows.

As mentioned, for samples we use the sigma factor dependent promoter sequences in *B. subtilis*. We use the book "*Bacillus Subtilis and Its Closest Relatives: From Genes to Cells*" as reference on concepts and terminology [27]. Biosequence samples and consensus patterns are from the on-line database DBTBS [43].

A *regulon* is a set of promoter sequences that is regulated by the same sigma factor. Among the sigma regulons in *B. subtilis* we use only those regulated by the factors σ^D, σ^E, σ^H and σ^W (that is, the sigD, sigE, sigH, sigW regulons respectively) because the sample sizes are relatively larger and there are relatively long contiguous conserved regions within their consensus patterns. Size and length details of these samples are as follow:

regulon	sample size	average sequence length	maximum sequence length
sigD	33	51	59
sigE	62	44	51
sigH	48	37	50
sigW	32	42	45

To evaluate how a pattern p in an *mmg* fits a sample S, we use the number of sequences in S which matches the pattern p, i.e. $card(S \cap L_\epsilon(p))$. We also evaluate p using its coverage, $coverage(\{p\}, n)$, as defined in the beginning of the chapter. Here, we let n be the longest length of the sequences in the sample, and for convenience refer to the coverage value computed using such an n as simply the *coverage of p*.

6.2.3 Comparison results

Table 6.2, 6.3 and 6.4 show, for each pattern class from (A) to (E), the **mmg** discovered patterns from sigD, sigE, sigH and sigW, along with the (more contiguous parts of) known consensus patterns from DBTBS. We report only patterns in an *mmg* with either very high number of matching sequences or with very low coverage, with emphasis on the former. That is, we do not show patterns which match very few sequences, even when they have very low coverages. Each pattern listed is followed by two values (x, y), where x is the number of sequences in S which matches the pattern, and y is the coverage of the pattern (for simplicity, range specifiers have been ignored in the computation of these coverages).

Table 6.2 shows the results obtained using the pattern classes (A), (B) and (C). These are the results that more closely resemble the known consensus patterns, and we highlight the regions within them with high similarity. Here we see that the use of range specifiers enables **mmg** to not mix up between the two segments in the pattern, which is most evident in the sigD patterns. Without the use of range specifiers, the patterns obtained from fixed form patterns (class (B)) are often not much better than those obtained from regular patterns (class (C)). We also note that good correspondence with the known consensus pattern in Table 6.2 almost always come from patterns with "high" number of matching sequences and "low" coverage, even though we currently lack a method to decide the ideal range of these values.

Table 6.3 and 6.4 show the results for sort regular patterns and sort regular patterns with range specifiers (pattern classes (D) and (E)) respectively. We show only the results using $k = 10$. Results obtained from these classes are typically complicated and resembles very little of the known consensus patterns. However, while these patterns bear little resemblance to the known consensus patterns, it is highly likely that they will function well as gene function predictors, due to their relatively low coverages, similar to what has been observed in Section 6.1. Regrettably due to lack of samples we are unable to test this hypothesis.

Another point to note is that, using more restrictive classes of languages in the **mmg** algorithm gave us consensus patterns that matches the known consensus patterns fairly reasonably, while more general classes tend to yield less resembling patterns. We note that it is actually not uncommon for consensus pattern finders to make assumptions regarding the pattern to look for. Many pattern finding algorithms allow the specifications of various search criteria, such as the maximum length of constant segments in the target patterns [79, 16]. This gives us some legitimacy for our use of more restrictive classes in **mmg**. Of course, one should not interpret this success in the restrictive classes to mean that more restrictive classes necessarily produce better patterns. For example the class generated by "$*(0, x)\ v\ w\ *(0, x)$" where v, $w \in \Sigma^*$, though more restrictive than the class "$*(0, x)\ v\ *(0, \infty)\ w\ *(0, x)$", would not produce the consensus patterns we want in our test cases.

6.3 Conclusions

We have argued that the notion of coverage is closely related to how general or specific is a set of consensus patterns, making the measure an important factor to consider when constructing consensus patterns to use as signal peptide predictors.

We have also seen that such coverage is related to how a pattern obtained from **mmg** resembles well established consensus patterns. This further supports the suggestion that the measure could be important when considering whether to accept a pattern as consensus pattern.

We note also that such coverage can be computed for many kinds of patterns studied in molecular biology [15], for instance the PROSITE patterns [12]. We hope that the measure can be adopted in the analysis of these other types of patterns.

	regular pattern of fixed form with range specifiers (A)		regular pattern of fixed form (B)		regular pattern with at most 3 variable occurrences (C)	
	$k=5$	$k=10$	$k=5$	$k=10$	$k=5$	$k=10$
sigD	*AGC*CG* (14, 2.1×10^{35}) *CAC*CG* (11, 2.0×10^{35}) *AAC*CCGATA* (10, 1.8×10^{33}) *TAC*CCGA* (10, 2.9×10^{34})	*AAC*CCGATAT* (8, 4.3×10^{32}) *AC*ACGA* (7, 6.4×10^{34}) *TTA*CCGA* (5, 8.2×10^{33}) *ATAC*CCGATA* (5, 5.0×10^{32})	*CCGA*C* (21, 8.2×10^{34}) *CG*CGATA* (17, 1.7×10^{34})	*CCGATA*GC* (9, 4.1×10^{33}) *ACG*CGATAT* (8, 1.8×10^{33}) *GCCGATA*AC* (7, 1.0×10^{33})	*TAC*CG* (18, 2.1×10^{35}) *AAAC*CGATA* (11, 2.1×10^{33}) *AGC*CGATA* (9, 7.2×10^{33})	*AAAC*CGATAT* (8, 5.0×10^{32}) *ATACA*CG* (7, 1.7×10^{34}) *TTAC*CCG* (6, 2.9×10^{33})
sigE	*CA*CATA* (42, 8.0×10^{29}) *TAAA*ATA* (20, 3.2×10^{29}) *CA*AATA* (18, 7.9×10^{29})	*TAAA*ATA* (20, 3.2×10^{29}) *CA*AATA* (18, 7.9×10^{29}) *CA*TCATA* (17, 2.0×10^{29}) *CAT*CATAT* (16, 8.3×10^{28}) *TCATA*CATA* (12, 2.3×10^{28})	*CATA*TA* (48, 8.0×10^{29}) *AATA*AAG* (18, 3.3×10^{29}) *TATA*TAG* (8, 3.1×10^{29})	*TCATA*ATA* (23, 7.9×10^{28}) *AATA*AAG* (18, 3.3×10^{29}) *GCATA*ATA* (13, 7.9×10^{28}) *CATA*TGTA* (13, 9.2×10^{28})	*CATA*TA* (48, 8.0×10^{29}) *AATA*TAG* (18, 3.3×10^{29})	*AATA*AG* (22, 7.9×10^{29}) *CATA*TGTA* (13, 9.2×10^{28}) *GCATA*TATA* (13, 7.9×10^{28})
sigH	*AGGA*GAAT* (23, 2.2×10^{28}) *GA*TAA* (22, 6.7×10^{29}) *AGGA*AAAT* (16, 2.2×10^{28}) *GGGAT*GAAT* (7, 5.4×10^{27})	*AGGA*TAA* (16, 7.9×10^{28}) *AAGGA*AAAT* (12, 5.4×10^{27}) *AAGGA*CGAAT* (7, 1.3×10^{27}) *AGGA*TGAAT* (7, 5.4×10^{27}) *GGGAT*GAAT* (7, 5.4×10^{27})	*AA*AAA* (32, 4.2×10^{29}) *GAA*GAAT* (23, 8.0×10^{28}) *AAAGG*GAAT* (11, 5.4×10^{27}) *TAAT*GAAT* (8, 2.2×10^{28})	*AAAA*AAA* (12, 4.9×10^{28}) *AAAGG*GAAT* (11, 5.4×10^{27}) *GAAG*GAAT* (10, 2.2×10^{28}) *TAAT*GAAT* (8, 2.2×10^{28}) *GAATT*AAAT* (7, 5.4×10^{27})	*AA*AAA* (32, 4.9×10^{29}) *GAA*GAAT* (23, 8.0×10^{28}) AAAGG*AT*GAAT* (8, 1.6×10^{26})	*GAA*AAAT* (16, 8.0×10^{28}) *GGAA*GAAT* (13, 2.2×10^{28}) AAAGG*AT*GAAT* (8, 1.6×10^{26}) *GAAGG*GAAT* (8, 5.4×10^{27})
sigW	*TGAAAC*CGTA* (21, 9.9×10^{23}) *TGAAAC*TAG* (11, 3.7×10^{24}) *GAAACTTT*ATAT* (8, 5.3×10^{22})	*TGAAACC*ACGTAT* (8, 1.4×10^{22}) *TTGAAAC*TAG* (6, 8.9×10^{23}) *TGAAAC*TACGTATA* (5, 3.2×10^{21}) *GAAACTTTTT*ATAT* (5, 2.9×10^{21})	Same as patterns obtained from regular patterns of fixed form with range specifiers		*TGAAAC*GTA* (23, 3.7×10^{24}) *TGAAAC*TA*G* (12, 2.4×10^{24}) *GAAACTTT*ATAT* (8, 5.3×10^{22})	A*TGAAAC*TA*G (8, 5.7×10^{23}) *TGAAACC*CCGTAT* (8, 1.4×10^{22}) *TGAAAC*CTA*G (5, 8.9×10^{23}) *GAAACTTTTT*ATAT* (5, 2.9×10^{21})

Consensus patterns sigD: *TAAA*GCCGATAT*; sigE: *CATAT*CATACA*, or *ATATT*CATACA*;
sigH: *AGGTATT*GAATT*; sigW: *TGAAACN*CGTA*

Table 6.2: **mmg** results obtained from pattern classes (A), (B) and (C)

	$k = 10$		Consensus pattern
sigD	*[TA][GTA][TA]AC*CGATAT[TA]*[AGTC]	$(13,\ 6.2 \times 10^{32})$	*TAAA*GCCGATAT*
	*GCC*G[TA]T[TA][CTA]A*[GCA][TA]	$(\ 9,\ 7.2 \times 10^{33})$	
sigE	*CATA[GTC][AC][ATC][AT]*A[GC]*	$(19,\ 1.3 \times 10^{29})$	*CATAT*CATACA*
	*[AGTC]ATA[AT][AGT][AG]T*A[TC]*	$(16,\ 1.7 \times 10^{29})$	*ATATT*CATACA*
	*G[AGC]ATA[GC][AT][AT][GT]*AT*	$(10,\ 3.4 \times 10^{28})$	
sigH	*GAAG[AG][AGT][AGT]*[AGT][AT]AT*	$(14,\ 2.5 \times 10^{28})$	*AGGTATT*GAATT*
	[AGT]GAAT[AT][AGT][AT][AT]AT*	$(13,\ 1.6 \times 10^{28})$	
	*AAA[GC][AG][AG][GT]*AAT*[ATC]	$(11,\ 1.3 \times 10^{28})$	
sigW	*[AC][AGC][TG]GAAAC[ATC]TTT*TA[AT]*	$(10,\ 4.0 \times 10^{22})$	*TGAAACN*CGTA*
	T[AT]TGAAAC[TGC]T[TC][GC]TA[AT]*	$(\ 9,\ 3.2 \times 10^{22})$	

Table 6.3: Results using sort regular patterns (D)

	$k = 10$		Consensus pattern
sigD	*CAAA[GTA][GT]*[GTA][AGTC][GC]CG*	$(\ 9,\ 1.3 \times 10^{34})$	*TAAA*GCCGATAT*
	*[CTA][AGTC]CTA[TA][TA][TA]*CCG*	$(\ 6,\ 9.7 \times 10^{33})$	
sigE	*[AT]AC[TC][ATC][TC][GTC]*ATA*	$(20,\ 3.1 \times 10^{29})$	*CATAT*CATACA*
	*[AGTC]CAT[AC][ATC][ATC][GT]*ATA*	$(19,\ 1.5 \times 10^{29})$	*ATATT*CATACA*
	*[AGTC]TAT[ATC][ATC][ATC][GT]*ATA*	$(14,\ 2.2 \times 10^{29})$	
sigH	*[ATC]AT[AT][GT][AG][AGT]*AAT*	$(14,\ 7.7 \times 10^{28})$	*AGGTATT*GAATT*
	*AC[AG][AGT][AGT][AGT]*AA[TC]*	$(14,\ 3.8 \times 10^{29})$	
	*[AGC]G[ATC]AT[AT][AGT]T[AGT]*AAT*	$(\ 9,\ 1.1 \times 10^{28})$	
	*G[GT]AT[GTC][GT][AGT][AG]*AA*	$(\ 8,\ 4.5 \times 10^{28})$	
sigW	*[AC][ATG][TG]GAAAC[AT]TTT*[GC]TA*	$(\ 8,\ 2.8 \times 10^{22})$	*TGAAACN*CGTA*
	*AA[AT]GAAAC[TC][TGC][TG][TG]*ATA*	$(\ 5,\ 3.0 \times 10^{22})$	

Table 6.4: Results using sort regular patterns with range specifiers (E)

Chapter 7

Conclusions and Future Work

We have hence, seen how a hypothesis that is minimal in the number of its constituent elements (called a fittest cover in this thesis) can be used in learning, and have seen how such a notion of minimality can be related to the minimality with respect to inclusion, via the idea of characteristic sets. This led us to the investigation of sufficient lengths for the elements of characteristic sets, which we did for the unions of some subclasses of the pattern languages, and for the unions of the tree pattern languages.

In this thesis we have also seen what can be potentially involved in a search for such fittest covers, and slightly touched upon their computational complexities. Finally, we have also seen how these fittest covers may be important in practical applications.

Overally, we find the fittest covers to be reasonable as hypotheses for learning, the use of which will let us infer most of the classes previously known to be inferable using *minl*s (that is, the classes with finite elasticity). In terms of computability, an advantage that the fittest covers have is that if for any given set S, there are only finitely many languages in the class that intersects non-emptily with S (that is, the class has finite thickness) and furthermore, these finitely many languages are enumerable, then the fittest covers can be computed. However, whether *minl*s can be computed remains questionable in such situations. In the following table we list some results regarding *minl*s and fittest covers for comparison.

	minl	fittest covers
Property of learnable classes	Finite elasticity (sufficient condition)	Witness of minimality (characterization) e.g. classes with finite elasticity
Computability	Questionable even for classes with finite thickness (and the candidates are enumerable)	Guaranteed if a class has finite thickness (and the candidates are enumerable)
Computational complexity	Efficient when *mmg*s can be efficiently found, and are *minl*s	Efficient for substring patterns; general case unknown

Many questions remain in the enquiry of the fittest covers. Besides the questions raised at the end of each of the chapters, several others come to mind:

1. The polynomial time algorithm **mmg** by Arimura *et al.*, discussed in Chapter 6, outputs one *mmg* in each computation. However, there are potentially many such *mmg*s in a given problem instance. For the purpose of our investigation in Chapter 6, it would have been more practical if we had an algorithm that enumerates all of these *mmg*s. Hence,
(i) what is the number of possible *mmg*s for a given instance of the **MMG** problem, and
(ii) whether these *mmg*s can be enumerated in time polynomial to the output size,
seem to us very interesting questions.

2. Yamamoto [85] suggested studying the use of hypotheses with maximum likelihood [20] in inductive inference. The idea is interesting because (1) the principle has a lot of influence in many scientific disciplines, especially in the biological sciences, and (2) many algorithms for finding such hypotheses have been suggested in the past. Other measures such as those of minimizing [41, 42] so-called Hamming distance [26] and Levenshtein distance [40] are also worthy of investigations. Likewise, further studies on the use of the MDL principle [38, 16], besides those [21, 19, 32] through the more general notion of program-size complexity [75, 76], may also yield interesting results.

3. Besides the classes of languages listed at the end of Chapter 4, many interesting classes of languages (e.g. some subclasses of the elementary formal systems [72] and Prolog programs [56]) have been shown to possess finite elasticity, and hence there is a characteristic set for each language in these classes within the class. Investigations on sufficient lengths for these characteristic sets, similar to those carried out in Chapter 4, would be interesting.

4. The study in Chapter 6 raises the question of comparison of expressiveness between subclasses of the pattern languages. More precisely, given a finite set of examples and a few classes of patterns, we want to be able to decide which among the classes is best suited for the examples. This is an important question in the sense that: while we answer questions of "which classes of languages are learnable?", we have to also bear in mind that not every class has the same expressive power, and hence may not be suitable for some purposes. Here, note that learnability in the PAC [81] sense is well characterized [14] in terms of the expressiveness measure of Vapnik-Chervonenkis dimension [82].

Bibliography

[1] A. V. Aho and M. J. Corasick. Efficient string matching: An aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.

[2] D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980. First appeared in STOC'79.

[3] D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.

[4] D. Angluin. Inference of reversible languages. *Journal of the ACM*, 29:741–765, 1982.

[5] S. Arikawa. Elementary formal systems and formal languages — simple formal systems. *Memoirs of the Faculty of Science, Kyushu University Series A, Mathematics*, 24:47–75, 1970.

[6] H. Arimura, R. Fujino, T. Shinohara, and S. Arikawa. Protein motif discovery from positive examples by Minimal Multiple Generalization over regular patterns. In *Proceedings of the Genome Informatics Workshop 1994*, pages 39–48. Universal Academy Press, 1994.

[7] H. Arimura, H. Ishizaka, and T. Shinohara. Learning unions of tree patterns using queries. *Theoretical Computer Science*, 185(1):47–62, 1997. Special Issue on ALT'95.

[8] H. Arimura, T. Shinohara, and S. Otsuki. A polynomial time algorithm for finding finite unions of tree pattern languages. In *Proceedings of Nonmonotonic and Inductive Logic, Second International Workshop*, volume 659 of *Lecture Notes in Artificial Intelligence*, pages 118–131. Springer-Verlag, 1991.

[9] H. Arimura, T. Shinohara, and S. Otsuki. Polynomial time inference of unions of two tree pattern languages. *IEICE Transactions on Information and Systems*, E75-D(7):426–434, 1992. First appeared in ALT'91.

[10] H. Arimura, T. Shinohara, and S. Otsuki. Finding minimal generalizations for unions of pattern languages and its application to inductive inference from positive data. In *Proceedings of STACS 94, 11th Annual Symposium on Theoretical Aspects of Computer Science*, volume 775 of *Lecture Notes in Computer Science*, pages 649–660. Springer-Verlag, 1994.

[11] H. Arimura, T. Shinohara, S. Otsuki, and H. Ishizaka. A generalization of the least general generalization. *Machine Intelligence 13*, 13:59–85, 1994. First appeared as RIFIS Technical Report in 1992.

[12] A. Bairoch. PROSITE: A dictionary of sites and patterns in proteins. *Nucleic Acids Research*, 25(19):2241–2245, 1991.

[13] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler. Genbank: update. *Nucleic Acids Research*, 32(Database-Issue):23–26, 2004.

[14] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

[15] A. Brāzma, I. Jonassen, I. Eidhammer, and D. Gilbert. Approaches to the automatic discovery of patterns in biosequences. *Journal of Computational Biology*, 5(2):277–304, 1998.

[16] A. Brāzma, E. Ukkonen, and J. Vilo. Discovering unbounded unions of regular pattern languages from positive examples. In *Proceedings of Algorithms and Computation, 7th International Symposium, ISAAC'96*, volume 1178 of *Lecture Notes in Computer Science*, pages 95–104. Springer-Verlag, 1996.

[17] J. Case, S. Jain, R. Reischuk, F. Stephan, and T. Zeugmann. A polynomial time learner for a subclass of regular patterns. *Electronic Colloquium on Computational Complexity (ECCC)*, 11(038), 2004.

[18] C. Chan, M. Garofalakis, and R. Rastogi. RE-tree: an efficient index structure for regular expressions. *The VLDB Journal*, 12(2):102–119, 2003.

[19] K. J. Chen. Tradeoffs in inductive inference of nearly minimal sized programs. *Information and Control*, 52:68–86, 1982.

[20] R. A. Fisher. On an absolute criterion for fitting frequency curves. *Messenger of Mathematics*, 41:155–160, 1912.

[21] R. Freivalds. Minimal Gödel numbers and their identification in the limit. In *Proceedings of Mathematical Foundations of Computer Science, 4th Symposium*, volume 32 of *Lecture Notes in Computer Science*, pages 219–225. Springer-Verlag, 1975.

[22] R. Freivalds and R. Wiehagen. Inductive inference with additional information. *Journal of Information Processing and Cybernetics (EIK)*, 15:179–195, 1979.

[23] M. Fulk and S. Jain. Learning in the presence of inaccurate information. *Theoretical Computer Science A*, 161:235–261, 1996.

[24] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[25] S. Goldman and S. Kwek. On learning unions of pattern languages and tree patterns. *Theoretical Computer Science*, 288(2):232–254, 2002. Special Issue on ALT'99.

[26] R.W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29:147–160, 1950.

[27] J. D. Helmann and C. P. Moran. *RNA Polymerase and Sigma Factors*, chapter 21, pages 289–312. American Society Microbiology, 2001.

[28] M. Hirao, H. Hoshino, A. Shinohara, M. Takeda, and S. Arikawa. A practical algorithm to find the best subsequence patterns. In *Proceedings of Discovery Science, Third International Conference, DS 2000*, volume 1967 of *Lecture Notes in Artificial Intelligence*, pages 141–154. Springer-Verlag, 2000.

[29] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[30] S. Jain and E. Kinber. Learning multiple languages in groups. In *Proceedings of Algorithmic Learning Theory, 16th International Conference, ALT 2005*, Lecture Notes in Artificial Intelligence, pages 256–268. Springer-Verlag, 2005.

[31] S. Jain, Y. K. Ng, and T. S. Tay. Learning languages in a union. *Journal of Computer and System Sciences.* to appear.

[32] S. Jain, D. Osherson, J. Royer, and A. Sharma. *Systems that Learn: An Introduction to Learning Theory.* MIT Press, Cambridge, Mass., second edition, 1999.

[33] K. Jantke. Monotonic and non-monotonic inductive inference. *New Generation Computing*, 8:349–360, 1991.

[34] T. Jiang, E. Kinber, A. Salomaa, K. Salomaa, and S. Yu. Pattern languages with and without erasing. *International Journal of Computer Mathematics*, 50:147–163, 1994.

[35] T. Jiang, A. Salomaa, K. Salomaa, and S. Yu. Decision problems for patterns. *Journal of Computer and System Sciences*, 50:53–63, 1995.

[36] I. Jonassen, C. Helgesen, and D. Higgins. Scoring function for pattern discovery programs taking into account sequence diversity. Technical report, University of Bergen, 1996.

[37] S. Kannan, Z. Sweedyk, and S. Mahaney. Counting and random generation of strings in regular languages. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 551–557. Society for Industrial and Applied Mathematics, 1995.

[38] P. Kilpeläinen, H. Mannila, and E. Ukkonen. MDL learning of unions of simple pattern languages from positive examples. In *Proceedings of the Second European Conference on Computational Learning Theory*, volume 904 of *Lecture Notes in Artificial Intelligence*, pages 252–260. Springer-Verlag, 1995.

[39] S. Kobayashi and T. Yokomori. Identifiability of subspaces and homomorphic images of zero-reversible languages. In *Proceedings of Algorithmic Learning Theory, 8th International Conference, ALT'97*, volume 1316 of *Lecture Notes in Artificial Intelligence*, pages 48–61. Springer-Verlag, 1997.

[40] V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics*, 10:707–710, 1966.

[41] M. Li, B. Ma, and L. Wang. Finding similar regions in many strings. In *Proceedings of the Thirty-First Annual ACM Symposium on the Theory of Computation*, pages 473–482. ACM, 1999.

[42] M. Li, B. Ma, and L. Wang. On the closest string and substring problems. *Journal of the ACM*, 49(2):157–171, 2002.

[43] Y. Makita, M. Nakao, N. Ogasawara, and K. Nakai. DBTBS: Database of transcriptional regulation in Bacillus Subtilis and its contribution to comparative genomics. *Nucleic Acids Research*, 32:75–77, 2004.

[44] A. Mitchell. Learnability of a subclass of extended pattern languages. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*. ACM Press, 1998.

[45] A. Mitchell, T. Scheffer, A. Sharma, and F. Stephen. The VC-dimension of subclasses of pattern languages. In O. Watanabe and T. Yokomori, editors, *Proceedings of Algorithmic Learning Theory, 10th International Conference, ALT'99*, volume 1720, pages 93–105. Springer-Verlag, 1999.

[46] T. Motoki, T. Shinohara, and K. Wright. The correct definition of finite elasticity: Corrigendum to identification of unions. In L. Valiant and M. Warmuth, editors, *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, page 375. Morgan Kaufmann, 1991.

[47] Y. Mukouchi. Characterization of finite identification. In K. Jantke, editor, *Analogical and Inductive Inference, Proceedings of the Third International Workshop*, pages 260–267, 1992.

[48] Y. Mukouchi. Containment problems for pattern languages. *IEICE Transactions on Information and Systems*, E75-D(7):420–425, 1992. Special Issue for ALT'01.

[49] Y. K. Ng, H. Ono, and T. Shinohara. Measuring over-generalization in the minimal multiple generalizations of biosequences. In *Proceedings of Discovery Science, 8th International Conference, DS 2005*, volume 3735 of *Lecture Notes in Artificial Intelligence*, pages 176–188. Springer-Verlag, 2005.

[50] Y. K. Ng and T. Shinohara. Inferring unions of the pattern languages by the most fitting covers. In *Proceedings of Algorithmic Learning Theory, 16th International Conference, ALT 2005*, volume 3734 of *Lecture Notes in Artificial Intelligence*, pages 269–282. Springer-Verlag, 2005.

[51] Y. K. Ng and T. Shinohara. Characteristic sets for inferring the unions of the tree pattern languages by the most fitting hypotheses. In *Proceedings of Grammatical Inference: Algorithms and Applications, 8th International Conference, ICGI 2006*, volume 4201 of *Lecture Notes in Artificial Intelligence*, pages 307–319. Springer-Verlag, 2006. to appear.

[52] Y. K. Ng and T. Shinohara. Finding consensus patterns in very scarce biosequence samples from their minimal multiple generalizations. In *Proceedings of Advances in Knowledge Discovery and Data Mining, 10th Pacific-Asia Conference, PAKDD 2006*, volume 3918 of *Lecture Notes in Artificial Intelligence*, pages 540–545. Springer-Verlag, 2006.

[53] H. Ono and Y. K. Ng. Best fitting fixed-length substring patterns for a set of strings. In *Proceedings of Computing and Combinatorics, 11th Annual International Conference, COCOON 2005*, volume 3595 of *Lecture Notes in Computer Science*, pages 240–250. Springer-Verlag, 2005.

[54] D. Osherson and S. Weinstein. Criteria of language learning. *Information and Control*, 52:123–138, 1982.

[55] G.D. Plotkin. A note on inductive generalization. In *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press, 1970.

[56] M. R. K. Krishna Rao. Some classes of prolog programs inferable from positive data. *Theoretical Computer Science A*, 241:211–234, 2000. Special Issue for ALT'96.

[57] M. R. K. Krishna Rao. A class of prolog programs with non-linear outputs inferable from positive data. In *Proceedings of Algorithmic Learning Theory, 16th International Conference, ALT 2005*, volume 3734 of *Lecture Notes in Artificial Intelligence*, pages 312–326. Springer-Verlag, 2005.

[58] D. Reidenbach. A negative result on inductive inference of extended pattern languages. In *Proceedings of Algorithmic Learning Theory, 13th International Conference, ALT 2002*, volume 2533 of *Lecture Notes in Artificial Intelligence*, pages 308–320. Springer-Verlag, 2002.

[59] D. Reidenbach. A discontinuity in pattern inference. In *Proceedings of STACS 04, 21st Annual Symposium on Theoretical Aspects of Computer Science*, volume 2996 of *Lecture Notes in Computer Science*, pages 129–140. Springer-Verlag, 2004.

[60] D. Reidenbach. On the equivalence problem for e-pattern languages over small alphabets. In *Proceedings of Developments in Language Theory, 8th International Conference, DLT 2004*, volume 3340 of *Lecture Notes in Computer Science*, pages 368–380. Springer-Verlag, 2004.

[61] D. Reidenbach. On the learnability of e-pattern languages over small alphabets. In *Proceedings of the Seventeenth Annual Conference on Computational Learning Theory*, volume 3120 of *Lecture Notes in Computer Science*, pages 140–154. Springer-Verlag, 2004.

[62] I. Rigoutsos and A. Floratos. Combinatorial pattern discovery in biological sequences: the TEIRESIAS algorithm. *Bioinformatics*, 14(1):55–67, 1998.

[63] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. Reprinted by MIT Press in 1987.

[64] H. Sakamoto. Finding a one-variable pattern from incomplete data. In *Proceedings of Algorithmic Learning Theory, 9th International Conference, ALT'98*, volume 1501 of *Lecture Notes in Artificial Intelligence*, pages 234–246. Springer-Verlag, 1998.

[65] M. Sato. Inductive inference of formal languages. *Bulletin of Informatics and Cybernetics*, 27(1):85–106, 1995.

[66] M. Sato, Y. Mukouchi, and D. Zheng. Characteristic sets for unions of regular pattern languages and compactness. In *Proceedings of Algorithmic Learning Theory, 9th International Conference, ALT'98*, volume 1501 of *Lecture Notes in Artificial Intelligence*, pages 220–233. Springer-Verlag, 1998.

[67] A. Shinohara. String pattern discovery. In *Proceedings of Algorithmic Learning Theory, 15th International Conference, ALT 2004*, volume 3244 of *Lecture Notes in Artificial Intelligence*, pages 1–13. Springer-Verlag, 2004.

[68] T. Shinohara. Polynomial time inference of extended regular pattern languages. In *RIMS Symposia on Software Science and Engineering, Kyoto, Japan*, volume 147 of *Lecture Notes in Computer Science*, pages 115–127. Springer-Verlag, 1982.

[69] T. Shinohara. Polynomial time inference of pattern languages and its application. In *Proceedings of the 7th IBM Symposium on Mathethematical Foundations of Computer Science*, pages 192–209, 1982.

[70] T. Shinohara. Inferring unions of two pattern languages. *Bulletin of Informatics and Cybernetics*, 20:83–88, 1983.

[71] T. Shinohara. Inductive inference of monotonic formal systems from positive data. *New Generation Computing*, 8:371–384, 1991. First appeared in ALT'90.

[72] T. Shinohara. Rich classes inferable from positive data: Length–bounded elementary formal systems. *Information and Computation*, 108:175–186, 1994. First appeared in COLT'90.

[73] T. Shinohara and H. Arimura. Inductive inference of unbounded unions of pattern languages from positive data. *Theoretical Computer Science A*, 241:191–209, 2000. Special Issue for ALT'96.

[74] C.J. Sigrist, L. Cerutti, N. Hulo, A. Gattiker, L. Falquet, M. Pagni, A. Bairoch, and P. Bucher. PROSITE: A documented database using patterns and profiles as motif descriptors. *Brief Bioinformatics*, 3:265–274, 2002.

[75] R. J. Solomonoff. A formal theory of inductive inference, Part I. *Information and Control*, 7:1–22, 1964.

[76] R. J. Solomonoff. A formal theory of inductive inference, Part II. *Information and Control*, 7:224–254, 1964.

[77] F. Stephan. Noisy inference and oracles. *Theoretical Computer Science A*, 185:129–157, 1997.

[78] T. Takae, T. Kasai, H. Arimura, and T. Shinohara. Knowledge discovery in biosequences using sort regular patterns. Workshop on Applied Learning Theory, 1998.

[79] B. Timothy and L. Elkan. The value of prior knowledge in discovering motifs with MEME. In *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology, ISMB'95*, pages 21–29. AAAI, 1995.

[80] J. Uemura and M. Sato. Compactness and learning of classes of unions of erasing regular pattern languages. In *Proceedings of Algorithmic Learning Theory, 13th International Conference, ALT 2002*, volume 2533 of *Lecture Notes in Artificial Intelligence*, pages 293–307. Springer-Verlag, 2002.

[81] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[82] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[83] K. Wright. Identification of unions of languages drawn from an identifiable class. In R. Rivest, D. Haussler, and M. Warmuth, editors, *Proceedings of the Second Annual Workshop on Computational Learning Theory*, pages 328–333. Morgan Kaufmann, 1989.

[84] M. Yamaguchi, S. Shimozono, and T. Shinohara. Finding minimal multiple generalization over regular patterns with alphabet indexing. In *Proceedings of the Seventh Workshop on Genome Informatics*, volume 7, pages 51–60. Universal Academy Press, 1996.

[85] A. Yamamoto. Generalization and likelihood as preference. In *Proceedings of the Workshop on Learning with Logics and Logics for Learning (LLLL)*, pages 53–56, 2005.

[86] T. Zeugmann and S. Lange. A guided tour across the boundaries of learning recursive languages. In K. Jantke and S. Lange, editors, *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *Lecture Notes in Artificial Intelligence*, pages 190–258. Springer-Verlag, 1995.