Proc. the 11th Symposium on Theoretical Aspect of Computer Science (STACS'94),
LNCS 775, Springer-Verlag, 649-660, Feb. 1994

Finding Minimal Generalizations for Unions of Pattern Languages and Its Application to Inductive Inference from Positive Data.

Hiroki ARIMURA Takeshi SHINOHARA Setsuko OTSUKI

Department of Artificial Intelligence
Kyushu Institute of Technology
Kawazu 680-4, Iizuka 820, JAPAN
{arim, shino, otsuki}@ai.kyutech.ac.jp

Abstract

A pattern is a string of constant symbols and variables. The language defined by a pattern p is the set of constant strings obtained from p by substituting nonempty constant strings for variables in p. In this paper we are concerning with polynomial time inference from positive data of the class of unions of a bounded number of pattern languages. We introduce a syntactic notion of minimal multiple generalizations (mmg for short) to study the inferability of classes of unions. If a pattern p is obtained from another pattern q by substituting nonempty patterns for variables in q, q is said to be more general than p. A set of patterns defines a union of their languages. A set Q of patterns is said to be more general than a set P of patterns if for any pattern p in P there exists a more general pattern q in Q than p. Clearly more general set of patterns defines larger unions. A k-minimal multiple generalization (k-mmg) of a set S of strings is a minimally general set of at most k patterns that defines a union containing S. The syntactic notion of minimality enables us to efficiently compute a candidate for a semantically minimal concept. We present a general methodology for designing an efficient algorithm to find a k-mmg. Under some conditions an mmg can be used as an appropriate hypothesis for inductive inference from positive data. As results several classes of unions of pattern languages are shown to be polynomial time inferable from positive data.

1 Introduction

Inductive learning or inductive inference is a process to find general concepts from their concrete examples. As such a process, we can think of program synthesis from examples, grammatical inference, knowledge discovery in databases, etc. Studies on inductive learning vary from theories to practices. One of the most important objectives might be to find frameworks for inductive learning that are well founded by theories and suitable for practical applications. A framework of polynomial time PAC (probably approximately correct) learning [Val84] has been paid much attention from both sides of theory and practice. In this paper, we adopt another framework called *polynomial time inference from positive data* based on identification in the limit [Gol67] which has also been considered as a useful model for practical applications [Shi82b, Shi82a, Nix83, TY92].

As concepts to be learned, in this paper, we are mainly concerned with pattern languages. A pattern is a string consisting of constant symbols and variables. For example, $p = 0x1x$ is a pattern, where 0 and 1 are constant symbols and x is a variable. The language defined by a pattern is the set of constant strings obtained from the pattern by substituting nonempty constant strings for each variable. The language defined by a pattern $p = 0x1x$ is $L(p) = \{\ 0w1w \mid w \in \Sigma^+\}$.

The class of pattern languages was introduced by Angluin [Ang79] as a class inferable from positive data. Although the class of pattern languages is so simple, it has been considered

as a very important class for studies on inductive learning [LW91, Muk92]. For example, elementary formal systems (EFSs for short), which were introduced by Smullyan [Smu61] and proposed as a unifying framework for language learning by Arikawa et. al. [ASY92], can be considered as natural extensions of patterns. Even from the viewpoint of practical applications, pattern languages have been paid much attention. Miyano et. al. [MSS91] considered the PAC learnability of EFS languages, and showed considerably successful experiments on some identification problems in Molecular Biology [AKM$^+$92].

The weakness of inductive learning from positive data relative to that from positive and negative data is well known since the seminal study by Gold [Gol67]. However, there might be so many cases where only positive data are available. Although the class of pattern languages is inferable from positive data indeed, it is too restricted to be applied to practical problems. Fortunately, unions of a bounded number of pattern languages construct a richer class inferable from positive data [Shi83, Wri89b]. In this paper, we consider several subclasses of unions of pattern languages for which an efficient learning algorithm exists.

One of the most important problems for algorithms to infer concepts from positive data is how to avoid *overgeneralizations*. A straightforward way to solve this is to find the least or minimal concepts (minl for short) containing a given examples. It is not the case that such a way results in a correct identification in the limit. If, however, the class in problem has a property called "finite thickness" or "finite elasticity", then we can construct an inference algorithm utilizing minl. We say a class has finite thickness when it has finitely many concepts containing an arbitrarily given example. For example, it is easy to see that the class of ground instances of first-order terms has finite thickness. In this paper first-order terms are called tree patterns and sets of their ground instances tree pattern languages. Since a minl of first-order terms can be uniquely determined as the least generalization in polynomial time [Plo70], the class of tree pattern languages is polynomial time inferable from positive data. On the other hand, for the class of pattern languages there are possibly many minl's. It should be noticed that the computability of finding one of the minl's does not always require the decidability of containment for the class. In reality, although the containment for pattern languages is undecidable [JSSY93], minl is computable but NP-hard [Ang79]. For some subclasses of pattern languages both containment and minl are polynomial time computable. A pattern is said to be one-variable if it contains at most one variable. A pattern is said to be regular if each variable appears at most once. Both the class of one-variable pattern languages and the class of regular pattern languages have a polynomial time minl algorithm.

Let us consider the class $\mathcal{C} = R_1, R_2, \ldots$ of sets of positive integers, where $R_i = \{i \times n \mid n = 1, 2, \ldots\}$. Clearly $\mathcal{C}$ has finite thickness. If a set of integers $S = \{2, 3, 4\}$ is given, the minimal set containing S within $\mathcal{C}$ is $R_1 = \{1, 2, 3, 4, 5, \ldots\}$. On the other hand, if we consider within the class of unions of two sets taken from $\mathcal{C}$, then $R_2 \cup R_3 = \{2, 3, 4, 6, \ldots\}$ is minimal. In this paper we consider the minl of this kind using unions of concepts instead of single concept. More precisely, we consider inductive learning and minl for the class of unions $\mathcal{C}^n = \{R_1 \cup R_2 \cup \ldots \cup R_n \mid R_i \in C\}$ for some class $\mathcal{C}$ and some integer n. Wright introduced the notion of finite elasticity which is a natural extension of finite thickness. He showed that finite elasticity is a sufficient condition for a class to be inferable from positive data and it is closed under unions [Wri89b]. From his study we know the class $\mathcal{C}^n$ is inferable for any n if $\mathcal{C}$ has finite thickness. Note that the class $\mathcal{C}^*$ of unions of unbounded finitely many concepts is out of the scope of Wright's theorem.

In this paper the class, from which unions are taken, is assumed to be constructed by using a generalization. We call such a framework a generalization system. A generalization system has a set D of descriptions partially ordered by an effectively decidable relation $\preceq$ with the greatest element $\top$. When $p \preceq q$, we say q is a generalization of p or p is an instance of q. The set U of minimal descriptions can be used as a universe of concepts. A member of U is also called an object. A description p in D represents a concept $L(p)$ consisting of objects that are instances of p. For example, the class of pattern languages, the class of tree pattern languages, and the class of integer multiples above can be considered as classes constructed by generalization systems. Clearly from the definitions, the more general description defines the

Pattern Language Class $\mathcal{C}$	$\mathcal{C}^1$	$\mathcal{C}^2$	$\mathcal{C}^k$
patterns	D [Ang79]	D [Shi83]	D [Wri89b]
one-variable patterns	P [Ang79]	P/2 [Wri89a]	P/k this work
regular patterns	P [Shi82b]	—	P/2km this work (m-variable regular)
tree patterns	P [Plo70]	P/2 [ASO92]	P/k [ASO93]

Figure 1: Summary of known results for inferability of unions of a bounded number of pattern languages from positive data, where D and P denote Effective and Polynomial time inferability, respectively, and P/c means that P holds when $\sharp\Sigma > c$.

larger concept, that is, $p \preceq q \Longrightarrow L(p) \subseteq L(q)$. In general the converse does not hold. If the converse holds, the generalization system is called complete. In many cases to find a minimal common generalization (mcg for short) is much easier than to find a minimal concept itself. For example, when we can make descriptions less general step by step, a simple greedy search method which starts from the greatest element $\top$ and tries to make it as specific as possible works as a correct mcg algorithm. We call such operations to make descriptions less general step by step a stepwise refinement. Clearly, if the generalization system is complete, then any mcg algorithm can be used as a minl algorithm. Our main purpose in this paper is to extend the scope of the mcg algorithm using generalization to the class of unions and establish a general method to design efficient inductive learning algorithms.

The most important thing is how to extend the generalization relation $\preceq$ to the unions. A set P consisting of at most k descriptions is called a k-multiple description. The concept $L(P)$ defined by a multiple description P is the union $\cup_{p\in P}L(P)$. In this paper we deal with the Hoare powerset ordering $\sqsubseteq$. For multiple descriptions P and Q, $P \sqsubseteq Q$ iff for any $p \in P$, $p \preceq q$ for some $q \in Q$. Clearly $P \sqsubseteq Q$ implies $L(P) \subseteq L(Q)$. The converse does not hold in general, even in a complete generalization system. A k-minimal multiple generalization (k-mmg for short) for a set S of objects is a k-multiple description P such that $S \subseteq L(P)$ and for no k-multiple description $Q \sqsubset P$, $S \subseteq L(Q)$. We can show that k-mmg for a class $\mathcal{C}$ is computable if $\mathcal{C}$ is an effective generalization system. However, it is not easy to find a k-mmg in polynomial time, even if mcg is polynomial time computable. Fortunately for the class of unions of tree pattern languages we already know an efficient k-mmg algorithm [ASO93]. When $L(p) \subseteq L(q_1) \cup \ldots \cup L(q_m)$ implies $L(p) \subseteq L(q_i)$ for some i, we say that the class of unions has compactness with respect to containment. The class of tree pattern languages is a complete generalization system and the class of unions of at most k tree pattern languages has compactness with respect to containment under the condition that the number of function symbols is more than k. Therefore using the k-mmg algorithm for tree pattern languages we can construct an efficient inference machine that infers unions of k tree pattern languages from positive data.

In this paper, we extract the essence of the efficiency of the polynomial time k-mmg algorithm for tree patterns to establish a general methodology of designing efficient k-mmg algorithms for generalization systems. Our algorithm searches multiple descriptions in the direction from the most general $\{\top\}$ to more specific ones. We say a multiple description P is reduced with respect to a finite set S of objects, if $L(P)$ contains all objects in S but $L(P')$ cannot contain some objects in S for any proper subset P' of P. In other words, reduced description has no redundancy in covering all the given objects. First we try to find a reduced multiple description P consisting of at most k descriptions. Then we try to refine P to be more specific as long as consistency with given examples is maintained. Furthermore, if the number of descriptions in P is less than k, we try to divide some component of P as long as the total number of descriptions does not exceeds the bound k.

Our method extracted from the k-mmg algorithm for tree patterns is not easily applied to

patterns. In general, the relation $\preceq$ on patterns including the membership problem for patterns is NP-complete. Furthermore $L(p) \subseteq L(q)$ does not always imply $p \preceq q$. Some subclasses such as one-variable patterns and regular patterns comprise an efficient and complete generalization system. Even such a subclass does not have compactness with respect to containment in general. However we can compute an mmg in polynomial time for both classes by using stepwise refinement. If the alphabet contains more than $2km$ constant symbols, then the class of unions of k regular pattern languages with at most m variables has compactness with respect to containment, and the k-mmg algorithm can work as a correct minl algorithm for unions. Therefore, the class of unions of at most k m-variable regular pattern languages is identifiable in the limit from positive data with consistent and conservative polynomial time update, if the number of constant symbols is greater than $2km$. Also a similar result can be obtained for the class of unions of one-variable pattern languages.

2 Preliminaries

First we introduce the class of pattern languages [Ang79], which is one of the most important generalization systems dealt with in this paper, and give the notion of identification in the limit from positive data [Gol67], which we adopt as a mathematical model of inductive learning. Furthermore we present generalization systems to capture properties common to systems such as patterns and tree patterns.

2.1 Pattern languages

For a set A of symbols, we denote by A^+ the set of nonempty finite strings over A, by $\sharp A$ the number of the elements in A. For a string w, we denote by $|w|$ the length of w. For a finite set S of strings, we denote by $|S|$ the total length of the strings in S. For $a \in A$, $A \setminus a$ denotes the set $\{x \in A \mid x \neq a\}$.

Let Σ be a finite set of *constant* symbols and X be a countable set of *variables* disjoint from Σ. A *pattern* is an elements in $(\Sigma \cup X)^+$. We denote the set of all patterns by $\mathcal{P}$. A *substitution* is a homomorphism from $\mathcal{P}$ to $\mathcal{P}$ that maps every constant symbol to itself. A set of replacements $\{x_1 := p_1, \ldots, x_n := p_n\}$ denotes the substitution that maps variable x_i to pattern p_i and any other variable to itself. By $p\theta$ we denote the image of a pattern p by a substitution θ. We define a binary relation $\preceq$ on $\mathcal{P}$ by $p \preceq q \Longleftrightarrow p = q\theta$ for some substitution θ. The language defined by a pattern p is the set $L(p) = \{w \in \Sigma^+ \mid w \preceq p\}$. Clearly $p \preceq q \Longrightarrow L(p) \subseteq L(q)$. If a language L is defined by some pattern p, then it is called a *pattern language*. A *one-variable pattern* is a pattern containing at most one variable. A *regular pattern* is a pattern where each variable occurs at most once [Shi82b]. Clearly a regular pattern defines a regular language. We denote by $\mathcal{P}$, $\mathcal{P}_1$, $\mathcal{P}_{\mathrm{reg}}$ the sets of patterns, one-variable patterns, regular patterns, respectively, and by $\mathcal{PL}$, $\mathcal{PL}_1$, $\mathcal{PL}_{\mathrm{reg}}$ the classes of pattern languages, one-variable pattern languages, regular pattern languages, respectively.

For example, let $\Sigma = \{0, 1\}$, $X = \{x, y, z, \ldots\}$. Then $p = xx1y1$ is a pattern that defines $L(p) = \{vv1w1 \mid v, w \in \Sigma^+\} = \{00101, 11101, 00001001, 01011011, \ldots\}$. Pattern p is neither one-variable nor regular. Pattern $q = xx1x1$ is a one-variable pattern, and $r = xy1z1$ is a regular pattern. Note that $L(q) \subsetneq L(p) \subsetneq L(r)$, because for substitutions $\theta_1 = \{y := x\}$ and $\theta_2 = \{y := x, z := y\}$, we have $q = p\theta_1$ and $p = r\theta_2$.

2.2 Inductive inference from positive data

Let U be a recursively enumerable set of *objects*. A *concept* is a subset of U. U is called the *universe* of concepts. An *indexed family of recursive concepts* or simply a *concept class* is a triple $\mathcal{C} = (C, D, L(\cdot))$, where C is a class of concepts, D is a recursively enumerable set of *descriptions*, $L(\cdot)$ is a mapping from D to C such that $C = \{L(p) \mid p \in D\}$, and there exists a

total computable function $f : U \times D \to \{0,1\}$ such that $f(w,p) = 1$ iff $w \in L(p)$. The function f is called a *membership* function. We sometimes do not distinguish $\mathcal{C}$ and C from each other.

Let $\mathcal{C} = (C, D, L(\cdot))$ be a concept class. A *positive presentation* of a concept $L \in \mathcal{C}$ is an infinite sequence $w_1, w_2, \ldots$ of objects drawn from a concept L such that $L = \{\ w \mid w = w_i$ for some $i\ \}$. An *inference machine* is an effective procedure M that runs in stages $1, 2, \ldots$, where M requests an *example* and computes a *hypothesis* based on the examples so far received. Let M be an inference machine and $h_i \in D$ be the hypothesis produced by M at stage i $(i = 1, 2, \ldots)$ when objects $w_1, w_2, \ldots$ are fed to M in this order. M is *consistent* if $L(h_i) \supseteq \{w_1, \ldots, w_i\}$ for any i. M is *conservative* if $h_{i-1} \neq h_i$ only if $w_i \notin L(h_{i-1})$. M is *polynomial time updating* if after receiving w_i M produces h_i within a polynomial time in $|w_1| + \ldots + |w_i|$. M on input $\sigma = w_1, w_2, \ldots$ *converges* to $p \in D$ iff after some finitely many stages it always produces p. M *identifies in the limit* or *infers* a concept L *from positive data* iff for any positive presentation σ of L, M on input σ converges to p with $L = L(p)$. M *infers* a concept class $\mathcal{C}$ *from positive data* iff M infers any concept L in $\mathcal{C}$. A concept class $\mathcal{C}$ is *inferable from positive data* when there exists an inference machine that infers $\mathcal{C}$.

Definition A concept class $\mathcal{C}$ is *polynomial time inferable from positive data* iff there exists a consistent, conservative and polynomial time updating inference machine M that infers $\mathcal{C}$ from positive data.

A concept class $\mathcal{C}$ has *finite thickness* iff $\sharp\{L \in \mathcal{C} \mid w \in L\}$ is finite for any object $w \in U$. A concept class $\mathcal{C}$ has *infinite elasticity* iff there exist infinite sequences $w_0, w_1, w_2, \ldots$ of objects and $p_1, p_2, \ldots$ of descriptions such that $\{w_0, \ldots, w_{i-1}\} \subseteq L(p_i)$ but $w_i \notin L(p_i)$ for any $i = 1, 2, \ldots$. A concept class $\mathcal{C}$ has *finite elasticity* iff $\mathcal{C}$ does not have infinite elasticity. Clearly from the definitions finite thickness implies finite elasticity. The *minl* problem is to find a description representing a minimal concept containing a given finite set of objects. Consider the following inference machine INFER utilizing minl, which receives an infinite sequence $w_1, w_2, \ldots$ of objects and produces an infinite sequence $p_1, p_2, \ldots$ of descriptions.

Algorithm INFER:
Stage 0. Set $S := \phi$, $h_0 :=$ "none", and $i := 1$. Goto stage 1.
Stage i. Receive the next example w_i and add w_i into S. If h_{i-1} is inconsistent with S, that is, $S \not\subseteq L(h_{i-1})$, then set h_i to be a description representing a minimal concept containing S. Otherwise, set $h_i := h_{i-1}$. Output h_i. Set $i := i + 1$. Goto Stage i.

Theorem 2.1 ([Ang79, ASO92]) *If a concept class $\mathcal{C}$ has finite elasticity and the minl problem for $\mathcal{C}$ are computable, then the procedure INFER shown above infers $\mathcal{C}$ from positive data consistently and conservatively. Furthermore, if the membership function and the minl problem are polynomial time computable, then $\mathcal{C}$ is polynomial time inferable from positive data by INFER.*

2.3 Generalization systems

A *generalization system* (*GS* for short) is a partially ordered set $(D, \preceq)$ with the greatest element $\top$. Elements in D are called *descriptions*. If $p \preceq q$ then we say p is a *refinement* of q, q is a *generalization* of p, p is more *specific* than q, or q is more *general* than p. We write $p \prec q$ if $p \preceq q$ but $q \not\preceq p$. In case where $(D, \preceq)$ is not partially ordered set but $\preceq$ is transitive, we take the set $D_{\equiv}$ of representatives of the equivalence classes of D modulo $\equiv$, where $p \equiv q \Longleftrightarrow p \preceq q$ and $q \preceq p$. Hereafter we identify descriptions which are equivalent to each other, and do not distinguish $(D, \preceq)$ and $(D_{\equiv}, \preceq)$ from each other. For any GS $(D, \preceq)$ we assume that the relation $\preceq$ is decidable and there exists recursive function *size* : $D \to N$ such that

- $p \prec q \Longrightarrow \mathrm{size}(p) > \mathrm{size}(q)$ for any $p, q \in D$,
- for almost all $p \in D, \mathrm{size}(p) \leq h(|p|)$ and $|p| \leq h'(\mathrm{size}(p))$ for some recursive functions h and h', and
- $\{p \in D \mid \mathrm{size}(p) \leq n\}$ is finite and computable for any $n \geq 0$,

where $|p|$ is the *length* of p as a representation. A GS $(D, \preceq)$ defines a concept class $(C, D, L(\cdot))$ such that

- The universe U of objects is the set of minimal elements in D.
- $C = \{L(p) \mid p \in D\}$, where the concept defined by $p \in D$ is $L(p) = \{w \in U \mid w \preceq p\}$.

Let S be a subset of D. If $q \preceq p$ holds for any $q \in S$, then p is called a *common generalization* (or *covering*) of S. If p is a common generalization of S and there is no common generalization of S that is properly more specific than p, then p is called a *minimal common generalization* (*mcg* for short) of S. The *mcg problem* is to find a minimal (with respect to $\preceq$) description covering a given finite set. Recall that the *minl problem* is to find a description that represents a minimal (with respect to $\subseteq$) concept. Clearly $p \preceq q \Longrightarrow L(p) \subseteq L(q)$ for any $p, q \in D$ but the converse does not hold in general. We say a GS $(D, \preceq)$ is *complete* iff $p \preceq q \Longleftrightarrow L(p) \subseteq L(q)$.

If p is a covering of S, then $\mathrm{size}(p) \leq \min\{\mathrm{size}(w) \mid w \in S\}$. Therefore, we can effectively find mcgs of S, for any finite set $S \subseteq D$, because there exist only finitely many coverings of S whose sizes are bounded by $\min\{\mathrm{size}(w) \mid w \in S\}$ and the relation $\preceq$ is decidable. We say a GS $(D, \preceq)$ is *efficient* iff

- $\preceq$ is polynomial time computable,
- for almost all $p \in D$, $\mathrm{size}(p) \leq h(|p|)$ and $|p| \leq h'(\mathrm{size}(p))$, for some polynomials h and h', and
- mcg is polynomial time computable.

Theorem 2.2 *Let $(D, \preceq)$ be an efficient GS and $(C, D, L(\cdot))$ be the concept class defined by $(D, \preceq)$. Then*

- *C has finite thickness, and*
- *there exists a polynomial time computable function which decides "$w \in L(p)$" or not.*

Furthermore, if $(D, \preceq)$ is complete, then

- *any mcg algorithm works as a correct minl algorithm, and*
- *C is polynomial time inferable from positive data.*

3 Multiple Generalization Systems

In this section we introduce multiple generalization systems by which unions of concepts are represented.

For a set D of descriptions, we define the set of *k-multiple descriptions* by $D^k = \{P \subseteq D \mid \sharp P \leq k\}$. A *$k$-multiple generalization system* (*k-MGS* for short) *derived from* a GS $(D, \preceq)$ is a partially ordered set $(D^k, \sqsubseteq)$, where the powerset ordering $\sqsubseteq$ is defined by $P \sqsubseteq Q$ iff for any $p \in P$, $p \preceq q$ for some $q \in Q$, $\mathrm{size}(P) = \Sigma_{p \in P}\, \mathrm{size}(p)$, and $|P| = \Sigma_{p \in P}\, |p|$. A k-MGS $(D^k, \sqsubseteq)$ defines a concept class $(C^k, D^k, L(\cdot))$ such that

- the universe U is the set of objects taken from D,
- $L(P) = \cup_{p \in P} L(p)$, and
- $C^k = \{L(P) \mid P \in D^k\}$.

A multiple description P is of *canonical form* if P contains no q such that $q \prec p$ for some $p \in P$. For any $P \in D^k$ there exists a unique $\hat{P}$ of canonical form such that $P \equiv \hat{P}$. Hereafter any multiple description is assumed to be of canonical form. Note that k-MGS $(D^k, \sqsubseteq)$ is a partially ordered set in this case.

A *k-multiple covering* of a finite set S of objects is a k-multiple description P such that $S \subseteq L(P)$. A *k-minimal multiple generalization* (*k-mmg* for short) is a k-multiple covering P of S such that $Q \not\sqsubset P$ for any k-multiple covering Q of S. In other words, the k-mmg is an mcg within a k-MGS $(D^k, \sqsubseteq)$. A *k-minl* is a minl for a concept class $(C^k, D^k, L(\cdot))$. A concept class $(C^k, D^k, L(\cdot))$ has *compactness with respect to containment* iff for any $p, q_1, \ldots, q_m$ $(1 \leq m \leq k)$, $L(p) \subseteq L(q_1) \cup \ldots \cup L(q_m)$ implies $L(p) \subseteq L(q_i)$ for some $1 \leq i \leq m$. Here we should note that compactness with respect to containment for unions and completeness for generalization guarantee $P \sqsubseteq Q \Longleftrightarrow L(Q) \subseteq L(P)$.

$$S = \left\{ \begin{array}{ll} 000, & 002 \\ 001, & 00011 \\ 011, & 02211 \end{array} \right\}, \qquad P = \left\{ \begin{array}{l} 00x \\ 0y1 \end{array} \right\}, \qquad p = 0x$$

Figure 2: A finite set S of strings over $\Sigma = \{0, 1, 2\}$, 2-minimal multiple generalization P of S for pairs of regular patterns, and a minimal common generalization p of S for regular patterns

Recall that in a GS $(D, \preceq)$ there exist only finitely many coverings of S for any finite set S of objects. On the other hand, usually in a k-MGS $(D^k, \sqsubseteq)$ there are infinitely many k-multiple coverings of S, that is, the concept class of unions does not have finite thickness. However, we can effectively find a k-mmg of S by restricting candidates for k-mmg to have no redundant description. A k-multiple covering $P \in D^k$ of S is said to be *reduced*, if $S \subseteq L(P)$ but $S \not\subseteq L(P')$ for any $P' \subsetneq P$.

Lemma 3.1 *Let S be a finite set of objects and P be a k-multiple covering of S that is of canonical form. If P is a k-mmg of S, then P is a reduced covering of S.*

Proof. Assume P is a k-multiple covering of S but not reduced. Then there exists a proper subset $Q \subsetneq P$ such that $S \subseteq L(Q)$. Since both P and Q are of canonical form, $Q \sqsubset P$. Therefore P cannot be a k-mmg of S. □

From Lemma 3.1, any k-mmg of S can be found out of reduced k-multiple coverings of S. If P is a reduced k-multiple covering of S, then for any $p \in P$ there exists at least one object $w \in S$ such that $w \preceq p$. Therefore the set of reduced multiple coverings of S is finite and computable. In fact, it is a subset of a finite set G^k, where $G = \{p \in D \mid w \in L(p) \text{ for some } w \in S\}$. Thus, a simple algorithm that searches for a minimal element in G^k with respect to $\sqsubseteq$ can effectively find a k-mmg of any finite set S. However, this algorithm belongs to exhaustive search methods and it cannot run so efficiently, even if a GS $(D, \preceq)$ is efficient. To realize a polynomial time algorithm to find a k-mmg, we need a stronger lemma than Lemma 3.1 which will be presented in the next section.

Theorem 3.2 *Let $(D^k, \sqsubseteq)$ be a k-MGS derived from an efficient and complete GS $(D, \preceq)$. Then*

- *$\sqsubseteq$ is polynomial time computable,*
- *for almost all $P \in D^k$, $\mathrm{size}(P) \leq h(|P|)$ and $|P| \leq h'(\mathrm{size}(P)))$ for some polynomials h and h',*
- *k-mmg is computable, and*
- *$(C^k, D^k, L(\cdot))$ has finite elasticity.*

Furthermore, if the concept class C^k has compactness with respect to containment and k-mmg is polynomial time computable, then

- *any k-mmg algorithm for $(D^k, \sqsubseteq)$ works as k-minl for C^k, and*
- *C^k is polynomial time inferable from positive data.*

4 Polynomial time minimal multiple generalization algorithm

In this section we present a characterization of the possible forms of k-mmgs and polynomial time algorithm to find a k-mmg based on a greedy search.

First we show a necessary and sufficient condition for a k-multiple description P consisting of *just* k descriptions to be a k-mmg of a finite set of objects. A k-multiple covering P of a finite set S of objects is said to be *tightest* iff any $p \in P$ is an mcg of $S - L(P \setminus p)$. Clearly, if P is a tightest covering of S, then P is of canonical form and reduced covering of S.

Theorem 4.1 *Let S be a finite set of objects, P be a reduced k-multiple covering of S, and $\sharp P = k$. Then P is a tightest k-multiple covering of S iff P is a k-mmg for S.*

Proof. If part is obvious. Let $P = \{p_1, \ldots, p_k\}$ be a tightest k-multiple covering of S and Q be a k-multiple covering of S such that $Q \sqsubseteq P$. Let $S_i = S - L(P \setminus p_i)$ for each $i = 1, \ldots, k$. Note that $S_i \neq \phi$ and $i \neq j \Longrightarrow S_i \cap S_j = \phi$.

Claim 1. For each $q \in Q$, $L(q)$ can have a nonempty intersection with at most one of S_1, S_2, ... , S_k. If $L(q)$ has a nonempty intersection with S_i, then $q \preceq p_i$.

(Proof of Claim 1) Since $Q \sqsubseteq P$, $q \preceq p_i$ for some $i = 1$, ... , k. Assume that $L(q)$ has a nonempty intersection with S_j. Then $\phi \subsetneq S_j \cap L(q) \subseteq S_j \cap L(p_i)$ because $L(q) \subseteq L(p_i)$. If $i \neq j$, then $S_j \cap L(p_i) = \phi$. Therefore we have $i = j$.

Claim 2. For each $i = 1$, ... , k, S_i has a nonempty intersection with $L(q)$ for some $q \in Q$.

(Proof of Claim 2) Clear from the condition $S \subseteq L(Q)$.

From these two claims we have $\sharp Q \geq \sharp P = k$. Since we assume $\sharp Q \leq k$, $\sharp Q = \sharp P = k$. For each $q \in Q$, $L(q)$ has a nonempty intersection with some unique S_i. For any $q' \in Q$ other than q, such S_i does not have nonempty intersection with $L(q')$. Therefore q should cover whole S_i. From Claim 1, $q \preceq p_i$. Since p_i is a mcg of S_i, $q \not\prec p_i$. Therefore $q = p_i$. Thus we have $Q \equiv P$. □

Theorem 4.1 does not fully characterize k-mmgs because the number of descriptions in some k-mmgs may be less than k. To explain such phenomena we introduce the notion of divisibility. Assume a GS $(D, \preceq)$ and a k-MSG $(D^k, \sqsubseteq)$ for some $k > 1$. Let S be a finite set of objects and p be a covering of S. For $k > 1$, a *k-division* of p *with respect to* S is a reduced k-multiple covering P of S such that $\sharp P > 1$ and $P \sqsubset \{p\}$. We say p is *k-divisible with respect to* S if there exists a k-division of p with respect to S. Note that k-divisibility implies k'-divisibility if $k \geq k'$ but it does not hold in general if $k < k'$. For example, consider a pattern $p = xy$ that covers a set $S = \{01, 12, 20\}$. Then p is not 2-divisible with respect to S. However, p is 3-divisible because S itself is a 3-division of p.

Theorem 4.2 *Let S be a finite set of objects, P be a reduced k-multiple covering of S. Then P is a k-mmg for S iff P is a tightest covering of S and any $p \in P$ is not Δk-divisible with respect to $S - L(P \setminus p)$, where $\Delta k = (k - \sharp P + 1)$.*

From Theorem 4.2 we see that there are *two* key problems to find a k-mmg; how to find a division of a description with respect to a finite set of objects, and how to get a tightest covering from a division.

Let $p \in D$ be a covering of a finite set S. A *partial covering* of S relative to p is a refinement q of p such that p is not a covering of S but $L(p)$ has a nonempty intersection with S, that is, $\phi \subsetneq S \cap L(p) \subsetneq S$. A *complete set of partial covering* (*cspc* for short) *of S relative to* a covering p is a set P of partial coverings of S relative to p such that for any partial covering q of S relative to p, there is some $q' \in P$ satisfying that $S \cap L(q) \subseteq S \cap L(q')$. We denote an arbitrary cspc of S relative to p by $\mathrm{cspc}(S, p)$. The following lemma is obvious from the definitions, which asserts that whenever a description p is a k-divisible covering of a finite set S of objects, we can effectively find a reduced k-multiple covering P of S.

Lemma 4.3 *Let $(D, \preceq)$ be a complete GS, S be a finite set of objects and p be a covering of S. Assume that Q is any cspc of S relative to p. Then p is k-divisible iff there exists a k-division P that is a subset of Q.*

A *tightest refinement* (*tr* for short) of a description p with respect to a finite set S of objects is a refinement of p that is an mcg of S. Suppose an algorithm that computes a tightest refinement of p with respect to S and denote by $\mathrm{tr}(p, S)$ the answer computed by the algorithm. Then, the following procedure, given a reduced k-multiple covering P of S, computes a tightest k-multiple covering of S by taking a tightest refinement of each member p in P. Hereafter, we denote by $\mathrm{TC}(S, P)$ the tightest covering computed by the procedure.

begin /* Computing the tightest covering of S less general than P */
  $Q := P$;
  while there exists some $q \in Q$ that is not marked **do**
    $r := \mathrm{tr}(q, S - L(Q \setminus q))$, and mark r;
    $Q := Q \backslash q \cup \{r\}$;
  endwhile;
  output Q;
end

Now, we have the algorithm MMG shown below to find a k-mmg of a finite set S of objects. Starting with the greatest multiple description $P = \{\top\}$, the algorithm MMG refines P by alternately applying two operations *dividing* at Line 4 and *tightening* at Line 5 to P. We claim that whenever it enters the while loop at Line 3, P is a tightest covering of S. If $P = \mathrm{TC}(T, Q)$ for some set T and a multiple covering Q of T, P is a tightest covering of T with just $\sharp Q$ members. Thus, P is a tightest covering of S when MMG enters the while loop at the first time. If P is a tightest covering of S and some $p \in P$ is Δk-divisible for some $\Delta k > 1$, then MMG finds a tightest covering Q of the set $S - L(P \setminus p)$ such that $Q \sqsubset \{p\}$, where $S - L(P \setminus p)$ is the set of strings in S that are covered by p but not by any other members in P. Then, we can show that the set $P \backslash p \cup Q$ obtained at Line 5 is also a tightest covering of S. Therefore P is being a tightest covering of S during the execution of the while loop. On the other hands, the number $\sharp P$ of members of P properly increases every time MMG executes the while loop, while $\sharp P$ cannot exceed k because Δk is defined as $\Delta k = k - \sharp P + 1$. Thus, MMG executes the loop at most $k - 1$ times and eventually terminates. Then, no member of P must be Δk-divisible for any $\Delta k > 1$ and P is a tightest covering of S. Hence, we have the correctness and the efficiency of the algorithm MMG from Theorem 4.2.

Theorem 4.4 *If cspc(S, p) and tr(p, S) are polynomial time computable for an efficient and complete GS $(D, \preceq)$, then a k-mmg of a finite set S of objects is computable in polynomial time in $|S|$ for the k-MGS $(D^k, \sqsubseteq)$.*

Algorithm MMG(k, S);
input: a positive integer k and a finite set S of objects;
output: a k-mmg P of S;
method:
  begin
1 $P := \mathrm{TC}(S, \{\top\})$;
2 $\Delta k := k$;
3 **while** $\Delta k \geq 2$ and there exists some $p \in P$ that is
        Δk-divisible with respect to $S - L(P \setminus p)$ **do**
4   find Δk-division ΔP of p with respect to $S - L(P \setminus p)$; /* dividing */
5    $P := P \backslash p \cup \mathrm{TC}(\Delta P, S - L(P \setminus p))$; /* tightening */
6    $\Delta k := k - \sharp P + 1$;
7 **endwhile**
8 output P ;
  end

5 Refinement operators

In the previous section, we show a sufficient condition for a concept class to have a polynomial time k-mmg algorithm. If we can design polynomial time algorithms to compute cspc and tr, then we construct a polynomial time k-mmg algorithm. In this section, we show algorithms for cspc and tr using refinement operators.

First we introduce the notion of refinement operator [Sha81, Lai88]. For a binary relation R, we denote the set $\{b \mid (a,b) \in R\}$ by $R(a)$ and the transitive closure of R by R^+. Let $(D, \preceq)$ be an efficient GS. A *refinement operator* is a subrelation of $\prec$. A refinement operator ρ is *complete* if $p \prec q \Longleftrightarrow p \in \rho^+(q)$ for any $p, q \in D$. A refinement operator ρ is *efficient* if $\rho(p)$ is polynomial time computable.

Clearly from the definitions, if ρ is a complete refinement for a complete GS, then the set $\{p \in \rho(p) \mid S \cap L(p) \neq \phi\}$ is the complete set of partial covering of S relative to p. Therefore if an efficient and complete refinement operator are available, a cspc is computable in polynomial time.

To compute $\mathrm{tr}(p, S)$, we can adopt a simple greedy search using a complete refinement operator. Consider the procedure shown below that, given a finite set S of objects and a covering p of S, computes a tightest refinement $\mathrm{tr}(p, S)$:

begin /* Computing a tightest refinement of p with respect to S */
  $q := p$;
  while there exists some $r \in \rho(q)$ such that $S \subseteq L(r)$ **do**
    $q := r$;
  output q
end

Let $n = \min\{\mathrm{size}(w) \mid w \in S\}$. Note that n is polynomial order of $|S|$ if the GS is efficient. If $r \in \rho(q)$ then $\mathrm{size}(r) > \mathrm{size}(q)$. If $\mathrm{size}(p) > n$ then $L(p)$ cannot contain any w such that $\mathrm{size}(w) \leq n$. Therefore the number of iterations on while loop in the procedure shown above is at most $n - \mathrm{size}(p)$. Hence, the procedure runs in polynomial time when both of GS and ρ are complete and efficient. From these observations we have the following theorem.

Theorem 5.1 *Let $(D, \preceq)$ be an efficient and complete GS, ρ be an efficient and complete refinement operator for $\prec$, S be a finite set of objects, and p be a covering of S. Then $cspc(S, p)$ and $tr(p, S)$ are computable in polynomial time in $|p|$ and $|S|$.*

6 Applications to pattern languages

First we present an efficient and complete refinement operator ρ for regular patterns [ASY92]. We define the *size* of a pattern p by $\mathrm{size}(p) = 2 \times |p| - \sharp v(p)$, where $v(p)$ is the set of variables appearing in p. Clearly $\mathrm{size}(p) \leq 2 \times |p|$ and $|p| \leq \mathrm{size}(p)$ for any p. A substitution θ is said to be *basic* for a regular pattern p, if θ satisfies one of the following:

- $\theta = \{x := a\}$, where $x \in v(p)$ and $a \in \Sigma$.
- $\theta = \{x := yz\}$, where $x \in v(p)$ and y, z are distinct variables not appearing in p.

We define ρ by $\rho(p) = \{q \mid q = p\theta, \theta \text{ is basic for } p\}$. Then we can show that ρ is an efficient and complete refinement for regular patterns. If $\sharp\Sigma > 2$, GS $(\mathcal{P}_{\mathrm{reg}}, \preceq)$ is efficient and complete [Shi82b, Muk92]. The following theorem is direct from Theorem 5.1.

Theorem 6.1 *If $\sharp\Sigma > 2$, then a k-mmg of a finite set S of strings is computable in polynomial time in $|S|$ for the k-MGS $(\mathcal{P}^k_{\mathrm{reg}}, \sqsubseteq)$.*

We cannot use our k-mmg algorithm as a k-minl for regular pattern languages, since $\mathcal{PL}^k_{\mathrm{reg}}$ does not have compactness with respect to containment. By a similar discussion as in [Muk92], we can show the compactness for the subclass $\mathcal{PL}^k_{\mathrm{reg}(m)}$ of unions of k m-variable regular pattern languages under $\sharp\Sigma > 2km$.

Lemma 6.2 *Let $k \geq 1$ and $m \geq 0$. If $\sharp\Sigma > 2km$, then for any m-variable regular patterns $p, q_1, \ldots, q_k$, $L(p) \subseteq L(q_1) \cup \ldots \cup L(q_k) \Longrightarrow L(p) \subseteq L(q_i)$ for some $1 \leq i \leq k$.*

However, we cannot use ρ as it is for $\mathcal{P}_{\rm reg}$ for m-variable regular patterns, because some $q \in \rho(p)$ contains more than m variables. To avoid this problem, we slightly modify ρ as ρ_m.

Let m be a positive integer. A substitution θ is said to be *m-basic* for a regular pattern p, if $\sharp v(p\theta) \leq m$ and θ satisfies one of the following:

- $\theta = \{x := a\}$, where $x \in v(p)$ and $a \in \Sigma$.
- $\theta = \{x := yz\}$, where $x \in v(p)$ and y, z are distinct variables not appearing in p.
- $\theta = \{x := ax\}$ or $\theta = \{x := xa\}$, where $x \in v(p)$ and $a \in \Sigma$.

We define ρ_m by $\rho_m(p) = \{q \mid q = p\theta, \theta \text{ is } m\text{-basic for } p\}$. Then we can show that ρ_m is an efficient and complete refinement for m-variable regular patterns.

Theorem 6.3 *If $\sharp\Sigma > 2km$, then the class $\mathcal{PL}^k_{\rm reg(}{}_{m)}$ of unions of k m-variable regular pattern languages is polynomial time inferable from positive data.*

An Example : Let S be the set { 0000, 00000, 0002, 000222, 00122, 002222, 00222, 1211, 112111, 2221 } of strings over $\Sigma = \{0, 1, 2\}$ and $k = 4$. The following table illustrates how the computation of a k-mmg of S for regular patterns proceeds, where a pattern underlined in some step indicates that it is selected to be divided at the step. Starting with the most general set $\{x\}$ at Step 1, the algorithm MMG refines the set of patterns step by step applying tightening and dividing operations alternately.

1: Tightening	{ x }	4: Dividing	{ $\underline{00zu}$, $x2z1$ }
2: Dividing	{ $\underline{xyzu}$ }	5: Tightening	{ $000u$, $00z2$, $x2z1$ }
3: Tightening	{ $0yzu$, $xyz1$ }	6: Solution	{ $000u$, $00y22$, $x2z1$ }

At Step 6, the algorithm terminates. Then, $\{000u, 00y22, x2z1\}$ is a k-mmg of S because the set is a tightest cover of S and none of $000u, 00y22, x2z1$ are Δk-divisible for $\Delta k = 4 - 3 + 1 = 2$.

7 Discussions

The main subject of this paper is to find an appropriate subclass of pattern languages whose unions form a class polynomial time inferable from positive data. We have generalized our former results for tree pattern languages [ASO93]. For inductive inference we need a description of a minimal concept explaining given the examples. Instead of the problem of finding minimal concept, we consider the problem of finding a syntactically minimal common generalization. Taking this approach we can show a uniform method to design efficient generalization algorithms. We also showed a condition for our algorithm finding a syntactically minimal description to be used as an algorithm finding a description of a minimal concept. As an application, we showed a class of unions consisting of at most k regular pattern languages with at most m variables is polynomial time inferable from positive data under the condition that the number of constant symbols is greater than $2km$.

For one-variable patterns, it seems to be hard to find an efficient and complete refinement operator. For example, the most general pattern x has infinitely many refinements xx, xxx, $xxxx$, ... Therefore if ρ is a complete refinement for $\mathcal{P}_1$, then $\rho(x)$ should contains these infinitely many refinements. However, we do not need all elements in $\rho(p)$ to compute a complete set of partial covering and a tightest refinement. For instance, we do not need any pattern q whose size is larger than the minimum size in S. To compute a tightest refinement, we can use a minl algorithm based on one-variable pattern automata [Ang79]. In reality we can efficiently compute both cspc and tr for $\mathcal{P}_1$. Therefore we can show that a k-mmg for one-variable patterns is also polynomial time computable for any $k \geq 1$. Angluin [Ang79] and Wright [Wri89a] showed the same result just for $k = 1, 2$, respectively. We can also show the compactness of unions of one-variable pattern languages under a similar condition on the number of constant symbols and the number of components of unions [Wri89a]. We will report more details on one-variable patterns elsewhere.

Theorem 7.1 *If $\sharp\Sigma > k$, then the class $\mathcal{PL}^k_1$ of unions of k one-variable pattern languages is polynomial time inferable from positive data.*

References

[AKM+92] S. Arikawa, S. Kuhara, S. Miyano, A. Shinohara, and T. Shinohara. A learning algorithm for elementary formal systems and its experiments on identification of transmembrane domains. In *Proc. of the 25th Hawaii International Conference on System Sciences*, pp. 675–684, 1992.

[Ang79] D. Angluin. Finding patterns common to a set of strings. In *Proceedings of the 11th Annual Symposium on Theory of Computing*, pp. 130–141, 1979.

[ASO92] H. Arimura, T. Shinohara, and S. Otsuki. Polynomial time inference of unions of two tree pattern languages. *IEICE Trans. Inf. & Syst.*, E75-D, pp. 426–434, 1992.

[ASO93] H. Arimura, T. Shinohara, and S. Otsuki. A polynomial time algorithm for finding finite unions of tree pattern languages. In *Proceedings of the Second International Workshop on Nonmonotonic and Inductive Logic*, pp. 118–131. LNAI 659, Springer, 1993.

[ASY92] S. Arikawa, T. Shinohara, and A. Yamamoto. Learning elementary formal systems. *Theoretical Computer Science*, Vol. 95, pp. 97–113, 1992.

[Gol67] E.M. Gold. Languages identification in the limit. *Information and Control*, Vol. 10, pp. 447–474, 1967.

[Lai88] P. D. Laird. *Learning from good and bad data.* Kluwer Academic, 1988.

[LW91] S. Lange and R. Wiehagen. Polynomial-time inference of arbitrary pattern languages. *New Generation Computing*, Vol. 8, No. 4, pp. 361–370, 1991.

[MSS91] S. Miyano, A. Shinohara, and T. Shinohara. Which classes of elementary formal systems are polynomial-time learnable? In S. Arikawa, A. Maruoka, and T. Sato, editors, *Proceedings of the Second Workshop on Algorithmic Learning Theory*, pp. 139–150, 1991.

[Muk92] Y. Mukouchi. Characterization of pattern languages. *IEICE Trans. Inf. and Syst.*, Vol. E75-D, No. 7, 1992.

[Nix83] R. Nix. Editing by example. Technical Report 280, Department of Computer Science, Yale University, 1983.

[Plo70] G. Plotkin. A note on inductive generalization. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence*, volume 5, pp. 153–163. Edinburgh Univ. Press, 1970.

[Sha81] E. Y. Shapiro. Inductive inference of theories from facts. Technical Report 192, Yale University, Department of Computer Science, 1981.

[Shi82a] T. Shinohara. Polynomial time inference of extended regular pattern languages. In *RIMS Symposia on Software Science and Engineering*, pp. 115–127. LNCS 147, Springer, 1982.

[Shi82b] T. Shinohara. Polynomial time inference of pattern languages and its applications. In *Proceedings of the 7th IBM Symposium on Mathematical Foundations of Computer Science*, pp. 191–209, 1982.

[Shi83] T. Shinohara. Inferring unions of two pattern languages. *Bulletin of Informatics and Cybernetics*, Vol. 20, pp. 83–88, 1983.

[Smu61] R. M. Smullyan. *Theory of Formal Systems.* Princeton Univ. Press, 1961.

[JSSY93] T. Jiang, A. Salomaa, K. Salomaa, and S. Yu. Inclusion is undecidable for pattern languages. In *Proc. 20th ICALP*, pp. 301–312, LNCS 710, Springer, 1993.

[TY92] N Tanida and T. Yokomori. Polynomial-time identification of strictly regular languages in the limit. *IEICE Trans. Inf. and Syst.*, Vol. E75-D, No. 1, pp. 125–132, 1992.

[Val84] L. G. Valiant. A theory of the learnable. *Comm. ACM*, Vol. 27, No. 11, pp. 1134–1142, 1984.

[Wri89a] K. Wright. *Inductive Inference of Pattern Languages.* PhD thesis, University of Pittsburgh, 1989.

[Wri89b] K. Wright. Identification of unions of languages drawn from an identifiable class. In *Proceedings of the 2nd Annual Workshop on Computational Learning Theory*, pp. 328–333, 1989.