RC 22387 (W0203-101) March 29, 2002
Computer Science

IBM Research Report

A Category-Theoretic Setting for Inductive Learning

Frank J. Oles
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

A Category-Theoretic Setting for Inductive Learning

Frank J. Oles

IBM T.J. Watson Research Center
Yorktown Heights, New York 10598
oles@us.ibm.com

March 28, 2002

Abstract

As a refinement of logical approaches to inductive learning, we propose category-theoretic inductive learning. We introduce precedence-inclusion patterns, which generalize constituent structure trees familiar to computational linguists. Categories of precedence-inclusion patterns have the requisite mathematical structure to support category-theoretic inductive learning. Thus, we obtain structures supporting a significant new theory of pattern generalization that directly speaks to the problem of relational learning in many settings.

1 Introduction

Extraction of structured information from text (see [3] for a survey article), based on inductive learning, is the down-to-earth motivation for the work described in this paper. For instance, one may wish to learn a pattern that enables one to search text for other instances of the $overdose(drug, symptom)$ relation from the example sentences (facts found in [7])

> Symptoms of erythromycin overdose included nausea.
> Other symptoms of Prozac overdose may include restlessness and agitation.

which themselves contain three instances of the $overdose(drug, symptom)$ relation. We also believe this work can be applied to the generalization of ontologies, particularly hierarchies for the classification of email, documents, or webpages. However, those are still only goals. This paper is concerned with new mathematical foundations for such applications.

The field of inductive learning is dominated by two general principled approaches: one statistical and the other logical. On one hand, statistical machine learning (e.g., see [4] or [10]) provides mathematical techniques for the solution of a vast array of practical problems. On the other hand, while inductive logic programming (e.g., see [2]) has not had the practical impact of statistical methods, it has extreme generality, thus having possible application to problems not amenable to statistical approaches.

In this paper, we want to propose a new direction that we call *category-theoretic inductive learning* and to claim that many inductive learning problems can profitably be approached from the point of view of category theory. Like inductive logic programming, the category-theoretic approach is symbolic, so in that sense it is as distinct from statistical methods as is inductive logic programming.

Let's consider the parallels and differences with ILP, while at the same time defining category-theoretic inductive learning. In category-theoretic inductive learning, for each problem area, we expect to identify a suitable category, roughly playing the role of a background theory in ILP. We will focus on the semantic as opposed to the syntactic, with primary attention paid to the objects of the category

instead of to the logical formulae describing the objects. Nonetheless, the objects – at least the finite ones – should have manageable representations. As in much of mathematics, the category, once chosen, will enable the development of methods specific to that category. Instead of addressing pattern generalization through proofs that one formula is implied by another, in the category-theoretic approach we will show that there is a morphism from one object to another. Our expectation is that the construction of such morphisms, informed by the definition of the category, will be more straightforward than theorem proving. Finally, in support of learning from examples, category-theoretic inductive learning replaces inductive logic programming's rather loose notion of least general generalizations, first identified in [8], with objects known as minimal most specific generalizations that, in the finite case, are often small, are readily computable, and are uniquely defined up to isomorphism. With these comparisons in mind, one might say that category-theoretic inductive learning is a refinement, not an antagonist, of logical approaches to inductive learning.

Our category-theoretic approach is not intended to have the full generality of inductive logic programming. At a high level, that's exactly why we might hope for some computational advantage. At a lower level, we have only examined certain kinds of pattern generalization problems in enough detail to feel confident that the category-theoretic inductive learning program can be carried out, at least insofar as there seem to be no mathematical obstacles. In particular, we wish to consider only certain problems of *relational learning*, by which we mean the induction from examples of some number of assertions that certain elements $x_1, x_2, \cdots$ of a structure S are in some particular relation $R(x_1, x_2, \cdots)$ to one another when the structure S is a specific instance of a more general pattern. Moreover, we wish to deal only with pattern generalization in the context of precedence-inclusion patterns, to be defined below. But, this is still a very large area, and it is relevant to many specific problems such the extraction of information from text.

We make no pretense that the category theory used in this paper is deep. What we are trying to do is to show the applicability of category-theoretic thinking to a new domain. No proofs are given due to space limitations.

We wish to thank Michael W. Mislove, David E. Johnson, Sylvie Levesque, and Thilo Goetz for useful conversations.

2 Precedence-Inclusion Patterns

We often encounter structured sets of entities, where, after some analysis, each entity may be labeled to reflect the set of properties it inherently possesses or the properties it possesses by virtue of its presence in, or perhaps by virtue of its position in, the set under consideration. Moreover, in such a structured set of entities, the structure generally serves to relate the individual entities to one another in familiar ways. Typically one entity may precede another one or one entity may include (or contain) another one. These observations are the starting point for the definition of categories of *precedence-inclusion patterns*. The resulting theory delivers a computationally tractable and mathematically well-founded approach to pattern generalization that is directly relevant to many relational learning problems. The key result, Theorem 7.2, is that, in a category of precedence-inclusion patterns, each finite set of finite precedence-inclusion patterns has a minimal most specific generalization that is unique up to isomorphism. We will argue that this best possible generalization can be readily computed, and we will make clear how this construction directly contributes to relational learning. Thus, we will have shown that the categories of precedence-inclusion patterns have the mathematical properties needed to support category-theoretic inductive learning.

Here's an example of the kind of generalization situation that motivates us. The pattern shown by the sentence

(A) *(Mary went (to the store).)*

is found in

(B) *(Mary went (to the store) today.)*

and is also found in

(C) *(Mary went quickly (to the store).)*

but that pattern is not found in

(D) *(Mary went (to the movies) (after the store closed).)*

To do this, we need a theory of pattern generalization that goes beyond word order, and that can make use of the results of partial parsing, but that does not necessarily require that sentences be fully parsed. The partial parsing of the sentences above is indicated by the matched pairs of parentheses inserted into the sentences. Each matched pair indicates an element of the pattern to which linguistic properties are assigned. Individual words are pattern elements, too, and properties, such as parts of speech, are assigned to them. Note that some linguistic elements precede other elements, while some elements include other elements. The approach we have in mind would say that there are pattern-preserving maps from (A) to (B) and from (A) to (C), but there is no pattern-preserving map from (A) to (D). In fact, we also want to say that (A) represents the minimal most specific generalization of (B) and (C).

It will come as no surprise to computational linguists that precedence-inclusion patterns generalize constituent structure trees. See Theorem 5.1. However, it may come as some surprise that precedence-inclusion patterns are truly more general. By small changes in the axiomatic definition of constituent structure trees, we obtain structures supporting a significant theory of pattern generalization, an element that is missing from constituent structure trees by themselves. In fact, we claim that in general one can't express the generalization of two constituent structure trees as a tree. An example of this is given in Section 6.

Also, we will get a new Cartesian closed category extending the category of sets. As interesting as this is, it doesn't seem to have any implications we now see for category-theoretic inductive learning.

Our formal discussion of precedence-inclusion patterns and their use in relational learning starts with assuming the existence of a set A, a single A-ary relation R (*the relation of interest*, a relation defined over the set of objects that may appear in structures exemplifying patterns, whatever those objects, structures and patterns may be), and a complete lattice L.

The set A is a set of *argument names*, each of which names an argument for R, the single relation of interest. In relational learning, we wish to determine which constituents of a structure are related by the specific relation R when that structure is an instance of a general pattern. We propose to do this by mapping elements of A to elements of the structure, thus instantitating the relation of interest. Our approach will be consistent with that idea that, in a single piece of text or of data, the relation R may be instantiated in multiple ways, albeit each instantiation will be represented by a different precedence-inclusion pattern based on the same text or data.

From a very formal point of view, we could actually dispense with R for the following reasons:

1. At no point in our work do we use any information about the logical properties of the relation R.

2. Only the set A enters into the definition of a precedence-inclusion pattern.

Nonetheless, keeping R in mind serves to remind us that, in any structure containing objects associated with the various argument names, those objects are supposed to be related by the specific relation R, which is supposedly of some special interest.

The complete lattice L is called the *property lattice*, and its elements intuitively correspond to the sets of properties that an entity in a structure may possess, with the minimal element being assigned to entities to which no properties are attributed. To prove that arbitrary products exist in categories

of precedence-inclusion patterns (Theorem 4.2), we make essential use of the assumption that L is a complete lattice.

There is no reason to assume either L or A is finite, although finiteness may often be expected in applications.

When L is degenerate (i.e., a one-element complete lattice), then patterns will be based purely on the relations of precedence and inclusion, and no other properties of the elements of structures will be relevant to pattern generalization except those that relate to instantiating the relation of interest.

When A is empty, or, equivalently, when the relation R is 0-ary, then R is either the logical constant `true` or the logical constant `false`. But, in our current perspective where we permit exactly one relation of interest, there is no intrinsic difference between these two situations. While the problem of determining specific elements related by R becomes moot when A is empty, this case enables us to see clearly how the problem of relational learning is a generalization of the problem of binary classification, i.e., finding rules that determine when a structure should be assigned to a particular class.

We will have a lot to say about irreflexive, transitive binary relations, but that phrase does not trip lightly off the tongue, so we will term such relations *strict partial orders.* Those well-schooled in ordered structures may think our preference for strict partial orders instead of partial orders is a bit odd. Rest assured that we know what we are doing: it is these relations that must be preserved by pattern-preserving maps.

The following definition offers a nontrivial extension of the concept of a transitive, binary relation to an ordered pair of binary relations. Let's say that an ordered pair $\langle E_1, E_2 \rangle$ of binary relations on a set P is *interactively transitive* if both E_1 and E_2 are transitive and, for all $x, y, z \in P$,

1. xE_1y and yE_2z implies xE_1z, and
2. yE_2x and yE_1z implies xE_1z.

We will say a set P is a *precedence-inclusion pattern* when it is provided with with two strict partial orders $\prec_P$, called *strict precedence,* and $\sqsupset_P$, called *strict inclusion,* along with a partial function $\alpha_P : A \dot{\to} P$, called the *argument naming function,* and a total function $\Lambda_L : P \to L$, called the *labeling function,* such that the ordered pair of relations $\langle \prec_P, \sqsupset_P \rangle$ is interactively transitive. Normally, we will just call a precedence-inclusion pattern simply a *pattern.* When there is no chance of confusion, we will dispense with the subscripts naming the pattern P, writing, for instance $x \prec y$ instead of $x \prec_P y$.

The function $\alpha_P : A \dot{\to} P$ is partial in order to deal with examples in which a relation is only partially instantiated, as well as to be able to talk about stripped patterns and stripping functors in Section 6.

When L is degenerate, we will call a precedence-inclusion pattern *purely positional.* When A is empty, we will say a precedence-inclusion pattern is a *classification pattern*. Clearly, for a purely positional pattern P, the labeling function Λ_P need not be given explicitly, because there is no choice in its definition. Similarly, for a classification pattern P, the argument naming function α_P need not be given explicitly.

If we forget about α and Λ for a moment – think of purely positional classification patterns – we can view the definition of a pattern as proposing a nontrivial modification of the definition of a strict partial order to fit an ordered pair of binary relations that allows the relations to interact, with the property that if either component of the ordered pair is the empty relation, then the only constraint on the other one is that it be a strict partial order.

Striving for the simplest nontrivial examples of precedence-inclusion patterns, consider purely positional classification patterns. For instance, based on a string $S = \langle a, b, c, d, e \rangle$ of length five, we can rather arbitrarily define a five-element purely positional classification pattern

$$W = \{\langle a, b, c\rangle, \langle a\rangle, \langle b, c\rangle, \langle b\rangle, \langle d, e\rangle\}.$$

The order of the elements of $S = \langle a, b, c, d, e \rangle$ and the fact that each string in W has a unique occurrence

as a substring of S determines both the strict precedence relation on W, which is given in detail by

$$\langle a, b, c\rangle \prec \langle d, e\rangle, \quad \langle a\rangle \prec \langle b, c\rangle \prec \langle d, e\rangle, \quad \langle a\rangle \prec \langle b\rangle \prec \langle d, e\rangle.$$

and the strict inclusion relation on W, which is given in detail by

$$\langle a, b, c\rangle \sqsupset \langle a\rangle, \quad \langle a, b, c\rangle \sqsupset \langle b, c\rangle \sqsupset \langle b\rangle.$$

Don't be misled into thinking all precedence-inclusion patterns are essentially textual. Geometric examples are given in Section 6.

If $f : X \dot{\to} Y$ is a partial function from a set X to a set Y, then we denote the *domain of definition* of f by dom f, i.e., dom $f = \{x \in X \mid f(x) \text{ is defined}\}$.

We say that a precedence-inclusion pattern Q is a *subpattern* of a pattern P if

1. as sets, $Q \subseteq P$,
2. $a \in \operatorname{dom} \alpha_Q$ iff $a \in \operatorname{dom} \alpha_P$ and $\alpha_P(a) \in Q$, and
3. for all $x, y \in Q$ and for all $a \in \operatorname{dom} \alpha_Q$,
   (a) $x \prec_Q y$ iff $x \prec_P y$,
   (b) $x \sqsupset_Q y$ iff $x \sqsupset_P y$,
   (c) $\Lambda_Q(x) = \Lambda_P(x)$, and
   (d) $\alpha_Q(a) = \alpha_P(a)$.

The reader should not regard the definition of subpattern as entirely routine. At this stage, another definition replacing biconditionals by implications is not unreasonable. The actual definition given is the right definition for stating the results of Sections 6 and 7.

3 Pattern-Preserving Maps

The concept of a morphism between patterns is the basis of understanding pattern generalization. Let Q and P be precedence-inclusion patterns. When there is a morphism from Q to P, then we may say Q *is a generalization of* P.

For the formal definition of a morphism, if $\operatorname{dom} \alpha_Q \not\subseteq \operatorname{dom} \alpha_P$, then there are no morphisms from Q to P. If $\operatorname{dom} \alpha_Q \subseteq \operatorname{dom} \alpha_P$, then a morphism $h : Q \to P$ from Q to P is a (total) function from Q to P satisfying, for all $x, y \in Q$ and for all $a \in \operatorname{dom} \alpha_Q$,

1. $x \prec_Q y$ implies $h(x) \prec_P h(y)$,
2. $x \sqsupset_Q y$ implies $h(x) \sqsupset_P h(y)$,
3. $\Lambda_Q(x) \leq \Lambda_P(h(x))$, and
4. $h(\alpha_Q(a)) = \alpha_P(a)$.

Thus, a morphism from a pattern Q to a pattern P describes how to find all the parts of Q within P in a structurally consistent way. A morphism between patterns is also called a *pattern-preserving map*.

A less artful definition of morphism might have replaced the third clause above by

$$\Lambda_Q(x) = \Lambda_P(h(x)),$$

but this would not be what we want. The definition as given implies that if P is more specific than Q, then not only must the elements of P that correspond to elements of Q have all the relevant properties of

corresponding elements of Q, but elements of P are permitted to have other properties, too. Moreover, with the different definition, we would not be able to prove that even binary products exist in categories of precedence-inclusion patterns (see Theorem 4.2) – and products are a key part of the theory of pattern generalization (see Theorem 6.1).

A bijective pattern-preserving map is not necessarily an isomorphism. We will give an example in Section 6. This is not really so weird, since it should remind us of the similar situation in topological spaces, where a bijective continuous map is not necessarily a homeomorphism. However, we do have the following easy proposition that applies to finite patterns. Along the road to the proof of Theorem 7.2, it is used several times.

Proposition 3.1 *A bijective endomorphism of a finite pattern is necessarily an automorphism.*

It is possible for a pattern-preserving epimorphism to fail to be surjective.

4 Categories of Patterns

Let ***Set*** denote the category of sets, and let ***SPO*** be the category of strictly partially ordered sets, with strictly monotone maps as morphisms. The results in this section have routine proofs.

Theorem 4.1 *For any set A of argument names and for any property lattice L, the categories **Set** and **SPO** can both be regarded as (i.e., are isomorphic to) full, coreflective subcategories of the category **PI**(A, L) of patterns.*

Theorem 4.2 *For any set A of argument names and for any property lattice L, arbitrary products exist in the category **PI**(A, L) of precedence-inclusion patterns.*

Theorem 4.3 *If the set A of argument names is empty, so that **PI**$(\emptyset, L)$ is a category of classification patterns, then arbitrary coproducts of patterns exist in **PI**$(\emptyset, L)$.*

Theorem 4.4 *The category of purely positional classification patterns is Cartesian closed.*

5 Constituent Structure Trees

Constituent structure trees give examples of classification patterns. (See Theorem 5.1.) Since the definitions of the two concepts bear some similarity, it is worth looking at the differences.

As a reference on mathematical concepts in computational linguistics, see [6]. With slight notational modification, we will take our definition of constituent structure tree from that source. Given a finite set Q of labels, a *constituent structure tree* is a finite set T together with a strict partial order $\prec$ (the precedence relation) on T, a partial order $\supseteq$ (the dominance relation) on T, and a labeling function $\lambda : T \to Q$ such that the following conditions hold:

Single Root Condition

$$(\exists r)(\forall x)(r \supseteq x)$$

Exclusivity Condition

$$(\forall x)(\forall y)((x \prec y \text{ or } y \prec x) \leftrightarrow (x \not\supseteq y \text{ and } y \not\supseteq x))$$

Nontangling Condition

$$(\forall w)(\forall x)(\forall y)(\forall z)((w \prec x \text{ and } w \supseteq y \text{ and } x \supseteq z) \rightarrow y \prec z)$$

where r, x, y, z, and w range over T.

There is nothing corresponding to R, the A-ary relation of interest, in the preceding definition. Also, in computational linguistics, constituent structure trees are not considered as objects of a category, as far as we know. However, the following important, but fairly obvious, theorem holds.

Theorem 5.1 *Every constituent structure tree gives rise to a classification pattern by keeping the precedence relation as what we have called strict precedence, by taking the strict inclusion relation to be the dominance relation minus the identity relation, by letting L be the power set of Q, and by letting $\Lambda(x) = \{\lambda(x)\}$.*

It is worth reflecting on the small differences between the definitions of classification patterns and constituent structure trees, and on the fairly large consequences of those differences:

1. Dominance is a partial order, whereas the definition of a pattern requires strict inclusion to be a strict partial order. This is just a nit.

2. Constituent structure trees are required to be finite, but patterns do not have such a requirement. Letting precedence-inclusion patterns be infinite is needed to argue convincingly that categories of patterns generalize the category of sets because then one can prove that the category of sets is isomorphic to a full, coreflexive subcategory of a category of patterns (Theorem 4.1).

3. Constituent structure trees have a finite set of labels, as opposed to an arbitrary complete lattice of properties. For patterns, the lattice of properties comes into its own when analyzing when one pattern is a generalization of another, a consideration that seems not to have arisen for constituent structure trees.

4. Constituent structure trees are rooted, but there is no analagous requirement on patterns. Removing the Single Root Condition results in the existence of arbitrary coproducts in categories of classification patterns (Theorem 4.3).

5. This the most subtle difference: *only the left-to-right implication in the biconditional Exclusivity Condition holds in general for patterns*. This loosens things up greatly by permitting patterns with a geometric flavor. The relaxation of the Exclusivity Condition is key to being able develop our theory of pattern generalization.

Finally, we remark on a similarity between precedence-inclusion patterns and constituent structure trees: the Nontangling Condition for constituent structure trees is basically the same as what we earlier termed interactive transitivity. We like our term better because a pattern can satisfy this condition while still being fairly tangled, as evidenced by the pattern M shown in Figure 9.

6 An Example of Pattern Generalization

At the start of Section 3, we explained that, for patterns P and Q, saying Q is a generalization of P means there exists a pattern-preserving map $f : Q \to P$. Now suppose I is an index set and $\mathcal{P} = \{P_i \mid i \in I\}$ is an I-indexed set of precedence-inclusion patterns. Then we say that Q is a *generalization* of $\mathcal{P}$ if there exists an I-indexed set of pattern-preserving maps $\{f_i : Q \to P_i \mid i \in I\}$. This asserts all of the structure found in Q can be found in each of the patterns P_i. A generalization Q of $\mathcal{P}$ is a *most specific generalization* of $\mathcal{P}$ if every generalization of $\mathcal{P}$ is also a generalization of Q. That most specific generalizations always exist in categories of patterns is taken care of by the Theorem 6.1, which is essentially obvious from the definition of a product.

Theorem 6.1 *If I is an index set and $\mathcal{P} = \{P_i \mid i \in I\}$ is an I-indexed set of precedence-inclusion patterns, then a product of $\mathcal{P}$ is a most specific generalization of $\mathcal{P}$.*

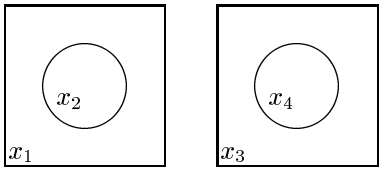

Figure 1: The pattern X, in which $R(x_2, x_4)$ holds.

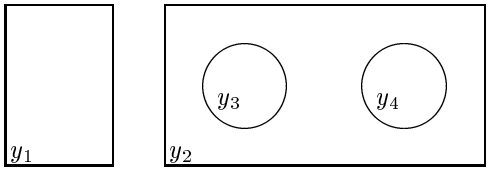

Figure 2: The pattern Y, in which $R(y_3, y_4)$ holds.

While Theorem 6.1 guarantees the existence of most specific generalizations, they are by no means unique up to isomorphism. For instance, in a category of classification patterns (so that, by Theorem 4.3, coproducts exist) then a coproduct of any number of copies of a most specific generalization of a set of patterns is also a most specific generalization of that set of patterns.

One might then think that the product of a set of patterns, given that products in a category are unique up to isomorphism, is the best generalization of a set of patterns, but this would be wrong, because in general we can do much better! We can get a minimal most specific generalization:

Definition Let I be an index set and let $\mathcal{P} = \{P_i \mid i \in I\}$ be an I-indexed set of precedence-inclusion patterns. A *minimal most specific generalization* of $\mathcal{P}$ is a most specific generalization M of $\mathcal{P}$ such that no proper subpattern of M is a most specific generalization of $\mathcal{P}$.

This concept will be illustrated by an example, and will be given additional mathematical substance in Theorem 7.2.

Consider two patterns $X = \{x_1, x_2, x_3, x_4\}$ and $Y = \{y_1, y_2, y_3, y_4\}$, based on the pictures shown in Figures 1 and 2. Here precise dimensions do not matter, and each pattern consists of two rectangular elements and of two circular elements. Thus, we take our property lattice L to be the power set of the two-element set $\{\Box, \circ\}$ where each of the eight picture elements is assigned by the appropriate labeling function Λ either the singleton set $\{\Box\}$ or the singleton set $\{\circ\}$, according to its shape. In all of the geometric patterns in this section, strict precedence $a \prec b$ means picture element a is entirely to the left of picture element b, while strict inclusion $a \sqsupset b$ means b is entirely within a. For example, $x_1 \prec_X x_3$ and $y_2 \sqsupset_Y y_3$. We could have described these particular patterns as partial parses of strings in various ways. For instance, X, to the extent that we have described it so far, might have been written

$$(x_1^{\Box}(x_2^{\circ}))(x_3^{\Box}(x_4^{\circ})),$$

which can be viewed as a decorated parse of a string of length four, in which names for elements exponentiated with their labels immediately follow the appropriate leading parenthesis. Similarly Y, to the extent that we have described it so far, might have been written

$$(y_1^{\Box})(y_2^{\Box}(y_3^{\circ})(y_4^{\circ})).$$

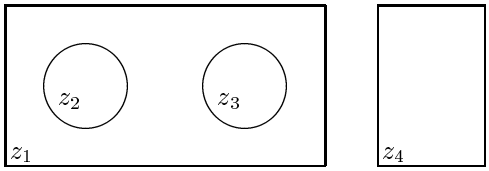

Figure 3: The stripped pattern Z. Does $R(z_2, z_3)$ hold?

Of course, not every geometric pattern is susceptible to this kind of linearization. We will also use these examples to illustrate relational learning, so suppose in pattern X, $R(x_2, x_4)$ is true, and that in pattern Y, $R(y_3, y_4)$ is true. The idea here is that $R(a, b)$ holds for elements of a geometric pattern if a and b have the right properties and reside in the correct places in a pattern that is a specialization of a suitable generalization of X and Y, such as $X \times Y$, the product of X and Y. However, based on the examples, we have the strong feeling that $R(a, b)$ should imply that both a and b are labeled as circular and that a strictly precedes b. To complete the formal definition of the patterns X and Y, let's name the first and second arguments of R respectively a_1 and a_2, so that $A = \{a_1, a_2\}$. Then $\alpha_X(a_1) = x_2$ and $\alpha_X(a_2) = x_4$, while $\alpha_Y(a_1) = y_3$ and $\alpha_Y(a_2) = y_4$. If we chose to present these patterns as parses, then we could have added the argument names to the exponents, obtaining

$$(x_1^{\square}(x_2^{\circ,\, a_1}))(x_3^{\square}(x_4^{\circ,\, a_2}))$$

for X and

$$(y_1^{\square})(y_2^{\square}(y_3^{\circ,\, a_1})(y_4^{\circ,\, a_2}))$$

for Y.

By a *stripped* pattern, we mean a pattern P for which the partial function α_P is empty. Of course, when working in a category of classification patterns, all of the objects are stripped patterns. This concept proves to be more useful in categories of patterns in which the set A of argument names is nonempty. In relational learning, the examples on which the learning is based are fully analyzed and unstripped (i.e., not stripped), such as X and Y above. However, the patterns to which we want to apply what is learned are stripped, and if an instance of the relation of interest is found in one of them, then an unstripped pattern can be created. For a pattern P, let $P^\dagger$ be the stripped pattern formed by replacing the argument naming function α_P by the empty function. For example, the description of X as

$$(x_1^{\square}(x_2^{\circ,\, a_1}))(x_3^{\square}(x_4^{\circ,\, a_2}))$$

leads to the description of $X^\dagger$ as

$$(x_1^{\square}(x_2^{\circ}))(x_3^{\square}(x_4^{\circ})).$$

Actually, what we are defining here is the *stripping functor*

$$(\cdot)^\dagger : \textbf{\textit{PI}}(A, L) \to \textbf{\textit{PI}}(\emptyset, L).$$

For a pattern-preserving map $f : P \to Q$ in $\textbf{\textit{PI}}(A, L)$, $f^\dagger = f$. The inclusion functor of the full subcategory $\textbf{\textit{PI}}(\emptyset, L)$ into $\textbf{\textit{PI}}(A, L)$ is left adjoint to the stripping functor, which shows that $\textbf{\textit{PI}}(\emptyset, L)$ is a full, coreflective subcategory of $\textbf{\textit{PI}}(A, L)$.

Consider the pattern Z shown in Figure 3, which has a linear representation as

$$(z_1^{\square}(z_2^{\circ})(z_3^{\circ}))(z_4^{\square}).$$

This is a stripped pattern. A question we would like to address is whether or not $R(z_2, z_3)$ holds in Z, on the basis of the unstripped examples X and Y. (Actually, the phrase "$R(z_2, z_3)$ holds in Z" doesn't

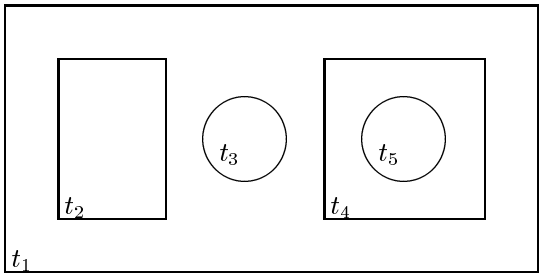

Figure 4: The stripped pattern T. Does $R(t_3, t_5)$ hold?

formally make sense because the argument naming function for Z is known to be empty. We should instead say "$R(z_2, z_3)$ holds in Z', where Z' satisfies $(Z')^\dagger = Z$." With this explanation, we intend to use the shorter, but technically inaccurate phrase, where the true meaning is clear.) We do see that both z_2 and z_3 are circular, and z_2 certainly strictly precedes z_3. However, a careful study of X and Y shows us that in both cases the second circle is strictly preceded by a rectangle, and this does not happen in Z. So we conclude that $R(z_2, z_3)$ does not hold in Z, at least on the basis of informal generalization from our two examples.

At this point we are in a position to appreciate the inadequacy of attempting to do construct a best generalization by removing elements from one pattern, say X, based on some sort of inconsistency with features of a second pattern, say Y. Just think about:

1. You can't remove either circle x_2 or circle x_4 if your goal is to learn the relation R.

2. Removing the rectangle x_1 causes loss of the property that, in both X and Y, the second circle is strictly preceded by a rectangle. This was the very fact used to resolve negatively the question of whether or not $R(z_2, z_3)$ holds in Z.

3. Removing the rectangle x_3 causes loss of the property that, in both X and Y, the second circle is strictly included in a rectangle.

Parallel reasoning indicates that you can't get away with only removing properties of elements instead of removing the element completely. The bottom line is that constructing a best generalization cannot be done by removal of parts from a pattern, as one might have conjectured from the linguistic example presented early in Section 2.

A trickier situation is given by the stripped pattern T, displayed in Figure 4 and possessing a linear representation as

$$(t_1^{\square}\,(t_2^{\square})(t_3^{\circ})(t_4^{\square}\,(t_5^{\circ}))).$$

Here the question is whether or not $R(t_3, t_5)$ holds in T. We claim that we should conclude that $R(t_3, t_5)$ holds in T because we will show that there is a minimal most specific generalization M of X and Y such that

1. the domain of the argument naming function α_M is all of $A = \{a_1, a_2\}$, which in conjunction with a pattern-preserving map $f : M^\dagger \to T$ enables us by replacing the empty function α_T with $\alpha_{T'} = f \circ \alpha_M$ to create from T an unstripped pattern T' such that $(T')^\dagger = T$ and such that $f : M \to T'$ – that's the same f as before – is a pattern-preserving map, and

2. inspection of T' shows $R(t_3, t_5)$ holds in T'.

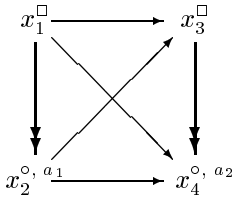

Figure 5: Directed acyclic graph representation of X.

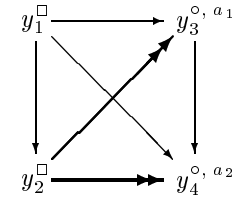

Figure 6: Directed acyclic graph representation of Y.

The first step in identifying M is to look closely at the product $X \times Y$ of X and Y. It cannot be depicted pictorially in the manner of the other patterns of this section, nor does it have a linear representation via a disciplined insertion of parentheses into a string. It can be represented by a edge-labeled directed acyclic graph. (Space does not permit the discussion of pattern representation, but there are interesting issues here. For instance, it turns out that finite patterns have unique minimal representations, but it is not yet clear if that is always the most computationally advantageous representation of a pattern.) To construct the graphical representation, we start by considering similar representations of X and Y, shown in Figures 5 and 6, respectively. In these graphs, we are using thin arrows to indicate strict precedence and double-headed thick arrows to indicate strict inclusion. In these graphs every ordered pair in both relations is depicted by an arrow. Using these two graphs, we can now depict $X \times Y$, as shown in Figure 7. In the graph, labels and argument names are placed at the vertices, with a bullet placed for those elements that have the empty set of labels and no argument names.

This all looks pretty complicated, but there is a particular kind of pattern-preserving map that helps us reduce the complexity of $X \times Y$ in small steps. If P is a pattern, a *retraction* of P is an idempotent endomorphism $r : P \to P$, i.e., for all $x \in P$, $r(r(x)) = r(x)$. A subpattern of P that is the image of a retraction, which is the same as being the set of fixed points of the retraction, is known as a *retract* of P. We use the phrases "R is a retract of P" and "P has a retract R" interchangably to express the fact that there is a retraction of P whose image is R. A retract of P is a *proper retract* if it is a proper subpattern of P. This concept is modeled on the similarly named concepts found in topology (see [5], p. 216) and in domain theory (see [9] or [1], p. 344). Proposition 6.2 tells why we like retracts of patterns.

Proposition 6.2 *Let R be a retract of a precedence-inclusion pattern P.*

1. *By composing with the corestriction of a retraction that defines R, we see that every generalization of P is a generalization, too, of R.*

2. *By composing with the inclusion map from R to P, we see that every pattern that is generalized by P is also generalized by R.*

Before constructing retracts, we need to understand some of their theory. For instance, the next proposition says being a retract is transitive.

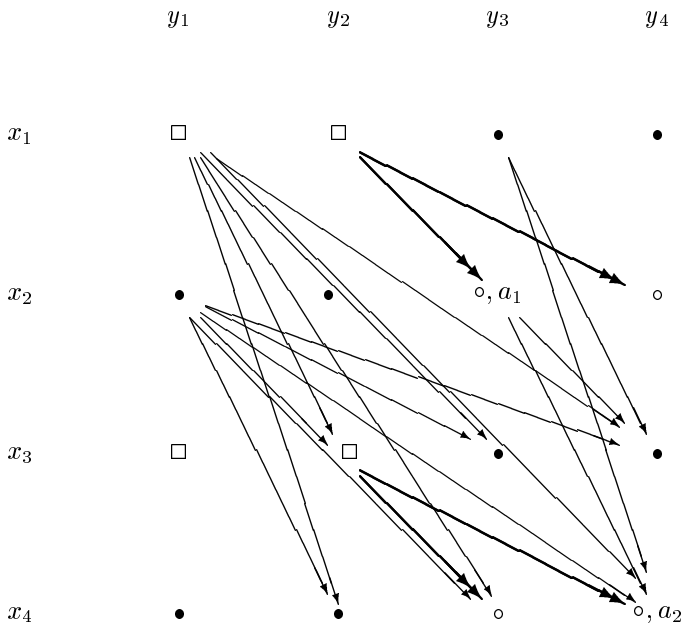

Figure 7: Directed acyclic graph representation of $X \times Y$.

Proposition 6.3 *If Q is a retract of a pattern P, and if R is a retract of Q, then R is a retract of P.*

It might look like the concept of a retract is overkill because perhaps the image Q of an arbitrary endomorphism $f : P \to P$ might serve the same purpose as a retract of P. However, at least for finite patterns P, Proposition 6.4 shows that each endomorphism is closely connected to a retraction. Just finding an endomorphism of a finite pattern whose image is a proper subset of the pattern leads directly to a proper retract of the pattern.

Proposition 6.4 *If $f : P \to P$ is an endomorphism of a finite pattern P, then there exists a positive integer n such that $f^n : P \to P$, the composition of f with itself n times, is a retraction.*

A proper retract of the product $X \times Y$ of our two example patterns will have the desired generalization properties that are possessed by $X \times Y$: (1) it will generalize both X and Y, and (2) any generalization of both X and Y will be a generalization of it. And, even better, it will be smaller! How small can it be? This, in a sense, is answered by Proposition 6.5, which first needs a definition: a pattern M is *fully retracted* if M has no nontrivial retraction (i.e., $1_M : M \to M$ is the only retraction of M). When we do speak of a *minimal retract of a pattern P,* we mean a retract of P that is fully retracted, and hence, by Proposition 6.3 properly contains no proper retract of P. Note that these definitions apply to infinite patterns as well as finite patterns. The proof of Proposition 6.5 is clear.

Proposition 6.5 *For every finite pattern P, there is a retraction $r : P \to P$ whose image is a minimal retract of P.*

Using a sequence of retractions each of which moves only one point, we can obtain the retract M of $X \times Y$ that is shown in Figure 8. (Getting such a sequence of one-point-moving retractions seems to happen regularly in examples, and it strikes us as somewhat remarkable, but we don't know how to delimit the phenomenon yet.) Starting with the observation that the elements of M that have argument names attached to them are fixed points of every endomorphism of M, it is easy to see that M has no

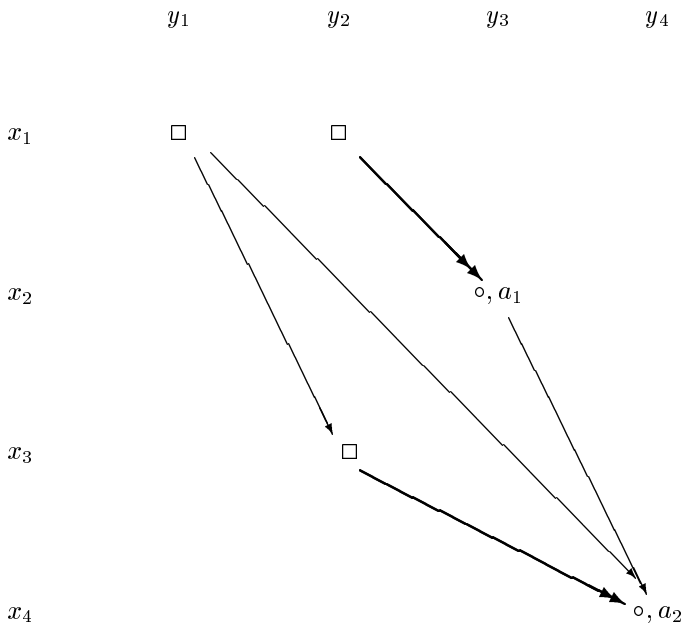

Figure 8: Directed acyclic graph representation of the retract M.

endomorphism except the identity function. Therefore, M is a minimal retract of $X \times Y$. It is also easy to see that M meets the definition of a minimal most specific generalization of X and Y.

In seeking to learn if a pattern P has all the structure common to both X and Y, it is vastly simpler to carry out a direct check for the existence of a pattern-preserving function $h : M \to P$, as opposed to a direct check for the existence of a pattern-preserving function $g : X \times Y \to P$, even though the existence of the one function is logically equivalent to the existence of the other.

It is possible to give a verbal description of the generalization M, which is a little surprising in light of the complexity of $X \times Y$. It reads something like this:

> There is a rectangle that strictly precedes a rectangle that strictly includes a circle that is the second argument to R and which is preceded by a circle that is the first argument to R, which is strictly contained in a rectangle.

In spite of this simple recital, it can be shown that M *cannot be described as a parse structure on a string, even though* X *and* Y *could be so described. This example suggests why constituent structure trees are defective as a carrier for a theory of generalization in computational linguistics.*

The pattern M has geometric representation as an arrangement of circles and rectangles, as shown in Figure 9, where we let

$$m_1 = \langle x_1, y_1 \rangle, \;\; m_2 = \langle x_1, y_2 \rangle, \;\; m_3 = \langle x_2, y_3 \rangle, \;\; m_4 = \langle x_3, y_2 \rangle, \;\; m_5 = \langle x_4, y_4 \rangle$$

Now consider the stripped pattern $M^\dagger$. The function $f : M^\dagger \to T$ given by

$$m_1 \;\mapsto\; t_2, \;\; m_2 \;\mapsto\; t_1, \;\; m_3 \;\mapsto\; t_3, \;\; m_4 \;\mapsto\; t_4, \;\; m_5 \;\mapsto\; t_5$$

is a pattern-preserving map (and, by the way, is an example of a bijective pattern-preserving map that is not an isomorphism, as is clear from the pictures of the patterns). Therefore, if we extend T to an unstripped pattern T' by defining the argument naming function $\alpha_{T'}$ as $\alpha_{T'} = f \circ \alpha_M$, then $f : M \to T'$ becomes a pattern-preserving map. Hence we conclude that $R(t_3, t_5)$ holds in T, as desired.

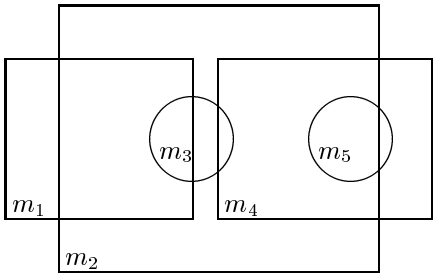

Figure 9: The minimal retract M, in which $R(m_3, m_5)$ holds.

Moreover, because there is no pattern-preserving map $h : M^\dagger \rightarrow Z$, we can say that on the basis of this formal analysis that $R(z_2, z_3)$ does not hold in Z based on generalizing from X and Y alone.

7 The Theory of Generalization

Much of the theory has already been laid out in the course of presenting the example in Section 6. Here we will pull it together by providing results that show that, for finite sets of finite patterns, there is only one minimal most specific generalization, up to isomorphism, and that any finite most specific generalization, not just a product of a set of patterns, can be used as a starting point in constructing a minimal most specific generalization.

Note that in Theorem 7.1, we do not assume that the individual patterns P_i are finite, nor that the index set I is finite. It actually is possible for an infinite set of infinite patterns to have a finite most specific generalization. (Consider an infinite set $\mathcal{P}$ of ordinary infinite sets considered as patterns, i.e., with empty strict precedence and strict inclusion relations. Then a singleton set, regarded as a pattern, again with empty relations, is a minimal retract and it is a most specific generalization of $\mathcal{P}$. This should not be surprising because pattern generalization is mostly, but not exclusivly, about relations that must be present, so no relations can lead to trivial generalizations.)

Theorem 7.1 *Let I be an index set and let $\mathcal{P} = \{P_i \mid i \in I\}$ be an I-indexed set of precedence-inclusion patterns. Suppose there is a finite pattern Q that is a most specific generalization of $\mathcal{P}$.*

1. *There is a retract M of Q such that M is finite, fully retracted, and a most specific generalization of $\mathcal{P}$.*
2. *If N is any fully retracted pattern that is also a most specific generalization of $\mathcal{P}$, then M is isomorphic to N.*

We now give the Theorem 7.2, the main goal of this section. For the finite case, it covers the existence and uniqueness of the minimal most specific generalization, and, implicitly, tells how to compute it.

Theorem 7.2 *Let I be a finite index set and let $\mathcal{P} = \{P_i \mid i \in I\}$ be an I-indexed set of finite precedence-inclusion patterns.*

1. *There exists a minimal most specific generalization M of $\mathcal{P}$.*
2. *M is finite and fully retracted.*

3. *Any minimal most specific generalization of $\mathcal{P}$ is isomorphic to M.*

4. *Any finite most specific generalization Q of $\mathcal{P}$ has a retraction $r : Q \to Q$ whose image is isomorphic to M.*

We end with the explicit description of the very simple procedure used in Section 6 that is guaranteed to return the minimal most specific generalization of a finite set $\{P_1, P_2, \cdots, P_n\}$ of finite patterns:

Minimal Most Specific Generalization Procedure

$M := P_1 \times P_2 \times \cdots \times P_n$;
while there exists a proper retract Q of M
    do $M := Q$;
return M;

Theorem 7.2 guarantees that the result of the nondeterministic procedure above is unique up to isomorphism. The last claim in Theorem 7.2 is significant. It says that, in the procedure, we do not need to start with the product of $\{P_1, P_2, \cdots, P_n\}$ for computing a minimal most specific generalization. Initialization using any other finite most specific generalization will suffice.

8 Summary

In this paper, we introduced the concept of category-theoretic inductive learning and categories of precedence-inclusion patterns. We have shown that categories of precedence-inclusion patterns have the requisite mathematical structure to support a theory of pattern generalization in line with category-theoretic inductive learning. In the future we hope to validate these ideas empirically with efforts to extract information from text.

References

[1] H.P. Barendregt. Functional programming and lambda calculus. In J. van Leeuvan, editor, *Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics*, pages 321–364. The MIT Press/Elsevier, 1990.

[2] F. Bergadano and D. Gunetti. *Inductive Logic Programming*. MIT Press, Cambridge, 1996.

[3] C. Cardie. Empirical methods in information extraction. *AI Magazine*, 18(4):65–80, 1997.

[4] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New York, 1996.

[5] J.R. Munkres. *Topology: A First Course*. Prentice Hall, Inc., Engelwood Cliffs, 1975.

[6] B.H. Partee, A. ter Meulen, and R.E. Wall. *Mathematical Methods in Linguistics*. Kluwer Academic Publishers, Dordrecht, 1990.

[7] PDR. *Physicians' Desk Reference*. Medical Economics Data, 1993.

[8] G.D. Plotkin. A note on inductive generalization. In B. Meltzer and D. Michie, editors, *Machine Intelligence 5*, pages 153–163. American Elsevier Publishing Co., Inc., 1970.

[9] D.S. Scott. Continuous lattices. In F.W. Lawvere, editor, *Toposes, Algebraic Geometry, and Logic*, volume 274 of *Lecture Notes in Mathematics*, pages 97–136. Springer-Verlag, 1972.

[10] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.