RC22312 (W0201-100) January 21, 2002
Computer Science

IBM Research Report

Precedence-Inclusion Patterns: A Fresh Approach to Relational 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

LIMITED DISTRIBUTION NOTICE: This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. Ithas been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distributionoutside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g, payment of royalties). Copies may be requested from IBM T. J. Watson Research Center , P. O. Box 218, Yorktown Heights, NY 10598 USA (email: reports@us.ibm.com). Some reports are available on the internet at http://domino.watson.ibm.com/library/CyberDig.nsf/home

Precedence-Inclusion Patterns: A Fresh Approach to Relational Learning

Frank J. Oles

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

January 18, 2002

Abstract

Precedence-inclusion patterns, a generalization of constituent structure trees in computational linguistics, possess a significant theory of pattern generalization that can be applied to the problem of relational learning in many settings, including learning from text, images, and video. When specialized to posets, the result is a new theory of poset generalization that may be applied to ontologies and hierarchical classification systems.

1 Introduction

We are interested in the problems of determining when a structure is an instance of a general pattern, as well as devising new methods for solving problems of relational learning. While we don't intend to consider these problems in complete generality, we do want methods that have a wide range of possible applications including extraction of information from text.

To be clear about it, by *relational learning* 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. Relational learning, then, is pattern generalization but with the aim of identifying those elements of a particular pattern that bear a special relationship to one another.

In text, we find two primary relations between linguistic elements: (1) an element may precede another one, or (2) an element may include or contain another one. Our goal has been to develop a theory of patterns based on these two relations. Of course, in computational linguistics there are constituent structure trees [6], which are already based on these relations. However, constituent structure trees are defective from the standpoint of generalization: there is no codified way of saying that one tree is more general than another, and there is certainly no way of computing a tree that is a best

generalization of a set of trees. We will show how to remedy these defects by defining *precedence-inclusion patterns* of which the concept of a constituent structure tree is special case. While we won't be able to compute trees as best generalizations of sets of trees, we will be able to compute finite patterns as best generalizations of finite sets of finite patterns.

We want the notion of the best generalization – actually, the concept is called the *minimal most specific generalization* – of a set of patterns to have a mathematically compelling definition. Additionally, we want the minimal most specific generalization to be another pattern, not just a logical summation of properties common to all the elements of the set. For this reason, we do not approach pattern generalization from the point of view of inductive logic programming (see [2] or the seminal paper [7]), because then, on the face of it, all we would get is what it is designed to produce, namely, a logical summation of common properties.

Precedence-inclusion patterns are based on axiomatizing the interactions between one object coming before another, perhaps in time, and one object being contained in another. These interacting relationships can be seen everywhere, so we expect the theory laid out in this paper to have many applications beyond text. For instance, in video, the ordering of the frames determines precedence, while within frames different picture elements can be components of larger elements. One can also apply it to the analysis of images, as should be clear from the examples we present.

Since the theory of pattern generalization we describe can be specialized to give a new theory of generalization for partially ordered sets, we believe this work can be applied in the future to the generalization of ontologies, including classification hierarchies for documents or webpages.

This paper is about giving a feeling for what precedence-inclusion patterns are and how to compute with them. We will also state and explain the main theorem underlying possible applications to relational learning. It asserts that any finite set of finite precedence-inclusion patterns has a minimal most specific generalization that is unique up to isomorphism. The space allocated is too little to give proofs of mathematical results. These will be presented elsewhere.

2 Motivating Problems

Here's an example of the kind of generalization situation we'd like to analyze in a principled way: 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

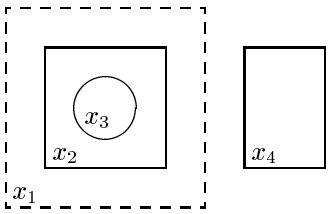

Figure 1: The pattern X, in which $R(x_2, x_4)$ holds.

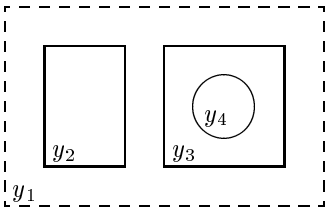

Figure 2: The pattern Y, in which $R(y_2, y_3)$ holds.

(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 does not necessarily require that sentences be fully parsed. 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). Working out the details can be a fairly easy exercise for the reader once we define precedence-inclusion patterns and pattern-preserving maps.

As for a motivating problem for our approach to relational learning, the fact is that geometric examples are the most accessible, while at the same time providing easily stated problems with nonobvious solutions, so that is what we will give. Also, geometric examples also serve to demonstrate that precedence-inclusion patterns have a potential use in extracting information from images.

So, consider the arrangements of picture elements shown in Figures 1 and 2. In both of these patterns, the dimensions of the picture elements don't matter. The arrangement of the elements is what counts. Specifically, we are interested in whether or not an element a strictly precedes another one b (notation: $a \prec b$) by being entirely to the left of it in one of the figures, or whether or not a strictly includes b (notation: $a \sqsupset b$). Also, we pay attention to the gross properties of picture elements. In this case, these properties state whether an element is circular or retangular, and if it is rectangular, whether its boundary is a dashed line or a solid line. In each of X and Y, two elements are identified as being in the relation R to one another. The problem is this:

> On the basis of the evidence provided by X and Y, can we claim in a principled way that another instance of the relation R occurs in the pattern

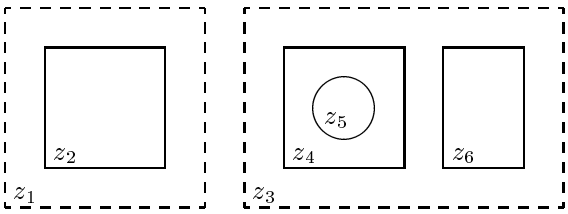

Figure 3: The pattern Z. Does $R(a, b)$ hold for $a, b \in Z$?

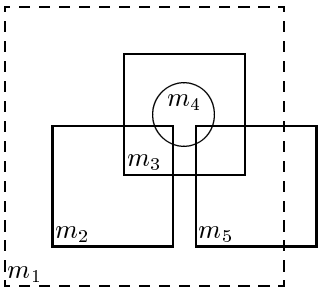

Figure 4: The pattern M, in which $R(m_2, m_5)$ holds.

Z shown in Figure 3?

To answer this question, later we will compute the pattern M, represented in Figure 4, from X and Y. In the pattern M, the relation $R(m_2, m_5)$ holds. and M is a minimal most specific generalization of X and Y. Note that M is really a pattern, not just a logical clause (although we were somewhat lucky that we could make a such a nice drawing portraying it). Why do we even say that M is a generalization of, say, X? The reason is that there is a pattern-preserving map from M into X. Think of it as a movement and distortion of picture elements. This is a mapping that preserves the gross properties of picture elements and that does not break any precedence or inclusion relations present in M. In this case, the mapping is not one-to-one. Because M has more elements than either X or Y, M can't be obtained by discarding elements of, say, X that are incompatible with Y. By constructing a pattern-preserving map from (a pattern essentially the same as) M into Z that sends m_2 to z_4 and m_5 to z_6, we can assert that $R(z_4, z_6)$ holds on the basis of generalizing from X and Y. However, generalization from X and Y does not permit us to assert $R(z_2, z_4)$ because there is no pattern-preserving map that sends m_2 to z_2 and m_5 to z_4. Determining whether $R(z_2, z_6)$ holds on the basis of X and Y is left as what should now be an easy exercise for the reader.

Note that the patterns of precedence and inclusion in our relational learning example could have been presented diffently to make them look more as though they were based on partially parsed strings. Thus, we could have written X more opaquely, but

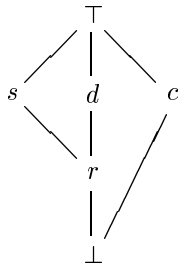

Figure 5: The lattice L for our geometric examples.

quite compactly, as

$$X = (x_1^d(x_2^{s,1}(x_3^c)))(x_4^{s,2}).$$

Here the exponents $d, s,$ and c stand for "dashed rectangle," "solid rectangle," and "circle," respectively, while 1 and 2 name the arguments that make R true. Similarly, the pattern Y can be expressed linearly as

$$Y = (y_1^d(y_2^{s,1})(y_3^{s,2}(y_4^c))).$$

However, it can be proved that the pattern M cannot be expressed in such a linear fashion. We can't get out of this bind by picking a different minimal most specific generalization. As patterns, they are all isomorphic. Since constituent structure trees are linear in the sense that they correspond to parses of strings, this example illustrates the root cause of why constituent structure trees form an unsatisfactory setting for a theory of generalization.

3 Basic Definitions

Let L be a complete lattice, whose elements correspond to sets of properties that may be possessed by the elements of patterns. (Recall that a *complete lattice* is a partially ordered set every subset of which has a greatest lower bound. In this paper we will use almost no lattice theory, but as a lattice theory reference, we recommend [3].) The complete lattice L is called the *property lattice.* The power set of a set, ordered by set inclusion, is a complete lattice. In the degenerate case, a complete lattice may have a single element; it can arise as the power set of the empty set. The degenerate case is the sole case in which the top and the bottom of a complete lattice can coincide. The reason L taken to be a complete lattice instead of just a lattice is to be able to prove arbitrary products exist (for the definition, see [4], p. 69) in categories of precedence-inclusion patterns.

For the examples $X, Y, Z,$ and M we will take L to be the complete lattice whose Hasse diagram is shown in Figure 5. Here the lattice elements d, s, c and r signify "dashed rectangle," "solid rectangle," "circle," and "rectangle," respectively, where the presence of the property r allows us to support patterns in which an element can be instantiated in an example by rectangle with either kind of boundary. The lattice element

$\bot$ is the label applied to a pattern element to which no other property is attributed. For example, in a particular pattern, the presence of an element labeled $\bot$ might signify simply that something whose properties are immaterial must precede or must include something else. The lattice element $\top$ is the label assigned to a pattern element having all properties, but in these particular examples, some properties are inconsistent with others, so $\top$ won't be used as an actual pattern element label.

Let A be a set of *argument names*, and let R, *the relation of interest*, be an A-relation defined over a set of objects that may appear in patterns. In our geometric examples, R is binary, so for those examples we take $A = \{1, 2\}$. We will map elements of A to elements of a structure that exemplifies a pattern in order to indicate those elements of the structure for which the relation of interest holds. Sometimes the mapping will only be defined for some elements of A. Keeping in mind that the aim of relational learning is often to support *template filling*, we use partial functions defined on A to support *partial template filling*. For instance, in text one might be looking to extract the name of a company matched with the company's law firm and the company's auditor. But maybe only two of these three items would be found in a particular example, in which case we would like to extract as much information as we can through partial template filling.

By a *strict partial order* we mean an irreflexive, transitive, binary relation. (The definition is fairly standard, but not universal. For instance, since asymmetic and transitive is equivalent to irreflexive and transitive, the same concept is termed a strict partial order in [9], p. 72.) Strict partial orders are closely related to partial orders: the union of a strict partial order on a set with the identity relation on that set yields a partial order, while subtracting the identity relation from a partial order yields a strict partial order ([9], p. 73).

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 definitions permit the set A, the lattice L, and prcedence-inclusion patterns themselves all to be infinite. However, we anticipate that finite patterns will find the most applications. When A is empty, we will call a pattern a *classification pattern* because the issue of identifying elements within the pattern becomes moot and the pattern can only be used to determine membership of the structure as a whole in a class

of interest. When L is the degenerate one-point complete lattice, then we call a pattern *purely positional* because the labeling of elements does not count. In this case only the arrangement of elements matters.

Note that the relation R is only in the background because it did not explicitly enter into the definition of precedence-inclusion pattern. This is because we make no use of the logical properties of R, and because we assume that there is only one relation of interest. (If there were more than one, we might have to worry about how they interact, and we want to avoid the complexity that would entail.)

4 Constituent Structure Trees

Constituent structure trees give examples of classification patterns. 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 $\lambda : T \rightarrow Q$ 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. However, the following important, but fairly obvious, theorem holds.

Theorem 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 a pattern requires strict inclusion to be a strict partial order. This is just a nit.

2. Constituent structure trees have a requirement of finiteness. Letting precedence-inclusion patterns be infinite is needed to argue convincingly that the category of patterns generalizes the category of sets because then one can prove that category of sets is isomorphic to a full, coreflexive subcategory (see [4], p. 89) of the category of patterns.

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 (see [4], p.62) in categories of classification patterns.

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, for transitive relations, 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 4.

5 Mathematical Theory of Patterns

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

Let P and Q be patterns. We say that a precedence-inclusion pattern P is a *subpattern* of a pattern Q if

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

Pattern generalization is based on the concept of a pattern-preserving map. A *pattern-preserving map* shows how to find all of the structure in its domain in its codomain. In more detail, if $\mathrm{dom}\,\alpha_P \not\subseteq \mathrm{dom}\,\alpha_Q$, then there are no pattern-preserving maps from P to Q. If $\mathrm{dom}\,\alpha_P \subseteq \mathrm{dom}\,\alpha_Q$, then a pattern-preserving map $f : P \to Q$ from P to Q is a (total) function from P to Q satisfying, for all $x, y \in P$ and for all $a \in \mathrm{dom}\,\alpha_P$,

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

We say P *is a generalization* Q if there exists a pattern-preserving map $f : P \to Q$. Suppose $\mathcal{Q} = \{Q_i \mid i \in I\}$ is an I-indexed family of patterns. We say P *is a generalization of* $\mathcal{Q}$ if, for all $i \in I$, P is a generalization of Q_i. We say P *is a most specific generalization of* $\mathcal{Q}$ if P is a generalization $\mathcal{Q}$ and any generalization of $\mathcal{Q}$ is a generalization of P. We say P *is a minimal most specific generalization of* $\mathcal{Q}$ if P is a most specific generalization of $\mathcal{Q}$ and no proper subpattern of P is a most specific generalization of $\mathcal{Q}$.

Theorem 2 *Every finite set of finite patterns has a minimal most specific generalization, which is unique up to isomorphism.*

The procedure for computing the minimal most specific generalization of a finite set of finite patterns depends on a refinement of the concept of a pattern-preserving map. A pattern-preserving map $g : P \to P$ is a *retraction of* P if it is idempotent, i.e., for all $x \in P$, $g(g(x)) = x$. A subpattern Q of P is said to be a *retract of* P if it is the image of a retraction of P. By a *proper retract of* P, we mean a retract of P that is a proper subset of P. These concepts have analogs in topology (see [5], p. 216) and in domain theory (see [1] or [8]).

The other ingredient for computing minimal most specific generalizations is the product of a set of patterns. We shall give the details the construction of products only for an ordered pair of patterns, but the same idea works for any size index set. As a set, $P \times Q$ is the Cartesian product of the sets of elements of P and of Q. The relations of strict precedence and strict inclusion are defined componentwise, so

1. $\langle x_1, x_2\rangle \prec_{P\times Q} \langle y_1, y_2\rangle$ iff $x_1 \prec_P y_1$ and $x_2 \prec_Q y_2$, and
2. $\langle x_1, x_2\rangle \sqsupset_{P\times Q} \langle y_1, y_2\rangle$ iff $x_1 \sqsupset_P y_1$ and $x_2 \sqsupset_Q y_2$.

For the labeling function, we use the greatest lower bound operation of the lattice:

$$\Lambda_{P\times Q}(\langle x_1, x_2\rangle) = \Lambda_P(x_1) \wedge \Lambda_Q(x_2).$$

For the argument naming function, we let $\mathrm{dom}\,\alpha_{P\times Q} = \mathrm{dom}\,\alpha_P \cap \mathrm{dom}\,\alpha_P$, and, where defined, we let

$$\alpha_{P\times Q}(a) = \langle \alpha_P(a), \alpha_Q(a)\rangle.$$

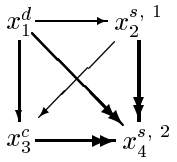

Figure 6: Directed acyclic graph representation of X.

We can now give a very simple procedure that returns 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;

Finding proper retracts of patterns, the main step in the procedure, is not hard in practice. Our experience indicates that almost always a finite pattern of cardinality m that has a proper retract also has a proper retract of cardinality $m-1$. In other words, in executing the procedure usually we can get a sequence of ever-smaller retracts by using retractions that move only a single point. Such pattern-preserving maps are easy to find. However, algorithmic details will be a subject for future research.

The patterns $P_1, \cdots, P_n$ that provide the input to the procedure are generally *unstripped*, i.e., their argument naming functions are nonempty because they contain instances of the relation of interest. A pattern S to which we wish to apply the results of the procedure is initially *stripped*, i.e., has an empty argument naming function. Any pattern P can obviously be turned into an stripped pattern $P^\dagger$. For each pattern-preserving map $f : M^\dagger \to S$ that we can construct, we can define an argument naming function on S, which makes a pattern S_f such that $S_f^\dagger = S$ and $f : M \to S_f$ is pattern-preserving, which results in the identification of an instance of the relation R in the pattern S. When we introduced the motivating examples and referred to "(a pattern essentially the same as) M," we were referring to $M^\dagger$.

6 The Minimal Most Specific Generalization of X and Y

To actually determine that M, shown in Figure 4, is the minimal most specific generalization of X and Y, the procedure tells us to start with $X \times Y$. Observe that we can represent X as a directed acyclic graph with two varieties of edges, as shown in Figure 6. In the graph, thicker, double-headed arrows indicate strict precedence and thinner, single-headed arrows indicate strict inclusion. Similarly, we depict Y by the graph shown in Figure 7. From these graphical representations of X and Y, we can obtain a representation of $X \times Y$ as shown in Figure 8. Note that some of the elements

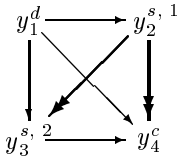

Figure 7: Directed acyclic graph representation of Y.

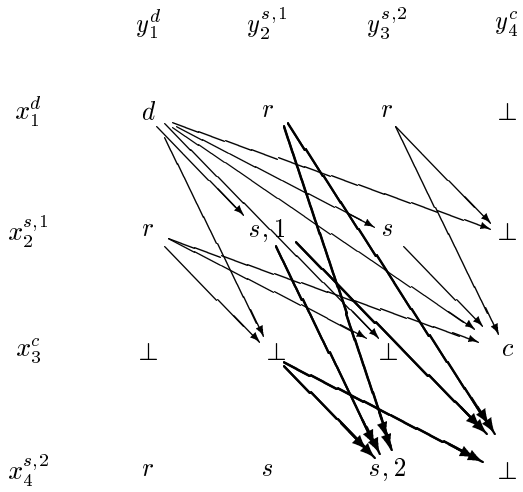

Figure 8: Directed acyclic graph representation of $X \times Y$.

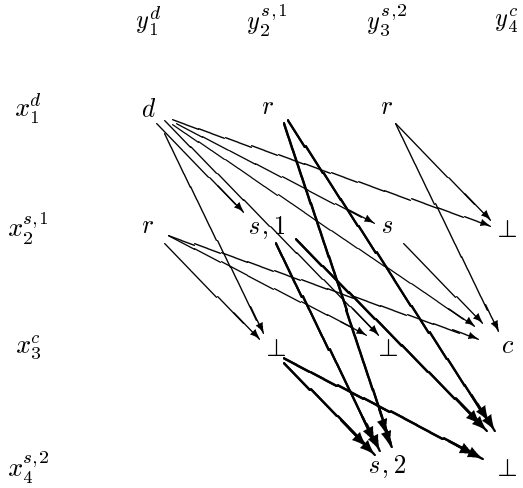

Figure 9: Directed acyclic graph representation of M_5.

of $X \times Y$ have the label $r = d \wedge s$, and others have the label $\bot = c \wedge d = c \wedge s$. Since $\bot \leq r$, observe that mapping $\langle x_1, y_4 \rangle$ to $\langle x_1, y_3 \rangle$, while leaving all other points fixed, is a retraction of $M_1 = X \times Y$ whose image is M_2. There is a retraction of M_2 that moves $\langle x_4, y_1 \rangle$ to $\langle x_2, y_1 \rangle$, and leaves all other points fixed, whose image is M_3. Moving $\langle x_3, y_1 \rangle$ to $\langle x_2, y_1 \rangle$ gives the next retract M_4, and moving $\langle x_4, y_2 \rangle$ to $\langle x_2, y_2 \rangle$ gives the retract M_5, shown in Figure 9. We have now eliminated all the "isolated" elements of $X \times Y$. Since $r \leq d$, and since $\langle x_1, y_1 \rangle$ is related by the same relations to the all of the elements to which $\langle x_1, y_3 \rangle$ is related, M_5 has a retraction that moves $\langle x_1, y_3 \rangle$ to $\langle x_1, y_1 \rangle$, and leaves all other points fixed. The image of this retraction is M_6, shown in Figure 10. Then M_7 comes from moving $\langle x_1, y_2 \rangle$ to $\langle x_2, y_2 \rangle$. M_8 comes from moving $\langle x_2, y_1 \rangle$ to $\langle x_1, y_1 \rangle$. M_9 comes from moving $\langle x_2, y_4 \rangle$ to $\langle x_2, y_3 \rangle$. M_{10} comes from moving $\langle x_3, y_3 \rangle$ to $\langle x_3, y_2 \rangle$. M_{11} comes from moving $\langle x_3, y_2 \rangle$ to $\langle x_2, y_2 \rangle$. (Note that moving $\langle x_3, y_2 \rangle$ to $\langle x_2, y_2 \rangle$ and leaving all other points fixed is not a retraction of our original pattern $X \times Y$, so new opportunities for finding simple retractions can open up as the process continues.) M_{12} comes from moving $\langle x_4, y_4 \rangle$ to $\langle x_4, y_3 \rangle$. Then $M \cong M_{12}$ has no proper retracts, and is pictured in Figure 11. One can easily see that the patterns represented in different ways in Figures 4 and 11 are isomorphic.

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, Cambridge, 1990.

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

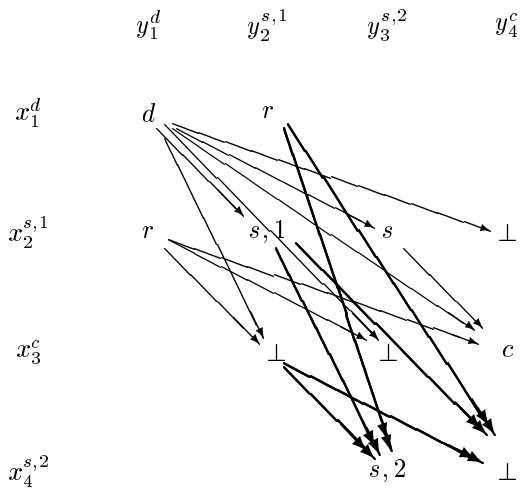

Figure 10: Directed acyclic graph representation of M_6.

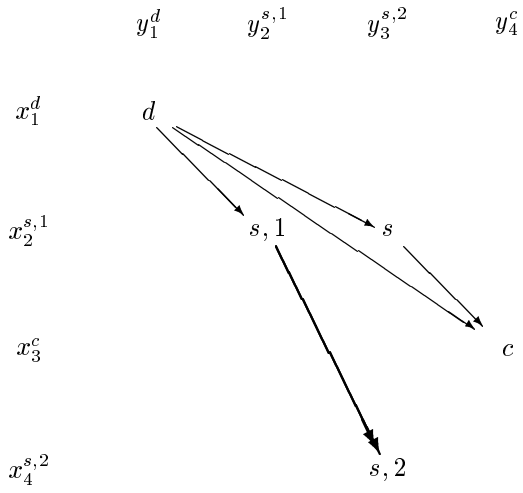

Figure 11: Directed acyclic graph representation of M_{12}.

[3] B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, 1990.

[4] S. MacLane. *Categories for the Working Mathematician*. Springer-Verlag, New York, 1971.

[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] 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.

[8] 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, New York, 1972.

[9] P. Suppes. *Axiomatic Set Theory*. D. Van Nostrand Co., Inc., Princeton, 1960.