Pattern Recognition, Vol. 10, pp. 223–236.
Pergamon Press Ltd. 1978. Printed in Great Britain.

0031-3203/78/0601-0223 $02.00/0

STRUCTURAL PATTERN RECOGNITION, HOMOMORPHISMS, AND ARRANGEMENTS*†

ROBERT M. HARALICK

Department of Electrical Engineering, Department of Computer Science,
University of Kansas, Lawrence, KS 66045, U.S.A.

(*Received* 28 *November* 1977; *received for publication* 3 *January* 1978)

Abstract – This paper discusses the general supervised pattern discrimination problem from a structural point of view. We show that both the problem of determining a decision rule and the problem of applying a decision rule are problems of finding homomorphisms, whether the pattern data structure is an N-tuple as in statistical pattern recognition, or a string or its generalizations as in syntactic pattern recognition. We then introduce the concept of an arrangement, which is a labeled N-ary relation, as a more complex pattern data structure and show how decision rules can be constructed and applied to arrangements using the homomorphism concept. The methodology suggested in the paper provides a structural pattern recognition generalization to phrase-structured syntactic pattern recognition.

Pattern inference Grammatical inference Statistical pattern recognition Syntactic pattern recognition Theory of covers Homomorphisms Relations Arrangements Structural pattern recognition

1. INTRODUCTION

The purpose of this paper is threefold: (1) To provide a unified treatment of N-tuple and string pattern recognition which clarifies their common structural basis; (2) To introduce the arrangement as a pattern data structure; and (3) To show how to use the arrangement in an analogous fashion to the use of N-tuple and string in structural pattern recognition.

Statistical pattern recognition uses the N-tuple for the basic data structure of a pattern. Each pattern must be an ordered list of values and the lists must be the same length. If the values are real numbers, the concept of distance yields linear or quadratic decision rules to determine the category to which an N-tuple should be assigned.[11] If the values are non-numeric symbols, then decision rules using cylinder set covers can be used.[14,19]

Syntactic pattern recognition uses the string for the basic data structure of a pattern. Each pattern is called a sentence and must be the concatenation of symbols from a given symbol set. If the set of sentences associated with a category is a finite state language (those generated by a regular phrase structure grammar), then a decision rule which decides category assignments can be a finite state automaton.[12]

Much work has been done to generalize the string data structure of phrase structure syntactic pattern recognition. Data structures of trees,[2,5] arrays,[3,15,21] plexes,[6,7,22,26] and webs[24] have all been described. However, in this paper, when discussing phrase structure grammar, we will restrict ourselves to strings only.

In structural pattern recognition, each pattern class is characterized by a unique set of relationships between the parts of the patterns in the class. The emphasis is on the combinatorial complexity and uniqueness which this relational organization can have. Structural pattern recognition assumes the combinatorial complexity is high enough to permit an almost 100% correct classification rate. Therefore, instead of the error rate analysis methods typical of statistical pattern recognition, structural pattern recognition utilizes what amounts to covering techniques to generate decision rules. The set covering methodology of Michalski[19] or the production rules of MYCIN[4,27] and the grammatical inference procedures surveyed by Fu and Booth,[10] Biermann and Feldman[1] as well as those described by Feldman,[8] and Pao [23] are all special cases of the general covering paridigm which we believe characterizes structural pattern recognition.

There are four basic problems which structural pattern recognition faces: (1) the choice of data structure for the pattern (a design problem); (2) the translation of the real world measurements to the data structure of the pattern (a feature extraction problem); (3) the construction of the decision rule (a generalization problem); and (4) the application of the decision rule (an implementation problem).

In this paper we show that from a structural pattern recognition perspective, problem (3), the construction of the decision rule is really a problem expressible in terms of determining covers by homomorphisms and determining decision rules, utilizing these covers by homomorphisms. Once this is understood to be the same general kind of problem, whether the data structure for the pattern is an N-tuple, string, or the

† Support for this work was provided by grant AFOSR 77-3307.

string generalizations mentioned earlier, we can naturally inquire about the possibilities of structural pattern recognition with other kinds of data structures. To this end we introduce the labeled N-ary relation as a data structure which is more complex than the string or N-tuple and which includes the N-tuple, the string, and its generalizations, as special cases. We call the labeled N-ary relation an arrangement and illustrate how decision rule construction can take place using the arrangement as a pattern data structure.

Section 2 discusses how structural pattern recognition is a problem of finding homomorphisms. Section 3 illustrates a set covering example using the structural pattern recognition methodology, while Section 4 interprets a syntactic grammatical inference example as one which fits the structural pattern recognition methodology. Section 5 describes the arrangement data structure and illustrates its use in a structural pattern recognition approach.

2. HOMOMORPHISMS AND STRUCTURAL PATTERN RECOGNITION

In this section we describe how the decision rule construction problem is a problem of finding homomorphisms. The mathematical description given here is designed to be general enough for all structural pattern recognition and is written with the aim of unifying on a theoretical level all the basic processes involved. Some of the early ideas which motivated this section can be found in Haralick.[13] We begin with some definitions.

Let P be the set of patterns and C be the set of categories. Let $P' \subseteq P$ be the set of observed patterns. A training data set is a binary relation $T \subseteq P' \times C$ which pairs observed patterns with their true category identifications. Typically, the fraction of patterns in P which are observed is very small. Also each category in C is associated with at least one observed pattern; hence $T(P') = C$ so that T is onto. Because we assume no error in the case of structural pattern recognition, if an observed pattern is paired to a category, it is paired to at most one category. That is, T is single-valued; $(p,c) \in T$ and $(p,c') \in T$ imply $c = c'$. Given some kind of structure on the set of patterns P, the decision rule construction problem is to find a binary relation D, called the decision rule, which pairs each pattern in P to categories in C and which is much larger than T; thus we must have $T \subseteq D \subseteq P \times C$. Note that we do not require D to be single-valued.

The structure on the pattern set P is a given collection of subsets of P which cover it. We denote such a collection by $\mathscr{L}$; $\mathscr{L}$ is some given subset of the set of all subsets (the power set) of P. The structure $(P, \mathscr{L})$ may be a neighborhood space or a topological space. The partial ordering on P induced by $\mathscr{L}$ may be a lattice, or as emphasized in this paper, $\mathscr{L}$ may be the collection of homomorphic images of elements of P.

The decision rule construction problem is how to assign to some category patterns which have not been observed. In other words, it is how to generalize from the training set. In structural pattern recognition, the generalization takes place through the cover $\mathscr{L}$. The decision rule pairs a pattern $p \in P$ with a category $c \in C$ when there exists a subset $L \in \mathscr{L}$ satisfying the generalizing conditions

(1) $L \cap P' \neq \phi$

(2) $p \in L \cap P'$ implies $(p,c) \in T$.

The first condition states that generalization can only take place through a subset L which contains one of the observed patterns. We only generalize from patterns for which there is some category identification information. The second condition forces uniqueness: all observed patterns in L must have the same category identifi ation by T. We call a subset $L \in \mathscr{L}$ that satisfies (1) and (2) a generalizing set. In the remainder of this section we will analyze generalizing sets and illustrate the role which homomorphisms play.

We define a restriction of $\mathscr{L}$ to be that collection $\mathscr{L}'$ which contains only those members L of the collection $\mathscr{L}$ which satisfy the generalizing conditions (1) and (2). That is,

$$\mathscr{L}' = \{L \in \mathscr{L} \mid \text{(1) } L \cap P' \neq \phi \text{ and (2) There exists a } c \in C \text{ satisfying } (p,c) \in T \text{ for every } p \in L \cap P'\}.$$

Associated with $\mathscr{L}$ is the natural binary relation $F \subseteq P \times \mathscr{L}$ which pairs each pattern with each member of $\mathscr{L}$ to which the pattern belongs.

$$F = \{(p,L) \in P \times \mathscr{L} \mid p \in L\}.$$

Let F' be the restriction of F to $\mathscr{L}'$; $F' = F \cap (P \times \mathscr{L}')$. The members of $\mathscr{L}'$ must be sufficient to allow generalization to take place using all the information in the training set. This means that for each observed pattern, there must exist a consistent subset in $\mathscr{L}'$ containing the observed pattern. Thus we expect that for each observed pattern p, there exists an $L \in \mathscr{L}'$ satisfying $(p,L) \in F'$; hence, F' is defined everywhere on P'.

To show how a decision rule can be constructed which utilizes the cover $\mathscr{L}'$, we need to define a particular kind of relation composition and a homomorphism based on this composition. Let $E \subseteq A \times B$, $G \subseteq A \times C$, and $H \subseteq B \times D$. This kind of composition translates pairs of E through G and H to pairs in $C \times D$. The translation takes place componentwise. The first component is translated through G, and the second component is translated through H.

Definition 1

We denote the composition of E with (G,H) by $E \cdot (G,H)$ and define $E \cdot (G,H) = \{(c,d) \in C \times D \mid$ for some $(a,b) \in EF$, $(a,c) \in G$ and $(b,d) \in H\}$.

Figure 1 illustrates the composition idea using a commutative diagram. Notice that if $A = C$ and G is the identity on A, then the resulting composition corresponds to the usual idea of function composition.

Our natural concept of homomorphism is a

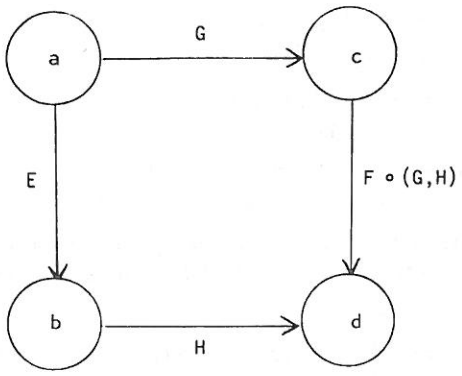

Fig. 1. Illustration of the relation composition concept. If $(a,b)\in E$, and $(a,c)\in G$ and $(b,d)\in H$, then the pair (a,b) gets translated to (c,d) by the composition of E with (G,H).

structure-preserving mapping from one set to another. The mapping of structure is achieved by the composition just defined.

Definition 2

(G,H) is a homomorphism from $E\subseteq A\times B$ into $W\subseteq C\times D$ if and only if $E\cdot(G,H)\subseteq W$.

In other words, every pair in E must be translated to some pair in W. However, there may be pairs in W which are not the translation of any pair in E.

We use the concept of relational composition and homomorphism in the following way. We define the inclusion relation $I_{P'}$ by $I_{P'}=\{(p,q)\in P\times P'|p=q\}$. The association Q between members of $\mathscr{L}'$ with categories of C can then be obtained as the relation composition of $I_{P'}$ with (F',T): $Q=I_{P'}\cdot(F',T)$. Q is a homomorphic image of the inclusion relation $I_{P'}$. As illustrated in Fig. 2, this means that a generalizing set L is paired by Q with a category c if there is an observed pattern $p\in P$ which is a member of L, $(p,L)\in F'$, and whose true category identification is c, $(p,c)\in T$.

The generalizing is taking place through the sets in $\mathscr{L}'$, and we must expect that the relationship between a pattern p and a generalizing set L as determined by F' must bear a similarity to the relationship of the pattern p to its category identification (if any) as

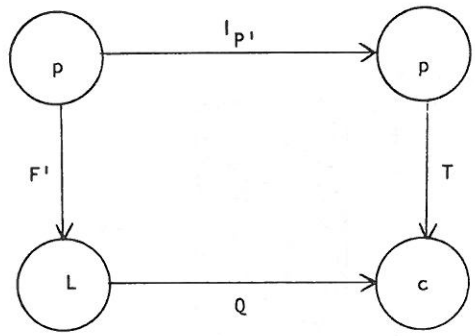

Fig. 2. Commutative diagram defining the relation Q by $Q=I_{P'}\cdot(F',T)$. Also, it should be apparent that under reasonable conditions, T is the homomorphic image of F'; $T=F'\cdot(I_{P'},Q)$.

determined by T. In some sense F' must contain at least as much information as T does. Given F' we must be able to translate it and produce T. In fact, it is the case that T is the homomorphic image of F' obtained with the homomorphism $(I_{P'},Q)$. Proposition 1 proves that $Q\subseteq I_{P'}\cdot(F',T)$ implies $F'\cdot(I_{P'},Q)\subseteq T$. Proposition 2 proves the converse and the theorem states that $Q=I_{P'}\cdot(F',T)$ implies $T=F'\cdot(I_{P'},Q)$.

Proposition 1. Suppose $Q\subseteq I_{P'}\cdot(F',T)$. Then $F'\cdot(I_{P'},Q)\subseteq T$.

Proof. Let $(p,c)\in F'\cdot(I_{P'},Q)$. Then there exists $(p',L)\in F'$ such that $(p',p)\in I_{P'}$ and $(L,c)\in Q$. But $(p',p)\in I_{P'}$ implies $p=p'$. Since $(L,c)\in Q$ and $Q\subseteq I_{P'}\cdot(F',T)$, there exists $(q,q)\in I_{P'}$ such that $(q,L)\in F'$ and $(q,c)\in T$. Since $L\in\mathscr{L}'$, there exists a $c'\in C$ satisfying $(p^*,c')\in T$ for every $p^*\in L\cap P'$. But $q\in L\cap P'$ so that $(q,c')\in T$. But T is single-valued so that $(q,c')\in T$ and $(q,c)\in T$ imply $c=c'$. Finally, $(p',L)=(p,L)\in F$ and $(p,p)\in I_{P'}$ imply $p\in L\cap P'$ so that $(p,c)\in T$.

Proposition 2. Suppose $I_{P'}\cdot(F',T)\subseteq Q$. Then $T\subseteq F'\cdot(I_{P'},Q)$.

Proof. Let $(p,c)\in T$. Then $(p,p)\in I_{P'}$. Also, since F' is defined everywhere, there exists a $L\in\mathscr{L}'$ such that $(p,L)\in F'$. Hence, $(L,c)\in I_{P'}\cdot(F',T)\subseteq Q$. Now, $(p,L)\in F'$, $(p,p)\in I_{P'}$, and $(L,c)\in Q$ imply $(p,c)\in F'\cdot(I_{P'},Q)$. Thus $T\subseteq F'\cdot(I_{P'},Q)$.

Theorem 1. $Q=I_{P'}\cdot(F',T)$ implies $T=F'\cdot(I_{P'},Q)$.

Proof. Propositions 1 and 2.

The theorem basically means that once given a training data relation T and a cover $\mathscr{L}$, the structural pattern recognition paradigm seeks to determine a relation F' which is the inverse homomorphic image of T. Generalizing means finding inverse homomorphic images of the training data.

Once the generalizing has been done, it is a simple matter to construct the decision rule. We need only translate the relation F' to a relation that pairs patterns with categories. We do this in the following way. Let I_P be the identity on P. We define the decision rule D as a binary relation from P into C by $D=F'\cdot(I_P,Q)$. Whereas the training data relation T is the homomorphic image of F' under homomorphism $(I_{P'},Q)$, the decision rule is the homomorphic image of F' under the homomorphism (I_P,Q). The decision rule pairs a pattern p with a category c if the pattern is in some generalizing set L which contains an observed pattern paired with category c by T.

To summarize, the structural pattern recognition paradigm begins with a cover $\mathscr{L}$ on the set of patterns P. Then using the training data relation T, a relation F' is determined which pairs patterns to generalizing sets in the cover $\mathscr{L}$. This relation F' is an inverse homomorphic image of the training data relation. The decision rule D is constructed as a homomorphic image of F'.

In Sections 3 and 4 we explain N-tuple and string structural decision rule construction in terms of these kinds of homomorphisms. In Section 5 we introduce the arrangement data structure for a pattern and

illustrate how decision rules for it, too, can follow the same structural pattern recognition paradigm.

3. N-TUPLE PATTERN DISCRIMINATION USING COVERS

In this section we give an example of the construction of a decision rule using the structural pattern recognition technique of Section 2. We show how $\mathscr{L}$ can be defined as the homomorphic image of elements of P, where the structure preserved by the homomorphism relates to a pseudo-metric on the power set of P. Then we show the comparison between this technique and the Michalski covering technique,[17–20] which is a top down version of the Quine–McCluskey technique for minimization of Boolean variables[16,25] for non-Boolean variables. Haralick[14] gives a description of some of the covering ideas discussed in this section.

We begin our discussion with definitions for Cartesian products, metrics, cylinder operators, and homomorphisms.

Definition 3

Let $J = \{j_1, j_2, \ldots, j_N\}$ be a linearly ordered finite set whose elements satisfy $j_n < j_{n+1}$, $n = 1, \ldots, N-1$. Then, we define the *Cartesian product* with respect to an indexing set J of a selected group of sets $D_1, D_2, \ldots, D_K$ by $\underset{j\in J}{\chi} D_{j_1} = D_{j_2} \times \ldots \times D_{j_N}$.

Definition 4

A real valued function ρ defined on $P \times P$ is a *metric* if and only if

(1) $\rho(x,y) \geq 0$ with equality if and only if $x = y$ (non-negativity)
(2) $\rho(x,y) = \rho(y,x)$ (symmetry)
(3) $\rho(x,y) \leq \rho(x,y) + \rho(y,z)$. (triangular inequality)

The pair (P, ρ) is called a *metric space.*

A function ρ which satisfies the weakened condition

(1) $\rho(x,y) \geq 0$ and $x = y$ implies $\rho(x,y) = 0$
(2) $\rho(x,y) = \rho(y,x)$

is called a pseudo-metric.

When the set of patterns P is a Cartesian product set $\underset{j\in J}{\chi} D_j$, a natural metric ρ can be defined on P which counts the number of components in which two patterns differ.

Proposition 3. Let ρ be a function defined on $(\underset{i\in I}{\chi} D_i) \times (\underset{i\in I}{\chi} D_i)$ given by $\rho((x_1, \ldots, x_N), (y_1, \ldots, y_N)) = \#\{i \in I | x_i \neq y_i\}$. Then ρ is a metric on $\underset{i\in I}{\chi} D_i$.

Proof. Clearly ρ satisfies the non-negativity and symmetry conditions. To show the triangular inequality, let $(z_1, \ldots, z_N) \in \underset{i\in I}{\chi} D_i$. Then

$$\begin{aligned}
&\rho((x_1, \ldots, x_N), (y_1, \ldots, y_N)) \\
&\quad = \#\{i \in I | x_i \neq y_i\} \\
&\quad = \#(\{i \in I | x_i \neq y_i,\ y_i \neq z_i,\ x_i \neq z_i\} \\
&\qquad \cup \{i \in I | x_i \neq y_i,\ y_i \neq z_i,\ x_i = z_i\} \\
&\qquad \cup \{i \in I | x_i \neq y_i,\ y_i = z_i,\ x_i \neq z_i\} \\
&\qquad \cup \{i \in I | x_i \neq y_i,\ y_i = z_i,\ x_i = z_i\}.
\end{aligned}$$

Since $\{i \in I | x_i \neq y_i, y_i = z_i, x_i = z_i\} = \phi$, we may union the first two sets and the first and third sets to obtain

$$\begin{aligned}
&\#\{i \in I | x_i \neq y_i\} \\
&\quad = \#(\{i \in I | x_i \neq y,\ y_i \neq z_i\} \cup \{i \in I | x_i \neq y_i,\ x_i \neq z_i\}) \\
&\quad \leq \#(\{i \in I | y_i \neq z_i\} \cup \{i \in I | x_i \neq z_i\}) \\
&\quad \leq \#(\{i \in I | y_i \neq z_i\} + \{i \in I | x_i \neq z_i\}).
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\rho((x_1, \ldots, x_N), (y_1, \ldots, y_N)) \\
&\quad \leq \rho((x_1, \ldots, x_N), (z_1, \ldots, z_N)) \\
&\qquad + \rho((z_1, \ldots, z_N), (y_1, \ldots, y_N)).
\end{aligned}$$

It is easy to verify that the function ρ' defined by

$$\rho'(A,B) = \min_{x\in A} \min_{y\in B} \rho(x,y),$$

is a pseudo metric on the power set $\mathscr{P}(\underset{i\in I}{\chi} D_i)$.

Functions which are distance decreasing function on a metric or pseudo-metric space are called contraction mappings, and for us contraction mappings are playing the role of homomorphisms.

Definition 5

Let (P', ρ') be a metric or pseudo-metric space. Let $h: P' \to P'$ satisfy $\rho'(x,y) \geq \rho'(h(x), h(y))$. Then h is called

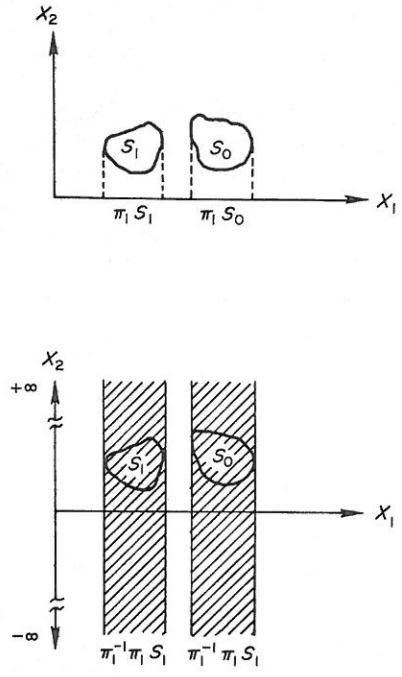

Fig. 3. Illustration of the projection and inverse projection operators for two subsets S_0 and S_1 of $D_1 \times D_2$.

a *contraction mapping* on P', and for our application we call h a *homomorphism.*

Corresponding to the natural metric ρ on a Cartesian product set $\underset{i\in I}{\times} D_i$, there is a natural class of functions from the power set $\mathscr{P}(\underset{i\in I}{\times} D_i)$ to $\mathscr{P}(\underset{i\in I}{\times} D_i)$ that we call cylinder operators. The cylinder operator extends any set to a neighborhood which contains it. The extension is achieved by taking each N-tuple in the set and letting designated coordinates run free.

Definition 6

Let $J \subseteq I$. The *cylinder operator* Ψ_J from $\mathscr{P}(\underset{i\in I}{\times} D_i)$ to $\mathscr{P}(\underset{i\in I}{\times} D_i)$ is defined by

$$\Psi_J(A) = \{(y_1, \ldots, y_N) \in \underset{i\in I}{\times} D_i \mid \text{for some } (x_1, \ldots, x_N) \in A, \; x_j = y_j \text{ for all } j \in J\}.$$

The subset $\Psi_J(A)$ is called the *J-cylinder set* of A. The *order* of the cylinder set or the order of the cylinder operator is the number $\#J$. See Fig. 3 for an example of some order 1 cylinder sets.

Proposition 4. Let $J \subseteq I$. The cylinder operator Ψ_J is a homomorphism on $\mathscr{P}(\underset{i\in I}{\times} D_i)$.

Proof. Let $A, B \in \mathscr{P}(\underset{i\in I}{\times} D_i)$. Since $A \subseteq \Psi_J(A)$ and $B \subseteq \Psi_J(B)$, $\min_{x\in\Psi_J(A)} \min_{y\in\Psi_J(B)} \rho(x,y) \leq \min_{x\in A} \min_{y\in B} \rho(x,y)$. Hence, by definition of ρ', $\rho'(\Psi_J(A), \Psi_J(B)) \leq \rho'(A,B)$; and this makes Ψ_J a homomorphism.

For structural pattern recognition with N-tuples, the collection $\mathscr{L}$ can be defined as containing all images of elements of the Cartesian product pattern set $P = \underset{i\in I}{\times} D_i$ by any cylinder operator having order less than or equal to some convenient constant k.

$$\mathscr{L} = \{L \subseteq P \mid L = \Psi_J(\{p\}) \text{ for some } p \in P = \underset{i\in I}{\times} D_i \text{ and } J \subseteq I \text{ satisfying } \#J \leq k\}.$$

Once the collection $\mathscr{L}$ is defined, the decision rule D can be determined by $D = F' \cdot (I_P, Q)$ where $Q = I_{P'} \cdot (F', T)$, T is the training data relation, and F' is the relation which pairs elements of p to generalizing subsets in $\mathscr{L}$.

We next illustrate this structural pattern recognition technique by an example.

We let pattern space or measurement space P be $D_1 \times D_2 \times D_3 \times D_4$ where

$$D_1 = \{x, y, z\}$$
$$D_2 = \{1, 2, 3\}$$
$$D_3 = \{A, B, R, S\}$$
$$D_4 = \{\alpha, \beta, \gamma\}.$$

We take $\mathscr{L}$ to be the set of all order 1 cylinder sets of P.

$$\mathscr{L} = \{L \subseteq P \mid \text{for some } p \in P, \; L = \Psi_J(p), j = 1, 2, 3, 4\}.$$

Let the set of categories be $C = \{0, 1\}$ and the set of observed patterns P' be

$$P' = \{(z,3,R,\alpha), (y,3,S,\alpha), (z,3,R,\beta), (y,3,B,\beta) \; (z,2,S,\alpha), (z,3,B,\gamma), (z,2,S,\beta), (z,3,A,\gamma)\}.$$

The training data relation $T \subseteq P' \times C$ is given in the table below.

T

Pattern	Category
$(z,3,R,\alpha)$	0
$(y,3,S,\alpha)$	0
$(z,3,R,\beta)$	0
$(y,3,B,\beta)$	0
$(z,2,S,\alpha)$	1
$(z,3,B,\gamma)$	1
$(z,2,S,\beta)$	1
$(z,3,A,\gamma)$	1

The order 1 cylinder sets which have non-empty intersection with patterns in P' are shown in the following list where $*$ means any legal value.

$$z***,\ y***,\ *2**,\ *3**,\ **A*,\ **B*,\ **R*,\ **S*,\ ***\alpha,\ ***\beta,\ ***\gamma.$$

Of these order 1 cylinder sets, those which are generalizing sets (involve only patterns from one category) are $y***$, $*2**$, $**R*$, $**A*$, $***\gamma$.

Notice that it is the case that each pattern in P' belongs to at least one of these cylinder sets. Hence the relation $F' \subseteq P \times \mathscr{L}'$ is, in fact, defined everywhere on P'. F' is given by the table below.

F'

Pattern	Cylinder set
$(z,3,R,\alpha)$	$**R*$
$(y,3,S,\alpha)$	$y***$
$(z,3,R,\beta)$	$**R*$
$(y,3,B,\beta)$	$y***$
$(z,2,D,\alpha)$	$*2**$
$(z,3,B,\gamma)$	$***\gamma$
$(z,2,S,\beta)$	$*2**$
$(z,3,A,\gamma)$	$**A*, ***\gamma$

The relation Q between the cylinder sets and the categories is defined by $Q = I_{P'} \cdot (F', T)$. Q is given in the table below.

Q

Cylinder set	Category
$y***$	0
$*2**$	1
$**R*$	0
$**A*$	1
$***\gamma$	1

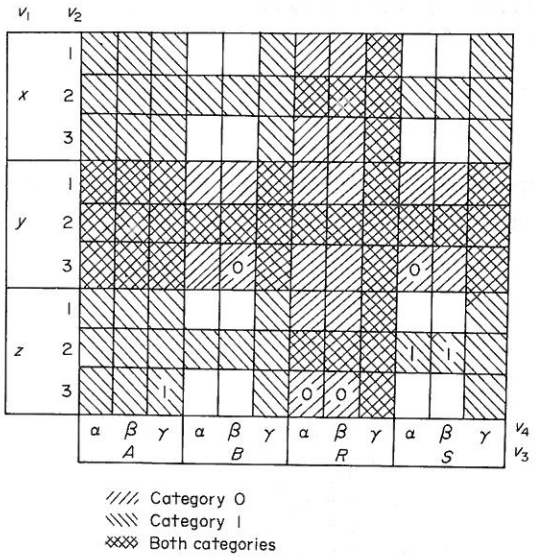

Fig. 4. A generalized Karnaugh map for the decision rule. Those areas hatched singly are patterns assigned to only one category. Those mean not hatched are not assigned to any category. Those areas which are doubly hatched are assigned to both categories.

The decision rule D which is a binary relation from the patterns to the categories is defined by $D = F' \cdot (I_P, Q)$. Hence

$$D = \{(p,c) \in P \times C \mid \text{if } c=0, \text{ then } p=(p_1,p_2,p_3,p_4)=y \text{ or } p_3=R; \text{ if } c=1, \text{ then } p_2=2, \text{ or } p_3=A, \text{ or } p_4=\gamma\}.$$

A measurement space picture of D appears in the Karnaugh map of Fig. 4. Notice that there may be patterns which are assigned no categories, patterns which are assigned to only one category, and patterns which are assigned to more than one category. Of course, training patterns are assigned uniquely to one category as required.

The relationship of this technique to the covering technique of Michalski is straightforward. First we define a cover of a set S_0 against a set S_1 and then an order n cover of S_0 against S_1.

Definition 7

A cover of S_0 against S_1 is any set $\mathscr{L}$ of cylinder sets satisfying $S_0 \subseteq \bigcup_{L \in \mathscr{L}} \subseteq S_1^c$, where S_1^c means the complement of set S_1.

The definition implies that if a collection of cylinder sets is to be a cover, its set theoretic union must completely cover the set S_0 and it must not cover any of the set S_1.

Definition 8

Another order n cover of S_0 against S_1 is any collection C of cylinder sets satisfying

(1) $\bigcup_{L \in \mathscr{L}} L \supseteq S_0$

(2) $\bigcup_{L \in \mathscr{L}} L \subseteq S_1^c$

(3) $L \in \mathscr{L}$ implies the order of L is less than or equal to n.

The covering technique requires the determination of a near minimal order cover of S_0 against S_1. As stated in Theorem 2, to construct a cover $\mathscr{L}$, we need only consider cylinder sets L satisfying $L \cap S_1 = \phi$ and $L \cap S_0 \neq \phi$. Then of those cylinder sets satisfying these conditions, we successively select those from lower order to higher order which cover patterns in S_0 not already previously covered by those sets selected[14].

Theorem 2. Let $\mathscr{L}_n = \{L \mid L \text{ is a cylinder set of order } \leq n, L \cap S_1 = \phi \text{ and } L \cap S_0 \neq \phi\}$. If $S_0 \subseteq \bigcup_{L \in \mathscr{L}_n} L$ then $\mathscr{L}_n$ is a cover of S_0 against S_1.

Proof. By hypothesis, $\bigcup_{L \in \mathscr{L}} L \subseteq S_0$ and the order of $L \in \mathscr{L}_n$ is less than or equal to n. Since $L \in \mathscr{L}_n$ implies $L \cap S_1 = \phi$, we must have $L \subseteq S_1^c$. But this is true for all $L \in \mathscr{L}_n$. Hence $\bigcup_{L \in \mathscr{L}_n} L \subseteq S_1^c$. Therefore, $\mathscr{L}_n$ is a cover of S_0 against S_1.

The requirement that $L \cap S_0 \neq \phi$ is our requirement that generalization can be made only through sets containing some observed patterns. The requirement that $L \cap S_1 = \phi$ is our requirement that sets in $\mathscr{L}'$ must be generalizing sets so that all observed patterns in an $L \in \mathscr{L}'$ have the same category identification. For the example problem, an order 1 cover of S_0 against S_1 is determined by $**R*$ and $y***$, exactly the pair of cylinder sets associated with category 0 by the relation Q.

The same result of associating $**R*$ and $y***$ with category 0 could be obtained using a generalized version of the iterative Quine–McCluskey technique. However, for large pattern sets of high dimensional order, that grouping technique is likely to involve more memory and computation time than beginning with order 1 cylinder sets and then going successively to order k cylinder sets where k is the smallest constant which makes the generalizing subsets cover the pattern set.

4. STRING PATTERN DISCRIMINATION USING GRAMMATICAL INFERENCE

Determining decision rules when patterns are strings involves using more algebraic structure than for the case when patterns are N-tuples. In this section we put the syntactic grammatical inference methods for regular grammars into the structural perspective established in Section 2. We will show first how an incompletely specified finite state acceptor can be constructed from the training data relation, and second how the collection $\mathscr{L}$ can consist of the equivalence classes of the input monoids of any acceptors, which by a specified type of homomorphism is a homomorphic image of the acceptor constructed from the training data relation. Then we illustrate this method by a simple syntactic pattern recognition example. The ideas in this section are motivated by

Fu,[9] Fu and Booth,[10] Biermann and Feldman,[1] and Pao.[23]

We begin with some notation and definitions. Let Σ be the set of characters which can make up pattern strings. Let Σ^* be the set of all strings of characters from Σ. The set of patterns P is Σ^*, an infinite set even if Σ is finite. As is well known, Σ^* is the free monoid generated by Σ under the string concatenation operation. Let $\Sigma^{*\prime} \subseteq \Sigma^*$ be the set of observed patterns and $C = \{1, 2, \ldots, K\}$ be the set of K recognition categories. We assume all patterns in $\Sigma^{*\prime}$ are of finite length. The training data relation T is a subset of $\Sigma^{*\prime} \times C$; $T \subseteq \Sigma^{*\prime} \times C$.

We desire to define $\mathscr{L}$ a collection of subsets of Σ^*. However, the infiniteness of Σ^* makes it difficult to define such a collection directly. Our technique will be to define the collection $\mathscr{L}$ indirectly via finite state acceptors for the observed strings in $\Sigma^{*\prime}$. A K-category acceptor is essentially an automaton which starting from a designated starting state will transit by a string $\sigma \in \Sigma^*$ to a terminal state associated with the category of σ.

Definition 9

A *finite state acceptor* $\mathscr{A}$ for K categories is a $(K + 4)$-tuple $\mathscr{A} = (S, \Sigma, \delta, s_0, A_1, \ldots, A_K)$, where S is a finite set of states, $\delta \subseteq (S \times \Sigma) \times S$ is the transition relation, $s_0 \in S$ is the starting state, and $A_k \subseteq S$, $k = 1, \ldots, K$ are mutually exclusive subsets of terminal states associated with categories $1, \ldots, K$, respectively. Hence, $i \neq j$ implies $A_i \cap A_j = \phi$. When δ is defined everywhere on $S \times \Sigma$, $\mathscr{A}$ is said to be *completely specified*. Otherwise $\mathscr{A}$ is *incompletely specified*. When δ is single-valued, $\mathscr{A}$ is said to be *deterministic*. Otherwise $\mathscr{A}$ is said to be *nondeterministic*. If there is a category k such that a string $\sigma \in \Sigma^*$ can reach a state in A_k by the transition relation δ, then σ is said to be *accepted* by category k. Otherwise it is *rejected*.

A finite state acceptor can be constructed immediately from the training data relation T. If s_0 is the starting state and $(\sigma_1 \sigma_2 \ldots \sigma_N, k) \in T$ is the first string category pair in T, then we define the transition relation δ to include (s_{i-1}, σ_i, s_i), $i = 1, \ldots, N$ and we make A_k include s_N. Then we take the next string category pair in T and do the same thing beginning with s_0 and following with the next not yet used state in place of s_1. The resulting K-category finite state acceptor will be incompletely specified and nondeterministic in general. The acceptor can easily be made deterministic by iteratively merging pairs of states r and t if $(s, \sigma, r) \in \delta$ and $(s, \sigma, t) \in \delta$. The grammar associated with this finite state acceptor is known as the canocial definite finite state grammar.[10]

The acceptor constructed as above will correctly associate each observed pattern string in $\Sigma^{*\prime}$ with its correct category identification. However, strings not in $\Sigma^{*\prime}$ will be rejected. Grammatical inference is concerned with generalizing the constructed finite state acceptor so that it can accept many more strings than those initially observed. The basis of grammatical inference for regular grammars is the merging or combining of states in the initial K-category finite state acceptor,[23] or non-terminals in the production rules of the grammar.[8] These are two essentially equivalent processes, and we will concentrate on the state merging technique. The rule for merging states is that two states can be combined if under some input, their successors can be combined and the resulting acceptor is deterministic and acceptor states associated with different categories never get combined.

We can view the combining of states of S as being done by a function $h: S \to W$. If for some $w \in W$ there exists s, $s' \in S$ satisfying $h(s) = w = h(s')$. Then we say states s and s' are combined by h. Of course, not all pairs of states can be combined if the resulting acceptor is to be deterministic. The resulting acceptor will be deterministic if, and only if, the combining function is transition preserving. A function is transition preserving if, and only if, when it combines two states which have successors under a string $\sigma \in \Sigma^*$, then it must also combine the successors.

Definition 10

Let $\delta \subseteq (S \times \Sigma) \times S$ and $h: S \to W$. h is called a transition preserving function of δ if and only if $(s_1, \sigma, s_2) \in \delta$, $(s_1', \sigma, s_2') \in \delta$, $w_1 = h(s_1)$, $w_2 = h(s_2)$, and $w_1 = h(s_1')$ imply $w_2 = h(s_2')$.

The following proposition states that the resulting acceptor is deterministic if, and only if, the combining function is transition preserving.

Proposition 5. Let $\delta \subseteq (S \times \Sigma) \times S$ and $h: S \to W$. Define $\gamma \subseteq (W \times \Sigma) \times W$ by $\gamma = \{(w_1, \sigma, w_2) \in W \times \Sigma \times W \mid$ there exists $s_1, s_2 \in S$ such that $(s_1, \sigma, s_2) \in \delta$ and $w_1 = h(s_1)$ and $w_2 = h(s_2)\}$.

Then h is a transition preserving function of δ if, and only if, γ is single-valued.

Proof. Suppose h is transition preserving and $(w_1, \sigma, w_2) \in \gamma$ and $(w_1, \sigma, w_2') \in \gamma$. Then there exists s_1, $s_2 \in S$ such that $(s_1, \sigma, s_2) \in \delta$, $w_1 = h(s_1)$, $w_2 = h(s_2)$ and there exists s_1', $s_2' \in S$ such that $(s_1', \sigma, s_1') \in \delta$, $w_1 = h(s_1')$ and $w_2' = h(s_2')$. Since h is transition preserving, $(s_1, \sigma, s_2) \in \delta$, $(s_1', \sigma, s_2') \in \delta$, $w_1 = h(s_1) = h(s_1')$ and $w_2 = h(s_2)$ imply $w_2 = h(s_2')$. Hence, $w_2 = w_2'$ making γ single-valued.

Suppose γ is single-valued and $(s_1, \sigma, s_2) \in \delta$, $(s_1', \sigma, s_2') \in \delta$, $w_1 = h(s_1) = h(s_1')$ and $w_2 = h(s_2)$. Let $w_2 = h(s_2')$. Then by definition of γ $(w_1, \sigma, w_2) \in \gamma$ and $(w_1, \sigma, w_2') \in \gamma$. Since γ is single-valued, $w_2 = w_2'$. This makes h a transition preserving function of δ.

Functions which are transition preserving and do not combine terminal states associated with different categories not only ensure that the resulting acceptor will be single-valued, but also ensure that the entire transition structure is preserved. Such functions are called homomorphisms.

Definition 11

Let $\mathscr{A} = (S, \Sigma, \delta, s_0, A_1, \ldots, A_K)$ and $\mathscr{B} = (T, \Sigma, \gamma, t_0, B_1, \ldots, B_K)$ be two K-category finite state acceptors. A function $h: S \to T$ is a homomorphism from $\mathscr{A}$ into $\mathscr{B}$ if, and only if,

(1) $(s_1, \sigma, s_2) \in \delta$ and $h(s_2) = w_2$ imply there exists a $w_1 \in W$ such that $(w_1, \sigma, w_2) \in \gamma$ and $w_1 = h(s_1)$
(2) $t_0 = h(s_0)$
(3) $B_k = h(A_k)$, $k = 1, \ldots, K$.

Proposition 6. Let $\mathscr{A} = (S, \Sigma, \delta, s_0, A_1, \ldots, A_K)$ be a finite state acceptor for K categories. Let $h: S \to W$ be an onto transition preserving function which satisfies $i \neq j$ implies $h(A_1) \cap h(A_j) = \phi$. Define $\gamma \subseteq (W \times \Sigma) \times W$ by $\gamma = \{(w_1, \sigma, w_2) \in W \times \Sigma \times W |$ there exists $s_1, s_2 \in S$ such that $w_1 = h(s_1)$, $w_2 = h(s_2)$ and $(s_1, \sigma, s_2) \in \delta\}$. Then $\mathscr{B} = (W, \Sigma, \gamma, h(s_0), h(A_1), \ldots, h(A_K))$ is a deterministic finite state acceptor which is a homomorphic image of $\mathscr{A}$.

Proof. By proposition 5, γ is single-valued making β deterministic. Let $(s_1, \sigma, s_2) \in \delta$ and $h(s_2) = w_2$. Let $w_1 = h(s_1)$. Then $(s_1, \sigma, s_2) \in \delta$, $h(s_2) = w_2$, and $h(s_1) = w_1$ imply by definition of γ that $(w_1, \sigma, w_2) \in \gamma$. Finally, by hypothesis $i \neq j$ implies $h(A_i) \cap h(A_i) = \phi$. Thus β is a deterministic finite state acceptor.

To show β is a homomorphic image of ß, suppose that $(s_1, \sigma, s_2) \in \delta$ and $h(s_2) = w_2$. Let $w_1 = h(s_1)$. Then by definition of γ, $(w_1, \sigma, w_2) \in \gamma$. Therefore, h satisfies conditions (1), (2), and (3) of the definition and h is a homomorphism from $\mathscr{A}$ into $\mathscr{B}$. Since h is onto W, $\mathscr{B}$ is the homomorphic image of $\mathscr{A}$ under h.

The importance of combining states by a homomorphism is apparent when we consider that the strings accepted by any category of the homomorphic image acceptor must include those accepted by the corresponding category of the original acceptor. Thus to generalize the set of pattern strings accepted by any finite state acceptor, we need only find a homomorphism which combines states in a transition preserving manner. The following proposition proves than the set of pattern strings accepted by a homomorphic image acceptor is at least as large as the set of pattern strings accepted by the original acceptor.

Proposition 7. Let h be a finite state acceptor homomorphism from $\mathscr{A} = (S, \Sigma, \delta, s_0, A_1, \ldots, A_K)$ to $\mathscr{B} = (T, \Sigma, \gamma, t_0, B_1, \ldots, B_K)$. Suppose for some $\sigma \in \Sigma^*$, $(s_0, \sigma, s_f) \in \delta$ and for some $k \in \{1, \ldots, K\}$, $s_f \in A_k$. Then there exists $t_f \in T$ such that $(t_0, \sigma, t_f) \in \gamma$ and $t_f \in B_k$.

Proof. Let $t_f = h(s_f)$. Since $t_0 = h(t_0)$ and $(s_0, \sigma, s_f) \in \delta$ and h is a homomorphism from $\mathscr{A}$ into $\mathscr{B}$, $(t_0, \sigma, t_f) \in \gamma$. Since h a homomorphism implies $B_k = h(A_k)$ and $s_f \in A_k$, we must have $t_f = h(s_f) \in B_k$.

With this knowledge about finite state acceptors and their homomorphisms, we are ready to understand how the cover $\mathscr{L}$ of Σ^* can be constructed. It is well known that associated with each finite state acceptor is a partition of Σ^* having the property that two strings are in the same cell of the partition if their state transitions are identical (Proposition 8). It is also true that the cells of such a partition are a homomorphic image of Σ^* (Propositions 9 and 10). Thus, we can define $\mathscr{L}$ to contain all the cells from all partitions of Σ^* determined by finite state acceptors, which are defined everywhere homomorphic images of the finite state acceptors determined by the training data relation. We can, if we wish, restrict the number or kind of homomorphisms. $\mathscr{L}$ can consist of all the cells from all partitions of Σ^* determined by homomorphisms in a specified class of the homomorphisms on the original finite state acceptor. Because the finite state acceptor homomorphism was defined not to combine terminal states associated with different categories, the restriction $\mathscr{L}'$ of $\mathscr{L}$ is in this instance equal to $\mathscr{L}$.

Proposition 8. Let $\delta \subseteq (S \times \Sigma) \times S$. Define $R = \{(a, b) \in \Sigma^* \times \Sigma^* | (s, a, t) \in \delta$ if and only if $(s, b, t) \in \delta\}$. Then R is an equivalence relation on Σ^*.

Proof. R is reflexive, symmetric, and transitive.

Proposition 9. Let $\delta \subseteq (S \times \Sigma) \times S$. Define $R = \{(a, b) \in \Sigma^* \times \Sigma^* | (s, a, t) \in \delta$ if, and only if, $(s, b, t) \in \delta\}$. Let $[a]$ designate the equivalence class for the element a. Then the equivalence classes of R form a monoid under the binary operation defined by $[a] \cdot [b] = [ab]$.

Proof. First, the operation is well defined since if $a' \in [a]$ and $b' \in [b]$ we must have $a'b' \in [ab]$. To prove this, let $a' \in [a]$ and $b' \in [b]$. Then $(s, a, r) \in \delta$ if and only if $(s, b, r) \in \delta$ and $(r, a', t) \in \delta$ if and only if $(r, b', t) \in \delta$. Notice that $(s, ab, t) \in \delta$ implies there exists a $r \in S$ satisfying $(s, a, r) \in \delta$ and $(r, b, t) \in \delta$. Since $(s, a, r) \in \delta$ implies $(s, a', r) \in \delta$ and $(r, b, t) \in \delta$ implies $(r, b', t) \in \delta$, we have $(s, a', r) \in \delta$ and $(r, b', t) \in \delta$. But $(s, a', r) \in \delta$ and $(r, b', t) \in \delta$ implies $(s, a', b', t) \in \delta$. Thus (s, ab, t) implies $(s, a'b', t) \in \delta$. A similar argument shows the converse. Therefore, $(s, ab, t) \in \delta$ if and only if $(s, a'b', t) \in \delta$ so that $a'b' \in [ab]$.

Second, the operation is associative since

$$\begin{aligned}([a] \cdot [b]) \cdot [c] &= ([ab]) \cdot [c] = [(ab)c] \\ &= [a(bc)] = [a] \cdot ([bc]) \\ &= [a] \cdot ([b] \cdot [c]) .\end{aligned}$$

Finally, the identify is $[\lambda]$ since

$$[a] \cdot [\lambda] = [a\lambda] = [a]$$

and

$$[\lambda] \cdot [a] = [\lambda a] = [a] .$$

Proposition 10. Let $\delta \subseteq (S \times \Sigma) \times S$. Define $R = \{(a, b) \in \Sigma^* \times \Sigma^* | (s, a, t) \in \delta$ if, and only if, $(s, b, t) \in \delta\}$. Let $\mathscr{E}$ be the set of equivalence classes of R and $[\sigma]$ designates the equivalence class for σ. Then the function $h: \Sigma^* \to \mathscr{E}$ defined by $h(\sigma) = [\sigma]$ is a homomorphism from the monoid Σ^* into the monoid $\mathscr{E}$.

Proof. $h(ab) = [ab] = [a] \cdot [b] = h(a) \cdot h(b)$.

We next illustrate this structural pattern recognition technique by an example. We let $\Sigma = \{A, B, \lambda\}$, where λ is the null string, and we let the pattern space be Σ^*. The set of observed patterns $\Sigma^{*\prime}$ is given by

$$\Sigma^{*\prime} = \{A, AB, BA, BBA, AAA, ABA, ABAA, AA, B, \lambda, ABAB\}.$$

We suppose there are two recognition categories $C = \{0, 1\}$ and the data training relation $T \subseteq \Sigma^{*\prime} \times C$ is given in the table below.

T	
Pattern	Category
A	0
AB	0
BA	0
BBA	0
AAA	0
ABA	0
$ABAA$	0
AA	1
B	1
λ	1
$ABAB$	1

The state diagram for a deterministic finite state acceptor which can be derived directly from the training data relation T is shown in Fig. 5. We will construct the cover $\mathscr{L}$ of Σ^* by putting into $\mathscr{L}$ the equivalence classes of the input monoids for two out of the 17 defined everywhere finite state acceptors, which are homomorphic images of the original one.

The two homomorphisms and the homomorphic image acceptors are shown in Fig. 6. The equivalence classes of the input monoids for these two homomorphic image acceptors are listed in Fig. 7. The square bracket designates the equivalence class of the pattern string inside. The subscript on the bracket specifies whether the equivalence class is for homomorphic image acceptor 1 or 2. The notation of Fig. 7 also gives for each equivalence class a regular expression denoting the set of its pattern strings. To simplify the terms in the regular expression, equivalence class names whose regular expressions have been previously defined are used in place of those regular expressions when it is convenient to do so.

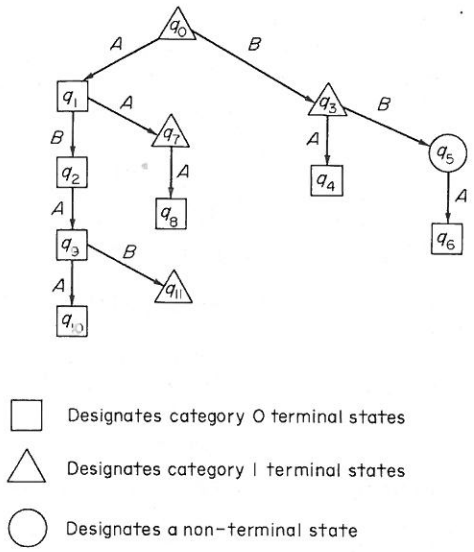

Fig. 5. A state diagram for the finite state acceptor determined by the data training relation.

With the exception of equivalence classes, $[BB]_1$, $[ABB]_1$, $[AAB]_1$, $[BAB]_2$, $[BABA]_2$, $[ABABA]_2$ and $[BBAB]_2$, all the equivalence classes have non-empty intersection with the observed pattern string set. Furthermore, each equivalence class having non-empty intersection with the observed pattern string set is a generalizing set because it contains observed

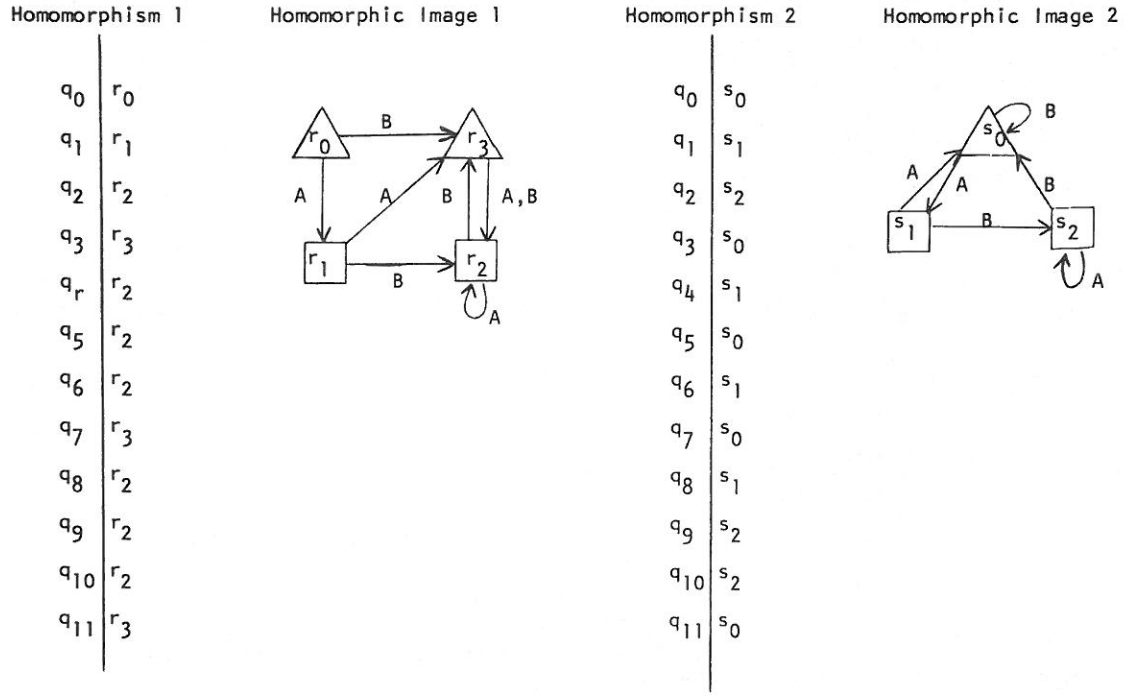

Fig. 6. Two homomorphisms and their homomorphic images of the finite state acceptor determined by the training data relations (see Fig. 5).

pattern strings from only one category. The relation $F' \subseteq \Sigma^{*\prime} \times \mathscr{L}'$, which pairs observed pattern strings with generalizing sets, is given by the table below.

F'	
Pattern	Covering set
A	$[A]_1, [A]_2$
AB	$[AB]_1, [AB]_2$
BA	$[BA]_1, [BA]_2$
BBA	$[BA]_1, [BBA]_2$
AAA	$[BA]_1, [A]_2$
ABA	$[BA]_1, [ABA]_2$
AAA	$[BA]_1, [AB]_2$
AA	$[AA]_1, [\lambda]_2$
B	$[B_1, [B]_2$
λ	$[\lambda]_1, [\lambda]_2$
$ABAB$	$[BAB]_1, [ABAB]_2$

The relation Q pairing generalizing sets and categories is defined by $Q = I_{\Sigma^{*\prime}} \circ (F', T)$. Q is given in the following table.

Q	
Covering set	Category
$[\lambda]_1$	1
$[A]_1$	0
$[B]_1$	1
$[AB]_1$	0
$[AA]_1$	1
$[BA]_1$	0
$[BAB]_1$	1
$[\lambda]_2$	1
$[A]_2$	0
$[B]_2$	1
$[AB]_2$	0
$[BA]_2$	0
$[ABA]_2$	0
$[ABAB]_2$	1
$[BBA]_2$	0

The decision rule D, which is a binary relation from the patterns to the categories, is defined by

$$[\lambda]_1 = \lambda$$

$$[A]_1 = A$$

$$[B]_1 = B(BB)^*$$

$$[BB]_1 = [B]_1[B]_1$$

$$[AB]_1 = A[B]_1$$

$$[ABB]_1 = AB[B]_1$$

$$[AA]_1 = AA(BB)^*$$

$$[AAB]_1 = [AA]_1B$$

$$[BA]_1 = ([AB]_1 + [B]_1 + [BB]_1 + [AA]_1)\ A(A^* + B(A + B))^*$$

$$[BAB]_1 = [BA]_1\ [B]_1$$

$$[\lambda]_2 = (AA)^*$$

$$[A]_2 = A(AA)^*$$

$$[B]_2 = [\lambda]_2\ B(([A]_2\ B[A]_2\ B[\lambda]_2) + [\lambda]_2)^*$$

$$[AB]_2 = [A]_2\ [B]_2$$

$$[BA]_2 = [B]_2\ [A]_2$$

$$[BAB]_2 = [BA]_2\ [B]_2$$

$$[BABA]_2 = [BAB]_2\ [A]_2$$

$$[ABA]_2 = [AB]_2\ [A]_2$$

$$[ABAB]_2 = [ABA]_2\ [B]_2$$

$$[ABABA]_2 = [ABAB]_2\ [A]$$

$$[BB]_2 = [A]_2[B]_2\ ([B]_2 + [A]_2[B]_2[B]_2)\ ((B + (AA)^*)^*[A]_2[B]_2A^*[B]_2)^* + [B]_2\ ([B]_2 + [A]_2[B]_2[B]_2)$$

$$[BBA]_2 = [BB]_2\ [A]_2$$

$$[BBAB]_2 = [BBA]_2\ B\ A^*$$

Fig. 7. The equivalence classes and their corresponding regular expressions of the input monoid for the two finite state acceptors of Fig. 6.

$D = F' \cdot (I_{\Sigma^*}, Q)$. We therefore can write D as a set of all string category pairs satisfying certain conditions.

$$D = \{(\sigma, c) \in \Sigma^* \times C |$$
$$\text{if } c = 0, \text{ then } \sigma \in [A]_1, [AB]_1, [BA]_1, [A]_2, [AB]_2, [BA]_2, [ABA]_2, [BBA]_2$$
$$\text{if } c = 1, \text{ then } \sigma \in [\lambda]_1, [B]_1, [AA]_1, [BAB]_1, [\lambda]_2, [B]_2, [ABAB]_2\}.$$

Because not all the equivalence classes are guaranteed to have observed patterns, the equivalence classes missing in the specification of D can cause some pattern strings in Σ^* to be paired with no category. Since the equivalence classes determined from different homomorphisms can overlap, it is also possible for D to pair some pattern strings with more than one category.

The relationship of the construction of the decision rule to the grammatical inference techniques of Pao[23] and Feldman[8] is straightforward. These techniques begin with either the finite state acceptor determined by the training data relation, or a grammar which produces only the observed pattern strings. Grammatical inference takes place by merging the states of the finite state acceptors as we have illustrated or by merging non-terminals of the grammar. Because of the natural isomorphism between finite state acceptors and regular grammars, these two merging techniques amount to the same basic process even though the merging of non-terminals can lead to non-deterministic finite state acceptors.

5. ARRANGEMENT PATTERN DISCRIMINATION

In this section we introduce the arrangement[14] as a pattern data structure and illustrate how it can be used just like the string and N-tuple in a structural pattern recognition approach.

5.1 *The arrangement*

Let A be the set of elements whose arrangement is being described. Each group of related elements from A is given a label from the label set L. Let R be the labeled N-ary relation which consists of labeled N-tuples of related elements from A.

Definition 12

A *simple order-N arrangement* is a triple (R, A, L) where $R \subseteq A_N \times L$. We can use R to specify the arrangement (R, A, L) when the sets A and L are understood.

An N-tuple is a special case of an order 1 arrangement. To see this, suppose $(x_1, \ldots, x_N)$ is the N-tuple. Take L to be the label set having integers for labels. Define the set $A = \{x_1, x_2, \ldots, x_N\}$ and the relation $R \subseteq A \times L$ by $R = \{(\alpha, \ell) \in A \times L | x_\ell = \alpha\}$. A string is really just an N-tuple with variable length so it, too, is a special case of an order-1 arrangement.

We define a general arrangement as a coordinated set of arrangements having the same label set. This concept allows relationships of different orders on the same set to be handled together.

Definition 13

A *general arrangement* is a set of simple arrangements, each simple arrangement being of different order, being defined on the same set, and having the same label set. If there are K simple arrangments in the arrangement A, then $A = \{R_1, R_2, \ldots, R_K;\ A, L\}$ where $R_k \subseteq A^{N_k} \times L$, $k = 1, \ldots, K$.

Since arrangements are going to serve as our patterns and since we have illustrated that structural pattern recognition can be viewed as generalizing through homomorphic images of the observed training patterns, we will need to define a composition operation and a homomorphism. For this purpose we will use the same general idea we used in Section 2. The composition of an arrangement by a binary relation will produce an arrangement which contains all the labeled N-tuples of the given arrangement translated component by component through the binary relation. The only difference between this composition and that introduced in Section 2 is that in Section 2 the translation could take place using a different binary relation for each of the components of the ordered pairs in the given relation. Here we allow the given relation to be N-ary and insist that all components get translated the same way through the binary relation during the composition.

Definition 14

Let $A = \{R_1, \ldots, R_K; A, L\}$ be an arrangement and $H \subseteq A \times B$. The *composition* of arrangement A with H results in an arrangement B which we define as

$$A \cdot H = B = \{S_1, S_2, \ldots, S_K; B, L\},$$

where

$$S_k = \{(b_1, b_2, \ldots, b_{N_k}, \ell) | (a_1, \ldots, a_{N_k}, \ell) \in R_k \text{ and } (a_n, b_n) \in H,\ n = 1, \ldots, N_k\}.$$

The concept of homomorphism introduced in Section 2 was that the image of the composition had to be contained in a given relation. We employ exactly the same concept here put into the arrangement perspective.

Definition 15

An arrangement $A = \{R_1, \ldots, R_K;\ A, L\}$ is *contained* in an arrangement $D = \{T_1, \ldots, T_K;\ A, L\}$ if and only if $R_k \subseteq T_k$, $k = 1, \ldots, K$. In this case we write $A \subseteq D$.

Definition 16

Two arrangements $A = \{R_1, \ldots, R_K;\ A, L\}$ and $B = \{S_1, \ldots, S_N;\ B, M\}$ are *comparable* if the number of relations in each arrangement is the same ($K = N$), if the label sets are the same ($M = L$), and if the relation R_k has the same order as the relation S_k:

$$(R_k \subseteq A^{N_k} \times L \text{ and } S_k \subseteq B^{N_k} \times L).$$

Combining Function F	Image of Arrangement T_0 Under Function F	Image of Arrangement T_1 Under Function F
F_1: S → E	$A_1 \subseteq E \times E \times K$	$A_5 \subseteq E \times E \times K$
a,b → v	vv0	vv0
c,d → w	wv0	wx0
e → x	ww1	yv0
f → y	ww1	vw1
		ww1
		xw1
		yw1
F_2: S → F	$A_2 \subseteq F \times F \times K$	$A_6 \subseteq F \times F \times K$
a,c,e → v	vv0	vv0
b,d,f → w	wv0	wv0
	vw1	vw1
	ww1	ww1
F_3: S → H	$A_3 \subseteq H \times H \times K$	$A_7 \subseteq H \times H \times K$
a,b → p	pp0	pp0
c,d → q	qp0	qr0
e,f → r	pq1	rp0
	qq1	pq1
		qq1
		rq1
F_4: S → G	$A_4 \subseteq G \times G \times K$	$A_8 \subseteq G \times G \times K$
a,d,f → v	vv0	vv0
b,c,e → w	wv0	wv0
	vv1	ww0
	wv1	vw0
		vv1
		wv1

Fig. 8. Illustration of how functions F_1, F_2, F_3, and F_4 combine elements of S together and thereby form homomorphic images A_1–A_8 of the observed training arrangement.

Definition 17

Let $A = \{R_1, \ldots, R_K; A, L\}$ and $B = \{S_1, \ldots, S_K; B, L\}$ be two comparable arrangements. Let $H: A \to B$. The function H is a homomorphism from arrangement A to arrangement B if, and only if, $A \cdot H \subseteq B$.

We will discuss structural pattern recognition using the simple arrangement for the pattern data structure rather than the general arrangement. We can take the pattern set P to be the set of all arrangements and the collection $\mathscr{L}$ to have members which are sets of arrangements, each of which has the property that it is the preimage of a homomorphism to a feature arrangement associated with the set. The feature arrangements are homomorphic images of the observed arrangement patterns. The restriction of $\mathscr{L}$ to $\mathscr{L}'$, the relation Q and the decision rule D can all be determined as in Section 2.

For our example, we take the category set C to be $\{A, B\}$, the pattern set to be all second order arrangements on a set $S = \{a, b, c, d, e, f\}$, with labels $K = \{0, 1\}$, the observed arrangement patterns to be $P' = \{T_0, T_1\}$, and the training data relation $T \subseteq P' \times C$.

The definition for the arrangements T_0 and T_1 and the training relation T are given below.

$T_0 \subseteq S \times S \times L$	$T_1 \subseteq S \times S \times L$
*aa*0	*aa*0
*ba*0	*ba*0
*ca*0	*ce*0
*da*0	*de*0
*ad*1	*ea*0
*bd*1	*fa*0
*cd*1	*ad*1
*dd*1	*bd*1
	*cd*1
	*dd*1
	*ed*1
	*fd*1

T	
Arrangement patterns	Category
T_0	A
T_1	B

Figure 8 illustrates how four functions which combine elements of S together can form homomorphic images of the observed training arrangements. We name the homomorphic image arrangements $A_1, A_2, \ldots, A_8$. Notice that arrangements A_1, A_2, A_3, and A_6 are either identical or isomorphic.

We define the collection $\mathscr{L}$ to consist of sets, each of which contains all arrangements having one of the feature arrangements for its homomorphic image.

$$\mathscr{L} = \{L \subseteq P \mid \text{for every arrangement } R \in L, \text{ there exists some function } f \text{ and feature arrangement } B \text{ satisfying } R \cdot f = B\}.$$

On the basis of the four combining functions chosen and the 8 feature arrangements they determine, the collection $\mathscr{L}$ could contain as many as 8 sets. However, since 4 of the feature arrangements are isomorphic, $\mathscr{L}$ contains only 5 sets of arrangements L_1–L_5 defined as follows:

$$L_1 = \{R \in P \mid \text{there exists some function } f \text{ satisfying } R \cdot f = A_1\}$$

$$L_2 = \{R \in P \mid \text{there exists some function } f \text{ satisfying } R \cdot f = A_5\}$$

$$L_3 = \{R \in P \mid \text{there exists some function } f \text{ satisfying } R \cdot f = A_5\}$$

$$L_4 = \{R \in P \mid \text{there exists some function } f \text{ satisfying } R \cdot f = A_7\}$$

$$L_5 = \{R \in P \mid \text{there exists some function } f \text{ satisfying } R \cdot f = A_8\}.$$

Because L_1 contains observed training arrangements for more than one category, the restriction $\mathscr{L}'$ of $\mathscr{L}$ consists of only L_2–L_5 for its generalizing sets. The relation F' which pairs arrangement patterns with generalizing sets is defined by

$$F' = \{(p, L) \in P \times \mathscr{L}' \mid p \in L\}.$$

The relation Q, which pairs generalizing sets with categories, is defined by

$$Q = I_{P'} \cdot (F', T) = \{(L_2, A), (L_3, B), (L_4, B), (L_5, B)\}.$$

The decision rule D, which pairs patterns to categories, is defined by $D = F' \cdot (I_{P'}, Q)$ and can be written as

$$D = \{(p, c) \in P \times C \mid \text{if } c = A, \text{ then } p \in L_2 \text{ if } c = B, \text{ then } p \in L_3 \cup L_4 \cup L_5\}.$$

6. CONCLUSIONS

In this paper we have described structural pattern recognition using a relational algebra. We have suggested that the generalization of the observed training patterns takes place by the implicit or explicit use of a restricted cover on the pattern space. The cover is restricted to insure that only one category will be asociated with each set in the cover. The form of the cover depends on the data structure of the patterns: cylinder sets for N-tuples, and equivalence classes of input monoids for strings. The cover may be given prior to knowing the pattern training relation or after knowing the pattern training relation.

In the relational setting in which we have described this process, the natural relation F' pairing patterns to sets in the restricted cover to which they belong is an inverse homomorphic image of the pattern training relation. The decision rule D, which pairs patterns to categories, is a homomorphic image of the relation F'. Not only do homomorphisms enable us to produce D given the pattern training relation and a cover of the pattern space, but as suggested for the N-tuple and string patterns, the cover itself can be defined in terms of homomorphisms on the pattern space. Thus homomorphisms play an essential role in the construction of structural pattern recognition decision rules.

Finally, we have illustrated that once structural pattern recognition is viewed from this perspective, other kinds of relational pattern data structures can be invented and naturally used. To illustrate this, we defined an arrangement as a labeled N-ary relation and showed how arrangements can be used as a pattern data structure. In this situation, the pattern space is the set of all labeled N-ary relations. The sets in the cover are defined by selected homomorphic images of the training pattern arrangements. The decision rule is then a homomorphic image of the relation which pairs patterns to sets in the restricted cover.

Acknowledgement – I would like to thank Linda G. Shapiro for the help and ideas she shared with me while I was preparing the paper and Lynn Ertebati for typing the manuscript.

REFERENCES

1. A. W. Biermann and J. A. Feldman, On the synthesis of finite state machines from samples of their behavior, *IEEE Trans. Comput.* **21**, 592–597 (1972).
2. W. S. Brainerd, Tree generating regular systems, *Inf. Control* **14**, 217–231 (1969).
3. M. F. Dacey, The syntax of a triangle and some other figures, *Pattern Recognition* **2**, 11–31 (1970).
4. R. Davis, B. Buchanan and E. Shortliffe, Production rules as a representation for a knowledge-based consultation program, *Art. Intelligence* **8**, 15–45 (1977).
5. J. E. Donor, Tree acceptors and some of their applications, *J. Comput. Syst. Sci.* **4**, 406–451 (1970).
6. J. Feder, Plex languages, *Inf. Sci.* **3**, 225–241 (1971).
7. J. Feder, Languages of encoded line patterns, *Inf. Control* **13**, 230–244 (1968).
8. J. A. Feldman, First thoughts on grammatical inference, Stanford Artificial Intelligence Project Memo 55, Stanford University, Stanford, California (1967).
9. K. S. Fu, *Syntactic Methods in Pattern Recognition.* Academic Press, New York, 295 pp (1974).
10. K. S. Fu and T. L. Booth, Grammatical inference: introduction and survey – I, *IEEE Trans Syst., Man, Cybern.* **5**, 95–111, January (1975).
11. K. Fukanaga, *Introduction to Statistical Pattern Recognition.* Academic Press, New York (1972).
12. S. Ginsburg, *The Mathematical Theory of Context-Free Language.* McGraw-Hill, New York (1966).
13. R. M. Haralick, The pattern recognition problem from the perspective of relation theory, *Pattern Recognition* **7**, 67–79 (1975).

14. R. M. Haralick, Structural pattern recognition, arrangements and theory of covers, *IEEE Conf. on Pattern Recognition and Image Processing*, Troy, New York, June (1977).
15. R. A. Kirsch, Computer interpretation of English text and patterns, *IEEE Trans. electron. Comput.* **13**, 362–376 (1964).
16. E. J. McCluskey, Jr., Minimization of boolean functions, *Bell System Tech. J.* **35**, 1417–1444, November (1956).
17. R. S. Michalski, On the quasi-minimal solution of the general covering problem, *Proc. 5th International Symposium*, Yugoslavia, Bled, 8–11 October, 1969.
18. R. S. Michalski and B. H. McCormick, Interval generalization of switching theory, *Proc. 3rd Ann. Houston Conf. on Computer and System Science*, Houston, Texas, pp. 231–226 (1971).
19. R. S. Michalski, AQVAL/1 – Computer implemental of a variable-valued logic system VL_1 and example of its application to pattern recognition, *Int. Jnt Conf. on Pattern Recognition*, Washington, D.C., pp. 3–17 (1973).
20. R. S. Michalski, Variable-valued logic: System VL_1, *Proc.* 1974 *Int. Symp. on Multiple-Valued Logic*, West Virginia State University, Morgantown, West Virginia, pp. 323–346 (1974).
21. D. M. Milgram and A. Rosenfeld, Array automata and array grammars, *IFIP Congr.* **71**, Booklet TA-2. North-Holland, Amsterdam, pp. 166–173 (1971).
22. R. N. Narasimhan, Syntax-directed interpretation of classes of pictures, *Comm. ACM* **9**, 166–173 (1966).
23. T. W. Pao, A solution to the syntactical induction-inference problem for a non-trivial subset of context-free languages, Interim Technical Report 69–19, Moore School of Engineering, University of Pennsylvania, Philadelphia, Pennsylvania (1969).
24. J. L. Pfaltz and A. Rosenfeld, Web grammars, *Proc. 1st Int. Jnt Conf. on Artificial Intelligence*, Washington, D.C., May pp. 609–619 (1969).
25. W. V. Quine, A way to simplify truth function, *Am. Math. Monthly* **62**, 627–631, November (1955).
26. A. C. Shaw, A formal picture description scheme as a basis for a picture processing system, *Inf. Control* **14**, 9–52 (1969).
27. E. H. Shortliffe, *Computer-Based Medical Consultation: MYCIN*. Elsevier, New York, 264 pp. (1976).