SCENE MATCHING PROBLEMS

Robert M. Haralick
University of Kansas
Lawrence, Kansas 66045

May 1978

I. INTRODUCTION

One of the important problems in scene analysis is the comparing of two scenes or a scene with a prototype scene. The purpose of the comparison is to answer questions about the similarity of the scenes. The problem is a structural one. In this paper we give some examples of the problem, translate the problem into its general structural form, and give an algorithm for solving it.

We show that the mathematical form of the problem is one of finding homomorphisms from one relation to another. We show that the relation homomorphism problem is a consistent labeling problem. Finally, we illustrate how a tree search with a look-ahead discrete relaxation operator can solve the consistent labeling problem and, therefore, the relation homomorphism problem.

II. IMAGE MATCHING

By image matching we mean how we can tell two images are of the same kind of thing. For this to happen, all the parts of one image must have similar parts in the other and the relationships between parts in one must be the same as the relationship between the associated parts in the other.

We will illustrate how this problem can be posed as a relation homomorphism problem. Suppose we have a segmented image, and we are able to characterize each segment in terms of certain basic attributes, for example, shape discriminators. Using these attributes, we could assign a shape value to each of the segments. To define a N-ary relation from these values, we can group related segments together, N at a time, and form the corresponding set of N-tuples of their values. Then we append to the N-tuple an (N+1)-component which is the name or label of the group of related segments whose shape values are the first N components. One possible kind of label on the (N+1)st component is a counter index. We can assign the integer label "1" to all N-tuples arising from a group of segments the first time the N-tuple is encountered. The label "k" can be assigned the kth time the same kind of N-tuple is encountered.

One criterion by which segments can be considered related is spatial connectivity or nearness. Two segments are eligible to be included in the same related group when their interaction lengths overlap (when they are close enough). To make things simple in our examples, we will use interaction lengths of zero. Thus, two segments will be related only when they are touching.

As a specific example, one might consider a missile launching complex as described in terms of its constitutent image phonemes. These might include railroad spurs, roads, power lines, buildings, radar antennas, support vehicles, etc. In terms of the stylized examples which we will present for purposes of simplicity and generality, such specific components are represented by circles, squares, triangles, etc.; however, it should be kept in mind that these "geometrical objects" are generic prototypes and always represent actual image components, shapes, attributes, subattributes, etc.

The example in Figure 1 illustrates an image which has four basic kinds of figures: squares, triangles, circles, and arrows. A quadruple whose first three components are these shapes taken in the order square, triangle, circle, and arrow will be considered to belong to the relation defined by the image if all three shapes touch each other in a pairwise manner. In general, we may use the criterion: consider any N-tuple if enough of its components interact in a pairwise or K-wise manner. The fourth component of the quadruple is a label which just counts the number of distinct times that the shapes its first three components represent occur in a related manner in the image.

In Figure 1, there are four drawings. Each drawing has two triangles, one circle, one square, and one arrow. Using the relation idea, there are two pairs of drawings whose arrangements are isomorphic by the identity function. The drawings themselves, however, have their parts placed differently in absolute position and orientation. This isomorphism becomes clear upon examination of Figure 2 which shows the possible arrangements for the drawings. The drawings on the left each define the relation labeled A. Hence, they are isomorphic. The drawings on the right each define the relation labeled B. Hence, they are isomorphic.

Matching can mean matching in the sense of isomorphism or can mean matching in the looser sense of similarity. In the next section, we define the general concept of similarity in terms of relation homomorphisms and in the following sections we continue the image matching example using the relation homomorphism concept.

III. THE RELATIONAL HOMOMORPHISM PROBLEM

For an N-ary relation $R \subseteq A^N$ and a function $f:A \to B$ from set A to set B, we define the composition of R with f, $R \circ f$, as the relation $R' \subseteq B^N$ where $R' = \{(b_1,\ldots,b_N) \in B^N \mid$ there exists $(a_1,\ldots,a_N)$ in R with $f(a_i) = b_i$, $i = 1,\ldots,N\}$. Let $T \subseteq A_N$ and $S \subseteq B^N$ be two N-ary relations. A function $f:A \to B$ which satisfies $T \circ f \subseteq S$ is called a relational homomorphism. Given two arbitrary N-ary relations the relational homomorphism problem is the problem of determining all relational homomorphisms between them. The image matching problem and the scene labeling problem are examples of relational homomorphism problems. Automata homomorphisms, graph homomorphisms, and graph colorings are other examples of relational homomorphism problems.

Let $A = \{w,v,0,1\}$ and R be a ternary relation defined on the set A; $R \subseteq A \times A \times A$. Suppose R is given by:

vv0
wv0
vwl
wwl

Let B = {a,b,c,d,0,1} and h:A → B be defined by:

h	
w	a
v	d
0	1
1	0

Then R∘h has the triples:

ddl
adl
da0
aa0

Let S be a ternary relation defined on the set B; S ⊆ B x B x B. Suppose S is given by the triplets:

aa0
ba0
ca0
da0
adl
bdl
cdl
ddl

Since R∘h ⊆ S, h is a homomorphism from R into S. Since h is a one-one function, S contains a copy of R. The function h is called a homomorphism.

Let f:A → B be defined by:

f	
a	v
b	v
c	w
d	w
0	0
1	1

Then S∘f has the triplets:

vv0
wv0
vwl
wwl

Notice that S∘f ⊆ R, making f a homomorphism of S into R. Since it is the case that S∘f = R, R is a homomorphic image of S and f is said to be onto R.

Let T be the ternary relation defined on B consisting of the following triplets:

aal
bal
ab0
bb0

Let g:A → B be defined by:

	g
w	b
v	a
0	1
1	0

Then R∘g = T. This makes g a one-one onto homomorphism. One-one onto homomorphisms are called isomorphisms. Two relations which are isomorphic are exactly the same except for the name of the symbols used. To convert one relation to the other, we need just translate all symbols in the first relation through the isomorphism and we will obtain the second relation.

IV. HOMOMORPHISMS FOR IMAGE MATCHING

In this section, we wish to continue the image matching problem first with isomorphic images and then with homomorphic images. Our isomorphism example will be more complicated here than in Section II, where it was the simple identity function.

Our example is illustrated in Figure 3 which has four drawings. Each drawing has two squares, one circle, one hexagon, and one triangle. Taking the component order as square, hexagon, triangle, and circle and using the relation concept, there are two pairs of drawings in Figure 3 whose corresponding relations are isomorphic. Also the relation for each drawing in Figure 3 is isomorphic to the relation for one of the drawings in Figure 1. The isomorphism, however, is not the identity function: a square stays square, a hexagon becomes a triangle, a triangle becomes an arrow, and a circle remains a circle.

More complicated still is the case where the correspondence between one drawing and another is by a homomorphism which does not establish a one-one correspondence. Such a case is illustrated in Figure 4 which depicts two drawings. Taking the component order as hexagon, circle, triangle, arrow, and square and using the name or label 1 for all triplets except the triplet (arrow, triangle, square) which gets the label 2, we may use the relation concept to establish the correspondence between one of the drawings (the one on the right) in Figure 4 and two of the drawings in Figure 1 (the ones on the left). The correspondence is a homomorphism and finding it, although easy, should begin to give the reader some idea of the combinatorial problems involved. The drawings on the left of Figure 4 is homomorphic to neither of the drawings in Figure 1.

The problem of finding homomorphisms is truly one of establishing the correspondence using relationships. Figure 5 shows the quadruples in the relation for the right-hand drawing of Figure 4 and the relation for a left-hand drawing of Figure 1. The homomorphism between the relations appears in the central bottom part of Figure 5.

V. HOMOMORPHISMS FOR SCENE LABELING

Suppose a scene has been divided into segments $S = \{s_1,\ldots,s_K\}$. A low level feature extractor with decision rule using gray tone, color, shape, and texture of each segment assigns some possible description from a set of D descriptions to each segment. This operation defines a segment-description relation $F \subseteq S \times D$.

The problem with this low-level assignment is that each segment may be associated with multiple descriptions. The desired labeling of the scene would have each segment described unambiguously. We would, therefore, like to use some a priori information to reduce the ambiguity.

A similar situation arises in the line labeling problem of Waltz[14]. Here, S is the set of line segments found in a scene and D is a set containing labels that can be associated with any line. The labels in D could be, for example, convex, concave, occluding left, occluding right (see Figure 6). The segment-description relation F, determined from low level processes, associates with each line in S one or more labels from D. The desired line labeling would be some subset of F that associates each line with only one label.

One way of reducing the possibly ambiguous description a line or segment initially has is to use constraints from a higher level world model. Such a model can specify labeling constraints for each group of related segments or lines. To employ such a model, related (ordered) sets of N segments or lines must be determined. Segments can be related on the basis of their relative spatial positions. Lines can be related on the basis of the junctions they form. Then for each kind of relationship the model can specify a constraint which the labels of each kind of related segments or lines must satisfy.

For instance, pairs of segments in S could be related if they mutually touch each other. There could be different kinds of touching such as to the left, to the right, above, below, in front of, in back of, supported by, and contained in. Suppose L is the set of such relationship labels. Then the set of spatially related segments or lines could be specified by the relation $A \subseteq S \times S \times L$, where $(s,t,i) \in A$ if and only if label i describes the way segment s relates to segment t. In the general case, the relationships in L can describe the way N segments or lines are related so that the relation A is a labeled N-ary relation: $A \subseteq S^N \times L$.

The world model contains constraining information. For example, pairs of segments whose relationship label is i can be constrained by the world model to have associated with them only certain allowable description pairs. In this case the world model is specified as a relation $C \subseteq D \times D \times L$, where $(d_1,d_2,i) \in C$ if and only if it is legal for a pair of segments s_1 and s_2 having relation i to have respective descriptions d_1 and d_2. In general, the relation C is a labeled N-ary relation, $C \subseteq D^N \times L$ which includes in it all labeled N-tuples of compatible descriptions for an ordered set of N related segments.

To summarize the information we have available:

(1) $F \subseteq S \times D$, the assignments of descriptions given by a low level operation;

(2) $A \subseteq S^N \times L$, the labeled sets of related N-tuples of segments;

(3) $C \subseteq D^N \times L$, the N-ary relational labeling constraints specified by the world model.

The scene labeling problem is to use F, A, and C to determine a new labeling relation G which contains fewer ambiguous descriptions than F and which is consistent with the constraints specified by the world model.

To express this problem in terms of relation homomorphisms, extend the relation F to the relation F', $F' \subseteq (S \frown L) \times (D \frown L)$, by:

$$F' = \{(s,d) \mid (s,d) \in F \text{ or } s = d \in L\}$$

F' has the same mapping that F does plus the addition of the identity mapping on L. To keep F' essentially the same as F, we will assume that S, L, and D are mutually exclusive.

Consider the relation $A \subseteq S^N \times L$ as an (N+1)-ary relation on $(S \bigcup L)$ since it is certainly true that $A \subseteq (S \bigcup L)^{N+1}$. Likewise, consider the relation $C \subseteq D^N \times L$ as an (N+1)-ary relation on $(D \bigcup L)$; $C \subseteq (D \bigcup L)^{N+1}$.

Our problem is to find all functions $G:(S \bigcup L) \rightarrow (D \bigcup L)$ satisfying:

$$(1)\ G \subseteq F'$$

$$(2)\ A \circ G \subseteq C$$

Note that this discussion of scene labeling is more general than that of Rosenfeld, Hummel, and Zucker[12] who consider only binary relational constraints.

VI. THE CONSISTENT LABELING PROBLEM

In this section we formulate a general network constraint analysis problem which we call the labeling problem. The labeling problem is a generalization of specific problems from each of several different specialty areas. Some of these specific problems include the subgraph isomorphism problem (Ullman[13]), the graph homomorphism problem (Harary[8]), the automata homomorphism problem (Ginzberg[5]), the graph coloring problem (Harary[8]), the relational homomorphism problem (Haralick and Kartus[6]), the packing problem (Deutsch[3]), the scene labeling problem (Barrow and Tenenbaum[1]), the shape matching problem (Davis[2]), the Latin square puzzle (Whitehead[15]), constraint satisfaction problems (Fike[4]), and theorem proving (Kowalski[9]). The generalized problem involves a set of units which usually represent a set of objects to be given names, a set of labels which are the possible names for the units, and a compatibility model containing ordered groups of units which mutually constrain one another and ordered groups of unit-label pairs which are compatible. The compatibility model is sometimes called a world model. The problem is to find a label for each unit such that the resulting set of unit-label pairs is consistent with the constraints of the world model.

Before we can fully state the labeling problem, we need some concepts and definitions. Let $U = \{1,\ldots,M\}$ be a set of M units and let L be a set of labels. A function $f:U \rightarrow L$ is called a labeling of U. The labeling problem is to use the world model to find a particular kind of labeling called a consistent labeling for all M units in U.

The problem of labeling is that not all of the labelings are consistent because some of the units are a priori known to mutually constrain one another. If an N-tuple of units $(u_1,\ldots,u_N)$ are known to mutually constrain one another, then not all labelings are permitted or legal for units $(u_1,\ldots,u_N)$. The compatibility model tells us which units mutually constrain one another N at a time and which labelings are permitted or legal for those units which do constrain one another. One way of representing this compatibility model is by a quadruple (U,L,T,R) where $T \subseteq U^N$ is the set of all N-tuples of units which mutually

constrain one another and the constraint relation $R \subseteq (U \times L)^N$ is the set of all 2N-tuples $(u_1,\ell_1,\ldots,u_N,\ell_N)$ where $(\ell_1,\ldots,\ell_N)$ is a permitted or legal labeling of units $(u_1,\ldots,u_N)$. We call T the unit constraint relation and R the unit-label constraint relation.

A labeling $f:U \to L$ is a consistent labeling with respect to the compatibility model (U,L,T,R) if and only if $(u_1,\ldots,u_N) \in T$ implies $(u_1,f(u_1),\ldots,u_N,f(u_N)) \in R$; that is, for each $(u_1,\ldots,u_N) \in T$, the labeling $(f(u_1),\ldots,f(u_N))$ is a permitted or legal labeling of units $(u_1,\ldots,u_N)$. When U and L are understood, such a labeling is called a (T,R)-consistent labeling. The consistent labeling problem is to find all consistent labelings with respect to the compatibility model (U,L,T,R). We denote the set of all (T,R)-consistent labelings by $\mathcal{L}(T,R)$.

In the following theorem we prove that the relational homomorphism problem can be expressed as a consistent labeling problem. Hence, all relational homomorphism problems can be solved by solving the consistent labeling problem.

Theorem 1

The relational homomorphism problem can be expressed as a consistent labeling problem.

Proof

Let $U = \{1,\ldots,M\}$ be a set of units, $T \subseteq U^N$, and $S \subseteq L^N$. Define $R \subseteq (U \times L)^N$ by $R = \{(u_1,\ell_1,\ldots,u_N,\ell_N) \in (U \times L)^N \mid (u_1,\ldots,u_N) \in T$ and $(\ell_1,\ldots,\ell_N) \in S\}$. Let f be a function from U to L. We will show that the labeling f is consistent with respect to the compatibility model (U,L,T,R) if and only if $T \circ f \subseteq S$.

Suppose f is a (T,R)-consistent labeling. Let $(\ell_1,\ldots,\ell_N) \in T \circ f$. Then there exists $(u_1,\ldots,u_N) \in T$ such that $\ell_n = f(u_n)$, $n = 1,\ldots,N$. But since f is a consistent labeling, $(u_1,f(u_1),\ldots,u_N,f(u_N)) \in R$. Now by definition of R, $(f(u_1),\ldots,f(u_N)) \in S$. Hence, $(\ell_1,\ldots,\ell_N) \in S$ and we obtain $T \circ f \subseteq S$.

Suppose $T \circ f \subseteq S$. Let $(u_1,\ldots,u_N) \in T$. Since $T \circ f \subseteq S$ and f is a function defined everywhere on U, $(f(u_1),\ldots,f(u_N)) \in S$. Now by definition of R, $(u_1,\ldots,u_N) \in T$ and $(f(u_1),\ldots,f(u_N)) \in S$ imply $(u_1,f(u_1),\ldots,u_N,f(u_N)) \in R$. Hence, f is a (T,R)-consistent labeling.

VII. FINDING CONSISTENT LABELINGS

Given the compatibility model (U,L,T,R), where $T \subseteq U^N$ and $R \subseteq (U \times L)^N$, the problem is to find all labelings in the set $\mathcal{L}(T,R)$. To motivate a procedure for doing this, we first investigate the cases where it is easy to compute it. Then we will look for ways to operate on the model, reducing it to a set of simpler models for which it is easier to find consistent labelings. To help us do this, we need two additional concepts.

We define the projection πR by:

$$\pi R = \{(u,\ell) \in U \times L \mid \text{for some } (u_1,\ell_1,\ldots,u_N,\ell_N) \in R,\ (u,\ell) = (u_n,\ell_n) \text{ for some } n\}$$

And for any N-tuple $(u_1,\ldots,u_N)$ of units we define the block $R(u_1,\ldots,u_N)$ by:

$$R(u_1,\ldots,u_N) = \{(\ell_1,\ldots,\ell_N) \in L^N \mid (u_1,\ell_1,\ldots,u_N,\ell_N) \in R\}.$$

Proposition 1 proves that if the relation πR is a function and if $(u_1,\ldots,u_N) \in T$ implies $R(u_1,\ldots,u_N) \neq \phi$, then πR is a (T,R)-consistent labeling. Furthermore, since $f \in \mathcal{L}(T,R)$ implies $f \subseteq \pi R$, if πR is a function, then $f = \pi R$. This implies $\mathcal{L}(T,R)$ is the singleton set $\{\pi R\}$. Hence, our simple procedure for finding consistent labelings will depend on determining the projection πR and to make sure that R does not have extraneous elements: $(u_1,\ell_1,\ldots,u_N,\ell_N) \in R$ implies $(u_1,\ldots,u_N) \in T$.

Proposition 1

Let $T \subseteq U^N$ and $R \subseteq (U \times L)^N$. Suppose πR is a function and $(u_1,\ldots,u_N) \in T$ implies $R(u_1,\ldots,u_N) \neq \phi$. Then πR is a (T,R)-consistent labeling.

Proof

Let $(u_1,\ldots,u_N) \in T$. Since $R(u_1,\ldots,u_N) \neq \phi$, there exists $\ell_1,\ldots,\ell_N$ such that $(u_1,\ell_1,\ldots,u_N,\ell_N) \in R$. By definition of πR, $(u_n,\ell_n) \in \pi R$, $n = 1,\ldots,N$. Since πR is a function, it is single-valued. Hence, $\ell_n = \pi R(u_n)$, $n = 1,\ldots,N$. Now, $(u_1,\ell_1,\ldots,u_N,\ell_N) \in R$ and $\ell_n = \pi R(u_n)$ implies $(u_1,\pi R(u_1),\ldots,u_N,\pi R(u_n)) \in R$. By definition of (T,R)-consistent labeling, πR is a (T,R)-consistent labeling.

This fact suggests a search procedure in which the set $\mathcal{L}(T,R)$ of consistent labelings is successively partitioned by reducing R until the resultant R either has no consistent labelings or πR is the consistent labeling. Such a search procedure can have two components. One part can reduce an R at any stage by removing from R those easy to find unit-label N-tuples which contribute to no consistent labelings. We call this part the look-ahead part. Another part can divide R into two relations such that the consistent labelings for the pair of relations constitute a partition for the consistent labelings of the original relation. We call this part the tree search part.

VII.1 Tree Search

The tree search is based on the idea that the easiest way to find the labelings in $\mathcal{L}(T,R)$, is to break that problem in two parts by finding an $R_1 \subseteq R$ and $R_2 \subseteq R$ so that the labelings in $\mathcal{L}(T,R_1)$ and $\mathcal{L}(T,R_2)$ are easier to find, do not duplicate one another, and exhaust the labelings in $\mathcal{L}(T,R)$.

One way of doing this is to break one of the blocks of R into two pieces and define R_1 to be R with one piece of the broken block and R_2 to be R with the other piece of the broken block. Let $(w_1,\ldots,w_N) \in T$ be given. Let $\{P_1,P_2\}$ be a partition of the block $R(w_1,\ldots,w_N)$. Define:

$$R_i = \{(u_1,\ell_1,\ldots,u_N,\ell_N) \in R \mid (u_1,\ldots,u_N) = (w_1,\ldots,w_N) \text{ implies } (\ell_1,\ldots,\ell_N) \in P_i\},\ i = 1 \text{ and } 2.$$

From Proposition 2, we know that $R_1,R_2 \subseteq R$ implies $\mathcal{L}(T,R_1) \cup \mathcal{L}(T,R_2) \subseteq \mathcal{L}(T,R)$. Proposition 3 (part 1) proves that the labelings in $\mathcal{L}(T,R_1)$ and $\mathcal{L}(T,R_2)$ exhaust the labelings in $\mathcal{L}(T,R)$; hence, $\mathcal{L}(T,R_1) \cup \mathcal{L}(T,R_2) = \mathcal{L}(T,R)$. Furthermore, the fact that $\{P_1,P_2\}$ is a partition of $R(w_1,\ldots,w_N)$ forces $\mathcal{L}(T,R_1) \cap \mathcal{L}(T,R_2) = \phi$ (part 2). Therefore, we obtain that

$\{\mathcal{L}(T,R_1), \mathcal{L}(T,R_2)\}$ is a partition of $\mathcal{L}(T,R)$. Because $R_1, R_2 \subseteq R$, we have reduced the original problem to two smaller problems.

Proposition 2

$R \subseteq S$ implies $\mathcal{L}(T,R) \subseteq \mathcal{L}(T,S)$.

Proof

Let $f \in \mathcal{L}(T,R)$. Let $(u_1,\ldots,u_N) \in T$. Since $f \in \mathcal{L}(T,R)$ and $(u_1,\ldots,u_N) \in T$, $(u_1,f(u_1),\ldots,u_N,f(u_N)) \in R$. But $R \subseteq S$. Hence, $(u_1,f(u_1),\ldots,u_N,f(u_N)) \in S$. Now by definition of $\mathcal{L}(T,S)$, $f \in \mathcal{L}(T,S)$.

Proposition 3

Let $T \subseteq U^N$ and $R \subseteq (U \times L)^N$. Let $(w_1,\ldots,w_N) \in T$. Let $\{P_1,P_2\}$ be a partition of the block $R(w_1,\ldots,w_N)$. Define:

$$R_i = \{(u_1,\ell_1,\ldots,u_N,\ell_N) \in R \mid (u_1,\ldots,u_N) = (w_1,\ldots,w_N) \text{ implies } (\ell_1,\ldots,\ell_N) \in P_i\},\ i = 1 \text{ or } 2.$$

Then

(1) $\mathcal{L}(T,R) \subseteq \mathcal{L}(T,R_1) \bigcup \mathcal{L}(T,R_2)$

(2) $\mathcal{L}(T,R_1) \bigcap \mathcal{L}(T,R_2) = \phi$

Proof

(1) Let $f \in \mathcal{L}(T,R)$. Since $(w_1,\ldots,w_N) \in T$ and $f \in \mathcal{L}(T,R)$, $(w_1,f(w_1),\ldots,w_N,f(w_N)) \in R$. By definition of R_1 and R_2, $(w_1,f(w_1),\ldots,w_N,f(w_N)) \in R_1$ or R_2. Let $(u_1,\ldots,u_N) \in T$. Since $f \in \mathcal{L}(T,R)$, $(u_1,f(u_1),\ldots,u_N,f(u_N)) \in R$. Either $(u_1,\ldots,u_N) = (w_1,\ldots,w_N)$ or not. If $(u_1,\ldots,u_N) = (w_1,\ldots,w_N)$, we have by assumption either $(u_1,f(u_1),\ldots,u_N,f(u_N)) \in R_1$ or R_2. If $(u_1,\ldots,u_N) \neq (w_1,\ldots,w_N)$, then by definition of R_1 and R_2, $(u_1,f(u_1),\ldots,u_N,f(u_N)) \in R$ and $(u_1,\ldots,u_N) \neq (w_1,\ldots,w_N)$ imply $(u_1,f(u_1),\ldots,u_N,f(u_N)) \in R_1$ and R_2. Hence, $(u_1,\ldots,u_N) \in T$ implies $(u_1,f(u_1),\ldots,u_N,f(u_N)) \in R_1$ or $(u_1,\ldots,u_N) \in T$ implies $(u_1,f(u_1),\ldots,u_N,f(u_N)) \in R_2$. By definition of consistent labeling, $f \in \mathcal{L}(T,R_1) \bigcup \mathcal{L}(T,R_2)$.

(2) Suppose $f \in \mathcal{L}(T,R_1) \bigcap \mathcal{L}(T,R_2)$. Since $(w_1,\ldots,w_N) \in T$, $f \in \mathcal{L}(T,R_1)$, and $f \in \mathcal{L}(T,R_2)$, $(w_1,f(w_1),\ldots,w_N,f(w_N)) \in R_1$ and $(w_1,f(w_1),\ldots,w_N,f(w_N)) \in R_2$. This implies that $(f(w_1),\ldots,f(w_N)) \in P_1 \bigcap P_2$. But this is impossible, since $\{P_1,P_2\}$ is a partition of $R(w_1,\ldots,w_N)$. Hence $\mathcal{L}(T,R_1) \bigcap \mathcal{L}(T,R_2) = \phi$.

VII.2 Look-Ahead

The idea behind the look-ahead is to locate any N-tuples of unit-label pairs in R which do not contribute to consistent labelings. Some such N-tuples are easy to find. Others are difficult. By looking ahead for the easy ones and removing them, we guarantee that no matter what N-tuple in T is used to divide R into R_1 and R_2, we will not discover that the sets $\mathcal{L}(T,R_1)$ and $\mathcal{L}(T,R_2)$ are empty. This single idea can be quite powerful because the efficiency of the tree search depends, in part, on which N-tuple in T is used to do the division. At any step, choosing the "right" N-tuple could show that $\mathcal{L}(T,R_1)$ and $\mathcal{L}(T,R_2)$ are empty. Constantly choosing the "wrong" N-tuple (a situation which Mackworth[10] shows leads to thrashing behavior) might mean that it is not until the bottom of the tree search that we discover that $\mathcal{L}(T,R_1)$ and $\mathcal{L}(T,R_2)$ are empty and that we may have to rediscover that there are no consistent labelings again and again all for the same reason: that there is some N-tuple in T such that from where we are (near the top of the tree) there is no way it can participate in a consistent labeling.

To help us define the look-ahead operator ϕ, we need to define the idea of restriction. If $T \subseteq U^N$ and $V \subseteq U$, we define the restriction of T by V as:

$$T|_V = \{(u_1,\ldots,u_N) \in T \mid u_n \in V,\ n = 1,\ldots,N\}$$

If $R \subseteq (U \times L)^N$ and $V \subseteq U$, we define the restriction of R by V as:

$$R|_V = \{(u_1,\ell_1,\ldots,u_N,\ell_N) \in R \mid u_n \in V,\ n = 1,\ldots,N\}.$$

Proposition 4 proves that f, a consistent labeling with respect to (U,L,T,R), implies $f|_V$ is a consistent labeling with respect to $(V,L,T|_V,R|_V)$ where $V \subseteq U$.

The look-ahead operator ϕ throws out of R all those N-tuples of unit-label pairs which cannot be extended to consistent labelings through each N-tuple in T. It is defined by:

$$\phi R = \Big\{(u_1,\ell_1,\ldots,u_N,\ell_N) \in R \mid (w_1,\ldots,w_N) \in T \text{ implies that there exists a consistent labeling f with respect to the compatibility model } (V,L,T|_V,R|_V) \text{ where } V = \{u_1,\ldots,u_N,w_1,\ldots,w_N\}\Big\}$$

Theorem 2 proves that $\mathcal{L}(T,R) = \mathcal{L}(T,\phi R)$ from which it follows by induction that $\mathcal{L}(T,R) = \mathcal{L}(T,\phi^\infty R)$.

Proposition 4

Suppose $V \subseteq U$. Then f, a consistent labeling with respect to (U,L,T,R), implies that $f|_V$ is a consistent labeling with respect to $(V,L,T|_V,R|_V)$.

Proof

Suppose $V \subseteq U$ and f is a consistent labeling with respect to (U,L,T,R). Let $(v_1,\ldots,v_N) \in T|_V$. Certainly $(v_1,\ldots,v_N) \in T$. Since f is a consistent labeling with respect to (U,L,T,R), $(v_1,f(v_1),\ldots,v_N,f(v_N)) \in R$. But $(v_1,\ldots,v_N) \in T|_V$ implies $v_n \in V$, $n = 1,\ldots,N$. Hence $(v_1,f(v_1,\ldots,u_N,f(u_N)) \in R|_V$ and

$f(v_n) = f|_V(v_n)$, $n = 1,\ldots,N$. Therefore, $(v_1,f|_V(v_1),\ldots,v_N,f|_V(v_n)) \in R|_V$.

Theorem 2

$$\mathcal{L}(T,R) = \mathcal{L}(T,\phi R).$$

Proof

By definition of ϕR, $\phi R \subseteq R$. By Proposition 2, $\phi R \subseteq R$ implies $\mathcal{L}(T,\phi R) \subseteq \mathcal{L}(T,R)$.

Suppose $f \in \mathcal{L}(T,R)$. Let $(u_1,\ldots,u_N) \in T$. Then $(u_1,f(u_1),\ldots,u_N,f(u_N)) \in R$ since $f \in \mathcal{L}(T,R)$. We need to show that $(u_1,f(u_1),\ldots,u_N,f(u_N)) \in \phi R$. So let $(w_1,\ldots,w_N) \in T$. Define $V = \{u_1,\ldots,u_N,w_1,\ldots,w_N\}$. By Proposition 4, f is a consistent labeling with respect to the compatibility model (U,L,T,R) implies that $f|_V$ is a consistent labeling with respect to $(V,L,T|_V,R|_V)$. Thus, by definition of ϕ, $(u_1,f|_V(u_1),\ldots,u_N,f|_V(u_N)) \in \phi R$. Since $u_n \in V$, $n = 1,\ldots,N$, $f|_V(u_n) = f(u_n)$. Hence, $(u_1,f(u_1),\ldots,u_N,f(u_N)) \in \phi R$. Then by definition of $\mathcal{L}(T,\phi R)$, $f \in \mathcal{L}(T,\phi R)$.

To help make this discussion concrete, we give the example of Figure 7 which shows a tree search from beginning to end. The compatibility model consists of the unit set U = {1,2,3,4,5}, label set L = {a,b}, unit constraint relation T = {(1,3), (1,4), (2,3), (3,4), (4,5)}, and unit-label constraint relation R = {(1,a,3,a), (1,a,3,b), (1,a,4,a), (1,a,4,b), (2,a,3,a), (2,a,3,b), (3,a,4,b), (3,b,4,a), (4,a,5,a), (4,b,5,b)}. For this R, $\phi R = R$ so that the look-ahead does not help at the top of the tree. The tree consists of one division using the block R(1,3) = {(a,a), (a,b)}. This division forms the relations R_1 and R_2. After applying two iterations of the look-ahead operator to R_1 and R_2, we reach a fixed point and the projections $\pi\phi^2R_1$ and $\pi\phi^2R_2$ consistent labelings.

VIII. LOOK-AHEAD OPERATORS

Waltz[14], Montanari[11], Mackworth[10], and Haralick and Kartus[6] all give examples of some look-ahead operators. In this section we describe a look-ahead operator of the basic type and power used by the above researchers. Haralick and Shapiro[7] has a detailed discussion of this kind of 2 parameter look-ahead operator. It is defined by:

$$\phi_{KP}R = \{(u_1,\ell_1,\ldots,u_N,\ell_N) \in R \mid \text{for every combination } j_1,\ldots,j_K \text{ of } 1,\ldots,N \text{ and for every } u'_{K+1},\ldots,u'_p \in U \text{ there exists } \ell'_{K+1},\ldots,\ell'_p \in L \text{ such that f defined by } f(u_{jk}) = \ell_{jk},\ k = 1,\ldots,K \text{ and } f(u'_\rho) = \ell'_\rho,\ \rho = K+1,\ldots,P \text{ is a (T,R)-consistent labeling}\}$$

Figure 8 shows a tree search employing this look-ahead operator. Notice that it requires one more iteration to reach a fixed point. For binary compatibility models, ϕ_{23} operator will be weaker than the ϕ operator of the previous section.

IX. COMPLEXITY ANALYSIS

Each iteration of ϕ requires checking each N-tuple of R to see if it can be extended to a consistent labeling. Thus, each N-tuple of R must be extended by each N-tuple of T. Then a brute force procedure would generate at most $\#L^N$ labelings to be checked. At most, each such check requires determining whether for each N-tuple in T, the corresponding N-tuple of unit-label pairs is in R. Therefore, each iteration of ϕ requires at most $\#R\ \#T\ \#L^N\ \#T$ operations.

The greatest number of nodes in any branch can be $\log_2 \#R$. This represents the most number of times ϕ reaches a fixed point down a branch. In addition, there cannot be a combined total of any more than #R more iterations in the branch, since each iteration of ϕ not reaching a fixed point takes at least one N-tuple of unit-label pairs out of R. Hence, the number of iterations in any branch is at most $(\log_2 \#R + \#R)$.

We define the complexity, α, of a consistent labeling problem to the ratio of the number of branches the tree search has divided by the number of consistent labelings. Hence, the total number of operations is:

$$\alpha[1 + \#\mathcal{L}(T,R)][\#R + \log_2 \#R]\ \#R\ \#T\ \#L^N\ \#T$$

Of course, the NP-completeness of the problem implies that in the worst case we expect α to be exponential in the number of units. However, practical problems seem to have parameter values for α which must be low-order polynomials in number of units. This behavior in practice is similar to the behavior of other algorithms that solve NP-complete problems. For example, the simplex algorithm for linear programming hardly ever exhibits the worst case behavior in practice.

X. CONCLUSION

In this paper we have illustrated how a segmented scene can be translated into a relational structure and how scene matching can be accomplished by finding homomorphisms from one relation to another.

We have described the consistent labeling problem and shown how solving it solves the homomorphism problem. Finally, we described a procedure involving tree search and look-ahead operators for solving the consistent labeling problem.

REFERENCES

1. Barrow, H.G. and J.M. Tenenbaum. MYSYS: A System for Reasoning About Scenes, SRI AI Technical Report No. 121, Stanford Research Institute, Menlo Park, California, 1976.

2. Davis, L.S. Shape Matching Using Relaxation Techniques, TR-480, Computer Science Center, University of Maryland, September 1976.

3. Deutsch, J.P.A., "A Short Cut for Certain Combinatorial Problems," British Joint Computer Conference, 1966.

4. Fike, R.E., "REF-ARF: A System for Solving Problems Stated as Procedures," Artificial Intelligence, Vol. 1, 1970, pp. 27-120.

5. Ginzberg, A. Algebraic Theory of Automata, Academic Press, New York, 1968.

6. Haralick, R.M. and J. Kartus, "Arrangements, Homomorphisms, and Discrete Relaxation," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-8, No. 8, August 1978.

7. Haralick, R.M. and L.G. Shapiro, "The Consistent Labeling Problem," Remote Sensing Laboratory, University of Kansas Center for Research, Inc., Lawrence, Kansas, January 1978, 81 pages.

8. Harary, F. Graph Theory, Addison-Wesley Publishing Company, Massachusetts, 1969.

9. Kowalski, R., "A Proof Procedure Using Connection Graphs," Journal of the Association for Computing Machinery, Vol. 22, No. 4, October 1975, pp. 572-595.

10. Mackworth, A., "Consistency in Networks of Relations," Artificial Intelligence 8, 1977, pp. 99-118.

11. Montanari, U., "Networks of Constraints: Fundamental Properties and Applications to Picture Processing," Information Science 7, 1974, pp. 95-132.

12. Rosenfeld, A., R.A. Hummel, and S.W. Zucker, "Scene Labeling by Relaxation Operations," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-6, No. 7, June 1976, pp. 420-433.

13. Ullman, J.R., "An Algorithm for Subgraph Homomorphisms," Journal of the Association for Computing Machinery, Vol. 23, No. 1, January 1976, pp. 31-42.

14. Waltz, D.L. Generating Semantic Descriptions from Drawings of Scenes with Shadows, MIT Technical Report A1271, November 1972.

15. Whitehead, E.G., Jr. Combinatorial Algorithms, Courant Institute of Mathematical Sciences, New York University, 1972.

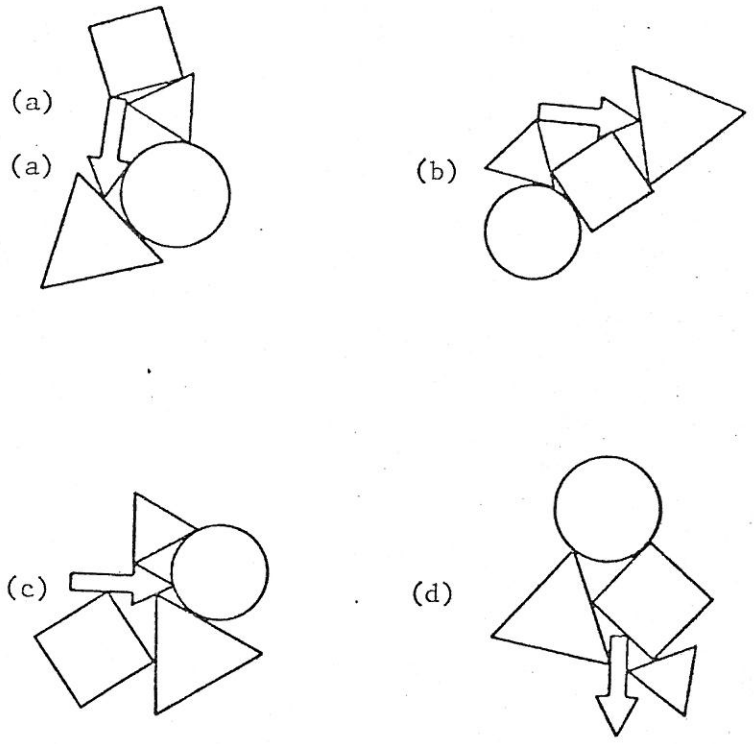


Figure 1 illustrates four drawings, each of which has two triangles, one square, one circle, and one arrow. Using the relation idea, there are 2 pairs of drawings whose arrangements are isomorphic.

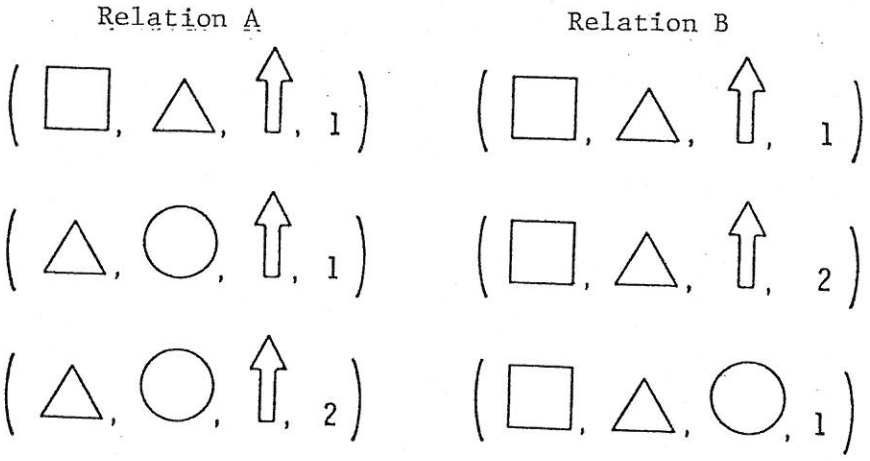


Figure 2 illustrates the quadruples for the relations defined by the drawings of Figure 1. The two drawings on the left in Figure 1 define Relation A. The two drawings on the right in Figure 1 define Relation B. The quadruple (□,Δ,↑,2) means that the drawing has a piece that consists of a square, triangle, and arrow pairwise touching each other and the label two designates that this is the second such piece in the drawing.

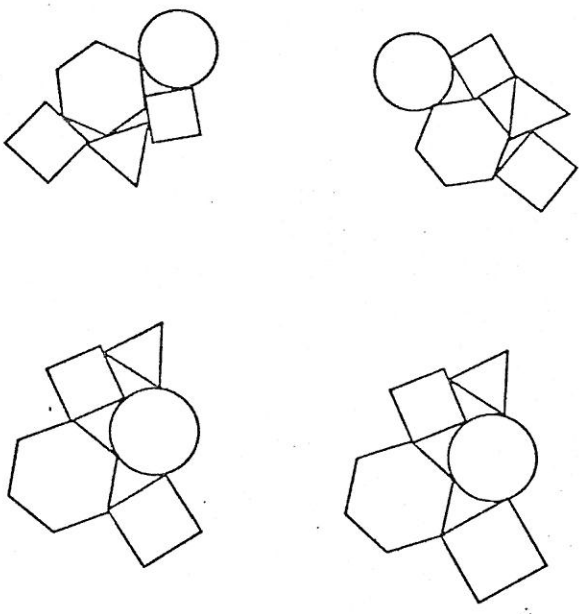


Figure 3 illustrates four drawings each of which has two squares, one circle, one hexagon, and one triangle. Using the relation concept, there are 2 pairs of drawings whose relations are isomorphic. The relations defined by each drawing here is isomorphic to the relations defined by one of the drawings in Figure 1.

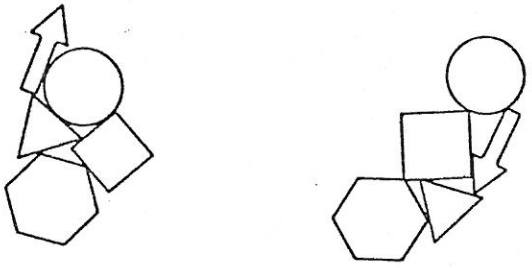
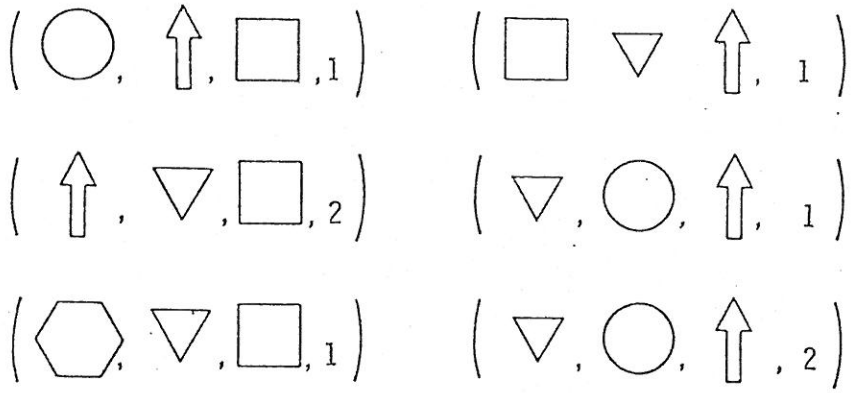
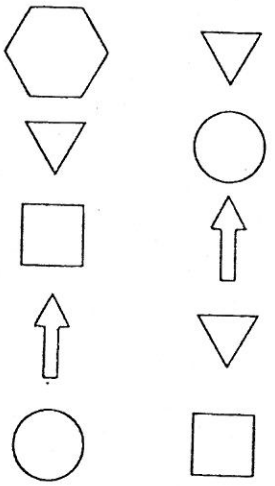


Figure 4 illustrates two drawings. Using the relation concept, labels of 1 or 2 can be assigned to each triplet of related shapes to make one of the drawings in Figure 1 a homomorphic image of one of these drawings.



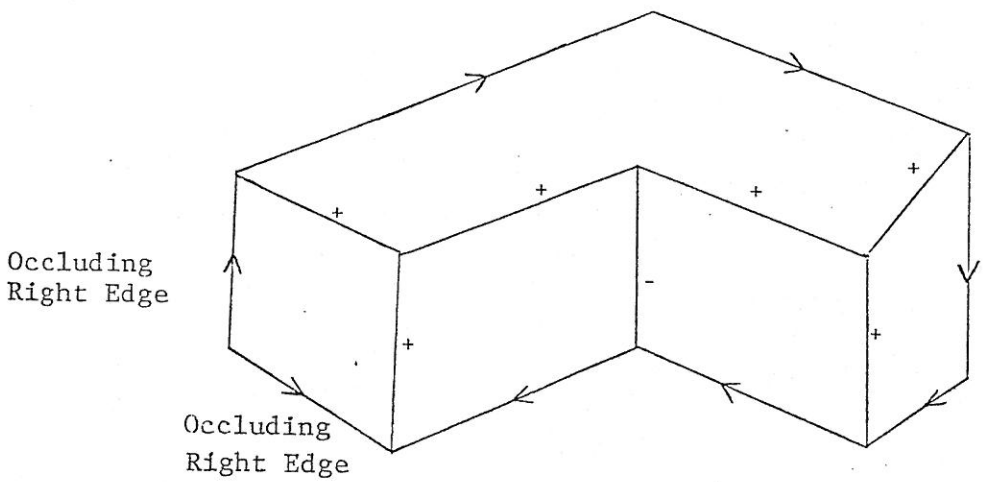
Relation for Right-Hand Drawing of Figure 4

Relation for Left-Hand Drawing for Figure 1



Homomorphism

Figure 5 illustrates the relation defined by one of the drawings in Figure 4 and the relation defined by one of the drawings in Figure 1. Below the relation is the homomorphism.



Key:

+	convex edge
-	concave edge
→	boundary edge with the face of the object to the right of the arrow

Figure 6 illustrates four possible labels for the edges of an object with trihedral vertices.

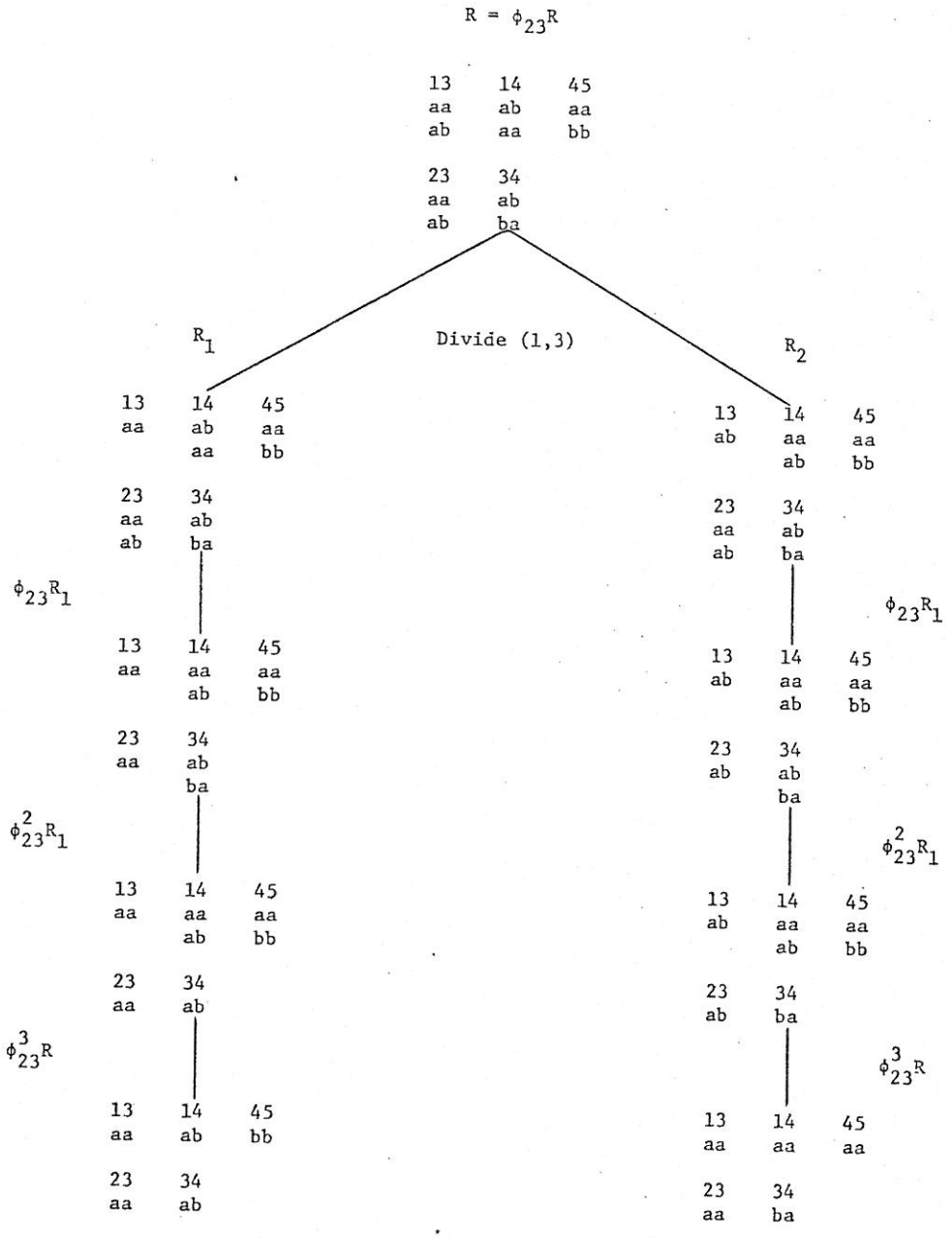


Figure 8 illustrates a simple tree search using the ϕ_{23} operator on the relations R and T from Figure 7.