The Consistent Labeling Problem: Part I

ROBERT M. HARALICK, SENIOR MEMBER, IEEE, AND LINDA G. SHAPIRO

Abstract—In this first part of a two-part paper we introduce a general consistent labeling problem based on a unit constraint relation T containing N-tuples of units which constrain one another, and a compatibility relation R containing N-tuples of unit-label pairs specifying which N-tuples of units are compatible with which N-tuples of labels. We show that Latin square puzzles, finding N-ary relations, graph or automata homomorphisms, graph colorings, as well as determining satisfiability of propositional logic statements and solving scene and edge labeling problems, are all special cases of the general consistent labeling problem. We then discuss the various approaches that researchers have used to speed up the tree search required to find consistent labelings. Each of these approaches uses a particular look-ahead operator to help eliminate backtracking in the tree search. Finally, we define the ϕ_{KP} two-parameter class of look-ahead operators which includes, as special cases, the operators other researchers have used.

Index Terms—Backtracking, consistent labeling, graph coloring, homorphisms, isomorphisms, look-ahead operators, matching, N-ary relations, relaxation, scene analysis, subgraph, tree search.

I. Introduction

IN THIS paper 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 [24], the graph homomorphism problem [16], the automata homomorphism problem [9], the graph coloring problem [16], the relational homomorphism problem [15], the packing problem [5], the scene labeling problem [1], the shape matching problem [4], the Latin square puzzle [27], constraint satisfaction problems [6], [7], and theorem proving [18]. 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 additional concepts and definitions. Let $U = \{1, \cdots, M\}$ be a set of M units and let L be a set of labels. If $u_1, \cdots, u_N \in U$ and $l_1, \cdots, l_N \in L$, then we call the N-tuple $(l_1, \cdots, l_N)$ a *labeling* of units $(u_1, \cdots, u_N)$. 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.

Manuscript received February 17, 1978; revised September 27, 1978.

R. M. Haralick is with the Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061.

L. G. Shapiro was with the Department of Computer Science, Kansas State University, Manhattan, KS 66506. She is now with the Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061.

The problem of labeling is that not all of the labelings in L^M are consistent because some of the units are *a priori* known to mutually constrain one another. If an N-tuple of units $(u_1, \cdots, u_N)$ are known to mutually constrain one another, then not all labelings are permitted or legal for units $(u_1, \cdots, u_N)$. 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 $(u_1, l_1, \cdots, u_N, l_N)$ where $(l_1, \cdots, l_N)$ is a permitted or legal labeling of units $(u_1, \cdots, u_N)$. We call T the *unit constraint relation* and R the *unit-label constraint relation*.

A labeling $(l_1, \cdots, l_P)$ is a *consistent labeling* of units $(u_1, \cdots, u_P)$ with respect to the compatibility model (U, L, T, R) if and only if $\{i_1, \cdots, i_N\} \subseteq \{1, \cdots, P\}$ and $(u_{i_1}, \cdots, u_{i_N}) \in T$ imply the $2N$-tuple $(u_{i_1}, l_{i_1}, \cdots, u_{i_N}, l_{i_N}) \in R$; that is, the labeling $(l_{i_1}, \cdots, l_{i_N})$ is a permitted or legal labeling of units $(u_{i_1}, \cdots, u_{i_N})$. Where U and L are understood, such a labeling $(l_1, \cdots, l_P)$ is called a *(T, R)-consistent labeling* of $(u_1, \cdots, u_P)$. The *consistent labeling problem* is to find all consistent labelings of units $(1, \ldots, M)$ with respect to the compatibility model.

In this paper we discuss the consistent labeling problem and define a two-parameter class of look-ahead operators that can be used to aid in finding solutions to a given labeling problem. In Section II we describe how a variety of combinatorial problems are special cases of the consistent labeling problem. In Section III we give a brief perspective of a procedure for solving the consistent labeling problem and discuss how researchers have used "relaxation" and "look-ahead" operators to accelerate the search time. Section III also gives a short summary of a recent paper [14] which addressed the labeling problem.

In Section IV we show the relationship between (T, R)-consistency as defined in this paper and global consistency with respect to R, as defined in the Haralick *et al.* paper, and we define the look-ahead operator ϕ_{KP} which generalizes the ϕ_P operator in the earlier paper. In Section V we show that the ϕ_{KP} operator can be implemented as a recursive procedure.

II. Some Examples of Labeling Problems

A. The Latin Square Puzzle

Latin square puzzles consist of a matrix and a set of objects to be arranged on the matrix. The objects each have a set of attributes, and there are constraints on which objects can be placed next to each other based on their attributes. One simple Latin square puzzle consists of a 4 X 4 matrix and 16 objects, each object having one of four possible colors and one of four possible shapes. The problem is to arrange the objects in the 4 X 4 matrix such that each row, each column, and each of the two main diagonals of the matrix contains exactly one object of each color and exactly one object of each shape.

In order to see how the Latin square puzzle reduces to a labeling problem, assume the colors are red, blue, yellow, and green; the shapes are circle, triangle, octagon, and square; and the elements of the matrix are numbered as in Fig. 1. The elements of the matrix can be thought of as the set of 16 units $U = \{1, \cdots, 16\}$. Each object can be described by a label consisting of the first letter of its color followed by the first letter of its shape. Thus, the red circle is *RC*, the green triangle is *GT*, and so on.

Let $L_S = \{C, T, O, S\}$ and $L_C = \{R, B, Y, G\}$ be the sets of shape labels and color labels, respectively. Let the label set L be defined by $L = L_C \times L_S$. Let $T \subseteq U^4$ consist of the 10 4-tuples of units which form the rows, columns, and diagonals of the 4 X 4 matrix and $R \subseteq (U \times L)^4$ consists of all 8-tuples of the form $(u_1, (c_1, s_1), u_2, (c_2, s_2), u_3, (c_3, s_3), u_4, (c_4, s_4))$ where

1) $(u_1, u_2, u_3, u_4) \in T$,
2) $(c_i, s_i) \in L$, $i = 1, \cdots, 4$, and
3) $i \neq j$ implies $c_i \neq c_j$ and $s_i \neq s_j$.

In order to examine the size of the compatibility model (U, L, T, R), consider the column consisting of units 1, 5, 9, and 13. There are 24 allowable 4-tuples of labels for these units. Consider one such 4-tuple (*RC*, *BT*, *YO*, *GS*). Putting in the labels together with the units, we see that the 8-tuple (1, *RC*, 5, *BT*, 9, *YO*, 13, *GS*) is a member of R. Hence, position 1 can be occupied by the red circle while position 5 can be occupied by the blue triangle, position 9 by the yellow octagon, and position 13 by the green square. According to the original rules of the Latin square puzzle, (1, *BT*, 5, *RC*, 9, *YO*, 13, *GS*) is also a valid 8-tuple since it represents unique labels for each of the four positions in a row. Thus, any permutation of the labels *RC*, *BT*, *YO*, *GS* can be combined with units 1, 5, 9, and 13 giving 24 8-tuples from (*RC*, *BT*, *YO*, *GS*). The constraints on one column of the array translate into $24^2 = 576$ 8-tuple in R. Similarly, the constraints on one row or one major diagonal of the array translate into 576 8-tuples in R. Since there are 4 rows, 4 columns, and 2 diagonals, the constraints on the array translate into 5760 8-tuples in R. The compatibility model, therefore, consists of a relation T having 10 4-tuples and a constraint relation R having 5760 8-tuples. The labels in Fig. 1 are one solution to the simple Latin square puzzle.

1 RC	2 YT	3 GO	4 BS
5 BO	6 GS	7 YC	8 RT
9 YS	10 RO	11 BT	12 GC
13 GT	14 BC	15 RS	16 YO

Fig. 1. Numbering of the matrix elements and a solution to the Latin square puzzle.

B. Scene Labeling

The scene labeling problem arises in the context where a picture is taken of a scene (such as an office) that has objects (like chairs, tables, desks, file cabinets, and so on) which need to be identified. A low-level computer vision system analyzes the picture, segments it, and perhaps even assigns one or more labels to some of the objects in the picture. Given the world model information which describes allowable spatial relationships among pieces of office furniture, and given the spatial relationships that exist among segments in the image, the scene labeling problem is to use the world model to find labels for each segment in the picture. This involves labeling those segments which have not been given labels and reducing the labeling ambiguities for those segments which have been given tenative labels by the low-level vision system.

Let $U = \{u_1, \cdots, u_M\}$ be the set of segments in the office image. By a spatial analysis of the image, we can produce a list of N-ary relationships that hold among the segments of U. Each item of the list can be expressed as a predicate and N possible segments for which the predicate holds. For example, with $N = 2$, the spatial analysis might discover that segment u_1 is above segment u_3, segment u_2 is on segment u_1, and segment u_5 is behind segment u_6. We write these kinds of relations in the shorthand form

ABOVE(u_1, u_3)
ON(u_2, u_1)
BEHIND(u_5, u_6)

The world model constraint is also a list of N-ary relationships. Each item of the list consists of a predicate and N possible object names for which the predicate holds. For example, with $N = 2$, the following constraints between object names may hold: pictures can be above chairs, books can be on desks, and chairs can be behind desks. In the shorthand form we write

ABOVE(PICTURE, CHAIR)
ON(BOOK, DESK)
BEHIND(CHAIR, DESK)

To put the scene labeling problem into the format of the general labeling problem, we must define the relation T of segments which mutually constrain one another and the relation R of constraints between segments and labels. We can construct the relations T and R as follows. Let P be a predicate. If for some segments $u_1, \cdots, u_N$, a spatial analysis of the image shows $P(u_1, \cdots, u_N)$ is true and if for some labels $l_1, \cdots, l_N$ the world model constraint permits $P(l_1, \cdots, l_N)$ to be true, and the low level vision analysis does not prohibit label l_n for segment u_n, $n = 1, \cdots, N$, then the N-tuple $(u_1, \cdots, u_N)$ is a member of T and the $2N$-tuple $(u_1, l_1, \cdots, u_N, l_N)$ is a member of R. For example, if one of the labels that the low-level vision system allows for segment u_1 is PICTURE and one of the labels it allows for segment u_3 is CHAIR and if P is the predicate ABOVE, since ABOVE(PICTURE, CHAIR) is true and ABOVE(u_1, u_3) is true, then T contains (u_1, u_3) and R contains $(u_1$, PICTURE, u_3, CHAIR). Each consistent labeling based on T and R is a possible labeling of the scene; and since consistent labelings are subsets of possible labelings, the number of labeling ambiguities will be reduced.

C. The Edge Orientation Problem

There are a variety of approaches to finding edges in a picture [22]. Most of them begin with the application of some local operator to determine the strength of an edge passing through each resolution cell in a particular direction. The problem with these local operators is that they tend to be noisy; their variance is high. Since most meaningful edges in real world images tend to be highly continuous with little curvature, it should be possible to combine the prior knowledge low curvature condition and the local gradient operator values to produce cleaner edges.

We define the orientation of an edge to be an angle between 0° and 360°. The edge lies along a line in the given angular direction and a person traveling along the edge in the angular direction of the edge will always find the darker side of the edge to his right. Low curvature edges mean that the maximum angle by which any small edge segment can bend with respect to its predecessor edge or successor edge segment is limited to some maximum bending angle which we call Δ (for example, 60°). With this kind of prior knowledge, we can formulate the edge orientation problem as a labeling problem.

Let $U = \{(i, j) \mid i = 1, \cdots, N \text{ and } j = 1, \cdots, M\}$ be the set of resolution cells of an N-row by M-column image. Let L be a set consisting of possible edge orientations (including the possibility of no edge). For example, L could be {none, 0, 45, 90, 135, 180, 225, 270, 315}. For each resolution cell (i, j) let $E(i, j) \subseteq L$ be the set of its possible edge orientations computed on the basis of the strength of some local edge operator. Let the neighborhood of resolution cell (i, j) be $N(i, j)$. $N(i, j)$ could be a 4-neighborhood, an 8-neighborhood, or perhaps something more complex. Only edge orientations of resolution cells in the neighborhood of a given resolution cell can constrain the edge orientation of the given resolution cell. On this basis we can define the compatibility model by (U, L, T, R) where

$$T = \{((i,j), (i',j')) \mid (i',j') \in N(i,j)\}$$

and

$$R = \{((i,j), l, (i',j'), l') \mid (i',j') \in N(i,j), l \in E(i,j), l' \in E(i',j') \text{ and } l, l' \neq \text{none implies } |l - l'| < \Delta\}.$$

D. The Edge Interpretation Problem

Once all edges have been identified, we might wish to discover what role each edge plays in a three-dimensional scene. In particular, suppose we have a scene consisting of an ideal line drawing representing a set of polyhedra. Clowes [2], Huffman [17], and Waltz [26] have investigated the use of constraints in the analysis of such scenes, and Rosenfeld *et al.* [22] presented a formal model of the process based on the work of Waltz. We will discuss the constraints that can be used in a simple scene and how they fit our model.

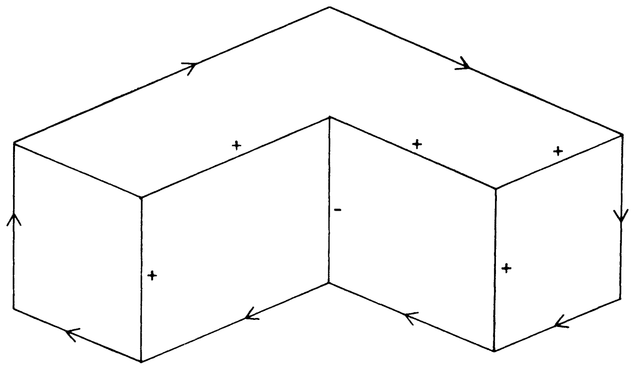

Fig. 2. Four possible labels for the edges of an object with trihedral vertices.

+ Convex edge.
– Concave edge.
→ Boundary edge with the face of the object to the right of the arrow.

Suppose we have a line drawing of a blocks world scene having only trihedral vertices (vertices where, at most, three surfaces meet). We would like to label each edge in the scene so that the label describes the physical role of the edge in the structure. For a scene with no cracks or shadows, Clowes and Huffman suggested that each edge can be labeled as either convex, concave, or a boundary edge of an object. Boundary edges can be divided further, depending on which side of the edge is a face of the object. A scene can be labeled by marking convex edges with the symbol +, concave edges with the symbol -, and boundary edges with an arrow directed so that the surface to the right of the arrow is part of the object. The object of Fig. 2 illustrates all four types of edges. For a scene with cracks and shadows, Waltz has suggested additional labels that can be added to the set.

The constraints come in when we look at the vertices of the scene. Guzman [10] gave names to each possible type of vertex that can appear in a scene. For example, the type of vertex in Fig. 3(a) is called a fork vertex. An analysis of the types of edges which can meet in a fork vertex in a trihedral block world scene, without shadows or cracks, shows that only certain combinations are physically possible. Fig. 3(b) shows the only five possible edge combinations for fork vertices in this type of scene.

An *order N vertex* is a vertex at which N edges meet. Each type of vertex is of a particular order; for example, the FORK vertex is of order 3. The set of order N vertices in a scene induces an Nth order constraint relation as follows.

Let V be the set of order N vertices in the scene, $E = \{1, \cdots, M\}$ be the set of edges in the scene, L be the set of edge labels, and X be the set of vertex labels. Let $f: V \to X$ be the function that assigns labels to vertices and $g: E \to \mathcal{P}(L)$ be the function that assigns sets of possible labels to edges. Let $h: X \to \mathcal{P}(L^N)$ be the function that specifies N-tuples of labels that can meet at each type of vertex. For each order N vertex type $x \in X$, we define T_x and R_x as follows.

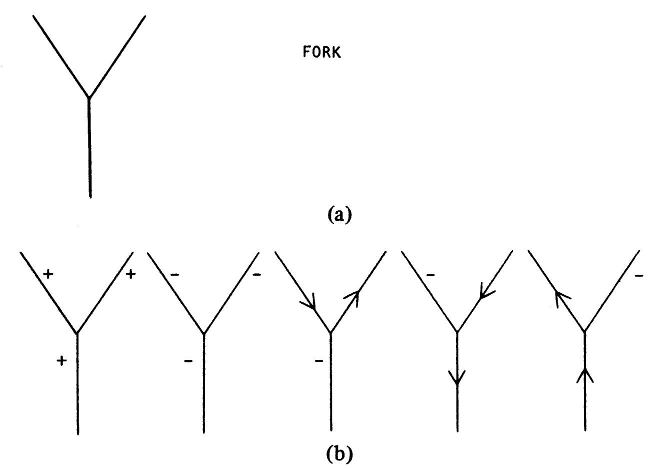

Fig. 3. FORK vertex (a) and the five physically possible edge combinations that can meet at a FORK vertex (b). These five combinations yield the 15 triplets of edge labels that the function h associates with the FORK vertex.

$T_x = \{(e_1, \cdots, e_N) \mid e_1, \cdots, e_N$ meet in vertex type x, edge e_{n+1} is the first edge in a clockwise direction after edge $e_n, n = 1, \cdots, N-1$ and e_1 is the designated starting edge for vertex type $x\}$.

$$R_x = \left\{(e_1, l_1, \cdots, e_N, l_N) \mid (e_1, \cdots, e_N) \in T_x \text{ and } (l_1, \cdots, l_N) \in h(x) \cap \bigtimes_{n=1}^{N} g(e_n)\right\}.$$

Thus, T_x consists of all N-tuples of edges that constrain each other because they meet in a vertex of type x, and R_x consists of all $2N$-tuples of edge-label pairs where the edges meet in a vertex of type x and the labels are a physically possible set of labels for a vertex of type x.

Let $T_N = \cup_{v \in V} T_{f(v)}$ and $R_N = \cup_{v \in V} R_{f(v)}$. T_N consists of all N-tuples of edges that constrain each other because they meet in some order N vertex and R_N is the corresponding constraint relation. Thus, for each N such that there is at least one order N vertex in the scene, there is a labeling probem with a compatibility model (E, L, T_N, R_N). A solution to the scene labeling problem is a labeling of edges $1, \cdots, M$ that is a solution to each of the order N labeling problems. That is, $(l_1, \cdots, l_M)$ is a scene-consistent labeling of edges $1, \cdots, M$ if $(l_1, \cdots, l_M)$ is (T_N, R_N)-consistent for each N such that there is an order N vertex in the scene.

E. 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 $R' = \{(b_1, \cdots, b_N) \in B^N \mid$ there exists $(a_1, \cdots, a_N)$ in R with $f(a_i) = b_i, i = 1, \cdots, N\}$. 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 scene labeling problem is an example of a relational homomorphism problem. Automata homomorphisms, graph homomorphisms, and graph colorings are other examples of relational homomorphism problems.

In the following theorem we prove that the relational homomorphism problem can be expressed as a consistent labeling problem.

Theorem 1: The relational homomorphism problem can be expressed as a consistent labeling problem.

Proof: Let $U = \{1, \cdots, M\}$ be a set of units, $T \subseteq U^N$, and $S \subseteq L^N$. Define $R \subseteq (U \times L)^N$ by $R = \{(u_1, l_1, \cdots, u_N, l_N) \in (U \times L)^N \mid (u_1, \cdots, u_N) \in T$ and $(l_1, \cdots, l_N) \in S\}$. Let f be a function from U to L. We will show that the labeling $(f(1), \cdots, f(M))$ of units $(1, \cdots, M)$ is consistent with respect to the compatibility model (U, L, T, R) if and only if $T \circ f \subseteq S$.

Suppose $(f(1), \cdots, f(M))$ is a consistent labeling with respect to (T, R) of units $(1, \cdots, M)$. Let $(l_1, \cdots, l_N) \in T \circ f$. Then there exists $(u_1, \cdots, u_N) \in T$ such that $l_n = f(u_n)$, $n = 1, \cdots, N$. But since $(f(1), \cdots, f(M))$ is a consistent labeling, $(u_1, f(u_1), \cdots, u_N, f(u_N)) \in R$. Now by definition of R, $(f(u_1), \cdots, f(u_N)) \in S$. Hence, $(l_1, \cdots, l_N) \in S$. Suppose $T \circ f \subseteq S$. Let $(u_1, \cdots, u_N) \in T$. Since $T \circ f \subseteq S$ and f is a function defined everywhere on U, $(f(u_1), \cdots, f(u_N)) \in S$. Now by definition of R, $(u_1, \cdots, u_N) \in T$ and $(f(u_1), \cdots, f(u_N)) \in S$ imply $(u_1, f(u_1), \cdots, u_N, f(u_N)) \in R$. Hence, $(f(u_1), \cdots, f(u_N))$ is a (T, R)-consistent labeling of units $(1, \cdots, M)$.

Because of the natural correspondence between graphs and binary relations, the subgraph isomorphism problem and the graph homomorphism problem can be expressed as relational homomorphism problems. Also, as proved in the next proposition, the automata homomorphism problem is equivalent to the relational homomorphism problem so that automata homomorphisms can be found by finding consistent labelings.

A *finite state automaton* is a triple (S, Σ, δ) where S is a finite set of states, Σ is a finite alphabet of input symbols, and $\delta: S \times \Sigma \to S$ is a function that maps state-input pairs into states. An *automata homomorphism* from an automaton (S, Σ, δ) to an automaton (Q, Σ, η) is a function $h: S \to Q$ such that $h(\delta(s, \sigma)) = \eta(h(s), \sigma)$ for every $s \in S$ and $\sigma \in \Sigma$.

Proposition 1: Let $A_1 = (S, \Sigma, \delta)$ and $A_2 = (Q, \Sigma, \eta)$ be two finite-state automata and suppose $\Sigma = \{\sigma_1, \cdots, \sigma_N\}$. Define $T = \{(s_0, \cdots, s_N) \in S^{N+1} \mid s_n = \delta(s_0, \sigma_n), n = 1, \cdots, N\}$ and $R = \{(q_0, \cdots, q_N) \in Q^{N+1} \mid q_n = \eta(q_0, \sigma_n), n = 1, \cdots, N\}$. Then $h: S \to Q$ is an automata homomorphism from A_1 to A_2 if and only if $T \circ h \subseteq R$.

Proof: Suppose $h(\delta(s, \sigma)) = \eta(h(s), \sigma)$ for all $\sigma \in \Sigma$ and $s \in S$. Let $(q_0, \cdots, q_N) \in T \circ h$. Then there exists $(s_0, \cdots, s_N) \in T$ such that $h(s_n) = q_n$, $n = 0, \cdots, N$. By definition of T, $(s_0, \cdots, s_N) \in T$ implies $s_n = \delta(s_0, \sigma_n)$, $n = 1, \cdots, N$. Now $q_n = h(s_n) = h(\delta(s_0, \sigma_n)) = \eta(h(s_0), \sigma_n) = \eta(q_0, \sigma_n)$, $n = 1, \cdots, N$. By definition of R, $q_n = \eta(q_0, \sigma_n)$, $n = 1, \cdots, N$ implies $(q_0, \cdots, q_N) \in R$. Hence, $T \circ h \subseteq R$.

Suppose $T \circ h \subseteq R$. Let $s_0 \in S$ be given. Define $s_n = \delta(s_0, \sigma_n)$, $n = 1, \cdots, N$. By definition of T, $s_n = \delta(s_0, \sigma_n)$, $n = 1, \cdots, N$ implies $(s_0, \cdots, s_N) \in T$. Let $q_n = h(s_n)$, $n = 0, \cdots, N$. By definition of $T \circ h$, $(s_0, \cdots, s_N) \in T$ and $q_n = h(s_n)$, $n = 0, \cdots, N$ implies $(q_0, \cdots, q_N) \in T \circ h$. By supposition

$T \circ h \subseteq R$. Hence, $(q_0, \cdots, q_N) \in R$. Now by definition of R, $q_n = \eta(q_0, \sigma_n)$, $n = 1, \cdots, N$. But $q_0 = h(s_0)$ so that $q_n = h(s_n) = h(\delta(s_0, \sigma_n)) = \eta(h(s_0), \sigma_n)$, $n = 1, \cdots, N$. Hence, $h(\delta(s, \sigma)) = \eta(h(s), \sigma)$ for all $\sigma \in \Sigma$ and $s \in S$.

The K-coloring graph problem can also be expressed as an N-ary relational homomorphism problem. Let $G = (V, E)$ be the graph to be colored. Let L be the set of K colors. Define $R \subseteq L \times L$ by $R = \{(l_1, l_2) \in L \times L \mid l_1 \neq l_2\}$. Then $h: V \to L$ is a homomorphism from E to R if and only if h is a K-coloring of G.

The packing problem [5] is defined as follows. Given a set of elements $E = \{e_1, \cdots, e_n\}$, a set of boxes $B = \{b_1, \cdots, b_m\}$, and a symmetric incompatibility relation $I \subseteq E \times E$, put each element of E in one box of B such that if e_i and e_j are incompatible ($(e_i, e_j) \in I$), they cannot be placed in the same box. The packing problem is equivalent to the graph coloring problem with the boxes taking the role of the colors.

F. Satisfiability and Consistent Labeling

The problem of satisfiability is as follows: given an expression in the propositional calculus, determine whether there is any assignment of the values true or false to each statement which makes the expression true. This *NP*-complete problem [3] is related to theorem proving, but it is simpler to understand and simpler to relate to the consistent labeling problem than theorem proving.

Whatever the given propositional expression may be, it is always possible to construct an equivalent expression which is in a conjunctive normal form (a conjunction of a set of clauses, each of which is a disjunction of simple statements and/or negation of simple statements). For example, the five clause expression $(L \vee K) \wedge (\overline{K} \vee \overline{M}) \wedge (M \vee L) \wedge (\overline{L} \vee K \vee \overline{M}) \wedge (\overline{K} \vee \overline{L} \vee \overline{M})$ is in conjunction normal form. To translate the satisfiability for this expression into a labeling problem, we will let the unit set be $\{1, 2, 3, 4, 5\}$ and associate the first clause $(L \vee K)$ with the unit 1, the second clause $(\overline{K} \vee \overline{M})$ with the unit 2, and so on. In order to make the expression true, it is necessary and sufficient for one term in each clause to take the value true. The constraint is that if a statement in one clause takes the value true (false), then its negation if it occurs in any clause cannot simultaneously take the value true (false).

The unit constraint relation T then consists of all pairs of units involving simple statements which are negatives of one another. The unit-clause association and the unit-constraint relation are listed below.

Unit-Clause Association	Unit-Constraint Relation	
1 $L \vee K$	(1, 2)	(2, 4)
2 $\overline{K} \vee \overline{M}$	(1, 4)	(3, 4)
3 $M \vee L$	(1, 5)	(3, 5)
4 $\overline{L} \vee K \vee \overline{M}$	(2, 3)	(4, 5)
5 $\overline{K} \vee \overline{L} \vee \overline{M}$		

The labels are the set of simple statements and their negations. In our example they are: $L, K, M, \overline{L}, \overline{K}, \overline{M}$. To make our tabulation shorter, we will define the unit-label constraint relation R by tabulating all elements of the form $(u, l, v, l') \notin R$ where $(u, v) \in T$.

Label Pairs for Constrained Units Not in the Unit-Label Constraint Relation R	
$(1, K, 2, \overline{K})$	$(3, M, 4, \overline{M})$
$(1, L, 4, \overline{L})$	$(3, L, 4, \overline{L})$
$(1, L, 5, \overline{L})$	$(3, L, 5, \overline{L})$
$(1, K, 5, \overline{K})$	$(3, M, 5, \overline{M})$
$(2, \overline{M}, 3, M)$	$(4, K, 5, \overline{K})$
$(2, \overline{K}, 4, K)$	

Each (T, R)-consistent labeling corresponds to an assignment of the value true to all those statements selected as labels to any of the units. In our example, the labeling $(K, \overline{M}, L, K, \overline{M})$ is a (T, R)-consistent labeling of units (1, 2, 3, 4, 5), and the value true for statements K, $\overline{M}$, and L will make the expression true. Hence, to satisfy the expression, take K true, M false, and L true.

To summarize, if the expression to be satisfied is

$$\bigwedge_{i=1}^{M} \left(\bigvee_{j=1}^{M_i} S_{ij} \right).$$

Then we define

$$U = \{1, \cdots, M\},$$

$$T = \{(i_1, i_2) \mid \text{for some } j_1 \text{ and } j_2, S_{i_1 j_1} = \overline{S_{i_2 j_2}}\},$$

$$L = \{S \mid S = S_{ij} \text{ for some } i \text{ and } j\},$$

and

$$R = \{(u_1, l_1, u_2, l_2) \mid (u_1, u_2) \in T \text{ and for some } j_1 \text{ and } j_2,$$
$$S_{u_1 j_1} = l_1,\ S_{u_2 j_2} = l_2 \text{ implies}$$
$$S_{u_1 j_1} \neq \overline{S_{u_2 j_2}}\}$$

where the equality of two statements means that the statements are symbolically identical and not that the truth values of the statements are the same. If $(l_1, \cdots, l_M)$ is a (T, R)-consistent labeling, then there exists indexes $j_1, \cdots, j_M$ such that $S_{ij_i} = l_i$, $i = 1, \cdots, M$ and the expression will be satisfied by each $S_{ij_i} = l_i$, $i = 1, \cdots, M$, taking the value true.

III. Related Literature

One general technique that can be used to solve a labeling problem is a depth-first search. The search procedure fixes labels to units as long as it can find a label for each new unit that is compatible, according to the constraint relation R, with the labels already fixed to previous units. Whenever the procedure cannot find a label for a new unit, it backtracks to the previous unit and tries to find a different label for that unit. If the procedure finds a label for all M units, it has found a consistent labeling. If the procedure backs up all the way to the first unit without finding any consistent labelings and there are no more possible labels for the first unit, the procedure fails and there exists no consistent labeling.

The depth-first search procedure suffers from thrashing. A poor choice of labels for one of the first units can cause failure of all paths stemming from that choice. To make the depth-first search more efficient, we must eliminate those paths which terminate because they are not contained in any consistent labeling. In Part II of this paper we will show that these terminating paths exist for one reason: there exists N-tuples of unit-label pairs in R which do not contribute to any consistent labeling. Thus, to do an efficient search, we must first remove from R all N-tuples of unit-label pairs which do not participate in a consistent labeling. Montanari [20] showed that this problem itself is NP-complete. However, compatibility relations can be graded with respect to the difficulty of removing those N-tuples. A *minimal* compatibility relation is a compatibility relation where the removal of any one N-tuple of unit-label pairs eliminates at least one consistent labeling with respect to that relation. The initial work by Waltz [26] on line labeling indicated that, although the compatibility relation he employed was not minimal, the amount of work to make it minimal using a sequentially implementable look-ahead operator was small. Gashnig [8] reported similar results. Hence, it may be that the worst case problems do not seem to arise very frequently in practice, and look-ahead operators of low-order complexity can be of great help.

To help us discuss some related literature, we need to define arc consistent and path consistent relations. A binary unit-label constraint relation R is *arc consistent* if, whenever the pair (u_1, l_1, u_2, l_2) is an element of R, then for every unit u, $(u_1, u) \in T$ implies $(u_1, l_1, u, l) \in R$ for some label l and $(u, u_2) \in T$ implies $(u, l', u_2, l_2) \in R$ for some label l'. A binary unit-label constraint relation R is *path consistent* if whenever the pair (u_1, l_1, u_2, l_2) is an element of R, then for every unit u, there exists a label l such that $(u_1, u) \in T$ implies $(u_1, l_1, u, l) \in R$ and $(u, u_2) \in T$ implies $(u, l, u_2, l_2) \in R$. The Waltz filtering algorithm [26], which was first described by Ullman [24a], uses a polynomial time look-ahead operator which reduces any binary compatibility relation to its maximal arc consistent piece. Montanari [20] discusses various aspects of the constrained labeling problem for binary relations. He shows that the largest path consistent relation contained in the compatibility relation can be computed in polynomial time by a look-ahead operator. Rosenfeld [21] and Rosenfeld *et al.* [22] show that arc and path consistent relations can be computed by look-ahead operators working in parallel. They name such look-ahead procedures discrete relaxation. Independently, Ullman [24] describes the efficacy of the Waltz look-ahead operator for the subgraph isomorphism problem. Haralick and Kartus [15] give a look-ahead operator generalization of arc consistency to N-ary relations, and Haralick [12] describes how the look-ahead operator can be used to help determine binary relational homomorphisms. Haralick [13] discusses the use of the look-ahead operator to find order-N arrangement homomorphisms and describe their applications in scene analysis. Mackworth [19] describes several implementations of look-ahead operators to compute arc and path consistent relations.

Rosenfeld *et al.* [22] also discuss fuzzy and probabilistic generalizations to discrete relaxation. Such generalizations have been used by Rosenfeld *et al.* [22], by Vanderbrug [25], by Hanson and Riseman [11] for edge enhancement, by Barrow and Tenenbaum [1] for scene analysis, by Davis [4] for shape matching, and by Zucker and Hummel [28] for clustering. However, since the probabilistic generalizations have not been shown to solve any optimization problem, it is not yet understood exactly what these relaxation generalizations do.

In a recent paper, Haralick *et al.* [14] describe a labeling problem and discuss look-ahead operators for reducing the size of the constraint relation. In that paper, a labeling problem is defined as follows. Given a set of M units $U = \{u_1, \cdots, u_M\}$, a set of labels L, and a constraint relation $R \subseteq (U \times L)^N$, find an M-tuple of labels $(l_1, \cdots, l_M)$ such that for every combination $i_1, \cdots, i_N$ of the integers $1, \cdots, M$, $(u_{i_1}, l_{i_1}, \cdots, u_{i_N}, l_{i_N})$ is an element of R. Such an M-tuple is called a *globally consistent* labeling of $u_1, \cdots, u_M$ and corresponds, in our definition of the labeling problem, to a consistent labeling of units $(u_1, \cdots, u_M)$ with respect to the compatibility model (U, L, U^N, R). We will present a brief summary of the main result of that paper.

Let $R \subseteq (U \times L)^N$ be a constraint relation. Denote by $\mathcal{L}(R)$ the set of globally consistent labelings defined by R. The minimal relation $S_R \subseteq R$ that has the same set of globally consistent labelings as R does is defined by

$$S_R = \cap \{R' \subseteq (U \times L)^N \mid \mathcal{L}(R') = \mathcal{L}(R)\}.$$

S_R can be characterized by the following properties:

1) $S_R = \{(u'_1, l'_1, \cdots, u'_N, l'_N) \in R \mid$ for some globally consistent labeling $(l_1, \cdots, l_N)$ and for some combination $i_1, \cdots, i_N$ of $1, \cdots, M$, $(u'_1, l'_1, \cdots, u'_N, l'_N) = (u_{i_1}, l_{i_1}, \cdots, u_{i_N}, l_{i_N})\}$.

2) $S_R \neq \phi$ iff there exists a globally consistent labeling of U.

3) S_R is symmetric.

A look-ahead operator for removing $2N$-tuples from R keeps only those $2N$-tuples of R that are extendable to consistent labelings of P-tuples for some $P > N$. To this end, the look-ahead operator $\phi_P(R)$ is defined by

$$\phi_P(R) = \{(u_1, l_1, \cdots, u_N, l_N) \in R \mid \text{for all } u_{N+1}, \cdots, u_p \in U \text{ there exist } l_{N+1}, \cdots, l_p \in L \text{ such that } (l_1, \cdots, l_p) \text{ is a consistent labeling of } (u_1, \cdots, u_p)\}.$$

The relationship of the ϕ_P operator to S_R can be characterized by the following properties:

1) $\phi_P(S_R) = S_R$.

2) $S_R \subseteq \phi_P(R) \subseteq R$.

3) $S_R \subseteq \phi_P^K(R)$ for all $K \geqslant 0$.

The remainder of this paper generalizes these results to the more general class of look-ahead operators which are of use in determining (T, R)-consistent labelings. The introduction of T into the compatibility model has direct consequences in implementation. Our results show that only the units which mutually constrain each other need to be stored and manipulated to achieve the same effect the ϕ_P operator has on the Haralick *et al.* model which assumes $T = U^N$ and requires more work.

IV. Finding (T, R)-Consistent Labelings with the Help of a Generalized Look-Ahead Operator ϕ_{KP}

A. Relationship Between Global Consistency and (T, R)-Consistency

In the Haralick *et al.* paper, an M-tuple of labels $(l_1, \cdots, l_M)$ is a globally consistent labeling of units $1, \cdots, M$ with respect to R if for *every* combination $i_1, \cdots, i_N$ of the integers

$1, \cdots, M$, the $2N$-tuple $(u_{i_1}, l_{i_1}, \cdots, u_{i_N}, l_{i_N})$ is an element of R. In a real-world labeling problem, it is often the case that a relatively small number of N-tuples of units constrain each other. This suggests that instead of checking every N-tuple of units, we need to only check those N-tuples of units that mutually constrain each other. We will say that the N-tuple of units $(u_1, \cdots, u_N)$ *mutually constrain* each other if and only if for some N-tuple of labels $(l_1, \cdots, l_N)$, the $2N$-tuple $(u_1, l_1, \cdots, u_N, l_N) \notin R$. The following proposition states that a labeling is a globally consistent labeling if and only if it is a (T, S)-consistent labeling, where T is the set of N-tuples that mutually constrain each other and S is that subset of R involving only N-tuples of unit-label pairs in which the corresponding N-tuples of units do mutually constrain each other.

This proposition tells us two important facts about the (U, L, T, R) model. First, the (U, L, T, R) model can find consistent labeling of units $1, \cdots, M$ faster than the Haralick *et al.* model, since only those N-tuples of units that constrain each other need be checked. Second, the (U, L, T, R) model often requires less memory since only those $2N$-tuples of R containing units that constrain each other need be stored. Thus, the (U, L, T, R) model leads to a more efficient implementation than the Haralick *et al.* model.

Proposition 2: Let $U = \{1, \cdots, M\}$ and $R \subseteq (U \times L)^N$. Let $S = \{(u_1, l_1, \cdots, u_N, l_N) \in R \mid$ for some $l'_1, \cdots, l'_N \in L$, $(u_1, l'_1, \cdots, u_N, l'_N) \notin R\}$, and $T = \{(u_1, \cdots, u_N) \in U^N \mid$ for some $l'_1, \cdots, l'_N \in L, (u_1, l'_1, \cdots, u_N, l'_N) \notin R\}$. Then $(l_1, \cdots, l_M)$ is a globally consistent labeling of units $(1, \cdots, M)$ with respect to R if and only if $(l_1, \cdots, l_M)$ is a (T, S)-consistent labeling of $(1, \cdots, M)$.

Proof: Suppose $(l_1, \cdots, l_M)$ is a globally consistent labeling of $(1, \cdots, M)$ with respect to R. To show that it is a (T, S)-consistent labeling, we need to show $(u_1, l_{u_1}, \cdots, u_N, l_{u_N}) \in S$ for every $(u_1, \cdots, u_N) \in T$. Let $(u_1, \cdots, u_N) \in T$. Then by definition of T, there exist labels $l'_1, \cdots, l'_N \in L$ such that $(u_1, l'_1, \cdots, u_N, l'_N) \notin R$.

Since $(l_1, \cdots, l_M)$ is a globally consistent labeling of units $(1, \cdots, M)$ with respect to R, $(u_1, l_{u_1}, \cdots, u_N, l_{u_N}) \in R$. Now by definition of S, $(u_1, l_{u_1}, \cdots, u_N, l_{u_N}) \in R$ and $(u_1, l'_1, \cdots, u_N, l'_N) \notin R$ imply $(u_1, l_{u_1}, \cdots, u_N, l_{u_N}) \in S$. Hence, labeling $(l_1, \cdots, l_M)$ is a (T, S)-consistent labeling of $(1, \cdots, M)$.

Suppose $(l_1, \cdots, l_M)$ is a (T, S)-consistent labeling of $(1, \cdots, M)$. To show it is globally consistent labeling with respect to R, we need to show $(u_1, l_{u_1}, \cdots, u_N, l_{u_N}) \in R$ for every $u_1, \cdots, u_N \in U$. Let $u_1, \cdots, u_N \in U$. Either $(u_1, \cdots, u_N) \in T$ or not. If $(u_1, \cdots, u_N) \in T$, then by definition of (T, S)-consistency, $(u_1, l_{u_1}, \cdots, u_N, l_{u_N}) \in S$. Since $S \subseteq R$, $(u_1, l_{u_1}, \cdots, u_N, l_{u_N}) \in R$. If $(u_1, \cdots, u_N) \notin T$, then by definition of T, for every $l'_1, \cdots, l'_N \in L$, $(u_1, l'_1, \cdots, u_N, l'_N) \in R$. In particular then, $(u_1, l_{u_1}, \cdots, u_N, l_{u_N}) \in R$. Hence, labeling $(l_1, \cdots, l_M)$ is a globally consistent labeling of $(1, \cdots, M)$ with respect to R.

B. The Generalized Look-Ahead Operator

The Haralick *et al.* paper defines a look-ahead operator ϕ_P which, when applied to a constraint relation R, removes some $2N$-tuples which do not contribute to a globally consistent labeling. We can generalize this look-ahead operator ϕ_P not only to work within the framework of the (U, L, T, R) model, but also to have an additional parameter K to give us more control over the use of the operator.

Let $U = \{1, \cdots, M\}$ be a set of units, L be a set of labels, $T \subseteq U^N$, and $R \subseteq (U \times L)^N$. Let $K \leqslant N \leqslant P$ with $K < P$. The *look-ahead operator* ϕ_{KP} is defined by $\phi_{KP} R = \{(u_1, l_1, \cdots, u_N, l_N) \in R \mid$ for every combination $j_1, \cdots, j_K$ of $1, \cdots, N$ and for every $u'_{K+1}, \cdots, u'_P \in U$, there exists $l'_{K+1}, \cdots, l'_P \in L$ such that $(l_{j_1}, \cdots, l_{j_K}, l'_{K+1}, \cdots, l'_P)$ is a (T, R)-consistent labeling of $(u_{j_1}, \cdots, u_{j_K}, u'_{K+1}, \cdots, u'_P)\}$.

Thus, to apply the ϕ_{KP} operator to R, we individually check each N-tuple of R. We fix any K of the N units $(u_{j_1}, \cdots, u_{j_K})$ to their labels in the N-tuple and check every set of $P - K$ units to determine if there are $P - K$ labels which make the K fixed labels $(l_{j_1}, \cdots, l_{j_K})$ plus the $P - K$ free labels $(l'_{K+1}, \cdots, l'_P)$ a consistent labeling of all P units $(u_{j_1}, \cdots, u_{j_K}, u'_{K+1}, \cdots, u'_P)$. We repeat this process for every combination of K out of the N units in the N-tuple. If for any combination of K fixed units and labels, there is a set of $P - K$ units for which $P - K$ labels, to make a consistent labeling, cannot be found, the $2N$-tuple is thrown out. Mackworth [19] uses the ϕ_{23} operator, the ϕ_P operator of Haralick *et al.* [14] is the ϕ_{NP} operator, Haralick and Kartus [15] use the ϕ_{lN} operator, and Ullman [21] uses the ϕ_{12} operator.

We illustrate the use of the ϕ_{KP} operator with an example where $N = 3$, $K = 2$, and $P = 4$.

C. Example

Let

$U = \{1, 2, 3, 4, 5\}$

$L = \{a, b\}$

$T = \{(1, 2, 3), (1, 2, 4), (1, 2, 5), (2, 3, 4), (2, 3, 5), (3, 4, 5)\}$

$N = 3, M = 5, K = 2, P = 4$

and

$$\begin{aligned} R = \{&(1, a, 2, a, 3, a)\\ &(1, a, 2, a, 4, a)\\ &(1, a, 2, a, 5, a)\\ &(1, a, 2, b, 3, a)\\ &(1, a, 2, b, 4, b)\\ &(1, b, 2, b, 5, b)\\ &(2, a, 3, a, 4, a)\\ &(2, a, 3, a, 5, a)\\ &(2, b, 3, a, 4, b)\\ &(3, a, 4, a, 5, a)\}.\end{aligned}$$

The results of examining three $2N$-tuples $(1, a, 2, a, 3, a)$, $(1, a, 2, b, 3, a)$, and $(1, a, 2, b, 5, b)$ are shown in Fig. 4. The $2N$-tuple $(1, a, 2, a, 3, a)$ passed all tests and is, therefore, an element of $\phi_{2,4} R$. The $2N$-tuple $(1, a, 2, b, 3, a)$ passed its first test since with the fixed unit-label pairs $(1, a)$ and $(2, b)$ and free units 3, 4 the 4-tuple (a, b, a, b) of labels is a consistent labeling of $(1, 2, 3, 4)$. However, with $(1, a)$ and $(2, b)$ still fixed and free units 3, 5, there is no consistent labeling of $(1, 2, 3, 5)$. (There is no label x with $2N$-tuple $(1, a, 2, b, 5, x)$

2N-Tuple	K = 2 Fixed unit-label pairs	P - K = 2 free units	P - K = 2 labels for the free units that contribute to a consistent labeling of all P = 4 units
(1,a,2,a,3,a)	1,a,2,a	3,4	3,a,4,a
		3,5	3,a,5,a
		4,5	4,a,5,a
	1,a,3,a	2,4	2,a,4,a
		2,5	2,a,5,a
		4,5	4,a,5,a
	2,a,3,a	1,4	1,a,4,a
		1,5	1,a,5,a
		4,5	4,a,5,a
(1,a,2,b,3,a)	1,a,2,b	3,4	3,a,4,b
		3,5	NONE
	No more combinations need be looked at for this 2N-tuple		
(1,b,2,b,5,b)	1,b,2,b	3,4	NONE
	No more combinations need be looked at for this 2N-tuple		

Fig. 4. Application of the ϕ_{KP} operator to three $2N$-tuples of R.

in R.) Thus $(1, a, 2, b, 3, a)$ is not in $\phi_{2,4}R$. Similarly, the $2N$-tuple $(1, b, 2, b, 5, b)$ fails its first test and is not in $\phi_{2,4}R$.

After testing each $2N$-tuple of R, we obtain

$$\begin{aligned}\phi_{2,4}R = \{&(1, a, 2, a, 3, a)\\ &(1, a, 2, a, 4, a)\\ &(1, a, 2, a, 5, a)\\ &(2, a, 3, a, 4, a)\\ &(2, a, 3, a, 5, a)\\ &(3, a, 4, a, 5, a)\}.\end{aligned}$$

At this point, every $2N$-tuple of R contributes to the one consistent labeling (a, a, a, a, a) of $(1, 2, 3, 4, 5)$.

In the above example, the look-ahead operator removed every $2N$-tuple of R that did not directly contribute to a consistent labeling of the units $(1, \cdots, M)$. We cannot expect ϕ_{KP} to always reduce every R to only those $2N$-tuples that contribute to a consistent labeling, since some relations are much more complex than the R in our example. Also, for large M and small P, ϕ_{KP} may not be as powerful as the $\phi_{2,4}$ above. However, we can expect that ϕ_{KP} will never remove a $2N$-tuple that does contribute to a consistent labeling of $(1, \cdots, M)$. The following proposition ensures this by showing that a labeling is (T, R)-consistent if and only if it is $(T, \phi_{KP}R)$-consistent. To make our notation easier, we let $\mathcal{L}(T, R)$ be the set of all (T, R)-consistent labelings.

Lemma 1: Let $U = \{1, \cdots, M\}$, $T \subseteq U^N$, $R \subseteq (U \times L)^N$, and $S \subseteq (U \times L)^N$. Then $R \subseteq S$ implies $\mathcal{L}(T, R) \subseteq \mathcal{L}(T, S)$.

Proposition 3: Let $U = \{1, \cdots, M\}$ be a set of units, L be a set of labels, and $R \subseteq (U \times L)^N$. Let $T \subseteq U^N, K \leqslant N \leqslant P$, $P > K$, and ϕ_{KP} be a look-ahead operator on R. Then $\mathcal{L}(T, R) = \mathcal{L}(T, \phi_{KP}R)$.

Proof: Since $\phi_{KP}R \subseteq R$, by Lemma 1, $\mathcal{L}(T, \phi_{KP}R) \subseteq \mathcal{L}(T, R)$.

Suppose $(l_1, \cdots, l_M)$ is a (T, R)-consistent labeling. Let $(u_1, \cdots, u_N) \in T$. Then, since $(l_1, \cdots, l_M)$ is a (T, R)-consistent labeling $(u_1, l_{u_1}, \cdots, u_N, l_{u_N}) \in R$. Let $u'_{K+1}, \cdots, u'_P \in U$ and $j_1, \cdots, j_K$ be a combination of $1, \cdots, N$. Since $(l_1, \cdots, l_M)$ is a (T, R)-consistent labeling of $(1, \cdots, M)$ and since $\{u_{j_1}, \cdots, u_{j_K}, u'_{K+1}, \cdots, u'_P\} \subseteq \{1, \cdots, M\}$, $(l_{u_{j_1}}, \cdots, l_{u_{j_K}}, l_{u'_{K+1}}, l_{u'_P})$ is a (T, R)-consistent labeling of $(u_{j_1}, \cdots, u_{j_K}, u'_{K+1}, \cdots, u'_P)$. Now by definition of the look-ahead operator ϕ_{KP}, $(u_1, l_{u_1}, \cdots, u_N, l_{u_N}) \in \phi_{KP}R$. Finally, since $(l_1, \cdots, l_M)$ is a labeling and $(u_1, \cdots, u_N) \in T$ imply $(u_1, l_{u_1}, \cdots, u_N, l_{u_N}) \in \phi_{KP}R$, we have by definition of a consistent labeling that $(l_1, \cdots, l_M)$ is a $(T, \phi_{KP}R)$-consistent labeling. Hence, $(l_1, \cdots, l_M) \in \mathcal{L}(T, \phi_{KP}R)$ and $\mathcal{L}(T, R) \subseteq \mathcal{L}(T, \phi_{KP}R)$.

We can summarize the results from Propositions 2 and 3 in the following theorem.

Theorem 2: Let $U = \{1, \cdots, M\}$ be a set of units, L be a set of labels, and $R \subseteq (U \times L)^N$. Let $T = \{(u_1, \cdots, u_N) \in U^N \mid$ for some $l'_1, \cdots, l'_N \in L$, $(u_1, l'_1, \cdots, u_N, l'_N) \notin R\}$ and $S = \{(u_1, l_1, \cdots, u_N, l_N) \in R \mid$ for some $l'_1, \cdots, l'_N \in L$, $(u_1, l'_1, \cdots, u_N, l'_N) \notin R\}$. Let $K \leqslant N \leqslant P$ and $P > K$. Let ϕ_{KP} be a look-ahead operator for (U, L, T, S) and ϕ_{KP*} be a look-ahead operator for (U, L, U^N, R). Then the following statements are equivalent:

1) $(l_1, \cdots, l_M)$ is a globally consistent labeling of $(1, \cdots, M)$ with respect to $\phi_{KP*}R$;

2) $(l_1, \cdots, l_M)$ is a globally consistent labeling of $(1, \cdots, M)$ with respect to R;

3) $(l_1, \cdots, l_M)$ is a (T, S)-consistent labeling of $(1, \cdots, M)$;

4) $(l_1, \cdots, l_M)$ is a $(T, \phi_{KP}S)$-consistent labeling of $(1, \cdots, M)$.

Proof: By Proposition 3, 1) holds if and only if 2) holds; also by Proposition 3, 3) holds if and only if 4) holds. By Proposition 1, 2) holds if and only if 3) holds. Therefore, statements 1)-4) are all equivalent.

V. A Recursive Approach to Applying the ϕ_{KP} Operator

The Haralick *et al.* paper outlined a method for incorporating the relaxation operators into a depth-first search procedure for finding globally consistent labelings. In this section we will discuss the incorporation of the ϕ_{KP} operator into a depth-first search and show that the ϕ_{KP} operator is really a recursive procedure.

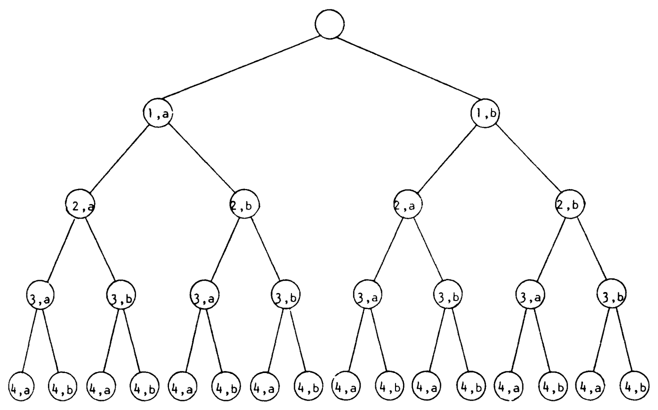

Fig. 5. Tree which must be searched to find a consistent labeling.

Suppose $U = \{1, 2, 3, 4\}$, $L = \{a, b\}$, $T \subseteq U^N$, and $R \subseteq (U \times L)^N$. Fig. 5 shows a tree structure which must be searched to find a consistent labeling of $\{1, 2, 3, 4\}$. At the first step of the search procedure, label a is fixed to unit 1, and R is checked to see if (a) is a consistent labeling of (1) with respect to (T, R). If so, the procedure continues down the leftmost branch of the tree. The next step is to fix label a to unit 2 and check if (a, a) is a consistent labeling of $(1, 2)$ with respect to (T, R). Again, if it is, the procedure continues down the leftmost branch, fixes the label a to unit 3 and checks if (a, a, a) is a consistent labeling of $(1, 2, 3)$ with respect to (T, R). The procedure continues down the left branch of the tree until it reaches a unit-label pair that is inconsistent with the other unit-label pairs already fixed higher in the tree. Then it backs up and tries the next label for the unit of the offending unit-label pair. For instance, if (a, a, a) is not a consistent labeling of $(1, 2, 3)$, the procedure would next fix label b to unit 3 and check if (a, a, b) is a consistent labeling of $(1, 2, 3)$. If it is, the procedure continues down the tree again. If not, since there are no more labels for unit 3, the procedure backs up another level, fixes label b to unit 2 and checks if (a, b) is a consistent labeling of $(1, 2)$. The procedure continues until it either finds a consistent labeling of $(1, 2, 3, 4)$ or backs all the way up to the root of the tree and has no more labels left to try for unit 1. In this case, it fails.

The tree search, without ϕ_{KP}, fixes a label l_n to unit n at level n in the tree and checks if $(l_1, \cdots, l_n)$ is a consistent labeling of $(1, \cdots, n)$ with respect to (T, R). The same tree search, with the ϕ_{KP} operator incorporated, fixes label l_n to unit n at level n, calculates $R_n = R$ restricted to those $2N$-tuples where l_i is the label for unit $i, i = 1, \cdots, n$, and applies the ϕ_{KP} operator to R_n until a fixed point is reached. Suppose the fixed point is reached in m applications of ϕ_{KP}. There are three possible results of applying ϕ_{KP} to the restricted relation R_n:

1) $\phi_{KP}^m R_n = \phi$.

2) $\phi_{KP}^m R_n \neq \phi$ and $\phi_{KP}^m R_n$ is a single valued relation; therefore, the set of labels that are left are a consistent labeling (this is easy to prove).

3) Neither of the above.

In case 1) there are no consistent labelings with respect to (T, R_n). The current path is abandoned, and the procedure backs up. In case 2) a consistent labeling has been found. The procedure can record its answer and either quit or continue looking for more consistent labelings. In case 3) the procedure must search further down the tree.

Example: Let

$$U = \{1, 2, 3, 4, 5\}$$
$$L = \{a, b\}$$
$$T = \{(1, 3), (2, 3), (3, 4), (4, 5)\}$$
$$R = \{(1, a, 3, a), (1, a, 3, b), (1, a, 4, a), (1, a, 4, b), (2, a, 3, a), (2, a, 3, b), (3, a, 4, b), (3, b, 4, a), (4, a, 5, a), (4, b, 5, b)\}$$
$$N = 2, K = 2, P = 4.$$

The search proceeds as follows. Unit 1 is fixed to label a. $R_1 = R$ since label a is the only label for unit 1 that can be found in R. $\phi_{24} R_1 = R_1$, which is neither single-valued and consistent nor null, so the procedure must search further down the tree. Unit 2 is fixed to label a. $R_2 = R_1$ since label a is the only label for unit 2 that can be found in R_1. Again $\phi_{24} R_2 = R_2$ and the search continues. Unit 3 is fixed to label a. The restricted relation R_3 is given by

$$R_3 = \{(1, a, 3, a,), (1, a, 4, a), (1, a, 4, b), (2, a, 3, a), (3, a, 4, b), (4, a, 5, a), (4, b, 5, b)\}.$$

Now $\phi_{24}^1 R_3 = \{(1, a, 3, a,), (1, a, 4, b), (2, a, 3, a), (3, a, 4, b), (4, b, 5, b)\}$ and $\phi_{24}^2 R_3 = \phi_{24}^1 R_3$. Since $\phi_{24}^1 R_3$ is single-valued, (a, a, a, b, b) is a (T, R)-consistent labeling of $(1, 2, 3, 4, 5)$.

If the procedure is going to find all consistent labelings, it now backs up and fixes unit 3 to label b. The restricted relation R_3 becomes

$$R_3 = \{(1, a, 3, b), (1, a, 4, a), (1, a, 4, b), (2, a, 3, b), (3, b, 4, a) (4, a, 5, a), (4, b, 5, b)\}.$$

This time $\phi_{24}^1 R_3 = \{(1, a, 3, b), (1, a, 4, a), (2, a, 3, b), (3, b, 4, a), (4, a, 5, a)\}$ and $\phi_{24}^2 R_3 = \phi_{24}^1 R_3$. Again, $\phi_{24}^1 R_3$ is single-valued. Thus, (a, a, b, a, a) is a second consistent labeling of $(1, 2, 3, 4, 5)$ with respect to (T, R). Now there are no more labels left to try for unit 3. The procedure backs up to level 2 in the tree and there are no more labels left to try for unit 2. The procedure backs up to level 1 in the tree, there are no more labels left to try for unit 1, and the procedure terminates.

Since the search procedure is a tree search, it is most naturally described by a recursive algorithm. The following algorithm, TREESEARCH, performs a depth-first search with the ϕ_{KP} operator incorporated. TREESEARCH expects as input the relations R and T, a unit u, which is the first unit to be fixed to a label, a list LIST of possible labels for u, and the parameters N, K, and P. Instead of fixing units to labels in order of unit number, TREESEARCH chooses some new unit at each level in the tree which has the least number, greater than one, of labels left in the current relation. TREESEARCH calls on three other procedures: REMOVE-ONE-LABEL which removes and returns the first label of a list, LABELING which returns the consistent labeling from a single-valued relation R, and FIXEDϕ which applies the ϕ_{KP} operator to a fixed point. TREESEARCH returns a consistent labeling if it finds one and *false* otherwise.

```
procedure TREESEARCH ($R,T,u$, LIST,$N,K,P$)
do while true
  begin
  if (LIST = null) then return (false);
  $l$ = REMOVE-ONE-LABEL (LIST);
  $R'$ = $R$ restricted to $N$-tuples where unit $u$ has label $l$;
  if $R'$ is single-valued then return (LABELING ($R'$));
  $\phi R'$ = FIXED$\phi$ ($R',T,N,K,P$);
  case
    ($\phi R'$ = null):
      continue;
    ($\phi R'$ is single-valued):
      return (LABELING($\phi R'$));
    ($\phi R'$ is not single-valued):
      begin
        $v$ = a unit such that $v$ has more than one possible
           label left in $\phi R'$ and has the smallest possible
           number of labels of all such units;
        NEWLIST = the list of possible labels for $v$;
        $CL$ = TREESEARCH ($\phi R', T, v$, NEWLIST, $N,K,P$);
        if ($CL$ = false)
        then continue
        else return ($CL$);
      end;
  end case;
  end
end TREESEARCH.
```

The procedure REMOVE-ONE-LABEL and CONSISTENT are dependent on the data structure used to represent T and R in the implemetation. The procedure FIXEDϕ calls on the ϕ_{KP} operator and is given by

```
procedure FIXED$\phi$ ($R,T,N,K,P$);
$R^* = R$;
do while (true)
  begin
    $\phi R = \phi_{KP}(R^*, T,N,K,P)$;
    if ($\phi R = R^*$ or $\phi R$ is single-valued)
    then return ($\phi R$)
    else $R^* = \phi R$
  end
end FIXED$\phi$
```

Now only the ϕ_{KP} operator is left to define. By definition, it must examine each $2N$-tuple in R. For each $2N$-tuple $(u_1, l_1, \cdots, u_N, l_N)$, it must try every combination $j_1, \cdots, j_K$ of $1, \cdots, N$. For each such combination, it must fix u_{j_i} to l_{j_i}, $i = 1, \cdots, K$ and try for every possible $P-K$ units $u'_{K+1}, \cdots, u'_P$ to determine if there is a (T, R)-consistent labeling of all P units $(u_{j_1}, \cdots, u_{j_K}, u'_{K+1}, \cdots, u'_P)$ such that u_{j_i} is fixed to l_{j_i}, $i = 1, \cdots, K$. But this last step is just the labeling problem all over again with P units instead of M and a restricted relation instead of R. Since it is efficient to use ϕ_{KP} on the larger labeling problem, we might expect that using $\phi_{KP}{}^*$ with $P^* < P$ will be efficient for solving the smaller labeling problem. We have not done enough experiments to determine the most efficient P^* and in our algorithms we show $P^* = P - 1$.

The following procedure implements the recursive algorithm for ϕ_{KP}:

```
procedure $\phi_{KP}$ ($R,T,N,K,P$)
if $P < N$ then return ($R$);
$\phi R$ = null;
do for each $2N$-tuple $NT = (u_1, l_1, \cdots, u_N, l_N) \in R$
  begin
    FLAG = true;
    do for each combination $\{j_1, \cdots, j_K\}$ of $\{1, \cdots, N\}$ while (FLAG $\neq$ false)
      begin
        UFIXED = $\{u_{j_1}, \cdots, u_{j_K}\}$;
        LFIXED = $\{l_{j_1}, \cdots, l_{j_K}\}$;
        do for each combination $\{i_1, \cdots, i_{P-K}\}$ of $\{1, \cdots, M\}$-UFIXED
            while (FLAG $\neq$ false)
          begin
            UFREE = $\{u_{i_1}, \cdots, u_{i_{P-K}}\}$;
            $\hat{R} = \{(u_1, l_1, \cdots, u_N, l_N)\} \in R \mid u_i \in$ UFIXED $\cup$ UFREE
                and $u_i \in$ UFIXED, implies $l_i$ is the corresponding
                element of LFIXED, $i = 1, \cdots, K\}$;
            $\hat{T} = \{(u_1, \cdots, u_N) \in T \mid u_i \in$ UFIXED $\cup$ UFREE$\}$;
            if ($\hat{R}$ is not single-valued)
```

```
                then begin
                    $v$ = a unit such that $v$ has more than one possible
                        label left in $R$ and has the smallest number
                        of possible labels of all such units;
                    NEWLIST = the list of all possible labels of $v$;
                    $P^* = P - 1$;
                    FLAG = TREESEARCH ($\hat{R}$, $\hat{T}$, NEWLIST, $N, K, P^*$)
                end
            end (of combination $i_1, \cdots, i_{P-K}$)
        end (of combination $j_1, \cdots, j_K$)
        if (FLAG $\neq$ false) then add $NT$ to $\phi R$
    end (of $N$-tuple $NT$);
    return ($\phi R$)
end $\phi_{KP}$
```

VI. Conclusion

In this first part of a two-part paper we have introduced a model for a general problem called the consistent labeling problem. We have shown that many different problems including puzzles, scene analysis problems, finding automata homomorphisms, and Boolean satisfiability are examples of the consistent labeling problem. We have defined a two-parameter look-ahead operator ϕ_{KP} that removes $2N$-tuples from the unit-label constraint relation R that do not lead to consistent labelings, and we have shown that ϕ_{KP} never removes any $2N$-tuples from R that do lead to consistent labelings. Thus, ϕ_{KP} can help solve a consistent labeling problem by reducing the size of R and thus reducing the size of the tree searched for a consistent labeling. Finally, we have illustrated how to incorporate ϕ_{KP} into a tree search and have shown that the ϕ_{KP} operator can be described by a recursive procedure.

In Part II we will present a number of theoretical results concerning the power of the ϕ_{KP} operator. We will introduce another look-ahead operator ψ_{KP} which is the generalization of several more relaxation operators used by other researchers, discuss its properties, and show its relation to ϕ_{KP}. Finally, we will discuss the complexity of the tree search with look-ahead operators.

References

[1] H. G. Barrow and J. M. Tenenbaum, "MYSYS: A system for reasoning about scenes," Stanford Res. Inst., Menlo Park, CA, SRI AI Tech. Rep. 121, 1976.

[2] M. Clowes, "On seeing things," *Artificial Intelligence*, vol. 2, pp. 79-116, 1971.

[3] S. A. Cook, "The complexity of theorem proving procedures," in *Proc. Annu. ACM Symp. Theory of Computing*, NY, 1971, pp. 151-158.

[4] L. S. Davis, *Shape matching using relaxation techniques*, Computer. Sci. Center, Univ. Maryland, TR-480, Sept. 1976.

[5] J. P. A. Deutsch, "A short cut for certain combinational problems," presented at the British Joint Comput. Conf., 1966.

[6] R. E. Fikes, "REF-ARF: A system for solving problems stated as procedures," *Artificial Intelligence*, vol. 1, pp. 27-120, 1970.

[7] E. C. Freuder, "Synthesizing constraint expressions," *Commun. Ass. Comput. Mach.*, vol. 21, no. 11, 1978.

[8] J. Gasching, "A constraint satisfaction method for conference making," presented at the 12th Annu. Allerton Conf. Circuit and System Theory, Univ. Illinois, 1974.

[9] A. Ginzberg, *Algebraic Theory of Automata.* New York: Academic, 1968.

[10] A. Guzman, "Decomposition of a visual scene into three-dimensional bodies," in *Automatic Interpretation and Classification of Images*, Graselli, Ed. New York: Academic, 1969, pp. 243-276.

[11] A. Hanson and E. Riseman, "Pattern-directed boundary formation via relaxation," presented at the *Machine Vision Workshop*, Univ. Massachusetts, Amherst, MA, June 1-3, 1977.

[12] R. M. Haralick, "The characterization of binary relation homomorphisms," *General Syst. J.*, vol. 4, pp. 113-121, 1978.

[13] ——, "Scene analysis, homomorphisms, and arrangements," in *Machine Vision*, Hanson and Reisman, Eds. New York: Academic, 1978.

[14] R. M. Haralick, L. Davis, A. Rosenfeld, and D. Milgram, "Reduction operations for constraint satisfaction," *Inform. Sci.*, vol. 14, pp. 199-219, 1978.

[15] R. M. Haralick and J. Kartus, "Arrangements, homomorphisms, and discrete relaxation," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, pp. 600-612, Aug. 1978.

[16] F. Harary, *Graph Theory.* Reading, MA: Addision-Wesley, 1969.

[17] D. Huffman, "Impossible objects as nonsense sentences," *Machine Intelligence*, B. Meltz and D. Michie, Ed., vol. 5. Edinburgh, Scotland: Edinburgh University Press, 1971, pp. 295-323.

[18] R. Kowalski, "A proof procedure using connection graphs," *J. Ass. Comput. Mach.*, vol. 22, pp. 572-595, Oct. 1975.

[19] A. Mackworth, "Consistency in networks of relations," *Artificial Intelligence*, vol. 8, pp. 99-118. 1977.

[20] U. Montanari, "Networks of constraints: Fundamental properties and applications to picture processing," *Inform. Sci.*, vol. 7, pp. 95-132, 1974.

[21] A. Rosenfeld, "Networks of automata: Some applications," *IEEE Trans. Syst., Man, Cybern.*, vol. 5, pp. 380-383, 1975.

[22] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene labeling by relaxation operations," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, pp. 420-433, June 1976.

[23] A. Rosenfeld and A. Kak, *Digital Picture Processing.* New York: Academic, 1976.

[24] J. R. Ullman, "An algorithm for subgraph homomorphisms," *J. Ass. Comput. Mach.*, vol. 23, pp. 31-42, Jan. 1976.

[24a] ——, "Associating parts of patterns," *Inform. Contr.*, vol. 9, pp. 583-601, 1966.

[25] G. Vanderbrug, "Experiments in iterative enhancement of linear features," in *LARS Symp. Proc. Machine Process. Remote Sensed Data*, Purdue Univ., Lafayette, IN, June 29-July 1, 1976.

[26] D. L. Waltz, "Generating semantic descriptions from drawings of scenes with shadows," MIT Tech. Rep. A1271, Nov. 1972.

[27] E. G. Whitehead, Jr., "Combinatorial algorithm," Courant Inst. Mathematical Sciences, New York Univ., NY, 1972.

[28] S. W. Zucker and R. A. Hummel, "Computing the shape of dot

clusters, I: Labeling edge, interior, and noise points," Univ. Maryland, College Park, MD, Tech. Rep. TR-543, May 1977.
[29] S. W. Zucker, R. A. Hummel, and A. Rosenfeld, "An application of relaxation labeling to line and curve enhancement," *IEEE Trans. Comput.*, vol. C-26, pp. 394–403, Apr. 1977.

Robert M. Haralick (S'62–S'67–M'69–SM'76) was born in Brooklyn, NY, on September 30, 1943. He received the B.S. degree and the Ph.D. degree from the University of Kansas, Lawrence, in 1966 and 1969, respectively.

He has worked with Autonetics and IBM. In 1965 he worked for the Center for Research, University of Kansas, as a Research Engineer, and in 1969 he joined the faculty of the Electrical Engineering Department where he served as a Professor from 1975 to 1978. In 1979 he joined the faculty of the Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, where he is now a Professor. He has done research in pattern recognition, multiimage processing, remote sensing, texture analysis data compression, clustering, artificial intelligence, and general systems theory. He has been responsible for the development of KANDIDATS (Kansas Digital Image Data System) a multiimage processing package which runs on a minicomputer system.

Dr. Haralick is a member of the Association for Computing Machinery, Sigma Xi, Pattern Recognition Society, and Society for General Systems Research.

Linda G. Shapiro was born in Chicago, IL, in 1949. She received the B.S. degree in mathematics from the University of Illinois, Urbana, in 1970 and the M.S. and Ph.D. degrees in computer science from the University of Iowa, Iowa City, in 1972 and 1974, respectively.

She was an Assistant Professor of Computer Science at Kansas State University, Manhattan, from 1974 to 1978. She is now an Assistant Professor of Computer Science at Virginia Polytechnic Institute and State University, Blacksburg. Her research interests include pattern recognition, scene analysis, computer graphics, data structures, and programming languages. She is currently working on an undergraduate textbook on data structures with R. Baron.

Dr. Shapiro is a member of the Association for Computing Machinery, the IEEE Computer Society, and the Pattern Recognition Society.

Experiments in Text Recognition with the Modified Viterbi Algorithm

RAJJAN SHINGHAL, MEMBER, IEEE, AND GODFRIED T. TOUSSAINT, MEMBER, IEEE

Abstract—**In this paper a modification of the Viterbi algorithm is formally described, and a measure of its complexity is derived. The modified algorithm uses a heuristic to limit the search through a directed graph or trellis. The effectiveness of the algorithm is investigated via exhaustive experimentation on an input of machine-printed text. The algorithm assumes language to be a Markov chain and uses transition probabilities between characters. The results empirically answer the long-standing question of what is the benefit, if any, of using transition probabilities that depend on the length of a word and their position in it.**

Index Terms—**Computational complexity, contextual information, cost, feature vector, *N*-gram probabilities, probability of misclassification, text recognition.**

Manuscript received March 17, 1978; revised September 14, 1978. This research was supported by the National Research Council of Canada under Grant NRC-A9293.

R. Shinghal is with the Department of Computer Science, Concordia University, Montreal, P.Q., Canada.
G. T. Toussaint is with the School of Computer Science, McGill University, Montreal, P.Q., Canada.

I. Introduction

ALGORITHMS using contextual information in spoken or printed text recognition fall into two main categories: dictionary look-up methods and Markov methods [20]. The dictionary look-up methods require that the words to be classified or recognized exist in a previously compiled dictionary available to the classifier [3], [4], [6], [8], [17], [18], [23], [24], and they represent a top-down philosophy in approaching the problem. The Markov methods [1], [5], [7], [9], [11], [15], [16], [18], [19] invoke the assumption that the language is a Markov source and use transition probabilities, thus representing a bottom-up approach.

It should be noted that other bottom-up approaches are possible; for example, using syntactic rules [12]. At the moment it appears difficult to determine whether the semantic-syntactic approach or the purely statistical approach is sounder [13]. It is of interest to determine how much can be gained with a purely statistical bottom-up approach such as the