# A LOGIC MODEL FOR CONSTRAINT PROPAGATION

Larry S. Davis

Computer Sciences Department
University of Texas
Austin, Texas

TR-137 February 1980

This research was supported in part by the National Science Foundation using funds derived from NSF Grant ENG-7904037.

## ABSTRACT

Relaxation algorithms are widely used in computer image analysis. This paper shows how discrete relaxation can be modeled as an inference process in predicate calculus. The advantages of a predicate calculus representation over a relational representation for constraints are discussed.

## 1. Introduction

During the past few years a class of algorithms, referred to as "relaxation" algorithms, have been used to solve a variety of computer vision problems. Relaxation algorithms attempt to reduce the ambiguity inherent in any context-free interpretation of picture segments by, effectively, introducing context sensitive knowledge, represented as binary constraints, and then iteratively applying those constraints to reduce the ambiguity of the initial context free interpretation. Davis and Rosenfeld [1] contain an overview of such processes and examples of their applications to specific computer vision problems.

There are two types of relaxation processes which have been investigated. The first is a "discrete" procedure where the context-free interpretation assigns a set of labels to each picture part, and the relaxation process reduces the size of the sets by eliminating interpretations which fail to satisfy the constraints (modeled by binary or higher-order relations). Discrete relaxation is a constraint propagation process. The second relaxation process is a "continuous" procedure where the context free interpretation associates a likelihood, or probability, with each label for each picture part, and the relaxation process attempts, e.g., to reduce the entropy of the probability functions at each picture part, while at the same time maintaining consistency between probabilities, as defined by the constraints. The constraints are modeled by compatibility functions, rather than as discrete binary relations.

This paper is concerned with discrete relaxation. In particular, we will show how discrete constraint propagation can be modeled as an inference process in logic. A discussion of the advantages of such a representation is deferred until Section 4. Section 2 presents discrete relaxation based on binary relations as a knowledge representation for constraints. The discussion in Section 2 closely follows the development in Rosenfeld et al, [2] and Zucker [3]. Section 3 presents a logic model for constraint propagation, based on a predicate calculus knowledge representation for constraints.

## 2. Discrete Relaxation

In this section we will describe the model proposed by Rosenfeld et al [2] for labeling a set of objects, or picture parts, subject to a set of binary constraints.

Let $A = \{a_1, \ldots, a_n\}$ be the set of objects to be labeled. A may be a set of image segments, image pixels, etc. Let $D = \{d_1, \ldots, d_m\}$ be a set of labels, or descriptors, which can be used to describe the $a_i$. A labeling, L, is a mapping:

$$L: A \rightarrow 2^D$$

which assigns a subset of labels to each object. Let $L_i$ denote $L(a_i)$.

Constraints specifying the allowable pairs of labels $(d,d')$ which can be simultaneously assigned to a given pair of objects $(a_i,a_j)$ are represented by a binary relation $R_{ij}$:

$$R_{ij} \subseteq D \times D$$

A labeling, L, is called consistent if, for all i,j:

$$d \in L_i \implies \text{there is a } d' \in L_j \text{ with } (d,d') \in R_{ij}$$

Rosenfeld et al [2] show that for any labeling, L, and constraints, $R_{ij}$, there is a labeling $L^* \subseteq L$ such that:

1) $L^*$ is consistent, and

2) for any consistent $L' \subseteq L$, $L' \subseteq L^*$.

i.e., $L^*$ is a greatest consistent labeling. Note that $L' \subseteq L$ means that $L_1' \subseteq L_1, \ldots, L_n' \subseteq L_n$.

A parallel, iterative algorithm exists for computing $L^*$ from L and it is called discrete relaxation. It is defined as follows. (Note: $L^r$ denotes the labeling computed after the $r^{th}$ iteration of the algorithm.)

```
1) $L^0 = L$, $r = 0$.
2) For $i = 1,\dots,n$ do
   $L'_i = L_i^r - \{d: d \in L_i^r$ and for some $j$, there is no $d'$ with
           a) $d' \in L_j^r$, and b) $(d,d') \in R_{ij}\}$
3) if $L'_i = L_i^r$, $i=1,\dots,n$, then set $L^* = \{L'_1,\dots,L'_n\}$ and stop.
4) $L^{r+1} = \{L'_1,\dots,L'_n\}$; $r=r+1$; go to (1).
```

Discrete relaxation can be illustrated using the following example from Rosenfeld et al [2]. The task is to label the sides of a triangle which has been (partially) cut out of a piece of paper. If the triangle has been completely cut out, then it can lie either entirely in front of, or behind, the original plane of the paper. Each edge of the triangle can have one of four possible interpretations:

1) convex edge: the edge may be attached to its surrounding, and the triangle is bent out of the plane of the paper along the edge, towards the viewer. Such an edge will be labeled with $\rightarrow$.

2) concave edge: similar to (1), except that the triangle is bent away from the viewer. The label is $\leftarrow$.

3) a positive occluding edge: the edge has been cut, and is in front of the plane of the paper. The label is $+$.

4) a negative occluding edge: the edge has been cut, and lies behind the plane of the paper. The label is $-$ .

Figure 1 contains all possible physically realizable labelings of such triangles. The constraint relation, R, is the same for all object pairs and can be read directly from Figure 1.

$$R_{ij} = \{(+,+), (-,-), (+,\rightarrow), (\rightarrow,+), (-,\leftarrow), (\leftarrow,-)\}$$

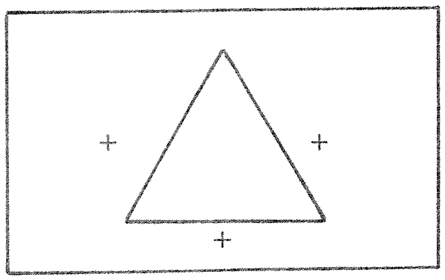
To illustrate the application of the relaxation algorithm, consider the labeling shown in Figure 2. Here, $L(a_3) = L^0{}_3 = \{+\}$, $L^0(a_2) = L^0{}_2 = \{+,-,\leftarrow\}$, $L(a_1) = L^0{}_1 = \{\rightarrow,+\}$. Then in step (2) of the algorithm:

1) $L'_3 = L^0{}_3$ since + at $a_3$ is consistent with + at $a_1$ and $a_2$.

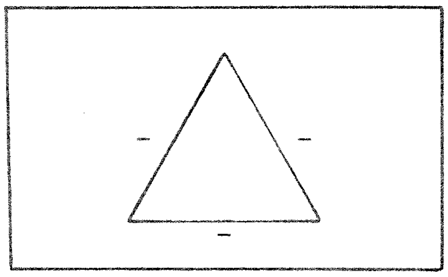
2) $L'_2 = \{+\}$ since $-$ at $a_2$ is not consistent with + at $a_3$ and $\leftarrow$ at $a_2$ is not consistent with + at $a_3$.

3) $L'_1 = L_1$ since both $\rightarrow$ and + at $a_1$ are consistent with + at $a_2$ and $a_3$,

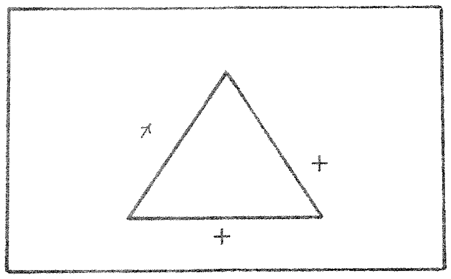
For this example, the algorithm terminates after one iteration, and $L^* = L^1 = \{L'_1, L'_2, L'_3\}$.



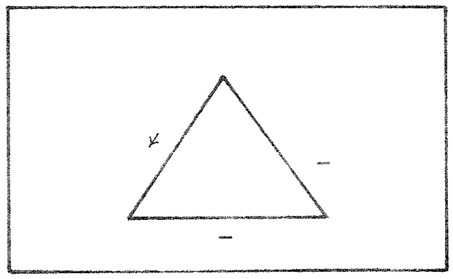
a) triangle "floats" in front of paper



b) triangle behind paper.



c) triangle bent out towards viewer



d) triangle bent away from viewer

Figure 1 - All physically realizable triangle labelings.

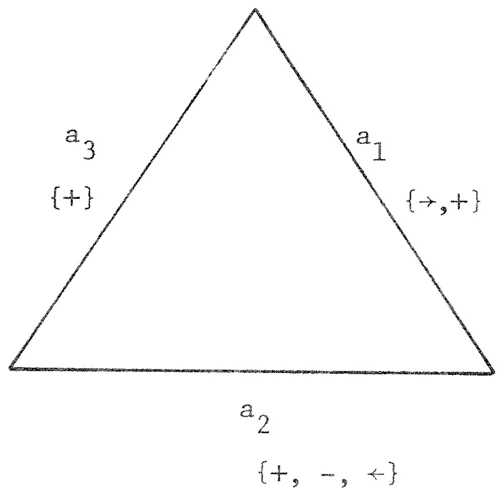


Figure 2 - An initial labeling.

## 3. A Predicate Calculus Model for Constraint Propagation

In this section we will describe a logic model for constraint representation and propagation. Again, let A be the set of objects to be labeled, and let D be the set of possible labels.

In Section 2, a labeling was a function. Here, a labeling, L, is a set of n formulae $\{L_1,\ldots,L_n\}$. Each $L_i$ is of the form:

$$[d_{i1}(a_i) \vee d_{i2}(a_i) \vee\ldots\vee d_{ip_i}(a_i)] \wedge \\ \sim d_{ip_{i+1}}(a_i) \wedge\ldots\wedge \sim d_{iq_i}(a_i) \tag{1}$$

Here, $p_i + q_i = m$, and each $d_i$ is mentioned exactly once in $L_i$. Each $L_i$ can be viewed as being composed of one positive clause and $q_i$ negative unit clauses. Each of the $q_i$ negative unit clauses represents the assertion that $a_i$ is not some specific label. The logic counterpart of discrete relaxation will manipulate this set of formulae by adding more negative unit clauses, and decreasing the size of the positive clauses. The positive clauses which remain when the logic algorithm terminates will correspond to the $L^*$ of discrete relaxation.

The binary constraints employed by the discrete relaxation are modeled as a set of Horn clauses [4]. We will introduce R as a binary relation indicating that two objects are related (e.g., they might be adjacent). It is not absolutely necessary to introduce such a predicate, but it makes what follows somewhat more intuitive. In a particular application several such predicates might be necessary (e.g., it might be necessary to distinguish between "above" and "inside").

For each predicate, R, there are m constraints, $C_1,\ldots,C_m$, one for each label in D. The form of $C_i$ is:

$$\forall j[d_i(a_j) \dashrightarrow \forall k[\ R(a_j,a_k) \dashrightarrow$$

$$d_{j_1}(a_k) \vee\ldots\vee\ d_{j_i}(a_k)]$$

For the purposes of applying the constraints, it is convenient to use the contrapositives of the expression enclosed in [ ]. Notice that when this is done, the inner universal quantifier becomes an existential quantifier. Doing this, and eliminating the universal quantifier and the second implication, we obtain:

$$\sim d_i(a_j) \leftarrow\!\!-\ \exists k[R(a_j,a_k) \wedge \sim d_{j_1}(a_k)\ \ \wedge\ldots\wedge\ \sim d_{j_i}(a_k)]$$

Finally, a <u>relation model</u>, $\mathcal{R}$, is a set of unit clauses of the form $R(a_i,a_j)$, and specifies which pairs of objects are in the relation R. Intuitively, we include $R(a_i,a_j)$ in the relation model if, in the relational model, $R_{ij} \subset D\times D$ - i.e., if some label on $a_i$ constrains the possible labels on $a_j$.

The constraint propagation mechanism in Section 2 can be modeled as in inference process. Consider the constraint:

$$\sim d_i(a_j) \leftarrow\!\!-\ \exists k[R(a_j,a_k) \wedge \sim d_{j_1}(a_k)\ \ \wedge\ldots\wedge\ \ \sim d_{j_i}(a_k)]$$

If, for some j and k:

1) $\sim d_i(a_j) \notin L$,

2) $R(a_j, a_k) \in \mathcal{R}$, and

3) for $r = j_1, \ldots, j_i$, $\sim d_r(a_k) \in L$

then $\sim d_i(a_j)$ can be inferred and added to L, and $d_i(a_j)$ can be removed from the positive clause in L corresponding to $a_j$. When no further $\sim d_i(a_j)$ can be inferred, then the resulting labeling is equivalent to the $L^*$ labeling which the discrete relaxation process computes.

As an example of how the logic process operates, consider the triangle example presented in Section 2. For this example, $L = \{\sim(-(a_3)),$ $\sim(\rightarrow(a_3)), \sim(\leftarrow(a_3)), \sim(\leftarrow(a_1)), \sim(-(a_1)), \sim(\rightarrow(a_2))\}$, plus the corresponding positive clauses (which we will ignore in the example); $R = \{R(a_1,a_2),$ $R(a_2,a_1), R(a_2,a_3), R(a_3,a_2), R(a_1,a_3)\ R(a_3,a_1)\}$ and $C = \{C_+, C_-, C_\rightarrow, C_\leftarrow\}$ where

1) $C_+ = \sim +(a_j) \longleftarrow \exists k[R(a_j,a_k) \wedge \sim +(a_k) \wedge \sim \rightarrow (a_k)]$

2) $C_- = \sim -(a_j) \longleftarrow \exists k[R(a_j,a_k) \wedge \sim -(a_k) \wedge \sim \leftarrow (a_k)]$

3) $C_\rightarrow = \sim \rightarrow(a_j) \longleftarrow \exists k[R(a_j,a_k) \wedge \sim +(a_k)]$

4) $C_\leftarrow = \sim \leftarrow(a_j) \longleftarrow \exists k[R(a_j,a_k) \wedge \sim -(a_k)]$

The constraint propagation process can, as in Section 2, be viewed as an iterative process which, at each iteration, considers all unit clauses $d_i(a_j)$ such that $\sim d_i(a_j) \notin L$, and attempts to derive $\sim d_i(a_j)$ from L, C and $R$. If successful, $\sim d_i(a_j)$ is added to L; when no further negative units can be added to L, the process terminates.

For the example, at iteration 1:

a) $\sim -(a_2)$ can be inferred by binding k to 3 and j to 2 in $C_-$.

b) $\sim \leftarrow(a_2)$ can be inferred by binding j to 2 and k to 3 in $C_{\leftarrow}$.

No other clauses can be inferred at this iteration, and, in fact, no new inferences can be made at the next iteration, so the process terminates with the same labeling computed in Section 2 by the discrete relaxation process.

## 4. Advantages of the Logic Model

The logic model for discrete relaxation introduced in Section 3 has several distinct advantages over the relational model presented in Section 2. First, logic is a far more natural representation for such constraints, since it is closer to the natural language statements which originally specify the constraints. Since a significant amount of research has been devoted to translating natural language into logic, "friendly" systems could be constructed for specifying a constraint propagation model for a particular image understanding task.

A related point is that the logic representation provides a more general point of view about the role of constraints in picture interpretation. The relational model simply requires that an interpretation for a picture has a consistent interpretation at each neighboring picture part. The logical model can certainly represent that requirement, but, more generally, it enables us to specify a set of locally verifiable conditions that must hold for a picture part to be labelled with a particular label. That is, we are not restricted to constraints of the form described in Section 3, but any formula of the form:

$$\sim d_i(a_j) \leftarrow F$$

can be used as a constraint.

In practice, restrictions are placed on the form of F both to guarantee that the inference process is computationally efficient and to limit the information flow between different objects. Ordinarily, a relaxation process is regarded as operating on a graph containing nodes, which correspond to the objects being labeled and edges connecting objects whose labellings constrain one another. The edges may be labeled to distinguish one class of neighbors from others (e.g., adjacent, above, etc.). Mose generally, at each iteration each node "reads" the state of the entire graph and updates its label set based on the distribution of labels around the graph. In order to limit the amount of information flowing into each node, and also to simplify the computations performed at each node, most relaxation algorithms constrain the flow of information to a node to include _only_ the label sets at each neighboring node and the labels on the edges to those neighbors. However, most relaxation algorithms also place severe limitations on the analysis of that information - e.g., the discrete relaxation algorithm outlined in Section 2. The logic formulation enables us to generalize this so as to allow for an arbitrary analysis of the information flowing into each node. In summary, then, F may, in theory, be any formula referencing the labeling at any node and the relationships between any nodes but in practice we constrain F to reference only the labelings of adjacent nodes and the relationships to those adjacent nodes.

Secondly, there are classes of constraints whose representation as relations would require very high-order relations, but whose representation in logic is very efficient. For example, consider the constraint : If a picture part is labelled with a $d_1$, then some adjacent picture part must be labelled with a $d_2$. This constraint cannot be represented using binary relations. In fact, the order of this constraint is not well-defined, but if a picture is decomposed into n parts, then for the analysis of that picture the constraint may require an (n-1)-ary relation, because the adjacency graph for the picture might be a star - i.e., suppose that $d_1$ is a possible guess for $a_1$, and that all of $a_2,\ldots,a_n$ are adjacent to $a_1$. Then, to check that at least one of $a_2,\ldots a_n$ has label $d_2$ associated with it, the relaxation procedures introduced by Rosenfeld et al must be generalized to n-ary relations (see Haralick et al [5]) and the n-ary relation must include all n-tuples of the form:

$$(d_1,*,\ldots,*,d_2,*,\ldots *)$$

where $d_2$ can appear in any position from 2 to n, and the * can represent any label. If there are m possible labels, then just to represent this single constraint requires a relation of size $m^n$, which is clearly unacceptable.

In logic, this constraint is simply represented as:

$$\sim d_1(a_i) \longleftarrow \forall_j[\sim d_2(a_j) \vee \sim \mathrm{ADJACENT}(a_i,a_j)]$$

Applying such a constraint can be accomplished as simply as applying the constraints discussed in Section 3.

A third point involves the use of negative context in constraint propagation. For example, consider the constraint:

A picture part is a $d_1$ only if no adjacent picture part is a $d_2$.

This type of negative context cannot be easily represented in a relational framework. Furthermore, the obvious generalization of discrete relaxation to include such constraints is to delete $d_1$ from any object a which is adjacent to any other object a' having possible label $d_2$; but this leads to obviously undesirable results, since it is possible that $d_2$ will subsequently be eliminated from a', thus removing the negative context which caused the removal of $d_1$ from a. Notice, that if $d_2$ were the <u>only</u> possible label for a', then it would have been correct to delete $d_1$ from a.

One possible solution to this problem is to simply not apply such constraints using discrete relaxation. But this does not address the more fundamental question of what classes of constraints should not be applied.

When the question is posed in the logic framework, then its answer becomes evident. There are no constraints which, theoretically, cannot or should not be applied. However, some constraints, such as the negative context constraint, may not immediately be applicable. Consider the logic representation of the above constraint:

$$\forall_i \; d_1(a_i) \dashrightarrow \; \forall_j [\sim\text{ADJACENT}(a_i,a_j) \vee \sim d_2(a_j)]].$$

which can be rewritten as:

$$\sim d_1(a_i) \leftarrow\!\!- \; \exists_j [\text{ADJACENT}(a_i,a_j) \wedge d_2(a_j)]$$

Notice the important difference between this constraint and the ones discussed in Section 3; namely that there is now a positive unit labeling clause ($d_2(a_j)$) to the right of the implication sign. In order for this constraint to be applicable, the formulae for $a_j$ in the labeling, L, must contain that positive unit clause - i.e., all other possible labels for $a_j$ must have been eliminated by the constraint propagation process.

One last point involves the design of hierarchical relaxation processes. Such processes have been discussed by Davis and Henderson [6] for shape recognition using discrete constraints, by Hayes for handwriting recognition using fuzzy constraints [7], and by Zucker for linear feature enhancement using fuzzy constraints [8]. An important aspect of such hierarchical systems is that not only are there constraints between labels at each level in the hierarchy, but there are a wide class of implicit constraints which are determined by the mapping from one level of the system to the next. It is important that these two classes of constraints be consistent with one another. In both Davis and Henderson and in Hayes this is accomplished by actually compiling the constraints between labels at a fixed level from a declarative representation of the mapping functions from one level to the next. However, neither system is capable of accepting in any uniform way, external knowledge in the form of additional constraints which are perhaps not implicit in the between level mapping model. For example, in [6], discrete constraints are compiled from a grammatical model describing the shape at airplanes in terms of adjacency of airplane pieces, geometric properties of pieces and geometric relationships between pieces. If external knowledge were available concerning the shape of an airplane, e.g., that the wings make an angle of 35° with the fuselage, there was no simple way to integrate it into the analysis (even though the grammar knew about axes of wings and fuselages) or to check that the knowledge was consistent with the current grammatical model. A hierarchical relaxation system which is based on a logic model of constraints would be able to accept such knowledge, and to check the consistency of that knowledge with its current knowledge base of constraints.

## REFERENCES

1. L. Davis and A. Rosenfeld, "Cooperative processes in low-level vision". Univ. of Texas Computer Sciences Dept. TR-123, Jan., 1980.

2. A. Rosenfeld, R. Hummel and S. Zucker, "Scene labeling by relaxation operations", IEEE Trans. on Systems, Man and Cybernetics, 6, 1976, pp. 420-433.

3. Zucker, S., "Relaxation labeling and the reduction of local ambiguities", in C. H. Chen (ed.), Pattern Recognition and Artificial Intelligence, Academic Press, N.Y., 1977.

4. A. Deliyanni and R. Kowalski, "Logic and semantic networks", Artificial Intelligence.

5. R. Haralick, L. Davis, A. Rosenfeld and D. Milgram, "Reduction Operators for constraint satisfaction", Information Sciences, 14, 1978, pp. 199-219.

6. L. Davis and T. Henderson, "Hierarchical constraint processes for shape analysis", Univ. of Texas Computer Sciences Dept. TR-115, Nov., 1979.

7. K. Hayes, "Reading handwritten words using hierarchical relaxation", Univ. of Maryland Computer Science TR-783, July, 1979.

8. S. Zucker, "Vertical and horizontal processes in low-level vision", in E. Riseman and A. Hanson (eds.), Computer Vision Systems, Academic Press, N.Y., 1978.