Arc Consistency for Factorable Relations

Mark W. Perlin

April, 1991

CMU-CS-91-128 3

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

An optimal arc consistency algorithm AC-4 was given by Mohr and Henderson [8]. AC-4 has cost $O(ea^2)$, and cost (na^2) for scene labelling. Although their algorithm is indeed optimal, under certain conditions a constraint satisfaction problem can be transformed into a less complex problem. In this paper, we present conditions and mechanisms for such transformations, and show how to factor relations into more manageable components. We describe how factorization can reduce AC-4's cost to O(ea), and apply this result to RETE match. Further, with our factorization, the cost of scene labelling is reduced to O(na).

This research was sponsored by the National Library of Medicine of the National Institutes of Health under Contract 5R29 LM04707-04, and by the

CR Classification

E.1	Graph Data Structures
F.2.2	Analysis of Computation on Discrete Structures
I.2.4	Knowledge Representation
I.2.8	Constraint Satisfaction
I.2.10	Vision and Scene Understanding

Table of Contents

1. Introduction

There are search problems that entail the generation and testing of the cartesian product of n sets. Following [8], let

$N = \{i, j, \ldots\}$ be the set of variables, $|N| = n$,

$A = \{b, c, \ldots\}$ be the possible values of the variables, $|A| = a$,

ib be the binding of variable i to value b,

R_i be a unary relation, where ib is admissible if $R_i(b)$,

$A_i = \{b \in A \mid R_i(b)\}$, be the i^{th} set following unary filtering,

R_{ij} be a binary relation, where (ib,jc) is admissible if $R_{ij}(b,c)$,

$|R_{ij}| = r_{ij}$, the number of pairs in R_{ij},

E be the binary relation $\{(i,j) \in N \times N \mid R_{ij}$ is not trivially true $\}$, $|E| = e$.

The binary *constraint satisfaction problem* (CSP) is to find all n-tuples in $\prod_N A_i$ which satisfy the binary relations.

Suppose a partial solution k-tuple, k<n, is not admissible. Then any extension of this k-tuple is not admissible. *Backtrack control* is a strategy that reduces search by eliminating these inadmissible extensions from the cartesian product computation.

Since CSP is NP-complete, additional preprocessing can be helpful [6], such as decreasing the initial size of each A_i set prior to forming the cartesian product. One mechanism for this filtering is *arc consistency*, which examines the R_{ij} to eliminate any $b \in A_i$ which lacks support in an A_j. Arc consistency is used, for example, in machine vision problems [17].

Waltz's original arc consistency algorithm for scene labelling [17] was based on the successive consideration of each variable's value set A_i. This had a computational complexity of $O(ena^3)$. Mackworth and Freuder [7] introduced algorithm AC-3, which used the edges in E, rather than the variables in N, to guide the filtering. This reduced the complexity to $O(ea^3)$. Mohr and Henderson [8] introduced AC-4, which used the edges between values (instead of variables) to direct the filtering, further reducing the cost to $O(ea^2)$. Since scene labelling employs a planar graph, $O(e) = O(n)$, and AC-4's complexity for scene labelling is $O(na^2)$.

In building a Waltz scene labelling application within a graphical user interface methodology [10], we implemented an O(na) arc consistency algorithm. In this paper, we first motivate our algorithm by considering the graph of supporting relations. We present AC-5, a version of AC-4 that explicates the underlying graph. We then examine how, under certain circumstances, edge relations can be *factorable*. For some problems, this can reduce the $O(ea^2)$ cost to O(ea). We apply this factorization to scene labelling, obtaining an O(na) cost, and also describe how our arc consistency method can be applied to RETE matching [3].

2. Support for Values

We write the set images under a binary relation B as:

$$B(x,*) = \{ y \mid B(x,y) \},$$
$$B(*,y) = \{ x \mid B(x,y) \}.$$

After filtering with the unary relations, we have the sets A_i. The relations R_{ij} are between the values in A_i and A_j. We construct the union of the R_{ij}, forming the relation R on values, defined for $b \in A_i$ and $c \in A_j$ as

$$R(ib,jc) \Leftrightarrow R_{ij}(b,c).$$

If it is not the case that $R_{ij}(b,c)$, then no n-tuple t with t(i)=b and t(j)=c is admissible. Identically, R(ib,jc) is necessary for a tuple t with t(i)=b and t(j)=c to be admissible.

Suppose that for no $c \in A_j$, R(ib,jc). Then no tuple t with t(i)=b will be admissible. Therefore, ib can be removed from A_i, and not affect the CSP. This can be used as a filtering strategy [17] for reducing the A_i. We define

$$\textit{supported-p}(ib) \Leftrightarrow \forall j \in E(i,*), \quad A_j \cap R(ib,*) \neq \varnothing.$$

When *supported-p*(ib) is false, value ib can be removed from A_i, and ib's relations removed from R.

Now, for efficient implementation, we view the set of values $\cup A_i$ as the nodes of a graph G, and the relation R as the links between them. Ordinarily in arc consistency algorithms [8], a graph G_0 is employed that uses the variables {i} as node set, and E as edge set. Our G, however, refines G_0 by using values instead of variables, linking the values in $A_i \times A_j$ according to R_{ij}.

Our algorithm AC-5 for arc consistency is given in Figure 1. It roughly replicates algorithm AC-4 [8]. However, instead of using counters, sets, and flags, AC-5 makes explicit use of G. Specifically, in our formulation, AC-4's

- Counter[(i,j),b] = # $A_j \cap R(ib,*)$,
- set $S_{jc} = R(*,jc)$,
- flag $M(i,b) = 0$, if $b \in A_i$, and 1 otherwise.

Our unit of complexity is the use of a node or link in G.

Step 1. INITIALIZE($\{A_i\}$, R)

1 Construct the nodes and links of the graph
    from the values in $\cup A_i$ and the relation R.
2 $\forall$ b $\in \cup A_i$
3 $\forall$ j $\in$ E(i,*)
4 **IF** $A_j \cap R(ib,*) = \emptyset$,
5 ENQUEUE-UNSUPPORTED(ib)

Step 2. FILTERING UNSUPPORTED VALUES

6 **WHILE** (jc $\leftarrow$ DEQUEUE-UNSUPPORTED())
7 $\forall$ ib $\in$ R(*,jc)
8 **IF** $A_j \cap R(ib,*) = \emptyset$,
9 ENQUEUE-UNSUPPORTED(ib)
10 SPLICE-OUT(jc)

Procedure for removing a value node and its relation links.

SPLICE-OUT(jc)
  $A_j \leftarrow A_j$ - {jc}
  $\forall$ ib $\in$ R(*,jc)
    $R_{ij} \leftarrow R_{ij}$ - (ib,jc)

Figure 1. The arc consistency algorithm AC-5. Its time and space cost is O(|G|).

In Step 1, the cost of initialization is proportional to the size of graph G, or O(|G|). Line 1, constructing G, has cost $O(na+\Sigma_E r_{ij})$, the number of nodes and links; this is precisely O(|G|). Line 2 iterates over the nodes, and line 3 iterates over the outdegree of each node, i.e., the links. This, again, has cost O(|G|). Step 4 performs the test

$A_j \cap R(ib,*) = \varnothing$,

which can be done in constant time by checking whether the set (e.g., a list) of links from value ib to variable j is empty.

When a node ib is found to violate *supported-p*(ib), it is placed on the control queue. The queue is maintained as a set (e.g., by marking deleted nodes). ENQUEUE-UNSUPPORTED(ib) and DEQUEUE-UNSUPPORTED() can be each done in constant time (e.g., by using a stack representation).

In Step 2, the cost of arc consistency filtering is also O(|G|). Step 6's WHILE loop considers each node jc at most once. Line 7 then examines all the links emanating from jc. These links can occur in this way at most once, since jc is to be deleted from the graph. Therefore, the constant time operation in Steps 8 and 9 can occur no more than |G| times. By similar counting, the SPLICE-OUT operation in line 10 can occur at most once for each node and link in the graph.

The total space cost is also O(|G|). The only two data structures are G, and the queue. G's representation is clearly O(|G|). The queue size is bounded by the number of nodes, hence by |G|.

3. Factoring Relations

The cost of our AC-5 algorithm (and the equivalent AC-4) is O(|G|), the size of the graph comprised of variable values and their relations. Further, as shown in [8], this is a minimal algorithm. How, then, is it possible to decrease the complexity of arc consistency?

If relation B between two sets is *factorable*, then B may be rewritten as the product P×Q, where P and Q are two new relations. In some cases, P and Q are sparser than B. If

$|P| + |Q| < |B|$,

then the factorization will reduce the total number of links connecting the two sets. Such factorizations appear in other divide-and-conquer [5] algorithms. For example, the Fast Fourier Transform works by factoring a matrix with n^2 nonzero entries into log(n) matrices, each having O(n) nonzero entries [1].

The relations R_{ij} used in arc consistency may be factorable in this way. Factorization introduces new variables, and extends the graph G to a new graph G'. Importantly, our arc consistency algorithm AC-5 (or AC-4) can operate on this new graph G' *without modification*. Since the complexity of arc consistency is bounded by the number of links in G, if G' has fewer links than G, the factorization may reduce the execution cost of AC-5.

In general, there may be no useful factorization of G's link relations. However, for certain key situations, such as scene labelling and RETE match, the relations *are* factorable. We now develop two general classes of factorizations.

3.1. Using Tuples as Values

In many situations, the values used as variable bindings are not atomic. Rather, they are actually formed as *tuples* of slot-values. This occurs when the role of each tuple component is to provide information about a neighboring edge. Specifically, for each edge j in E(i,*), the value $ib \in A_i$ has for its j^{th} component the slot-value ib(j). That is,

$$\pi^i_j: ib \in A_i \rightarrow ib(j) \in \text{Slot-Values}.$$

For example, as detailed below in Section 4.1 on scene labelling, if a variable i has h edge neighbors, its binding values are h-tuples of elements in the slot-values set $\{+,-,\rightarrow,\leftarrow\}$.

The key idea is to partition the elements of A_i (and A_j) into equivalence classes, and then connect the classes, instead of connecting the elements [14]. Given A_j, the inverse image of a slot-value v is

$$\pi^i_j{}^{-1}(v) = \{\, ib \in A_i \mid ib(j)=v \,\}.$$

If v and w are two compatible slot values, then R_{ij} connects every ib in the subset $\pi^i_j{}^{-1}(v)$ of A_i to every jc in the subset $\pi^j_i{}^{-1}(w)$ of A_j. However, relation R_{ij} between tuples can be compactly summarized by a core relation R^0_{ij} that connects compatible slot-values. The single link $R^0_{ij}(v,w)$ can then replace the complete bipartite graph between subsets $\pi^i_j{}^{-1}(v)$ and $\pi^j_i{}^{-1}(v)$.

Using this partition, we can map variable i's tuple values {ib} along each neighboring edge j into their slot-value equivalence classes. The function

$$\pi^i_j: ib \in A_i \rightarrow \pi^i_j(ib) \in \text{Slot-Values},$$

induces a sparse relation P_{ij}, having exactly $|A_i|$ links. Similarly, the function

$$\pi^j_i: jc \in A_j \rightarrow \pi^j_i(jc) \in \text{Slot-Values}$$

induces the sparse relation Q_{ij}, which maps tuples in A_j into their slot-value equivalence classes.

The core relation R^0_{ij} can then be formed, connecting the slot-value equivalence classes of $P_{ij}(A_i)$ with those of $Q_{ij}(A_j)$. This provides a factorization of R_{ij},

$$R_{ij} = P_{ij} \times R^0_{ij} \times Q_{ij}^T$$

shown in Figure 2. Here both P_{ij} and Q_{ij}^T are sparse relations formed from functions, with $|P_{ij}| \leq a$ and $|Q_{ij}^T| \leq a$. R^0_{ij} connects the slot-values of $P_{ij}(A_i)$ and $Q_{ij}(A_j)$.

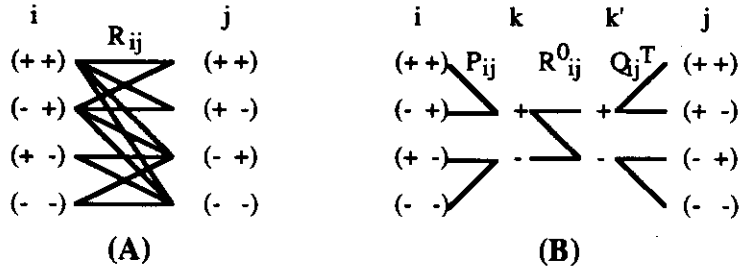

Figure 2. A. An example of relation R_{ij}'s links in graph G between variables i and j, where the variable binding values are tuples with component slot-values in {+,-}. **B.** The graph G' that results from factoring R_{ij} into $P_{ij} \times R^0_{ij} \times Q_{ij}^T$. Note how R_{ij}^0 clarifies the core slot-value relation {(+,+), (+,-), (-,-)}.

This factorization adds two new variables to G, k and k', whose binding values are slot-values. It also constructs additional links between k and k'. The utility of the factorization depends on $r^0_{ij} = |R^0_{ij}|$, since the number of links is now

$$p_{ij} + r^0_{ij} + q_{ij}^T \leq a + r^0_{ij} + a.$$

In the new graph G', the number of links is thus changed from a^2 to $2a + r^0_{ij}$. If r^0_{ij} is small, this may reduce the size of G.

3.2. Equality Relations

When R^0_{ij} represents an equality relation, then r^0_{ij} is small. In fact, R^0_{ij} then becomes the identity relation I. This is shown in Figure 3.A, where the factorization

$$R_{ij} = P_{ij} \times I_{ij} \times Q_{ij}^T$$

has the number of links

$$p_{ij} + k + q_{ij}^T \leq 2a + k,$$

where k is the number of common slot-values, $\#V_{ij}$, and $V_{ij}=P_{ij}(A_i) \cap Q_{ij}(A_j)$.

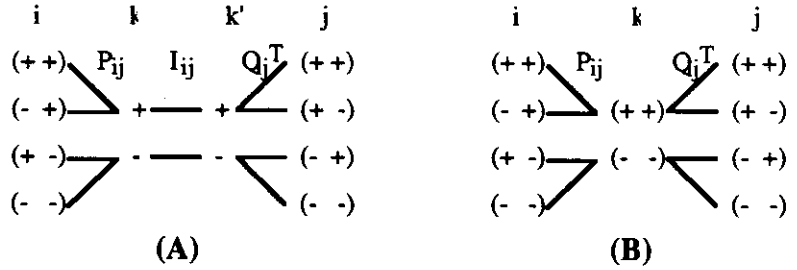

Figure 3. A. A factorization in which R^0_{ij} is an equality relation, i.e., the identity relation I_{ij}. **B.** Collapsing the identity I_{ij} into a single set.

We can do better. In Figure 3.B, we collapse the identity I_{ij} relation into the single set V_{ij}, and reduce the factorization to

$$R_{ij} = P_{ij} \times Q_{ij}^T.$$

Here, the number of links is

$$p_{ij} + q_{ij}^T \leq 2a,$$

Thus, when R_{ij} is an equality relation, the upper bound on the number of links is lowered from a^2 to 2a.

When *all* the R_{ij} are equality relations on tuple slots, then all the R_{ij} can be factored. This adds e variables, one for each factored edge, and describes a new constraint graph G'. The key property of G' is that every relation has at most a links between values in it. Therefore,

$$n' = n + e,$$

$$e' = 2e,$$

$$r_{ij}' \leq a.$$

Thus, the size of G' is at most

$$
\begin{aligned}
|G'| &= \#\text{nodes} + \#\text{links} \\
&= n' + \Sigma_{E'} r_{ij}' \\
&\leq (n + e) + (2e \times a) \\
&= O(ea).
\end{aligned}
$$

Therefore, we arrive at the reduced upper bound

$$O(|G'|) = O(ea).$$

And, for a graph with bounded out-degree (e.g., a planar graph),

$$O(|G'|) = O(na).$$

4. Applicable Situations

In applications of arc consistency, situations arise where

- variable bindings are comprised of tuples,
- the role of each tuple component is to provide information about a neighboring edge, and
- the slot values constrain tuples to have equal components along neighboring edges.

In these cases, factorization can construct a new graph G', for which the cost of AC-5 is O(ea).

4.1. Waltz Scene Labelling

In scene labelling [17], each junction in a line drawing is a variable i with a set A_i of physically realizable edge labellings. Suppose junction i has h neighboring edges. A value in A_i is then an h-tuple of possible labels for these edges. The slot values belong the set {+,-,→,←}, denoting convex, concave, or boundary edge labels.

Slot values along neighboring edges are constrained to be equal. Therefore, as shown in Figure 4, we can factor a scene labelling problem's graph G into a new graph G', in which

$$O(|G'|) = O(ea).$$

Since a line drawing is a two-dimensional projection, i.e., a planar graph, e is proportional to the number of variables n, and

$$O(|G'|) = O(na).$$

This linear dependence of arc consistency on the number of candidate bindings a is a new result for the scene labelling problem.

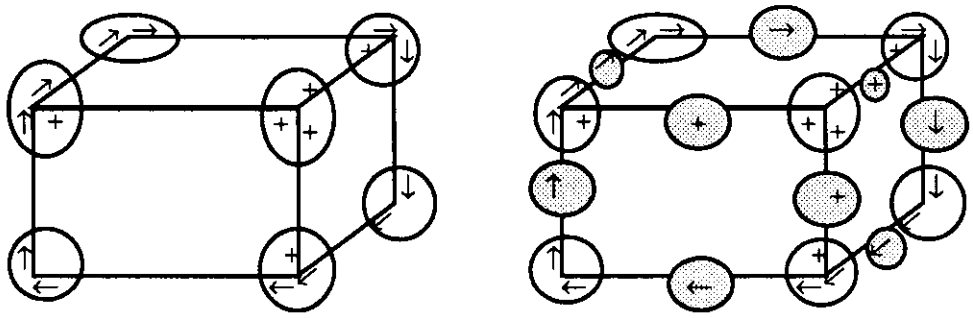

Figure 4. A. A 7-tuple CSP solution of 7 variables having tuple-valued bindings from graph G. Each circled vertex denotes a tuple-valued variable binding. **B.** Factoring each edge into two edges and a new variable to construct a new graph G'. The bindings of the 9 new edge variables are highlighted.

4.2. RETE Match

The conjunctive matching of rules against working memory (WM) employed in production systems [18] is a CSP that computes all solutions. In fact, in languages such as OPS-5, it is a binary CSP [12]. The usual recursive construction of conjunctive match uses a backtrack control to filter inadmissible extensions of k-tuples [13]. When this recursion is transformed into an incremental network program [11], the RETE match [3] algorithm results.

The key computation of RETE is incremental construction of the cartesian product $\Pi_N A_i$. Therefore, arc consistency can be of use: preliminary filtering of the A_i can reduce the cost of n-tuple formation. In RETE terminology, each variable i is called a condition element, and the A_i contain values called working memory elements (WMEs). The relations R_{ij} are determined by binary join tests between variables. One can construct a CSP graph having one edge per binary join test.

A variable binding value (or WME) is itself a tuple. Each join test compares the slot-value of a tuple binding of variable i with a tuple's slot-value from variable j. Therefore, the R_{ij} are *factorable* into $P_{ij} \times R^0_{ij} \times Q_{ij}^T$ via equivalence classes of slot-values, as in Section 3.1. (There are other optimizations for conjunctive match that

exploit slot-value equivalence classes, such as hashed. RETE [16], copy and constrain [9], and MatchBox [14].)

Interestingly, 90% of join tests are for equality [4]. If just these are used for arc consistency, then, by Section 3.2, the cost of AC-5 is reduced to O(ea). This is our improved linear bound, finding application to an important computation in AI: RETE match. Figure 5 shows the factoring of a rule's CSP.

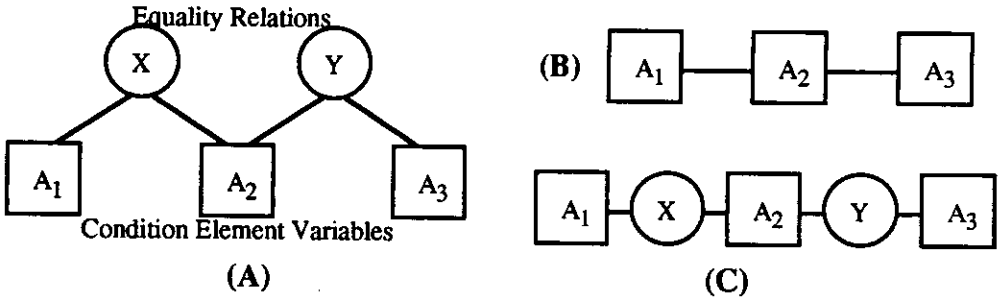

Figure 5. A. A graphical rule specification, following [15]. There are three variables (condition elements), and two equality relations (join tests). **B.** The graph G of the associated CSP. The relations are incorporated into the edges. **C.** The graph G' after factoring the equality relations. The join tests now explicitly appear as CSP variables. The topology is identical to the specification shown in (A).

5. Conclusions

Arc consistency is an important and ubiquitous algorithm in AI, used to reduce the combinatorics of cartesian product formation. There has been steady improvement in arc consistency algorithms. Mohr and Henderson [8] presented an optimal algorithm for arc consistency AC-4, that has cost $O(ea^2)$, and only $O(na^2)$ for scene labelling applications. We improved this bound to a linear cost in a for factorable relations. Recently, an analogous improvement [2], also applicable to scene labelling, was developed for Constraint Logic Programming; the authors exploit functions as we do, but do not describe factorable relations.

In this paper, we motivated and presented a variant of AC-4, called AC-5, that made explicit the use of the relation links between variable binding values, thus forming the graph G. We showed when and how the relations R_{ij} were factorable,

transforming G into a new G'. With equality relations, $|G'| \leq |G|$, and the cost of scene labelling is reduced to O(na). This result extends to other applications, such as RETE match, for which we showed a factorization reducing the arc consistency cost from $O(ea^2)$ to O(ea).

Acknowledgement

Sean Engelson programmed a version of our O(na) scene labelling user interface in 1988 on a SUN/3 computer. Discussions with Gene Freuder about our approach initiated the extensions described in this paper.

References

[1] J. M. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comp.*, vol. 19, pp. 297-301, 1965.

[2] Y. Deville and P. Van Hentenryck, "Efficient Arc Consistency Algorithm for a Class of CSP Problems," Department of Computer Science, Brown University, Technical Report CS-90-36, December, 1990.

[3] C. L. Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," *Artificial Intell.*, vol. 19, no. 1, pp. 17-37, 1982.

[4] A. Gupta, "Parallelism in Production Systems," Ph.D. dissertation, Department of Computer Science, Carnegie Mellon University, 1986.

[5] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms.* Rockville, Maryland: Computer Science Press, 1984.

[6] A. K. Mackworth, "Consistency in Networks of Relations," *Artificial Intelligence*, vol. 8, pp. 99-118, 1977.

[7] A. K. Mackworth and E. C. Freuder, "The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems," *Artificial Intelligence*, vol. 25, pp. 65-74, 1985.

[8] R. Mohr and T. C. Henderson, "Arc and Path Consistency Revisited," *Artificial Intelligence*, vol. 28, pp. 225-233, 1986.

[9] A. Pasik, "Improving Production System Performance on Parallel Architectures by Creating Constrained Copies of Culprit Rules," Department of Computer Science, Columbia University, Tech Report, 1987.

[10] M. W. Perlin, "Reducing Computation by Unifying Inference with User Interface," Carnegie Mellon University, Tech Report CMU-CS-88-150, June, 1988.

[11] M. W. Perlin, "Call-Graph Caching: Transforming Programs into Networks," in *Proc. of the Eleventh Int. Joint Conf. on Artificial Intelligence*, Detroit, Michigan, Morgan Kaufmann, August, 1989, pp. 122-128.

[12] M. W. Perlin, "Constraint-Based Specification of Production Rules," in *IEEE International Workshop on Tools for Artificial Intelligence*, Fairfax, VA, IEEE Computer Society Press, October, 1989, pp. 332-338.

[13] M. W. Perlin, "Transforming Conjunctive Match into RETE," School of Computer Science, Carnegie Mellon University, Chapter 1 in Tech Report CMU-CS-90-132, May, 1990.

[14] M. W. Perlin and J.-M. Debaud, "Match Box: Fine-Grained Parallelism at the Match Level," in *IEEE International Workshop on Tools for Artificial Intelligence*, Fairfax, VA, IEEE Computer Society Press, October, 1989, pp. 428-434.

[15] M. W. Perlin and P. Gaertner, "A Graphical Constraint-Based Production System Environment," in *Second Int. Conf. on Tools for Artificial Intelligence*, Washington, D.C., IEEE Computer Society, November, 1990.

[16] D. J. Scales, "Efficient Matching Algorithms for the Soar/Ops5 Production System," Master's thesis, Stanford University, 1986.

[17] D. Waltz, "Understanding line drawings of scenes with shadows," in *The Psychology of Computer Vision,* P. H. Winston, ed. New York: McGraw-Hill, 1975, pp. 19-91.

[18] D. A. Waterman and F. Hayes-Roth, ed., *Pattern Directed Inference Systems.* New York: Academic Press, 1978.