AN EFFICIENT RELEVANCE CRITERION FOR MECHANICAL THEOREM PROVING*

David A. Plaisted
Department of Computer Science
University of Illinois
Urbana, Illinois 61801

ABSTRACT

To solve problems in the presence of large knowledge bases, it is important to be able to decide which knowledge is relevant to the problem at hand. This issue is discussed in [1]. We present efficient algorithms for selecting a relevant subset of knowledge. These algorithms are presented in terms of resolution theorem proving in the first-order predicate calculus, but the concepts are sufficiently general to apply to other logics and other inference rules as well. These ideas should be particularly important when there are tens or hundreds of thousands of input clauses. We also present a complete theorem proving strategy which selects at each step the resolvents that appear most relevant. This strategy is compatible with arbitrary conventional strategies such as P_1-deduction, locking resolution, et cetera. Also, this strategy uses nontrivial semantic information and "associations" between facts in a way similar to human problem-solving processes.

I RELEVANCE FUNCTIONS

Definition. A support set for a set S of clauses is a subset T_i of S such that $S-T_i$ is consistent. A support class for S is a set $\{T_1,\ldots,T_k\}$ of support sets for S.

Definition. A (resolution) proof of C from S is a sequence $C_1,C_2,\ldots,C_n$ of clauses in which C_n is C and each clause C_i is either an element of S (an input clause) or a resolvent of two preceding clauses in S. (Possibly both parents of C_i are identical.) The length of such a refutation is n. A refutation from S is a proof of NIL (the empty clause) from S.

Definition. A relevance function is a function R which, given a set S of clauses, a support class T for S, and an integer n, maps onto a subset $R_n(S, T)$ of S having the following property:

> If there is a length n refutation from S, then there is a refutation from $R_n(S, T)$ of length n or less.

Thus if we are searching for length n refutations from S, we need only search for length n refutations from $R_n(S, T)$. In fact, the derivation from $R_n(S, T)$ will be a subderivation of the derivation from S, for all relevance functions considered here. Thus if there is a length n P_1-deduction from S, there will be a P_1-deduction of length n or less from $R_n(S, T)$, and similarly for other complete strategies.

Definition. Suppose S is a set of clauses. The connection graph of S, denoted G(S), is the graph whose nodes are the clauses of S, and which has a directed arc from C_1 to C_2 labeled (L_1, L_2) if there are literals $L_1 \in C_1$ and $L_2 \in C_2$ such that L_1 and $\bar{L}_2$ are unifiable. Such graphs have been introduced and discussed in [2]. Note that there will also be an arc labeled (L_2, L_1) from C_2 to C_1 in the above case.

Definition. A path from C_1 to C_n in G(S) is a sequence $C_1,C_2,\ldots,C_n$ of clauses of S such that there is an arc from C_i to C_{i+1} in G(S), for $1 \le i < n$. Also, the length of the path is n.

Definition. The distance $d(C_1, C_2)$ between C_1 and C_2 in G(S) is the length of the shortest path from C_1 to C_2 in G(S), and ∞ if no such path exists.

Definition. If S is a set of clauses, T is a support class for S, and n is a nonnegative integer, then $Q_n(S, T)$ is $\{C \in S\colon d(C, T_i) \le n$ in G(S) for all T_i in T$\}$, where $d(C, T_i)$ is $\min\{d(C, D)\colon D \in T_i\}$.

Intuitively, if $d(C_1, C_2)$ is small, C_1 and C_2 are "closely related." Also, $Q_n(S, T)$ is the clauses that are closely related to all the support sets. Typically we will know that several clauses are essential to prove a theorem, and each such clause by itself can be made into a support set.

Definition. A set S of clauses is fully matched if for all $C_1 \in S_1$ for all literals $L_1 \in C_1$ there exists $C_2 \in S$ and $L_2 \in C_2$ such that L_1 and $\bar{L}_2$ are unifiable.

* This research was partially supported by the National Science Foundation under grant MCS-79-04897.

Definition. $R_n(S, T)$ is the largest fully matched subset of $Q_n(S, T)$. Thus we obtain $R_n(S, T)$ from $Q_n(S, T)$ by repeatedly deleting clauses containing "unmatched" literals. This definition is not ambiguous, since if $Q_n(S, T)$ contains more than one nonempty fully matched subset, then $R_n(S, T)$ is the union of all such subsets.

Theorem 1. The function R is a relevance function. That is, if there is a length n refutation from S, and T is support class, then there is a refutation from $R_n(S, T)$ of length n or less. In fact, there is such a refutation from $R_{\lceil \frac{n}{2} \rceil}(S, T)$.

Proof. Assume without loss of generality that NIL appears only once in the refutation and that every clause in the refutation contributes to the derivation of NIL. Let S_1 be the set of input clauses appearing in the refutation. Then S_1 is connected, intersects all the support sets, and has at most n elements. Using properties of binary trees we can show that S_1 has at most $\lceil \frac{n}{2} \rceil$ elements. Note that $R_n(S, T)$ is a "global" relevance criterion. That is, it depends in a nontrivial way on all the input clauses and on interactions between all the support sets in T.

II EXAMPLES

Let S be the following set of clauses:

P1	$\overline{Q2}$ Q1	$\overline{Q4}$ P4
P2	$\overline{Q1}$ Q3	$\overline{Q5}$ Q4
$\overline{P1}$ $\overline{P2}$ P3	$\overline{Q6}$ P1	$\overline{P3}$ P4
$\overline{P1}$ Q1 Q7	$\overline{Q6}$	$\overline{P4}$
$\overline{P2}$ Q2	$\overline{P4}$ Q4 Q5	$\overline{Q7}$ P4

Here $\overline{P1}$ Q1 indicates {$\overline{P1}$, Q1}, i.e., $\overline{P1} \vee$ Q1 et cetera. Let T1 be {{P1}, {P2}}, let T2 be {{$\overline{P4}$}}, and T = {T1, T2}. Then $R_1(S, T) = R_2(S, T) = R_3(S, T) = \emptyset$ but $R_4(S, T)$ = {{P1},{P2}, {$\overline{P1}$,$\overline{P2}$,P3}, {$\overline{P3}$,P4}, {$\overline{P4}$}} which is in fact a minimal inconsistent subset of S.

For a second example, let S be the following:

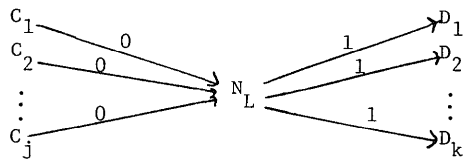
IN(a,box)
IN(x,box) ⊃ IN(x,room)
IN(x,room) ⊃ IN(x,house)
$\overline{\text{IN}}$(x,house)
ON(x,box) ⊃ $\overline{\text{IN}}$(x,box)
ON(x,street) ⊃ $\overline{\text{IN}}$(x,house)
IN(x,house) ⊃ IN(x,village)
$\overline{\text{IN}}$(house,box)
AT(house,street)
ON(b,box)
ON(c,street)
$\overline{\text{IN}}$(d,village)

Let T1 be {{IN(a,box)}} and let T2 be {{$\overline{\text{IN}}$(x, house)}}. Also, T = {T1, T2}. Then $R_1(S, T) = R_2(S, T) = R_3(S, T) = \emptyset$ but $R_4(S, T)$ = {{IN(a, box)}, {$\overline{\text{IN}}$(x,box), IN(x,room)}, {$\overline{\text{IN}}$(x,room), IN(x, house)}, {$\overline{\text{IN}}$(x,house)}}. This is a minimal inconsistent subset of S. Here "box", "room", "house", "a" et cetera are constants and x is a variable. Note that we cannot always guarantee that this relevance criterion will yield minimal inconsistent sets as in these examples.

III ALGORITHMS

Suppose S is a set of clauses. Let $||S||$ be the length of S in characters when written in the usual way. Let Lits(S) be the sum over all clauses C in S, of the number of literals in C.

If S is a set of propositional clauses and $||S|| = m$, then G(S) may have $O(m^2)$ arcs. However, we can construct a modified version $G_1(S)$ of G(S) which has the same distances between clauses as G(S) does but which has only O(m) arcs. The idea is to reduce the number of arcs as follows: Suppose $C_1, \ldots, C_j$ are the clauses containing L and $D_1, \ldots, D_k$ are the clauses containing $\bar{L}$. Then we add a node N_L and arcs as follows in $G_1(S)$:



The numbers indicate the lengths of the arcs. Similarly, there are arcs of the form $D_{i1} \xrightarrow{0} N_{\bar{L}} \xrightarrow{1} C_{i2}$. Although $G_1(S)$ is not a connection graph, and has arcs of length 0 and 1, it preserves distances between clauses as in G(S). Using this modified connection graph, we have linear time algorithms to do the following, if S is a set of propositional clauses, $T = \{T_1, \ldots, T_k\}$ is a support class, T_i is a support set, and n is a positive integer:

1. Construct $G_1(S)$ from S.
2. Find $\{C \in S: d(C, T_i) \leq n\}$.
3. Given $Q_n(S, T)$ for support class T, to find $R_n(S, T)$.

Since step 2 must be performed $|T|$ times to obtain $Q_n(S, T)$, the total algorithm to obtain $R_n(S, T)$ requires $O(|T| \cdot ||S||)$ time. (Here $|T|$ is the number of support sets.)

The algorithm to find $\{C \in S: d(C,T_i) \leq n\}$ is a simple modification of standard shortest path algorithms. For a presentation of these standard algorithms see [3]. This can be done in linear time because the edge lengths are all 0 and 1. We compute $R_n(S, T)$ as follows:

Definition. If S is a set of clauses, let M(S) be the largest fully matched subset of S. Note that $R_n(S, T) = M(Q_n(S, T))$.

The following algorithm M_1 computes M(S) for set S of propositional clauses in linear time. This algorithm can therefore be used to compute $R_n(S, T)$ if S is a set of propositional clauses. Note that t is a push-down stack.

```
procedure M1(S);
   t ← empty stack;
   for all L such that LεC or L̄εC for some CεS do
      clauses(L) ← {CεS:LεC};
      count(L) ← |clauses(L)| od;
   for all CεS do
      member(C) ← TRUE;
      for all LεC do
         if count (L̄) = 0 then
            push C on t; member(C) ← FALSE fi
      od
   od;
   while t not empty do
      pop C off t;
      for all LεC do
         count(L) ← count(L) - 1;
         if count(L) = 0 then
            for all C1 ε clauses(L̄) do
               if member(C1) then
                  push C1 on t;
                  member(C1) ← FALSE fi
            od
         fi
      od
   od;
   return({CεS:  member(C) = TRUE});
end M1;
```

If S is a set of first-order clauses, then G(S) can be constructed in O(Lits(S)*||S||) time using a linear unification method [4]. This bound results because a unification must be attempted between all pairs of literals of S. The number of edges in G(S) is at most $Lits(S)^2$. Given G(S), we can find $\{C \in S: d(C, T_i) \leq n\}$ in time proportional to the number of edges in G(S) (since all edge lengths are one). Also, given $Q_n(S, T)$, we can find $R_n(S, T)$ in time proportional to the number of edges in G(S) by a procedure similar to M_1 above. The total time to find $R_n(S, T)$ is therefore $O(|T|*Lits(S)^2 + ||S||*Lits(S))$. If $||S|| = m$ then the time to find $R_n(S, T)$ is $O(m^2|T|)$. By considering only the predicate symbols of literals, the propositional calculus algorithms can be used as linear time preprocessing step to eliminate some clauses from S. An interesting problem is to compute $R_n(S, T)$ efficiently for many values of n at the same time. There are methods for doing this, but we do not discuss them here.

IV REFINEMENTS

A. Connected Components

Proposition 1. If there is a length n refutation from S, and T is a support class for S, then there is a length n refutation from one of the connected components of $R_{\lceil \frac{n}{2} \rceil}(S, T)$. Also, the connected components can be found in linear time [3].

B. Iteration

Definition. If $T = \{T_1, T_2, \ldots, T_k\}$ is a support class for S and S_1 is a subset of S then $T /\!\!/ S_1$ (T restricted to S_1) is $\{T_1 \cap S_1, \ldots, T_k \cap S_1\}$. It may be that $T /\!\!/ R_n(S, T) \neq T$ or that distances in $G(R_n(S, T))$ are different than distances in G(S). This motivates the following definitions.

Definition. $R_n^0(S, T) = S$, $R_n^1(S, T) = R_n(S, T)$, and if i>1 then $R_n^i(S, T) = R_n(R_n^{i-1}(S, T), T /\!\!/ R_n^{i-1}(S, T))$. Also, $R_n^\infty(S, T)$ is the limit of the sequence $R_n^1(S, T)$, $R_n^2(S, T)$, $R_n^3(S, T)$, ...

Proposition 2. $R_n^{i+1}(S, T) \subseteq R_n^i(S, T)$ for $i \geq 1$. Therefore the limit $R_n^\infty(S, T)$ exists. Also, $R_n^\infty(S, T)$ can be computed in at most |S| iterations of $R_n(S, *)$. Can it be computed more efficiently than this?

Theorem 2. If there is a length n refutation from S, and T is a support class for S, then there is a length n refutation from one of the connected components of $R^\infty_{\lceil \frac{n}{2} \rceil}(S, T)$.

Proposition 3. For all i>0 there exist n, S and T such that $R_n^{i+1}(S, T) = R_n^i(S, T) \neq R_n^{i-1}(S, T)$. Thus this computation can take arbitrarily long to converge.

Proof. Let n = 2, $S = \{P_i \supset P_{i+1}: 1 \leq i \leq k\} \cup \{P_{i+1} \supset P_i : 1 \leq i \leq k\}$. Let $T = \{T_1, T_2, T_3\}$ where $T_j = \{P_i \supset P_{i+1}: i \equiv j (\text{mod} 3)\} \cup \{P_{i+1} \supset P_i : i \equiv j (\text{mod } 3)\}$. Then $R_2^a(S, T) = \emptyset$ if $2a \geq k$ but $R_2^a(S, T) \neq \emptyset$ if $2a < k$.

C. Selecting Support Clauses

We now give another approach.

Theorem 3. Suppose there is a length n refutation from S and $T = \{T_1, \ldots, T_k\}$ is a support class for S. Then there exist clauses $C_i \in T_i$, $1 \leq i \leq k$, such that

a) $\{C_1, C_2, \ldots, C_k\} \subseteq R^\infty_{\lceil \frac{n}{2} \rceil}(S, \{\{C_1\}, \{C_2\}, \ldots, \{C_k\}\})$ and
b) there is a length n refutation from $R^\infty_{\lceil \frac{n}{2} \rceil}(S, \{\{C_1\}, \ldots, \{C_k\}\})$.

Thus it is possible to select particular clauses from the support sets and use them to define R.

There may be many sets $R^\infty_{\lceil \frac{n}{2} \rceil}(S, \{\{C_1\}, \ldots, \{C_k\}\})$ satisfying condition a) above, but they may be smaller than the connected components of $R^\infty_{\lceil \frac{n}{2} \rceil}(S, T)$. It is possible to construct examples having this property. Therefore it may help to use the above sets rather than $R^\infty_{\lceil \frac{n}{2} \rceil}(S, T)$ when searching for refutations. Another advantage is that it is possible to examine the clauses $C_i \in T_i$ in some heuristically determined order. Furthermore, the above approach is useful when the support sets T_i are not known in advance but are obtained one clause at a time.

We now give a recursive procedure "rel3" for generating all the sets as in Thereom 3. The idea is to order the T_i so as to reduce the branching factor as much as possible near the beginning of the search. Thus we use the principle of "least commitment." This procedure has as input sets S_1 and S of clauses, integer n, and support class T for S. Let $S_1 = \{D_1,\ldots,D_j\}$ and $T = \{T_1,\ldots,T_k\}$. The procedure rel3(S_1,n,S,T) outputs all sets $R_n^\infty(S, \{\{D_1\},\ldots, \{D_j\}, \{C_1\}, \ldots, \{C_k\}\})$ having $\{D_1,\ldots,D_j,C_1,\ldots,C_k\}$ as a subset, for $C_i \in T_i$, $1 \le i \le k$. If there is a length n refutation from S, and T is a support class for S, then there will be a length n refutation from some set output by rel3($\emptyset$, $\lceil\frac{n}{2}\rceil$, S, T).

<u>Definition</u>. If $S = \{D_1,\ldots,D_j\}$ then Single(S) $= \{\{D_1\}, \ldots, \{D_j\}\}$.

<u>procedure</u> rel3(S_1,n,S,T)
  $S_2 \leftarrow R_n^\infty(S,\ T \cup \text{Single}(S_1))$
  <u>if</u> $S_1 \subset S_2$ then
    <u>if</u> $T = \emptyset$ <u>then</u> output (S_2) <u>else</u>
      $T_1 \leftarrow T/\!\!/S_2$;
      choose $T_2 \in T_1$ minimizing $|T_2|$;
      for all $C \in T_2$ <u>do</u>
        rel3($S_1 \cup \{C\}$,n,S_2,T_1-T_2)
      <u>od</u>;
    <u>fi</u>;
  <u>fi</u>;
<u>end</u> rel3;

D. <u>Center Clauses</u>

By using the idea that graphs have "centers," we can reduce the distance needed to search for relevant clauses by another factor of 2.

<u>Proposition 4</u>. Suppose S_1 is a connected subset of G(S), and if $C_1,C_2 \in S_1$ then $d(C_1,C_2) \le m$. Then there exists $C_3 \in S_1$ such that for all $C \in S_1$, $d(C,C_3) \le \lceil\frac{m+1}{2}\rceil$ in $G(S_1)$.

<u>Theorem 4</u>. Suppose there is a length n refutation from set S of clauses, and T is a support class for S. Then there exists a clause $C \in S$ and a set $S_1 \subset S$ having the following properties:

1. There is a length n refutation from S_1
2. S_1 is fully matched
3. S_1 intersects all the support sets in T
4. $C \in S_1$ and for all $C_1 \in S_1$, $d(C, C_1) \le \lceil\frac{n+2}{4}\rceil$ in $G(S_1)$.

<u>Proof</u>. Let S_1 be the input clauses actually used in some minimal refutation from S. Then $|S_1| \le \lceil\frac{n}{2}\rceil$. Choose a "center" C of S_1, and note that $\lceil\frac{1}{2}\lceil\frac{n}{2}\rceil + \frac{1}{2}\rceil = \lceil\frac{n+2}{4}\rceil$.

By searching for such sets S_1, we can sometimes obtain much better relevance criteria than by previous methods. The use of centers insures that elements of S_1 will be closer together than in previous methods.

To implement this method, let S_2 be $Q_{\lceil\frac{n+2}{4}\rceil}$(S, T). For each $C \in S_2$, let S_3 be $R^\infty_{\lceil\frac{n+2}{4}\rceil}(S,\{\{C\}\})$. If S_3 intersects all support sets, then it is a candidate set of input clauses for a length n refutation. Here S_2 is a set of possible centers. Note that two clauses of S_3 will have distance at most $\lceil\frac{n}{2}\rceil + 1$ in $G(S_3)$. Note also that if n=6 then $\lceil\frac{n+2}{4}\rceil = 2$ and if n = 10 then $\lceil\frac{n+2}{4}\rceil = 3$. Thus we can get somewhat nontrivial refutations with quite small distance bounds.

E. <u>Typing Variables</u>

For these relevance criteria to be useful, there must exist clauses C_1 and C_2 of S such that $d(C_1, C_2)$ is large. However, if the axiom $x=y \supset y=x$ is in S then two clauses of the form $t_1 = t_2 \vee D_1$ and $t_3 \neq t_4 \vee D_2$ will have distance 3 or less. This may cause everything to be close to everything else. To reduce this problem, we propose that all variables be <u>typed</u> as integer, Boolean, list, string, et cetera and unifications only succeed if the types match. Thus the above clauses would not necessarily be within distance 3 if t_1 and t_4 or t_2 and t_3 have different types. The use of types may increase the number of clauses, since more than one copy of some clauses may be needed. However, the overall effect may still be beneficial.

F. <u>Logical Consequences</u>

The preceding ideas can also be applied to derivations of clauses other than NIL from S.

<u>Definition</u>. A <u>support set</u> for S relative to C is a subset V of S such that C is not a logical consequence of S-V. A <u>support class</u> for S relative to C is a collection of support sets for S relative to C. For example, if I is an interpretation of S in which C is false, and V is the set of clauses of S that are false in I, then V is a support set for S relative to C.

<u>Definition</u>. M(S, C) is the largest subset of S in which all literals are matched, except possibly those having literals of C as instances.

<u>Definition</u>. $R_n(S,T,C)$ is $M(Q_n(S,T),C)$.

<u>Theorem 5</u>. If there is a length n derivation of something subsuming C from S, and T is a support class for S relative to C, then there is a length n derivation of something subsuming C from $R_{\lceil\frac{n}{2}\rceil}$(S, T,C).

As before, we can introduce $R^\infty_{\lceil\frac{n}{2}\rceil}(S,T,C)$ and other relevance criteria.

G. Procedures

To incorporate procedural and heuristic information, we may add clauses expressing the assertion $A(\bar{x}) \supset (\exists \bar{y}) B(\bar{x},\bar{y})$ where A and B are input and output assertions for the procedure and $\bar{x}$ and $\bar{y}$ are input and output variables. To account for the fact that heuristics may fail, we assign probabilities of truth to clauses. The task then is to find a set S_1 of clauses from which the desired consequence can possibly be derived, subject to the condition that the product of the probabilities of the clauses in S_1 is as large as possible. One way to do this is to run many trials, generating relevant subsets of S, where the clauses of S are chosen to be present or absent with the appropriate probability. We then select a relevant set of clauses from among those clauses that have been found to be relevant in many of the trials. Note that if procedures are encoded as above, then a short proof may correspond to a solution using a few procedure calls, but each procedure may require much time to execute.

H. Subgoals

If procedures are encoded as above, then each procedure may call the whole theorem prover recursively. This provides a possible subgoal mechanism. By storing the clauses from all subgoals in a common knowledge base, we may get interesting interactions between the subgoals. By noticing when subgoals are simpler than the original problem in some well-founded ordering, we may be able to get mathematical induction in the system. The use of clauses, procedures, subgoals, and relevance criteria as indicated here provides a candidate for a general top-level control structure for an artificial intelligence system.

V A COMPLETE STRATEGY

The following procedure attempts to construct a refutation from set S of first-order clauses:

```
procedure refute(S);
  for d = 1 step 1 until (NIL is derived) do
    for j = 1 step 1 until (j > d) do
      refl(S, j, d) od od;
end refute;

procedure refl(S, i, d);
  let T be a support class for S;
  R ← R_i^∞(S, T);
  if R is empty then return fi;
  V ← R ∪ { level 1 resolvents from R};
  if NIL ∈ V or d = 1 then return fi;
  for j = 1 step 1 until(NIL is derived) do
    refl(V, j, d - 1) od;
end refl;
```

This procedure selects at each step the clauses that seem most relevant and attempts to construct a refutation from them. Similar procedures can be given using other of the relevance functions described earlier.

A. Generating Support Sets

One way to generate support sets for the above procedure is to let each support set be the subset of S in which specified predicate symbols occur with specified signs. This would yield 2^n support sets for n predicate symbols. Of course, it is not necessary to use all of these support sets. A more interesting possibility is to have a collection $\{I_1, I_2, \ldots, I_k\}$ of interpretations of S and to let T_i be the set of clauses that are false in I_i. If I_i has a finite domain then T_i can be computed by exhaustive testing. Otherwise, special methods may be necessary to determine if a clause C is true in I_i. If I_i has an infinite domain, a possible heuristic is to let T_i be the set of clauses that are false on some finite subset of the domain. If f is an abstraction mapping or a weak abstraction mapping [5] and I is an interpretation, then $\{C \in S$: some clause in f(C) is false in I$\}$ is a support set for S. This approach may allow the use of nontrivial support sets which are easy to compute, especially if all elements of f(C) are ground clauses for all C in S. Note that T may include support sets obtained both syntactically and semantically. Although it may require much work to test if C is true in I_i, this kind of effort is of the kind that humans seem to do when searching for proofs. Also, this provides a meaningful way of incorporating nontrivial semantic information into the theorem prover. The arcs in the connection graph resemble "associations" between facts, providing another similarity with human problem solving methods.

REFERENCES

[1] Gallaire, H. and J. Minker, eds. Logic and Data Bases. New York: Plenum Press, 1978.

[2] Kowalski, R., "A Proof Procedure using Connection Graphs". J.ACM 22(1975)572-595.

[3] Reingold, E. M., J. Nievergelt, and N. Deo. Combinatorial Algorithms: Theory and Practice. Englewood Cliffs, New Jersey: Prentice-Hall, 1977.

[4] Paterson, M. S. and M. N. Wegman, "Linear Unification", IBM Research Report 5304, IBM, 1976.

[5] Plaisted, D., "Theorem Proving with Abstraction, Part I", Departmental Report UIUCDCS-R-79-961, University of Illinois, February 1979.