IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, VOL. 26, NO. 6, DECEMBER 1996

Correspondence

Knowledge Representation Using Fuzzy Deduction Graphs
Manohar Chandwani and Narendra S. Chaudhari

Abstract—A new knowledge representation model, known as fuzzy deduction graph (FDG), is introduced in this paper. An FDG can represent a knowledge base containing the fuzzy propositions and fuzzy rules. In an FDG, a systematic method of finding the fuzzy reasoning path (FRP) is given which is based on Dijkstra’s shortest path framework. The FRP gives a relationship between the antecedent (source) proposition and consequent (goal) proposition, such that the consequent proposition is reached with the greatest fuzzy value. The process of finding the FRP is illustrated with examples.

I. INTRODUCTION

We introduce a new knowledge representation model known as fuzzy deduction graph (FDG). The concept of FDG is derived from the (crisp) deduction graph due to Yang [13], [14] by incorporating fuzzy concepts [3], [15]. An FDG is a graphically structured subset of a rule base R with rules of fuzzy propositions represented in the form of Horn clauses/formulae. An FDG can be used to perform automated fuzzy reasoning. A fuzzy reasoning path (FRP) can be established to define the antecedent–consequent relationship between source and goal propositions. One way to establish FRP is through fuzzy logic. FRP defines an antecedent–consequent relationship of two propositions that leads to the greatest fuzzy value of the consequent proposition.

The reasoning with Petri nets [9], [11] and fuzzy Petri nets [1], [5] has received considerable attention from researchers in the past few years. Chen et al. [1] have presented an algorithm to find the reasoning path in a fuzzy Petri net representing the knowledge. Their algorithm selects the desired FRP between two propositions after finding a sprouting-tree of all paths from antecedent proposition to consequent proposition. This algorithm is good when the number of paths between two propositions is small. Moreover, the algorithm does not use any specific property of Petri net other than reachability. In this paper, we investigate the same path finding problem in a more systematic and efficient way. We follow Dijkstra’s shortest path framework on an FDG to establish FRP. In the crisp weighted graphs, Dijkstra’s algorithm is normally used to find the shortest path amongst many paths between two nodes as the minimum summation of weights labeling edges. Homg and Yang [2] used Dijkstra’s framework for construction of minimal deduction graph which represents an inference with minimum number of rules, when the weight labeling an edge in the graph is 0 or 1. Dijkstra’s framework presented in this paper establishes the reasoning path in FDG such that starting with source proposition, the goal proposition is reached with the greatest fuzzy value. The weights of edges in an FDG are real numbers in the fuzzy interval [0,1]. The maximum of multiplications is obtained on weights instead of minimum of summations of weights.

The rest of the paper is organized as follows: Section II briefly presents fuzzy knowledge and reasoning. In Section III, we give formal definition of the FDG and its construction issues. Section IV illustrates the FRP algorithm giving its performance with examples. The paper ends with conclusions in Section V.

II. FUZZY KNOWLEDGE AND REASONING

Most of the domain knowledge available from the application sources is imprecise, vague and/or ambiguous [4]. Mathematical quantification of such knowledge can be dealt using fuzzy theory due to Zadeh [15], leading to fuzzy knowledge. One way to represent fuzzy knowledge is through fuzzy propositions and fuzzy production rules. A fuzzy proposition involves a fuzzy term or its truth value is fuzzy. For instance, the proposition p: “John is brilliant” involves fuzziness due to degree of brilliance of John, that can be assigned a number in the interval [0,1], say 0.7 i.e.,

John is brilliant (0.7).

A fuzzy production rule defines the fuzzy relationship between antecedent and consequent propositions. If the fuzzy values of antecedent proposition and rule are known, then fuzziness of consequent can be determined in fuzzy logic. For instance, let p and q be fuzzy propositions in rule r: "If p Then q" and let the fuzzy value of p be denoted by f(p) and the fuzzy value associated with r be denoted by C(r), then fuzziness of q is computed by (1).

\[ f(q) = C(r)^* f(p). \]  \hspace{1cm} (1)

The above method of reasoning, i.e., finding the fuzzy relationship between antecedent and consequent propositions, can be generalized for the knowledge bases. We define fuzzy knowledge base (FKB) as a set of fuzzy propositions and fuzzy production rules. Formally, FKB consists of a set of ordered pairs

\[ FKB = \{(p_1, f(p_1)), (p_2, f(p_2)), \ldots, (p_m, f(p_m)) \} \]  \hspace{1cm} (2)

where each \( p_i \) (1 \( \leq \) i \( \leq \) m) is a proposition with fuzzy value \( f(p_i) \) and each \( r_j \) (1 \( \leq \) j \( \leq \) n) is a rule with certainty factor \( C(r_j) \). Let the general formulation of jth fuzzy rule be as follows:

\[ r_j: \text{ If } a_j \text{ Then } c_j(C(r_j)) \]  \hspace{1cm} (3)

where antecedent \( a_j \) and consequent \( c_j \) are finite conjunctions or disjunctions (or combinations of both) of fuzzy propositions.

The fuzzy value of the consequent \( c_j \) may be related to the fuzzy value of antecedent \( a_j \) and the certainty factor of rule \( r_j \), by extending (1) under various conditions as discussed in [7] and [8]

1) The antecedent \( a_j \) of rule \( r_j \) is a conjunction \( K \) over propositions \( p_i \) (1 \( \leq \) i \( \leq \) M) and the consequent \( c_j \) is a single
proposition. The fuzzy value of \( c_j \) is obtained as

\[
f(c_j) = C(r_j)^* \min_{p_i \in R} (f(p_i)). (4)
\]

2) The antecedent \( a_j \) of rule \( r_j \) is a disjunction \( D \) over propositions \( p_i(1 \leq i \leq M) \) and the consequent \( c_j \) is a single proposition. The fuzzy value of \( c_j \) is obtained as

\[
f(c_j) = C(r_j)^* \max_{p_i \in D} (f(p_i)). (5)
\]

3) The antecedent \( a_j \) of rule \( r_j \) is a disjunction of \( L(L \geq 2) \) conjunctions \( K_m(1 \leq m \leq L) \) over propositions \( p_i(1 \leq i \leq M) \) and consequent \( c_j \) is a single proposition. The fuzzy value of \( c_j \) can be expressed as

\[
f(c_j) = C(r_j)^* \max_{1 \leq m \leq L} \min_{p_i \in K_m} (f(p_i)). (6)
\]

4) The antecedent \( a_j \) of rule \( r_j \) is a conjunction \( K \) (Case A) or a disjunction \( D \) (Case B) over propositions \( p_i(1 \leq i \leq M) \) and the consequent \( c_j \) is a conjunction of propositions \( q_k(1 \leq k \leq N) \). The fuzzy values of each proposition \( q_k \) in \( c_j \) can be expressed as

\[
\begin{align*}
\text{Case A: } f(q_k) &= C(r_j)^* \min_{p_i \in K} (f(p_i)), \quad 1 \leq k \leq N \quad (7a) \\
\text{Case B: } f(q_k) &= C(r_j)^* \max_{p_i \in D} (f(p_i)), \quad 1 \leq k \leq N. \quad (7b)
\end{align*}
\]

5) The antecedent \( a_j \) of rule \( r_j \) is a disjunction of \( L(L \geq 2) \) conjunctions \( K_m(1 \leq m \leq L) \) over propositions \( p_i(1 \leq i \leq M) \) and consequent \( c_j \) is a conjunction of propositions \( q_k(1 \leq k \leq N) \). The fuzzy values of each proposition \( q_k \) in \( c_j \) can be expressed as

\[
f(q_k) = C(r_j)^* \max_{1 \leq m \leq L} \min_{p_i \in K_m} (f(p_i)), \quad 1 \leq k \leq N. (8)
\]

6) The antecedent \( a_j \) is a conjunction or a disjunction or a disjunction of conjunctions over propositions \( p_i(1 \leq i \leq M) \) and consequent \( c_j \) is a conjunction of propositions \( q_k(1 \leq k \leq N) \). However, the rules of this type are not used for making specific decision. Thus, such rules are not allowed in a knowledge base [1, p. 313].

A Horn formula \( H \) can be expressed as

\[
H: c_j \text{ if } a_j \quad (9)
\]

where \( a_j \) is a conjunction or a disjunction or a combination (both) of propositions called "body" and the consequent \( c_j \) is a conjunction of propositions called "head" of the formula. A fuzzy Horn formula \( H^f \) of fuzzy propositions can be specified as

\[
H^f: c_j \text{ if } a_j(C(H^f)) \quad (10)
\]

where \( a_j \) and \( c_j \) are the same as given in (9) and \( C(H^f) \) represents the fuzziness of \( H^f \).

If \( a_j \) is a conjunction and \( c_j \) contains only one proposition, then the Horn formula is degenerated to a (headed) Horn clause and the reasoning process is given by (4). If \( a_j \) is a disjunction of propositions (or conjunctions over propositions) and \( c_j \) contains only one proposition, then this Horn formula can be degenerated to a number of (headed) Horn clauses equal to the number of propositions (or conjunctions) in \( a_j \). Thus, the reasoning process given by (5) or (6) can be degenerated to the reasoning process of (4). Similarly, the reasoning process given by (8) can be degenerated to the reasoning process of (7a).

III. FUZZY DEDUCTION GRAPHS

Extending the concept of deduction graphs of Yang [13], an FDG can be considered as a graphically structured subset of a rule base \( R \) with fuzzy rules of fuzzy propositions. The rules are represented in the form of Horn clauses/formulae. A proposition in some rule of \( R \) is defined to be simple node in an FDG. A conjunction in the body of a rule in \( R \) is represented by a compound node in the graph. A rule of the form \( H \rightarrow B \) in represented by a full edge from node \( B \) to node \( H \). This full edge is labeled with weight equal to certainty factor of the rule. For a compound node with \( n \) components in its conjunction, there are \( n \) dotted edges from compound node to simple nodes representing components. Each dotted edge is labeled with weight 1, because of reflexivity of the clause i.e.,

\[ b_1 \cdots b_m \text{ implies } b \]

with fuzzy value 1 for \( 1 \leq i \leq m \).

Formally, we define a generalized FDG as a 4-tuple \( \text{FDG} = (V, E, C, f) \)

where

\[
\begin{align*}
V &= \{v_1, v_2, \cdots, v_m\} \text{ is a set of nodes;} \\
E &= \{e_1, e_2, \cdots, e_n\} \text{ is a set of directed edges;} \\
f: V \rightarrow [0,1] \text{ is a function from nodes to the real values in the fuzzy interval [0,1];} \\
C: E \rightarrow [0,1] \text{ is a function from edges to the real values in the fuzzy interval [0,1], such that}
\end{align*}
\]

\[
f(e') = f(v)^* C(e') \quad (11)
\]

where \( e' = \{(v, v')\} \) is a directed edge in \( E \) from node \( v \) to node \( v' \).

Equation (11) indicates that the fuzzy value of a node, on which an edge in incident, is multiplication of certainty factor labeling the edge with the fuzzy value of node from which the edge is emerging. When an FDG represents a subset of rule set \( R \) containing propositions, the following two functions may also be incorporated in the definition of FDG:

\[
g: V \rightarrow \{p_1, p_2, \cdots, p_m\} \text{ is a bijective function from nodes of } V \text{ to propositions;} \\
h: E \rightarrow \{r_1, r_2, \cdots, r_n\} \text{ is a bijective function from edges to rules.}
\]

However, we can notionally use propositions \( p_1, p_2, \cdots, p_m \) for nodes of \( V \) and rules \( r_1, r_2, \cdots, r_n \) for edges of \( E \).

We now illustrate the components of a fuzzy deduction graph with an example of knowledge base of Horn formulae and its graphical representation. Let a fuzzy Horn formula be given by

\[
H \text{ if } B(C)
\]

where head \( H \) is a conjunction of fuzzy propositions, body \( B \) is a conjunction or a disjunction of at least two conjunctions over fuzzy propositions and \( C \) is certainty factor of the formula.

Let \( H = h_1, h_2, h_3, \cdots, h_n \) and \( B = b_1, b_2, b_3, \cdots, b_m \), where each \( h_i(1 \leq i \leq n) \) and \( b_j(1 \leq j \leq m) \) are fuzzy propositions and comma denotes the conjunction. A graphical representation of above formula is given in Fig. 1(a). Here, \( b_1, b_2 \cdots \text{ etc. and } h_1, h_2 \cdots \text{ etc.} \)

are simple nodes; and \( B: b_1 \cdots b_m \) and \( H: h_1 \cdots h_n \) are compound nodes. A fuzzy Horn clause is a degenerated Horn formula where the head \( H \) is a simple node that can be represented graphically by Fig. 1(b).
Given a rule base \( R \) of rules in the form of Horn clauses/formulae, an FDG should be constructed algorithmically to prepare it as input to the algorithm for finding FRP. The construction procedures for an FDG are same as proposed by Horng and Yang for crisp deduction graphs [2]. These procedures are formally given in the Appendix. The first procedure constructs a dependency graph \( G_t \) with respect to the goal proposition \( t \) [12]. In the dependency graph of the rule set \( R \), each proposition is assigned a dependency value as follows:

1. \( D(p) = 0 \) if \( p \) is a base proposition i.e., it is not dependent on other propositions;
2. \( D(p) = \max\{D(q)|(q,p)\} + 1 \), if \( p \) is a derived proposition from \( q \).

The second procedure determines the minimal set \( R_e \) of rules from \( G_t \) which are required for the FRP. The third procedure constructs an array of linked adjacency list containing \( R_e \) to represent FDG. In all these procedures, there is no role of certainty factors of rules and fuzzy values of propositions, but they remain associated with them. In the algorithm for finding FRP, these factors are needed. Therefore, an FDG obtained by above procedures is labeled by the certainty factors and fuzzy values of rules and base propositions, respectively. The FRP is obtained as a sequence of labeled edges representing rules of \( R_e \) from a given source proposition to a given goal proposition.

IV. FUZZY REASONING PATH (FRP) ALGORITHM

A. Informal Description of the Algorithm

The algorithm presented here is based on the forward-chaining using Dijkstra’s shortest path framework and fuzzy reasoning processes discussed in Section II. At any instance, the algorithm determines the greatest fuzzy value of a proposition from the fuzzy values of predecessor propositions. The objective of the algorithm is to find the desired reasoning path in the FDG from \( R_e \), denoted by FRP \((s, t)\) where \( s \) is a node representing the source proposition and \( t \) is a node representing the goal proposition. In other words, the algorithm is required to prove the formula \((t \leftarrow s)\) with the greatest degree of truth value of goal node \( t \) from FDG. It is assumed here that there exists a path from the node \( s \) to the node \( t \). The algorithm starts from the source \( s \), labels it, reads its fuzzy value \( f[s] \), and assigns \( s \) to a variable \( RECENT \) representing current node. For each full edge from \( RECENT \) to a node \( h \), the algorithm assigns the fuzzy value of the node \( RECENT \) multiplied by certainty factor \( C(RECENT, h) \) to the node \( h \) because one rule has been used to derive \( h \). For each dotted edge, emerging from compound node \( u \), it is checked whether this component is reached from the source. When all the components of \( u \) are reached from the node \( RECENT \), the fuzzy value of the compound node \( u \) is found by taking the lowest fuzzy value amongst its components. The process is repeated till the goal node \( t \) is reached. The algorithm determines the set \( R_m \) containing the rules required in the desired FRP. This is done by tracing back from the goal node \( t \) using the recursive call of a function \( fuzzyvalue(t) \). At the end of algorithm, when the goal node \( t \) is reached, the FRP is established along with the greatest fuzzy value of goal node \( t \).

B. Data Structures

The data structures used in the proposed FRP algorithm are listed below.

1. \( V \): A set of nodes of both types: simple and compound in the representation of \( R_c \).
2. \( n \): The cardinality of \( V \) i.e., total number of nodes in the set \( V \) i.e., \( n = |V| \).
3. \( ADJ[v_1 \ldots v_n] \): This is an array of \( n \) linked lists, one for every node \( v_i \) (\( 1 \leq i \leq n \)).
4. \( f[v_1 \ldots v_n] \): This is an array of \( n \) elements where element \( f[v_i] \in [0,1] \), \( 1 \leq i \leq n \) represents the fuzzy value of node \( v_i \) in \( V \). This value is initially known for base nodes (propositions).
5. \( FLAG[v_1 \ldots v_n] \): This is an array of \( n \) elements. The element \( FLAG[v_i] \), \( 1 \leq i \leq n \) represents the status of the node \( v_i \) in \( V \) such that \( FLAG[v_i] \) is true if \( v_i \) is permanently labeled and is false otherwise.
6. \( PRED[v_1 \ldots v_n] \): This is an array of \( n \) elements. The element \( PRED[v_i] \), \( 1 \leq i \leq n \) stores the immediate predecessor of the node \( v_i \) in \( V \) if \( v_i \) is simple.
7. \( INDEG[v_1 \ldots v_n] \): This is an array of \( n \) elements. Each element \( INDEG[v_i] \), \( 1 \leq i \leq n \), stores the number of components in \( v_i \) if \( v_i \) is a compound node.
8. \( C(v_1, v_2) \): This represents a certainty factor in \([0,1]\) of the rule connecting node \( v_1 \) to node \( v_2 \).
9. \( RECENT \): This is a variable representing the current node that has been traversed.

C. Formal Presentation of the FRP Algorithm

Algorithm: FRP Computation using shortest path framework.

Input: A source node (proposition) \( s \) with its fuzzy value.

Outputs: 1) FRP from source node \( s \) to goal node \( t \) such that \( t \) attains the greatest fuzzy value.
2) Fuzzy value of the goal node \( t \).

Procedure FRP(s,t);
begin
for each node \( v \) in \( V \) do
begin
if (\( v \) is the node representing base proposition) then
read(\( f[v] \))
else
\( f[v] := 0 \);
\( PRED[v] := -1 \);
\( FLAG[v] := false \);
\( INDEG[v] := \) the number of components in \( v \), when \( v \) is compound
end;
/* loop */
FLAG[s] := true; RECENT := s;
while (FLAG[t] = false) do
  begin
    for each node u in the list ADJ[RECENT] do
      begin
        if (FLAG[u] = false) then
          begin
            PRED[u] := RECENT;
            temp := f[RECENT]* C(RECENT, u);
            if (f[u] < temp) then
              f[u] := temp;
          end
          else
            INDEG[u] := INDEG[u] - 1;
            if (INDEG[u] = 0) then
              begin
                FLAG[u] := false for all v in V;
                temp := fuzzyvalue(U);
                if (f[U] < temp) then
                  FLAG[u] := false;
            end
        end
      end;
  end
RECENT := a node p with the greatest fuzzy value f[p] and FLAG[p] being false;
FLAG[RECENT] := true;
end; /* of while */
R_m := { }; F := fuzzyvalue(t);
print("Fuzzy reasoning path found with value", F);
end; /* of Procedure */

Function fuzzyvalue(u);
begin
  if (FLAG[u] = true or (u = s) then
    return(f[u]);
  FLAG[u] := true; f := 1;
  if (u is compound) then
    for each component c of u do
      begin
        f := minimum(f, fuzzyvalue(the c))
      end;
  else begin
    f := fuzzyvalue(PRED[u])* C(PRED[u], u);
    R_m := R_m \ { u \leftarrow PRED[u]}
  end;
  return(f);
end; /* of function */
End of Algorithm.

D. Examples
Now we illustrate the applicability of the FRP algorithm with two examples to accomplish the desired FRP.

Example 1: Let R be the following set of fuzzy rules (Horn formulae/clauses) of propositions p_1, p_2, ..., p_8:

\[
R = \{ r_1: p_4 \rightarrow p_2 \ (C = 0.90) \\
    r_2: p_3 \rightarrow p_1, p_2 \ (C = 0.85) \}
\]

We want to establish the FRP between p_1 (source) and p_6 (goal) and determine the degree of truth of proposition p_6 using the FRP algorithm. We first apply Procedure 1 (cf. Appendix) to determine dependency graph G_t with respect to goal node p_6. The dependency values are assigned to the nodes in the dependency graph as indicated in Fig. 2. It may be noted that the rules r_7 and r_8 are not covered in G_t since these are not required in any path for goal node p_6.

The proposition p_1 is a base proposition and others are derived propositions. Now, the minimal set R_c from R with respect to the given goal node can be found using Procedure 2 (cf. Appendix). The set R_c is found as

\[
R_c = \{ r_1, r_2, r_3, r_4, r_5, r_6, r_7 \}
\]

and the set V_c of nodes involved in R_c is given by

\[
V_c = \{ p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8 \}.
\]

Using Procedure 3 (cf. Appendix), the linked adjacency lists can be constructed which represent the deduction graph of R_c. The FDG of R_c and its linked adjacency list representation are shown in Figs. 3 and 4, respectively.

Fig. 3 represents the set R_c in the form of graph G_c = (V_c, R_c). It may be noted that the real numbers in [0,1] labeling the edges
are $C$ values of the rules. Since there is only one base proposition with a given fuzzy value i.e., $p_1$, the node representing $p_1$ is labeled with its fuzzy value $f(p_1)$ initially. This value is supplied in the Procedure FRP(s,t). In Fig. 3, “and” and “or” mean “conjunction” and “disjunction” for compound nodes and simple nodes, respectively. The cardinality of $V_c$ is $n = |V_c| = 8$ and therefore, the graph $G_c$ is represented by an array of eight linked adjacency lists, as shown in Fig. 4. For distinguishing from simple nodes, the compound nodes are shown in square brackets in the list.

The execution of Procedure FRP(s,t) and function fuzzyvalue(t) can be observed by taking $f(p_1) = 0.8$. The output of the algorithm for above example is

Fuzzy reasoning path found with value 0.5187

and the set of rules $R_m$ required for this FRP is

$$R_m = \{r_1: p_2 \leftarrow p_1 \quad (C = 0.90)\}$$
$$r_2: p_3 \leftarrow p_2, p_6 \quad (C = 0.85)$$
$$r_3: p_6 \leftarrow p_3 \quad (C = 0.80).$$

The fuzzy reasoning path is shown by a trace with bold lines in Fig. 5. The each node in the trace is labeled by pair $(i, f)$ where $i$ is the iteration number and $f$ is the fuzzy value of the node obtained in the $i$th iteration.

**Example 2:** We now consider Chen’s example [1, p. 316] for comparison. Let $p_1, p_2, \cdots, p_9$ be nine propositions and $R$ be the set of fuzzy rules (Horn formulae/clauses) involving these propositions

$$R = \{r_1: p_2 \leftarrow p_1 \quad (C = 0.85)\}$$
$$r_2: p_3 \leftarrow p_2 \quad (C = 0.80)$$
$$r_3: p_4 \leftarrow p_2 \quad (C = 0.80)$$
$$r_4: p_5 \leftarrow p_4 \quad (C = 0.90)$$
$$r_5: p_6 \leftarrow p_1 \quad (C = 0.90)$$
$$r_6: p_4, p_9 \leftarrow p_6 \quad (C = 0.95)$$
$$r_7: p_7 \leftarrow p_1, p_8 \quad (C = 0.90)$$
$$r_8: p_4 \leftarrow p_7 \quad (C = 0.90)\}.$$

We want to find the FRP between $p_1$ and $p_4$, i.e., FRP($p_1$, $p_4$), the dependency graph with respect to $p_4$ is first obtained by Procedure 2 as given in Fig. 6. The dependency values $D(p)$ of the propositions are also indicated in Fig. 6. $p_1$ and $p_4$ are base propositions, their fuzzy values need to be given as input to FRP algorithm. Let $f(p_1)$ and $f(p_4)$ be 0.8 and 0.7, respectively.

Next, the minimal set $R_c$ is found using Procedure 2 as

$$R_c = \{r_1, r_2, r_3, r_4, r_5, r_7, r_8\}$$

with the node set

$$V_c = \{p_1, p_2, p_3, p_4, p_5, p_6, [p_4, p_9], [p_1, p_8]\}.$$

Now, using Procedure 3, the deduction graph $G_c = (V_c, E_c)$ in the form of linked adjacency list can be obtained. Figs. 7 and 8 show the FDG and its linked adjacency lists representation for rules of set $R_c$, respectively.
instance, the reasoning path should contain the sequence of those edges which have the certainty factors above bold-faced lines. The time-complexity of proposition to a goal (consequent) proposition. Our algorithm obtains the fuzzy reasoning path (FRP) using FDG. We have also developed a systematic framework (based on Dijkstra's shortest path) for finding the minimal number of edges.

Another condition may be to restrict the reasoning path to contain a threshold level for certainty factors of the rules for firing. For a given threshold, the FRP algorithm finds the reasoning path without finding other reasoning paths to the goal node. If some other reasoning path is desired, the appropriate conditions may be determined.

There are many expert systems [4], [10], expert control design [6], etc. which shall happen in the most of the cases, then the performance of the proposed FRP algorithm is better. The proposed concept of FDG and FRP algorithm can be useful in many AI applications such as expert systems [4], [10], expert control design [6], etc.

APPENDIX

Procedure 1: Construct dependency graph Gt with respect to goal node t.
Input: A set of rules and goal node t
Output: Dependency graph with Gt = (Vt, Et) begin

1: Vt := {t}; Et := {};
while (there is a rule in R whose head proposition is in Vt) do begin

2: select a rule r in R of the form
2.1: q if p1, p2, ..., pk such that q is in Vt;
3: R := R - {r};
4: Vt := Vt \ {q}; Et := Et \ {(p1, q), (p2, q), ..., (pk, q)};
end;
end; /* of Procedure 1 */

Procedure 2: Create minimal set Rc
Input: A set R of rules, goal node t, dependency graph with Gt = (Vt, Et) and the dependency value D(p) of each p in Vt
Output: Minimal set Rc begin

Rc := {}; S := {}; /* S is a work variable */
while (D(p) ≠ 0 for all p in S) do begin

1: select a proposition p in S with the maximum dependency value D(p);
2: add all the rules rj of R where head contains p into Rc;
3: R := R - {rj};
4: S := S \ {p};
5: S := S ∪ {q} if q is in body of rj;
end; /* of Procedure 2 */

Procedure 3: Create linked adjacency lists from Rc
Input: Minimal set Rc
Output: Linked Adjacency List ADJ[v1 ... vn] to represent Rc graphically.
begin
n := |Vc|;
for i := 1 to n do
ADJ[v[i]] := nil;
while (Rc ≠ {}) do begin

1: select any rule r in Rc of the form p → q;
2: Rc := Rc - {r};
3: insert a list node for p in the front of ADJ[q];
if q is compound node and is of the form "q1 ... qn" for n > 1 then
4: insert a list node for q in the tail of list ADJ[q] for 1 ≤ i ≤ n;
end; /* of Procedure 3 */
The authors wish to thank the anonymous reviewers for their comments.

REFERENCES


A Neural Architecture for a Class of Abduction Problems

Ashok K. Goel and J. Ramanujam

Abstract—The general task of abduction is to infer a hypothesis that best explains a set of data. A typical subtask of this is to synthesize a composite hypothesis that best explains the entire data from elementary hypotheses which can explain portions of it. The synthesis subtask of abduction is computationally expensive, more so in the presence of certain types of interactions between the elementary hypotheses. In this paper, we first formulate the abduction task as a nonmonotonic constrained-optimization problem. We then consider a special version of the general abduction task that is linear and monotonic. Next, we describe a neural network based on the Hopfield model of computation for the special version of the abduction task. The connections in this network are symmetric, the energy function contains product forms, and the minimization of this function requires a network of order greater than two. We then discuss another neural architecture which is composed of functional modules that reflect the structure of the abduction task. The connections in this second-order network are asymmetric. We conclude with a discussion of how the second architecture may be extended to address the general abduction task.

I. ABDUCTIVE INFERENCE

Abduction is inference to the best explanation for a given set of data. The general task of abduction takes as input a set of data and gives as output a hypothesis that can best explain the input data. A typical subtask of the abduction task is classification of the given data into potentially relevant elementary explanatory hypotheses stored in memory [3], [22]. In the classification task, the stored elementary hypotheses are matched with the data, and, depending on the degree of match, a prima facie belief value for each explanatorily relevant hypothesis is determined. For simple abductive problems, for instance diagnosis under the single fault assumption, the classification subtask may yield elementary hypotheses that can individually explain the entire data. For such problems, the elementary hypothesis with the highest belief value represents the best explanation.

In general, however, an elementary hypothesis that can explain the entire data may not be available in memory. Instead, a composite explanation has to be synthesized from elementary hypotheses that can explain various portions of the data [3], [22]. One composite hypothesis is operationally better than others depending on factors such as explanatory coverage, plausibility, parsimony, and internal consistency. Synthesizing a composite hypothesis that satisfies these criteria for the best explanation, however, is computationally very expensive, more so in the presence of certain types of interactions between the elementary hypotheses [1], [3]. This raises the issue of how to rapidly synthesize composite explanations from elementary explanatory hypotheses.

In [17] we proposed the exploitation of concurrency in the synthesis of composite explanations and described a distributed-memory message-passing architecture for this purpose. Our work on concurrent synthesis of composite explanations led us to think in terms of artificial neural networks for the abduction task. From a concurrent-processing viewpoint, neural networks form an attractive proposal for an efficient, fine grained, massively parallel machine dedicated

Manuscript received June 2, 1992; revised September 14, 1995.
A. K. Goel is with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280 USA (email: goel@gatech.edu).
J. Ramanujam is with the Department of Electrical Engineering, Louisiana State University, Baton Rouge, LA 70803 USA.
Publisher Item Identifier S 1083-4419/96/05412-X.