IEEE Region 10 Conference on Computer and Communication Systems, September 1990, Hong Kong

16

A graph Theoretic Approach for Knowledge Organisation in Rulebased Systems

H. Krishnamurthy
Supercomputer Edn. & Res. Centre

N.Balakrishnan
Department of Aerospace Engg.

Indian Institute of Science
Bangalore 560 012
INDIA

ABSTRACT

Expert systems are strongly characterised by their use of a large collection of domain specific knowledge acquired from human experts. Knowledge acquired from the expert over a length of time tends to be inconsistent and redundant thus requiring enormous amount of storage. In the case of rulebased systems knowledge has to be organised to make it complete, concise and consistent. This paper discusses a graph theoretic approach to achieve the same.

Introduction:

Knowledge organisation is one of the important issues addressed in the development of Expert Systems. For representing the domain knowledge in an expert system the approaches that are extensively used can be broadly classified into two categories viz. rule based approach and net based approach. In expert systems, knowledge is usually described as relations among concepts and then listed as rule bases or connected as semantic networks or grouped as frames. Expert systems currently developed use both these approaches for knowledge representation. In the case of net based systems, knowledge is represented as semantic nets and frames and these two knowledge structures have organisation embedded into them in some sense. Representation of semantic nets is similar to a directed graph and that of frames to a generalised tree structure. For these systems modification of the existing knowledge is done at an appropriate node and efficient graph and tree traversal algorithms exist for this. For rule based systems it is important to have a mechanism to organise the knowledge to improve the efficiency of the system. Knowledge organisation of rulebased systems aims at achieving a knowledge base which is complete, concise and consistent. Completeness of a knowledge base deals with the aspect of identifying whether all specified knowledge is included in a system. Given the initial assertion one should be able to reach all possible conclusions with the help of a reasoning mechanism. Conciseness of knowledge base is to eliminate unwanted information and also to combine simple rules to form complex ones so that the storage required for the knowledge base is minimum. Consistency of a knowledge base is an important area of research and deals with addressing several issues which can help in arriving at a knowledge base which removes even hidden inconsistencies. In the case of net based systems it is fairly easy due to the inherent structuring [1]. In a rulebased system, rules are independent of each other and their organisation plays an important rule in improving the search for a solution. Any addition, deletion and midification or knowledge to the existing knowledge base does not guarantee consistency. This paper discusses a graph theoretic algorithm for knowledge organisation of rule based systems.

Representation of knowledge using data-structures:

In constructing an expert sytem, a wide variety of knowledge must be represented so that it can be effectively used in solving a complex problem. The expert systems currently built use a combination of different types of knowledge structures for representation. This section discusses briefly the commonly used knowledge structures in the development of an expert system [2].

Production rules:

The type of knowledge structure that is chosen for an expert system depends on the task the system is expected to perform. If the expert system requires an evolutionary knowledgebase and support of interactive consultation as in MYCIN then production rules offer a knowledge representation that greatly facilitates the accomplishment of the above. If the number of rules required to solve a given complex task is more and each one of the rules is of a complex type then this datastructure can give good stylization and modularity but has to be organised to make storage and retrieval efficient. The later sections will discuss some of the examples of these type.

Semantic Nets:

A semantic network is a graph in which nodes are objects in some domain of discourse and each arc is labelled with a relation that exists between objects it connects. Hence it can be modelled as a labelled directed weighted graph. For example one can have a semantic network representation for the following example as in Fig. 1.

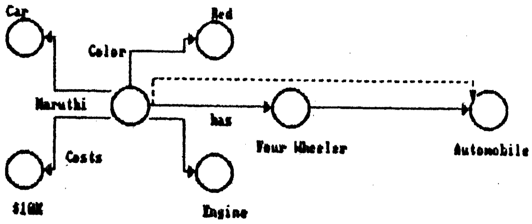

Figure 1

Maruthi is a car.
A car is a four wheeler.
A four wheeler is an automobile.
Maruthi has an engine.
It is RED in color.
It costs $10,000.

Frames:

A frame is a method of representing information. Frames are particularly useful in representing information that is relatively complex. Frame is a list of properties that a particular type of object possesses. For example, a frame or list of properties that an automobile possesses is

manufacturer
make
year
license
engine
electical system
fuel system etc.,

Storage and retrieval of knowledge can be done efficiently for either the framebased or netbased knowledge structures using efficient traversal algorithms. This will help in addition, deletion and modification of knowledge at the appropriate node. But the inference process to achieve reasonable conclusions can be a rulebase.

Necessity for knowledge organisation:

This section discusses the necessity for organising the rulebase of an expert system. Rulebase systems are those which have a certain number of IF-THEN rules which are assumed to be true. Based on some initial set of facts or assertions the rule deduces new conclusions. If the knowledge base isn't properly organised it may be necessary to start the rule checking procedure once again from the beginning each time a new conclusion is deduced. For example consider the following set of rules in a forward chaining approach.

R1 : X and Y --- Z
R2 : P and Q and R --- X
R3 : W --- Y

The assetions are P, Q, R and W.

The inference mechanism traverses the rules R1, R2 and R3 in the given order. It can be seen that R1 will not fire since X and Y are not available as assertions. R2 and R3 will fire and with the conclusions added to the assertion list, R1 will fire to decide that Z is true. From this simple example it can be seen that in a knowledge base if there are two rules R_i and R_j (i-j) then a proper organisation will lead to the fact that for firing R_i, R_j cannot be a precondition. Generalising this one can say that it is necessary, in the forward chaining algorithm to return to the FIRST rule each time a conclusion is added to the fact list. This approach will lead to examining the same set of rules again with new set of assertions. But in a large system it is not always possible to organise rules to avoid this condition because all the inputs are not available apriori. In the interactive situation after a rule R_a fires and the conclusion is a question to the user to provide his input then the next rule that can fire depends on the users response. So in such cases the rules have to be organised to avoid any unnecessary backtracking.

Graph Theoretic Approach for Knowledge Organisation:

The main thrust of this paper is to store the rules as a graph structure. In this approach, all the rules are stored in a graph in which each rule is represented by a special count node with premises coming in and conclusions leaving. The assertions are placed in a single linked list.

The algorithm explained in this procedure checks for redundant rules, conflicting rules subsumed rules, circular rules, unreachable and dead-end clauses. This brings to the fore the hidden inconsistencies in the knowledge base which are not detected by scanning these rules. For example a certain combination of rules which has a circular property is identified by a procedure which detects a directed circuit. Similarly procedures are implemented for checking other aspects like redundancy etc. This algorithm is implemented in Turbo Pascal on an IBM PC.

Outline of the algorithm:

The essential steps of the algorithm used to achieve the organisation of knowledge is as follows. It is assumed for illustrative purposes that the inference mechanism adopts a forward reasoning approach even though the algorithm will also work for goal-directed backward reasoning with minor modifications. In the forward reasoning approach the assertions are kept in a linked list [4]. The algorithm constructs a graph for the given rulebase. Redundant rules are represented in the graph by parallel edges. It is possible to identify parallel edges and remove one of them. Subsumed rules are identified by the fact that one rule is modelled as a subgraph of the other. A simple example is given in Fig.2.

A → B
A/\B → C

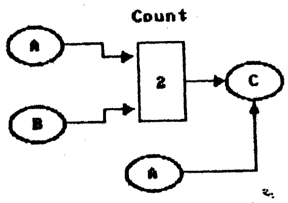

Figure 2

Circular rules correspond to the situation of chaining of the rules in a set forming a cycle an example of which is given in Fig. 3.

A → B
C → D
B/\D → E
E → F
F → A

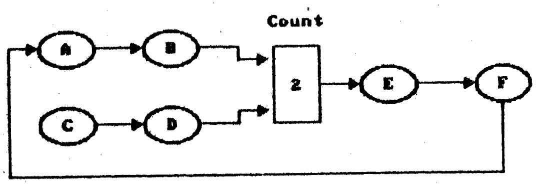

Figure 3

The graph traversal of the node set ABEFA will identify a directed circuit. In a large rulebase there may be hidden as well as explicit cycles which are due to a subset of the rulebase. The algorithm will identify that subset and the user can modify the knowledge base to break the cycle. Apart from these several other graph theoretic properties help us to identify unreachable and dead-end clauses. The nodes in the graph with zero in-degrees and non-zero out- degrees (called source nodes) are the facts corresponding to initial assertions. Similarly the nodes with zero out- degrees and non-zero in-degrees (called sink nodes) are the goal conditions. For example the graph given in Fig. 4 explains a simple situation.

INITIAL ASSERTIONS A,B,E

RULE BASE R1 : A → C
R2 : B → D
R3 : D/\E → F
R4 : F → G
R5 : F/\H → I
R6 : C/\G → K
GOAL CONDITIONS ARE I,K

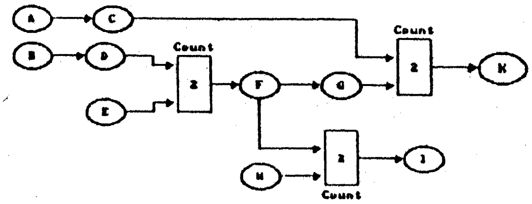

Figure 4

The algorithm which explains the methodology is given in Appendix.

Conclusions

One of the important aspects in the development of an Expert system is Knowledge organisation. The organisation of knowledge plays a key role in making the reasoning procedure efficient. Most of the commercially available expert systems use the rulebased approach as the main knowledge structure for representation. This paper discusses a graph theoretic approach to detect the inconsistencies in a knowledgebase and organise the same to make it complete, consise and consistent.

References

1) H. Krishnamurthy and M.N. Murty, A Conceptual clustering scheme for frame-based knowledge organisation, Proceedings of SPIE, Applications of Artificial Intelligence, Vol. 635, p 408, 1986.

2) D.A. Waterman, A Guide to Expert Systems, Addison-Wesley Pub. Co.

3) Y. Chang and K.S. Fu, Conceptual clustering in knowledge organisation, IEEET on PAMI, Vol. 7, Sept. 1985.

4) R.E. Neopolitan, Forward-chaining versus a graph approach as the inference engine in expert systems, Proceedings of SPIE, Application of AI, Vol. 635, 1986.

APPENDIX

Algorithm:

```
MAIN KNOWLEDGE_ORGANIZATION;

Begin
Read the rules;
   call REDUNDANT_SUBSUMED_RULES_
   DETECTION;
   call CONFLICTING_RULES_DETECTION;
   call CIRCULAR_RULES_DETECTION;
   call OPTIMIZE_RULES;
end.

REDUNDANT_SUBSUMED_RULES_DETECTION;

Begin
   Set I = First rule;
   while more rules do
   begin
     Set J = I + 1th rule;
     while more rules do
     begin
       If  number of conclusions in the
          Ith rule conclusion part is
          equal to the number of conclu-
          sions in the Jth rule conclu-
          sion part and Ith rule conclu-
          sion is the subset of Jth rule
          conclusion
       Then If number of premises in the
          Ith rule premise part is equal
          to the number of premises in
          the Jth premise part
       Then If Ith rule premise is the
          subset of Jth rule premise
       Then Print Rule I and J are
          redundant
       Else If number of premises in the
          Ith rule premise part is
          greater than number of prem-
          ises in the Jth rule premise
          part and Ith rule premise is
          the subset of Jth rule premise
          or Jth rule premise is the
          subset of Ith rule premise
       Then Print Rules I and J are
       subsumed rules;
     end;
  end;
end;

CONFLICTING_RULES_DETECTION;
begin
  Set I = First rule;
  while more rules do
  begin
    Set J = I+1th rule;
    while more rules do
       begin
         if  Ith rule's premise part is
            equal to Jth rule's prem
            ise part AND Ith rule's
            conclusion part is not equal
            to Jth rule's conclusion
            part
         Then Print Rules I and J are
            conflicting;
       end;
    end;
  end;

CIRCULAR_RULES_DETECTION;

Begin
  Set I = First rule;
  while more rules AND loop is not
  detected do
    begin
      Initialize the two dimensional loop
      array;
      while more conclusions for Ith
      rule do
      begin
         Initialize circular loop array;
         Set T = Ith rule's present
         conclusion;
         while more conclusions for T do
         begin
           Check whether the present
           conclusion of T is present
           in the premise part of any
           other rule (say K) and that
           rule is not present in the
           circular loop array;
           If K is equal to the current
           rule I
           Then Loop is detected, thus
              update the loop array
              with circular loop array;
           Else If K is not equal i.e
                conclusion is present in
                the premise part of any
                other rule other than
                current rule

              Then Add this rule to the
                circular loop array and
                update T as the Kth
                rule's conclusion part
              Else Set T as the conclu-
                 sion part of previous
                 rule found in the circ-
                 ular loop array
         end;
      end;
      Sort the loop array to get the
      innermost loop;
      Display the innermost loop obta-
      ined from loop array;
    end;
end;

OPTIMIZE_RULES;

Begin
   Initialize optimize array;
   Set I = first rule;
   while more rules do
   begin
      If Ith rule is not present in the
         optimize array
      Then Check whether the Ith rule's
         premise part is present in the
         conclusion part of any other
         rule;
         If no such rule
         Then Set CON = Ith rule's
            conclusion and Put Ith rule
            in the optimize array;
            Set J = First rule;
            While more rules do
            begin
              If Ith rule's premise
                 part is present in the
                 conclusion part of any
                 other rule
              Then Put Jth rule in the
                 optimize array and CON=
                 Jth rule's conclusion;
            end;
            Setup the optimize array
            for next set of optimize
            rules;
         end;
         Print the optimize array;
      end;
```