MTA SZTAKI Közlemények 37/1987 pp. 117-124

PESS - A PROLOG EXPERT SYSTEM SHELL USING INEXACT REASONING

HA HOANG HOP

Hanoi State's Committee for Science and Technology
Hanoi - Vietnam

1. INTRODUCTION

Inexact reasoning in expert systems (ESs) is one of the recent attentions in the area of designing ES. We have agreed that no single inference mechanism would appear to be more accepted in application than another. The natural consequence is to combine some inference mechanisms into an unified module which provides the user with various reasoning schemes. In this paper, PESS - a prototype of ES shell in PROLOG is presented to realize this idea. The emphasis is also aimed to PROLOG - a tentative tool for representation of uncertainty in ES. PROLOG has some significant features that aid the development of ESs. Expert system are computer programs intended to solve problems usually requiring human expertise. They typically consist of a knowledge base and an inference engine that provides mechanisms or schemes for symbolic reasoning, search and explanation.

Knowledge is usually represented in ES as production rules having the form:

IF condition THEN action or
IF antecendent THEN conclusion

One of the most common mode of inference in ES is backward changing - the input is viewed as a goal to satisfy, and the production rules are used to generate subgoals that must in turn be satisfied. The process terminates when it is asserted - either by facts in knowledge base or by the inference engine, that the goal is satisfied (succeeded) or failed.

In designing ESs, one has to consider processes of knowledge acquisition, knowledge representation, knowledge-based inference, and search-based computation along with facilities for explanation and development of expert system shells.

Building an ES shell is one of the current approaches to the problem of representing and propagating uncertainty through ESs. An ES shell can be considered as an ES without its expert knowledges, i.e., as a collection of inference mechanisms, and the facilities for providing explanation, development and debugging of the system. PESS is actually a PROLOG meta-interpreter that accounts for certainty factors by various inexact reasoning schemes.

2. INEXACT REASONING

The presence of human knowledge in an ES can be associated with at least four types of uncertainty [8]. The first type is related to the reliability of knowledge: uncertainty can be presented in factual knowledge as a result of ill-defined concepts in the observations. The second type of uncertainty is caused by the inherent imprecision of rule representation language relating to the problem of semantic matching which compares the approximate meaning of facts and premises. The third type appears when inference is based on incomplete information. And the fourth type arises from the aggregation of rules from different knowledge sources or different experts.

Here, we briefly review some inexact reasoning schemes that are used in PESS to provide a way of conbining these different types of uncertainty. We gradually consider 3 levels of representing knowledge: preliminary knowledge, rules and rules framework.

2.1. Preliminary factual knowledge

Preliminary factual knowledge is an ordered triple:

(attribute, object,value)

where value is an element of the attribute domain, that domain may be discrete or continuum set, and object may refer to as a combination of more elementary objects. A natural way to attach uncertainty to this triple is

(attribute, object, value) with CM

where CM is some certainty measure (for example, either of probability, necessity, possibility measures/distributions).

2.2. Representation of rules

Rule, in general, has the form:

IF condition THEN action or
IF antecedent THEN conclusion.

Suppose that condition (antecedent) and action (conclusion) are conjunction of triples. With this, rules are in higher levels of knowledge representation.

Uncertainity is fetched to rule by

IF condition THEN action with CM

2.3. Rule framework

In more complicated application of ES, we need more available and larger rule base, and so, we may pre-establish rule bases as framework [3],[4] with some meta-level inference.

In PRESS, we use a simple rule framework for providing the combination of certainty measures.

3. DESCRIPTION OF PESS

PESS consists of the following parts:

- Rule base in which each rule is associated with a certainty measure CM, a submodule of this rule base is served for meta-rules, in which, each meta-rule has a certainty measure that is computed by some predicates implemented in control block.

- Control block: the inference engine of PESS which uses the natural backward chaining inherited from PROLOG, besides, in this block, there exists an uncertainty handler allowing PESS to reason with different reasoning schemes (3 - valued logic reasoning [5], reasoning with threshold, EMYCIN like-reasoning [6], Bayesian). The uncertainty handler is implemented by modular and transparent manner for various reasoning schemes. See the Appendix for more details.

- User interface provides the user with window system (dialog box, pop-up menu for choosing inference mode and calls for macros/predicates to compute the combination of CM).

4. IMPLEMENTATION AND CONCLUSION

The first PESS's prototype is implemented in micro-PROLOG and for the time being, we are extending this prototype to have some debugging facilities.

This paper shows PESS - a ES shell in PROLOG that owns some good features:

(i) Different inexact reasoning schemes are combined in a single block.

(ii) The modular structure of PESS makes easy to expand with another reasoning schemes.

5. APPENDIX

The EMYCIN like-reasoning.

The rule has form:

IF antecedent THEN conclusion with CF (antecedent, conclusion), where CF is certainty factor that measures the degree to which the implication is valid.

To compute CF, we use the following expression:

CF(conclusion) = CF(antecedent) x CF(antecedent, conclusion), if CF(antecedent) > 0.2

if CF(antecedent) is less than 0.2, the rule is unappropriate, [6].

In the case, when antecedent is combined by more than one elementary antecedents Ai(i = 1,n) we use the expression below to compute CF(conclusion):

$$
\begin{aligned}
CF(conclusion) &= CF1(conclusion) + CF2(conclusion) - \\
&\quad - CF1(conclusion) \times CF2(conclusion) \\
&\quad \text{if } CF1(conclusion) >= 0 \\
&= CF1(conclusion) + CF2(conclusion) \\
&\quad + CF1(conclusion) \times CF2(conclusion) \\
&\quad \text{if } CF1(conclusion) \text{ and } CF2(conclusion) \\
&\quad \text{both less than } 0 \\
&= \frac{CF1(conclusion) + CF2(conclusion)}{1 - \min\{|CF(conclusion)| - |CF2(conclusion)|\}} \\
&\quad \text{if } -1 < CF_1(conclusion) \times CF2(conclusion) < 0 \\
&= 1 \\
&\quad \text{if } CF1(conclusion) \times CF2(conclusion) = -1.
\end{aligned}
$$

Repeat this expression for all antecedents Ai (i=1,n) we then receive the total CF(conclusion).

In general, antecedent in this mode of reasoning can be logical combination of factual informations. So, we can compute CF through the following formulas

$$CF(F1 \text{ or } F2) = \max\{CF(F1), CF(F2)\}$$
$$CF(F1 \text{ and } F2) = \min\{CF(F1), CF(F2)\}$$
$$CF(\text{not } F) = -CF(F)$$

where F1,F2 are factual informations conjuncting to form the antecedent A.

By PROLOG, we have defined 4 predicates for and, or, not combine operations.

```
and_op(cf(A),cf(B),cf(C)) :-min(A,B,C).
or_op(cf(A),cf(B),cf(C))  :-max(A,B,C).

not_op(cf(X),cf(Y)) :- Y  is  -X.
combine_op(cf(Conclusion),cf(Antecedent),cf(Rule),
                cf(NewConclusion) :-
                Medium is Antecedent*Rule,
                c(Conclusion,Medium,NewConclusion).

c(A,B,C) :-A >= O, B >= O,!,C is (A+B-A*B).
c(A,B,C) :-A >= O, B <  O,!,T is -B, min(A,T,M),
              C is (A+B)/(1-M).
c(A,B,C) :- A  <O,B>= O,!,T  is  -A,min(T,B,M),
              C is (A+B)/(1-M).
c(A,B,C) :- A < O,B < O,!,C  is  (A+B+A*B).
```

Another modes of reasoning can be found in [4].

REFERENCES

[1] B.G. Buchanan, E.H. Shortliffe, Rule-based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project, Addison-Wesley, Reading, MA (1984).

[2] W.F. Clocksin, C.S. Mellish, Programming in Prolog, Springer-Verlag (1981).

[3] F. Hayes-Roth et al., Building Expert Systems, Addison-Wesley, Reading, MA (1983).

[4] Ha Hoang Hop, Using PROLOG to represent uncertainty in Expert Systems, (forthcoming).

[5] S. Kleene, Introduction to Metamathematics, Van Nostrand, (1951).

[6] A.C. Scott et al., EMYCIN Manual, Stanford University (1981).

[7] G. Shafer, A Mathematical Theory of Evidence, Princeton University Press, Princeton, NJ (1976).

[8] Special Issues on Representation and Propagation of Uncertainty in Expert Systems, International Journal of Man-Machine Studies 22 (1985).

PESS - A PROLOG expert system shell using inexact reasoning

Ha Hoang Hop

Summary

This paper describes PESS - a PROLOG expert system which deals with the problem of representing and propagating uncertainty in expert systems /ESs/ in context of various inexact reasoning schemes.

PESS - egy PROLOG tipusu szakértő rendszer-váz pontatlan indoklási sémák esetében

Ha Hoang Hop

Summary

A cikkben a cimben emlitett szakértő rendszer-vázat ismerteti a szerző, különböző pontatlan indoklási sémák /reasoning schemes/ esetén. Főleg a szakértő-rendszerekben levő bizonytalanság reprezentálásának és terjedésének problémáját tárgyalja.