PERFORMANCE EVALUATION OF THE INFERENCE STRUCTURE IN EXPERT SYSTEM

Yang Tao He Zhijun Yu Ruizhao
Artificial Intel1igence Laboratory
Department of Computer Science and Engineering
Zhejlang University, Hangzhou, P.R. China

ABSTRACT

There are a variety of factors that affect the performance of an expert system. This paper presents an evaluating method focusing on the complexity, costs and efficiency of inference structure in a rule / Knowledge-source based expert system. The performance measuring factors proposed here, when fedback to Knowledge engineers, can help them gain a quantitative understanding of reasoning performance of Knowledge acquired at the (early) stages of the development life cycle, and enable them to make a numeric comparison and a choice among several structures, or have some refinements. This paper also presents some application examples of this evaluating method and describes an evaluating facility based on it.

I INTRODUCTION

In the last decade, expert systems and tools for building them have been being developed at a surprisingly rapid rate [Waterman, 1985]. This makes evaluation of expert system more and more important in its development process. As pointed out in [Gaschnig, et al., 1983], evaluations pervade the expert system building process and are crucial for improving system design and performance, but the existing techniques for evaluating these systems are few and primitive. Certainly many evaluating methods used in other software systems [Shneiderman, 1980] could be applied to expert systems, but an expert system is unique in the sense that it is a Knowledge-based system containing human expertise, and has many distinguishing features such as heuristic searching, reasoning, high level representation , etc.. Therefore, expert system should have its own evaluating techniques.

Gaschnig and others in their paper have presented many general evaluating criteria concerning on what, how, and when the evaluation should be .performed. Our approach , on the other hand, emphasizes the performance of inference structure which means complexity, costs or efficiency in the context of this paper, i.e, the degree of difficulty, effort, or the reasonable time for a system with such an inference structure to attain certain goals. As was proposed in [Newell, 1982], there does exist a distinct computer system level, the Knowledge level, lying immediately above the program level. Accordingly, our point is that this evaluation is performed at the Knowledge level whereas the others for common software systems are performed at the program level.

J. McDermott, et al.CMcDermott, et al., 1978] have first made an approach and presented several exciting results on the efficiency of certain production system implementations focusing on impacts of conflict set finding strategies. The evaluating method presented here is towards the inference structure itself, as the reasoning network constituted by the intrinsic relationships of Knowledge has a great influence on the performance of expert system.

The application significance of this evaluating method has been shown in the sense that it helps Knowledge engineers
(1) obtain a quantitative measure of inference performance of acquired Knowledge. Example 1, for instance, gives the evaluating result of an expert system.
(2) make a numeric comparison of performance among several structures (normally with the same function in problem solving). For instance, Example 2 analyzes the impacts of a generalization strategy on the inference performance, and Example 3 compares two mechanisms for pattern consistency checking.
(3)choose, refine or reorganize the

inference structure without the destruction of the function to enhance the performance of the system, and speed up the reasoning. For instance, Example 2 indicates the effectiveness of this generalization strategy for knowledge refinement, and Example 3 shows that a mechanism for pattern-consistency checking can be improved further.

The development of this model is motivated by our work on expert system development environment [He, 1986], as knowledge acquisition may be performed after or before the selection of supported system building tools. Also in KMIX [Yang, et al., 1986], it is possible to modify the control strategy statically or dynamically. That is why we need to make an estimation on inference structure itself, and have a prior probabilistic perspective of performance in control strategy applied. This will enable us to have an approximate preview of performance at early stages in the development life cycle.

II THE NETWORK MODEL OP INFERENCE STRUCTURE

The first stage of our evaluating method is to abstract the inference structure from a collection of knowledge piece embodied within an expert system . The following network model of inference structure is adapted from the theory of Petri Network [Peterson, 1981].

An inference structure, R, is defined as a four-tuple, R=(S, T, I, O), where $S=\{s_1, s_2, \ldots, s_n\}$ is a finite set of states ; $T=\{t_1, t_2, \ldots, t_m\}$ is a finite set of inference transitions, $S \cap T = \emptyset$; I is an input mapping function defined as $I: S \dashrightarrow 2^T$ or $T \dashrightarrow 2^S$; O is an output mapping function defined as $O: S \dashrightarrow 2^T$ or $T \dashrightarrow 2^S$. 2^S , 2^T are the power sets of S, T respectively.

The semantics of S and T are that S is the abstract specification of static properties of knowledge; T , on the other hand, their intrinsic inference relationships. If each state s_i in $I(t_k)$ satisfies the property predicate $P_{ki}(s_i)$, then each state s_j in $O(t_k)$ satisfies the property predicate $Q_{kj}(s_j)$.

An inference structure can be depicted via a graph: a circle represents a state, and a bar represents an inference transition.

An inference structure satisfies the following properties:

(1) $\forall\, i,j\ (\#(t_i, I(s_j))=\#(s_j, O(t_i)))$
$\forall\, i,j\ (\#(t_i, O(s_j))=\#(s_j, I(t_i)))$
where #(x,y) is the number of element x in y, y is a bag allowing the multiple appearance of an element.

(2) $\forall\, i\ (\ I(t_i)\neq\emptyset \wedge O(t_i)\neq\emptyset\ )$
$\exists\, j\ (\ I(s_j)\neq\emptyset \wedge O(s_j)=\emptyset\ )$
$\exists\, j\ (\ I(s_j)=\emptyset \wedge O(s_j)\neq\emptyset\ )$

(3) There exists no inference cycle, i.e., for any partial inference path $s_{i_1}, t_{j_1}, s_{i_2}, t_{j_2} \ldots s_{i_n}, t_{j_n}, s_{i_{n+1}}$, where t_{j_k} is in $O(s_{i_k})$ for $k=1,\ldots,n$ and s_{i_k} is in $O(t_{j_{k-1}})$ for $k=2,\ldots,n+1$, $s_{i_1} \neq s_{i_{n+1}}$.

Property 1 validates the correctness of function I and O; Property 2 warrants the initiality and terminality of reasoning; Property 3 excludes the self-reference of inference transitions. This constraint is used to prevent reasoning loops and is further discussed in Section IV. The latter two statements in Property 2 can be derived directly from Property 3 (the proof is omitted here).

In the rest of this section, we illustrate an inference structure Rc abstracted from rules 5, 6, 7, 8 and 23, and the related parameters specified in [Van Melle, Chapter 3], which is part of the inference structure in CLOT [Buchanan&Shortliffe, 1984]. The transition is named in accordance with the rule, so is the state to the parameter.

Rc = (S, T, I, O)
S = {clindef, family, sigbleed, onset, bldtype}
T = {t5, t6, t7, t8, t23}
I(clindef) = {t5, t6, t7, t8}
O(clindef) = $\emptyset$
I(family) = $\emptyset$
O(family) = {t5}
I(sigbleed) = {t23 }
O(sigbleed)={t5,t6,t7,t8}
I(onset) = $\emptyset$
O(onset) = {t6}
I(bldtype) = $\emptyset$
O(bldtype) ={t7,t8,t23}
I(t5) = {family, sigbleed}
O(t5) = {clindef}
I(t6) = {sigbleed, onset}
O(t6) = {clindef}
I(t7) = {sigbleed, bldtype}
O(t7) = {clindef}
I(t8) = {sigbleed, bldtype}
O(t8) = {clindef}
I(t23) = {bldtype}
O(t23) = {sigbleed}

Figure 1 shows the graphical representation of inference structure Rc.

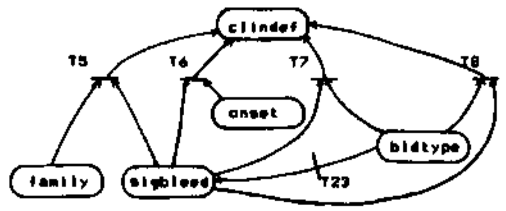

Fig. 1 Inference structure Rc.

III ESTIMATIONS OF THE PERFORMANCE MEASURING FACTORS

The next stage of our method is to derive the z-transform performance function of the inference structure abstracted.

Generally, the control strategies which can be applied to inference structure are basically data-driven, goal-directed, and Opportunistic searching. During their applications, the efforts of testing, matching, activating, acting, switching, etc. of inference transitions have the greatest influence on the speed of reasoning. Thus in our approach, each transition t in T is designated a performance factor p which is a measure of complexity, costs, or efficiency, whose value and semantics are determined depending on the detail situation confronted and required. Then the performance factor of a partial inference path, s_{i_1} t_{j_1} s_{i_2} t_{j_2} ... s_{i_n} t_{j_n} $s_{i_{n+1}}$, is defined as the sum of the performance factor of all transitions in this path, i.e., $\sum_k p_{j_k}$. As the number of partial inference paths routed with a specific performance factor p may be various from one reasoning time to another, N(p) stands for its mean value.

Let

$$f(z) = \sum_{p=0}^{\infty} N(p)*z^{p}$$

where f(z) is defined as the z-transform performance function of inference structure which involves the classical z-transform of feedback control theory [Katsuhiko, 1970].

$$G = \{x \mid x \in S \ (\ O(x)=\emptyset \ \wedge \ I(x)\neq\emptyset \ )\}$$

Let r(y,x) denote the partial inference relation between x and y, i.e. $y \in O(x)$ and $x \in I(y)$.

The derivation of the z-transform performance function f(z) is due to the propagation of the so-called performance accumulator A. The computational process is described as follows.

(1) For each inference transition t in T, designate $\pi(t_i)= z^{p_i}$ to t_i, where p_i is the performance factor of t_i ; designate performance accumulator A(s)=1 to every state s in G ; initialize a list called OPEN to be G.

(2) For any element x in OPEN, if A(x) is known, then x is replaced as I(x) added into OPEN after the application of propagating rules 1 and 3; if A(x) is unknown, but A(r(x,y)) is known for all y in O(x), then apply propagating rules 2 and 4 to compute A(x), where the statement 'A(x) is known' means that A(x) has been obtained.

(3) Repeat step (2) until $I(x)=\emptyset$ and A(x) is known for all x in OPEN.

(4) $f(z)= \sum_{x \in OPEN} A(x)$

Property 2 of R stated in Section II holds the condition of $G\neq\emptyset$ and $OPEN\neq\emptyset$ in (4). Property 3, however, warrants the terminality of computational loop mentioned in (2) and (3).

The propagating rules used above are listed as follows:

Propagating rule 1:
if x is in S and A(x) is known , then for all y in I(x), let

$$A(r(x,y)) = (1-(1-1/b)^{b})*A(x)$$

where $b=|I(x)|$

Propagating rule 2:
if x is in T, and for all y in O(x), A(r(y,x)) is known, then let

$$A(x) = \sum_{y \in O(x)} A(r(y,x))*\pi(x)$$

Propagating rule 3:
if x is in T and A(x) is known, then for all y in I(x), let

$$A(r(x,y)) = A(x)$$

Propagating rule 4:
if x is in S, and for all y in O(x) A(r(y,x)) is known, then let

$$A(x) = \sum_{y \in O(x)} A(r(y,x))$$

The computational model of the propagating rules are derived under the probabilistic consideration. The following shows the deduction process of rule 1.

Supposing that the transitions which can deduce the state x are t_1, $t_2, \ldots, t_b$, i.e. $I(x) = \{t_i \mid 1 \le i \le b\}$, then the average probability that t_i is used for the derivation of x is

$$q + (1-q)q + \ldots + (1-q)^{b-1} q$$
$$= 1-(1- 1/b)^b$$

where q is the mean probability of successful activation of a transition, which is approximated as 1/b. In the above statement, the first item denotes that t{ is selected first, and the other items denote t is selected after the unsuccessful activations of others in I(x). Thus the propagating factor used, in the rulel is weighted as 1-(1-1/b) .

The final evaluating stage is to compute performance measuring factors of inference structure R. Let the result of the above computation be

$$f(z) = \sum_{i=1}^{k} N(p)\, z^i$$

where $N(k) \neq 0$ and $N(i) = 0$ for $i>k$. Then let

$$BF = f(1)$$
$$DF = f'(1)/f(1)$$

where $f'(1)$ is the differentiation of $f(z)$ with $z=1$.

$$MDF = k$$

Here breadth factor BF is a measure of total searching breadth to achieve the final goal while reasoning; depth factor. DF is a measure of the mean performance factor of partial inference path routed while reasoning; MDF, then, is the maximum of the performance factors amongst such paths.

Other measuring factors are derived directly from the four-tuple of inference structure R, i.e., $|T|$, the total number of inference transitions in R ; $|S|$, the total number of states in R; $|F|$, where $F = \{ x \mid x \in S\ (I(x)=\emptyset \wedge O(x)\neq\emptyset)\}$.

As an example of the application of this evaluating method, the z-transform performance function of Rc illustrated in Section II is

$$f(z) = 2.73z + 2.73z^2$$

thus

$$|T| = 5 \quad |S| = 5 \quad |F| = 3$$
$$BF = 5.47 \quad DF = 1.5 \quad MDF = 2$$

IV SCOPE AND EXAMPLES OF THE APPLICATION

In this evaluating mod correlated with the effort of reasoning horizontally in matching, selection, confliction resolution, switching, etc., which is the most important factor affecting the reasoning speed of system. The greater BF is , the more effort is needed. DF and MDF, on the other hand, are measure of the vertically-accumulated effort in checking, matching, activation and action etc.. !T! and !Si are often associated with the size of storage. 1FI involves the number of primitive data which need to be acquired.

Commonly, it is easily found in most of the knowledge-based expert systems the track of reasoning network often constituted via a set of rules or knowledge sources. It is therefore tractable to abstract inference structure from them. Property 3 of an inference structure stated in Section II seems to be a major limitation of the application which excludes the self-referencing inference transitions in the system. When such a system like MYCIN is confronted, the additional mechanism is needed to avoid reasoning loops, so is the propagation process stated in Section III to prevent the computing loops. The similar problem has also been explored in [He&Yu, 1986], and the solution can be the marking method discussed in [Buchanan&Short life, 1984] or others. Another problem arises in determining the performance factor of transitions and the size of state, which requires the abstraction is performed under a fixed standard. One of the further direction of this approach is to explore the volume factor of inference structure, which has been investigated as f'(I) in our work, but its application significance remains to be clarified.

It should be noted that although this method is motivated for rule or knowledge-source based reasoning network, Example 3 below, however, is to evaluate two pattern-consistency checkers, which indicates that the inference structure is a kind of the cognitive modeling of human reasoning. The application scope of this method is therefore extended further.

The following examples show the application significance of this method. The performance factor of each transition is assigned as 1, *X* function is valued as z.

A. Example 1 : evaluating a geological expert system GPE

GPE is a geological prospecting system developed at Zhejiang University [He, et al., 1985] with a similar structure to PROSPECTOR. The results of evaluation are listed as follows:

$|T| = 86 \quad |S| = 120 \quad |F| = 49$
$BF = 59.09 \quad DF = 5.70 \quad MDF = 8$

B. Example 2 : comparing two kinds of inference structures

Here we apply a kind of generalization strategies used in knowledge learning and refinement [Michalski, et al. 1984] to inference structure R_{2a} , which results in structure R_{2b} as depicted in Figure 2. we only consider BF and DF which have the greatest impacts on the reasoning speed.

$BF_{2a} = 2n(1-(1-1/2n)^{2n}) \qquad DF_{2a} = 1$

$BF_{2b} = 3/2*n(1-(1-1/n)^{n}) \qquad DF_{2b} = 2$

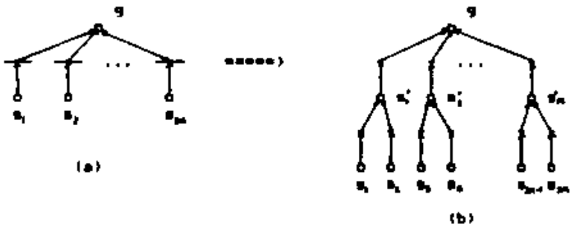

Fig.2 Inference structure R_{2a} is shown in (a), and a kind of its generalized sturcture, R_{2b} , is shown in (b).

Generally speaking, BF has more impact on the reasoning speed than DF. As the searching breadth of R_{2b} is about 25% less than that of R_{2a} . If this generalization in s'_1 , s'_2 ,...,s'_n is repeated, it will be diminished by about 43%, 57%, 68%, ... with respect to this evaluating model. Considering the finally-generalized structure is a full binary tree like inference structure with $n=2^{m-1}$,$|F|=2^m$, then for this structure, $BF=(3/4)^m*2^m$, and for R_{2a} , $BF=(1-(1-2^{-m})^{2^m})*2^m$. The former is about $1- e/(e-1)*(3/4)^m$ less than the latter, where e=2.7183.

C. Example 3 : refining a pattern-consistency checker

In the implementation of OPS5's high performance pattern matching mechanism Rete, it is necessary to make sure that the bound variables in patterns after matching are consistent with the same value. As stated in [Forgy, 1982], Rete uses a linear checker which is depicted in Figure 3(a) via an inference structure with four pattern variables. We think it was adopted because of its convenience in the implementation. The refinement proposed in [Zhang, et al., 1986] is to use binary checking method which is depicted in Figure 3(b) with four pattern variables. The former structure is called Rf, and the latter, Rz. When the number of pattern variables are n, the evaluating results are listed as follows.

For Rf
$|T| = 2(n-1) \quad |S| = 2n \quad |F| = n$
$BF \doteq 0.32n^2 \quad DF \doteq 0.33n \quad MDF = n$

For Rz
$|T| = 2(n-1) \quad |S| = 2n \quad |F| = n$
$BF \doteq o.63n*\log_2 n \quad DF \doteq 0.50*\log_2 n$
$MDF \doteq \log_2 n +1$

BF, DF, and MDF of Rz are only about $\log_2 n/n$ percent as many as those of Rf without the increase of |T|, |S| and |F|. We believe that it is more efficient to use such a refined structure for pattern consistency checking if rules are not allowed to be modified or added dynamically at running time.

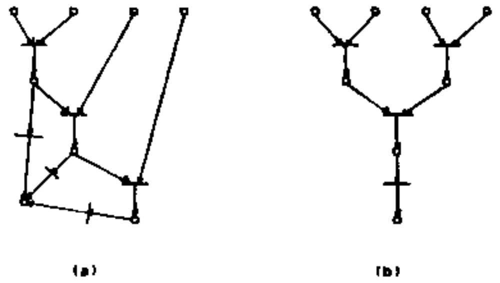

Fig. 3 Inference structure Rf is shown in (a), Rz is shown in (b).

V AN EVALUATING FACILITY

EVAL is an evaluating facility that gives the estimation results of evaluation for inference structure during the development of knowledge-based expert system in terms of the evaluating method described above. EVAL is an interactive system developed at Zhejiang University, and serves as a peripheral device for XMIX. XMIX is a general-purpose knowledge base system (i.e., a tool) which integrates and uniformly specifies the domain and control knowledge. As every knowledge entity in KMIX is specified as a state machine with constraints, it can be divided into a collection of substates, then knowledge sources constitute the inference relationships between those states. EVAL accepts a collection of knowledge entities, and abstracts an inference structure from them, then calculates its performance measure factors.

ACKNOWLEDGMENTS

We thank Wang Shenkang, Gao Ji, Cao Xuejun, Zhangwei, Wang Weining, Tong Xuejun, Zhang Lei, and Tongming at the Artificial Intelligence Laboratory of Zhejiang University, and Hang Jiangchao at the Academic Sinica, for commenting early drafts and providing valuable discussions.

REFERENCES

[1] D.A. Waterman, A Guide to Expert System, (Addison-Wesley, Mass., 1985)

[2] J. Gaschnig, P. Klahr, H. Pople, E. Shortliffe, and A. Terry, Evaluation of expert systems issues and case studies, in F. Hayes-Roth, D.A. Waterman and D.B. Lenat (Eds.), Building Expert Systems (Addision-Wesley, Mass., 1983)., 241-280

[3] E. Shneiderman, Software Psychology : Human Factors in Computer and Information Science, (Winthrop, Mass., 1980).

[4] J. McDermott, A. Newell and J. Moore, The efficiency of certain production system implementations, in D. A. Waterman and F. Hayes-Roth (Eds.),Pattern-Directed Inference Systems (Academic Press, NY, 1978), 155-176.

[5] A. Newell, The knowledge level, Artificial Intelligence 18 (1982) 87-127.

[6] B. G. Buchanan, E. H. Shortliffe (Eds.), Rule-Based Expert Systems The MYCIN Experiments of the Stanford Heuristic Programming Project, (Addison-Wesley, Mass., 1984).

[7] C.L. Forgy, Rete: a fast algorithms for the many patterns / many objects pattern match problem, Artificial Intelligence 19 (1982) 17-32.

[8] He Zhijun and Yu Ruizhao, (Eds.), Selected Papers on Artificial Intelligence, Computer Vision, and Intelligent CAD, (AI Lab/ Zhejiang University, China, 1986).

[9] He Zhijun, Expert system development environment, in [8] 1-21.

CIO] Yang Tao, He Zhijun, and Yu Ruizhao, Knowledge modeling and the KMIX general-purpose knowledge base system, in C8] 44-60.

[11] He Zhijun, Gao Ji, Lu Zhiqing, and Ru Zhiming, GPE: a geological prospecting expert system, in the 3rd National Conf. on Application of microcomputer, Nanjing, China, Oct., 1985.

[12] J.L. Peterson, Petri Net Theory and the Modeling of Systems, (Prentice-Hall, NJ, 1981)

[13] W. Van Melle, A domain-independent system that aids in constructing knowledge-based consulation programs, STAN-CS-820, Stanford University, 1980.

[14] Zhang Wei, Yu Ruizhao, and He Zhijun, Optimum networking in many patterns / many objects matching, Technical Report, Dept. of Computer Sci&Engr., Zhejiang University, Oct., 1986.

[15] R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), Machine Learning, (Tioga, CA, 1984)

[16] Katsuhiko Ogata, Modern Control Engineering, (Prentice-Hall, NJ, 1970).