RTESS: Real Time Expert System Shell

by Leo Santoso

Submission date: 15-Oct-2019 12:39PM (UTC+0700)
Submission ID: 1193136058
File name: Paper.pdf (165.27K)
Word count: 5414
Character count: 25343

RTESS: Real Time Expert System Shell

Leo Willyanto Santoso
Petra Christian University
Siwalankerto 121-131
Surabaya, 60236, Indonesia
+62 31 298 3455
leow@petra.ac.id

ABSTRACT
The improving performance of inference engine in expert system has become an important research in recent years. As it is not realistic to search through all production rule during each cycle using an exhaustive search. Expert systems with a large set of rules can be slow, and can not be suitable for real-time application. In this paper, new algorithm for forward chaining and backward chaining in inference engine was proposed. This algorithm accommodates balanced binary searched tree and binary tree sort that have good performance in large database. Moreover, this new inference engine is supported by certainty factor as well. Displaying image and other supporting materials as the answer is facilitated.

Categories and Subject Descriptors
I.2.1 [**Application and Expert Systems**]: Inference Engine; I.2.5 [**Programming Language and Software**]: Health Expert System.

General Terms
Algorithms, Performance, Design.

Keywords
Forward chaining, backward chaining, certainty factor, binary search tree, binary tree sort.

1. INTRODUCTION
During the past decades, expert system has been explore extensively. Expert system is a computer program that works in specific domain knowledge, exhibits a degree of expertise to solve the problem [5].

Inference engine is the brain of the expert system. Today, several inference engine programs that tries to derive answer from the knowledge base only accommodate one method to create decision, namely forward chaining or backward chaining. Inference engine of VP-Expert, one of the expert system shell, works only using backward chaining method to solve the problem [4, 5]. The development of new expert system cell which accommodate two methods, forward chaining and backward chaining is really needed.

The advantages of combining two methods are to reduce time consuming and to improve the confidence of the result. This illustration between general practitioner and medical patient will explain the situation. When a medical patient tells the condition of the body, in this task, forward chaining is used. Then, general practitioner predicts the disease from explained symptoms of medical patient. To make sure the diagnosis, general practitioner ask several questions to medical patient. In this task, backward chaining is implemented.

Sometimes an expert deal with uncertainty information, because of disguise or incomplete of information. There are two sources of uncertainty that must be encountered in an expert system, there are:

- Uncertainty with regard to the validity of knowledge base rule.
- Uncertainty with regard to the validity of user response.

Let's consider the example of the following question for medical patient: do you have coughing? Where the expected answer is either 'yes' or 'no'. A strictly 'yes' or 'no' response to the question may be unsatisfactory. The confidence factor or certainty factor (CF) is needed. A scale of 0 to 10 where a 0 represents a judgement that there is no coughing while a 10 indicates that the patient is experiencing the most intense pain imaginable. The question could be formed like do you have coughing (0/10). If the user responds with say a value of 9, then this is an indication that coughing at a very intense level.

An alternative question could be built in that situation that could be more appropriate or possibly even better that the use of scale or explicit confidence factor. The question is: Indicate the level of intensity of coughing? Extreme, very intense, moderate, minimal or none. In this example, the user only select the response that seems most appropriate rather than deal with a numeric value.

Furthermore, the other problem in expert system is ineffective search strategy [5, 10]. By implementing balanced binary search tree and binary tree sort, it can reduce time consuming in searching process [2, 8].

This paper presents a new expert system shell which has high-quality performance and fast to reach the solution. Previous real time expert system only deals with specific problem. For example real time expert system for fault diagnosis [1], real time expert system for computer network monitor and control [3], real time expert system for monitoring cardiag operated patients [12], and real time expert system for control of electrophysical complex [11]. By developing real time expert system shell, it can be used for broad problem. This new expert system shell has several method in inference engine, explanation capability, and certainty factor calculation. Next, this expert system shell will be a framework of real time expert system.
The remaining part of this paper is organized as follows. Section 2 presents an overview of current proposals for dealing with expert

systems. Section 3 depicts the approach that we have delineated to solve proposed problems. Section 4 discusses the performance of proposed method. Finally, section 5 concludes the paper.

2. BACKGROUND AND RELATED WORK

In this section, the previous work of backward chaining, forward chaining, binary tree sort and balanced binary search tree are presented.

2.1 Backward Chaining

Backward chaining is an inference method used in artificial intelligence. It is one of two methods of reasoning that uses inference rules – the other is forward chaining, also known as modus ponens.

Backward chaining starts with a list of goals (or a hypothesis) and works backwards from the consequent to the antecedent to see if there is data available that will support any of these consequents [4, 5, 10]. An inference engine using backward chaining would search the inference rules until it finds one which has a consequent (Then clause) that matches a desired goal. If the antecedent (If clause) of that rule is not known to be true, then it is added to the list of goals (in order for your goal to be confirmed you must also provide data that confirms this new rule). Figure 1 shows backward chaining diagram.

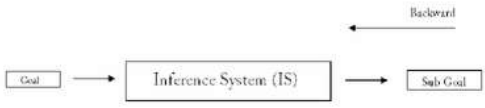

Figure 1. Backward chaining diagram

For example, suppose that the goal is to conclude the color of my pet Fritz, given that he croaks and eats flies, and that the rule base contains the following four rules:

If X croaks and eats flies – Then X is a frog

If X chirps and sings – Then X is a canary

If X is a frog – Then X is green

If X is a canary – Then X is yellow

This rule base would be searched and the third and fourth rules would be selected, because their consequents (Then Fritz is green, Then Fritz is yellow) matches the goal (to determine Fritz's color). It is not yet known that Fritz is a frog, so both the antecedents (If Fritz is a frog, If Fritz is a canary) are added to the goal list. The rule base is again searched and this time the first two rules are selected, because their consequents (Then X is a frog, Then X is a canary) match the new goals that were just added to the list. The antecedent (If Fritz croaks and eats flies) is known to be true and therefore it can be concluded that Fritz is a frog, and not a canary. The goal of determining Fritz's color is now achieved (Fritz is green if he is a frog, and yellow if he is a canary, but since he croaks and eats flies, he is a frog, and, therefore, he is green).

Because the list of goals determines which rules are selected and used, this method is called goal-driven, in contrast to data-driven forward-chaining inference. The backward chaining approach is often employed by expert systems.

2.2 Forward Chaining

Forward chaining is one of the two main methods of reasoning when using inference rules (in artificial intelligence). The opposite of forward chaining is backward chaining.

Forward chaining starts with the available data and uses inference rules to extract more data (from an end user for example) until a goal is reached [4, 5, 10]. An inference engine using forward chaining searches the inference rules until it finds one where the antecedent (If clause) is known to be true. When found it can conclude, or infer, the consequent (Then clause), resulting in the addition of new information to its data. Inference engines will iterate through this process until a goal is reached. Figure 2 shows forward chaining diagram.

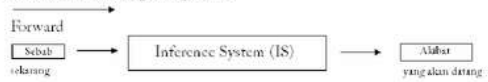

Figure 2. Forward chaining diagram

For example, suppose that the goal is to conclude the color of a pet named Fritz, given that he croaks and eats flies, and that the rule base contains the following four rules:

If X croaks and eats flies - Then X is a frog

If X chirps and sings - Then X is a canary

If X is a frog - Then X is green

If X is a canary - Then X is yellow

This rule base would be searched and the first rule would be selected, because its antecedent (If Fritz croaks and eats flies) matches our data. Now the consequents (Then X is a frog) is added to the data. The rule base is again searched and this time the third rule is selected, because its antecedent (If Fritz is a frog) matches our data that was just confirmed. Now the new consequent (Then Fritz is green) is added to our data. Nothing more can be inferred from this information, but we have now accomplished our goal of determining the color of Fritz.

Because the data determines which rules are selected and used, this method is called data-driven, in contrast to goal-driven backward chaining inference. The forward chaining approach is often employed by expert systems, such as CLIPS.

2.3 Certainty Factor (CF)

Certainty factor theory is a popular alternative to Bayesian reasoning. The basic principles of this theory were introduced by MYCIN, a diagnostic medical expert system [4, 10]. Certainty factors theory provides a judgmental approach to uncertainty management in expert system. An expert is required to provide a certainty factor, cf, to represent the level of belief in hypothesis H given that evidence E has been observed. The maximum value of the certainty factor was +1.0 (definitely true) and the minimum -1.0 (definitely false). Table 1 shows condition and the value of cf. The certainty factors method uses rules of the following form

IF E is true
THEN H is true {cf}

Certainty factors are used if the probabilities are not known or cannot be easily obtained. Certainty theory can manage incrementally acquired evidence, the conjunction and disjunction of hypotheses, as well as evidences with different degrees of belief. Table 1 shows some basic uncertain terms.

Table 1. Uncertain terms and their interpretation

Uncertain Term	CF
Definitely not	-1.0
Almost certainly not	-0.8
Probably not	-0.6
Maybe not	-0.4
Unknown	-0.2 to 0.2
Maybe	0.4
Probably	0.6
Almost certainly	0.8
Definitely	1.0

2.4 Binary Tree Sort

Binary tree sort is a sort algorithm that builds a binary search tree from the keys to be sorted, and then traverses the tree (in-order) so that the keys come out in sorted order [6].

Adding items to a binary search tree is on average an O(log(n)) process, so adding n items is an O(n log(n)) process, making Tree Sort a so-called, 'fast sort'. But adding item to an unbalanced binary tree needs O(n) time in the worst-case, when the tree resembles a linked list (degenerate tree), causing a worst case of O(n2) for this sorting algorithm. The worst-case behavior can be improved upon by using a Self-balancing binary search tree. Using a such a tree, the algorithm has an O(n log(n)) worst-case performance, thus being theoretically optimal. The algorithm of binary tree sort is as follows.

```
/*establish the first element as root */
tree = maketree(x[10]);
/* repeat for each successive element */
for (i = 1; i < n; i++) {
        y = x[i];
        q = tree;
p = q;
        /* travel down the tree until a leaf is reached */
        while (p != null) {
                q = p;
                if (y < info(p))
                        p = left(p);
                else
                        p = right(p);
        } /* end while */
        if (y < info(q))
                setleft(q,y);
        else
                setright(q,y);
} /* end for */
/* the tree is built, traverse it in inorder */
intrav (tree);
```

2.5 Binary Search Tree

The most efficient method of searching is the balanced binary searched tree [2, 8, 9]. Binary search tree itself is a binary tree data structure which has the following properties:

Each node (item in the tree) has a distinct value.

Both the left and right subtrees must also be binary search trees.

The left subtree of a node contains only values less than the node's value.

The right subtree of a node contains only values greater than the node's value.

The major advantage of binary search trees over other data structures is that the related sorting algorithms and search algorithms such as in-order traversal can be very efficient. Operations on a binary tree require comparisons between nodes. Searching a binary tree for a specific value can be a recursive or iterative process. In this research, a recursive method was chosen. The code of binary search tree is as follows.

```
binary_search_tree(node, key):
    if node is None:
        return None  # key not found
    if key < node.key:
        return binary_search_tree (node.left, key)
    elif key > node.key:
        return binary_search_tree (node.right, key)
    else:  # key is equal to node key
        return node.value
```

Balanced binary search tree is a binary search tree (BST) that attempts to keep its height, or the number of levels of nodes beneath the root, as small as possible at all times, automatically [8, 9]. It is one of the most efficient ways of implementing ordered lists and can be used for other data structures such as associative arrays and sets.

Most operations on a binary search tree take time directly proportional to the height of the tree, so it is desirable to keep the height small. Ordinary binary search trees have the primary disadvantage that they can attain very large heights in rather ordinary situations, such as when the keys are inserted in sorted order.

Balanced binary trees solve this problem by performing transformations on the tree (such as tree rotations) at key times, in order to reduce the height.

3. REAL TIME EXPERT SYSTEM SHELL

The combination of powerful searching and sorting algorithm in database, integrating two algorithms of knowledge acquisition in inference engine and supported by certainty factor calculation become a real time expert system shell (RTESS). The algorithm of RTESS can be seen in Figure 3.

Algorithm RTESS
Input: Rules
1. Error checking;
2. While (error=0)
3. if (option method=forward)
4. forward_chaining;
5. if (option method=backward)
6. backward_chaining;
7. if (option method=forward & backward)
8. forward&backward_chaining;

End

Figure 3. Algorithm of RTESS

In error checking procedure, this system checks the syntax of rule with cf and rule without cf. Syntax checking without cf is

checking process that gets key string of rule which entered into system. There are 13 key string, namely: ACTIONS, FIND, RULE, IF, '=', ASK, CHOICES, IMAGE, ';', ':', THEN, OR and AND. The combination of key strings in rule will be checked whether there is a syntax error or not. If there is a syntax error, then an informative error message will be displayed. Figure 4 and Figure 5 show the diagram of error checking without cf. The description of the state is described in Table 2. It describes the state, the condition and the error message which will be displayed when the input is invalid.

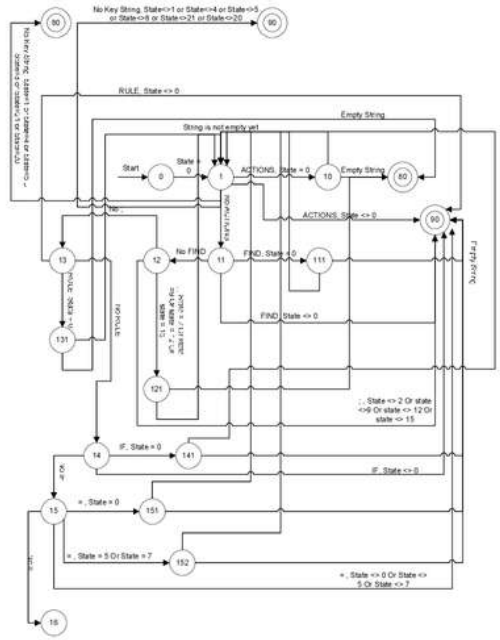

Figure 4. Diagram of error checking without cf

If the user chooses using cf or the rule using cf, then additional key strings for advanced checking are needed, there are: CF, '(', and ')'. Figure 6 and Figure 7 show the diagram of error checking with cf. The description of the state is described in Table 2

The correct syntax will determine the position when the knowledge acquisition process could be performed or not. If the position of syntax already at ASK position or more and there is no error syntax, then inference engine will be started. There is 9 order positions at inference engine, there are: ACTIONS position, FIND position, RULE position, IF THEN position, ANSWER RULE position, ASK position, CHOICES position, IMAGE position, and PATH IMAGE position.

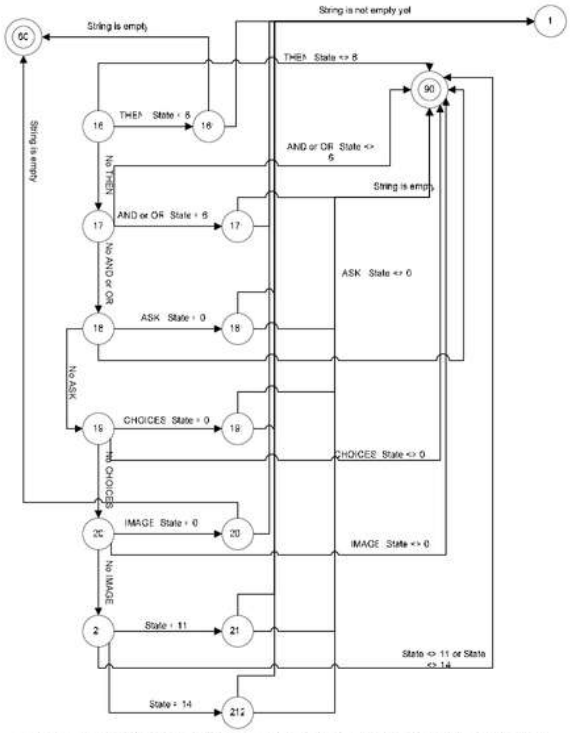

Figure 5. Diagram of error checking without cf (continue)

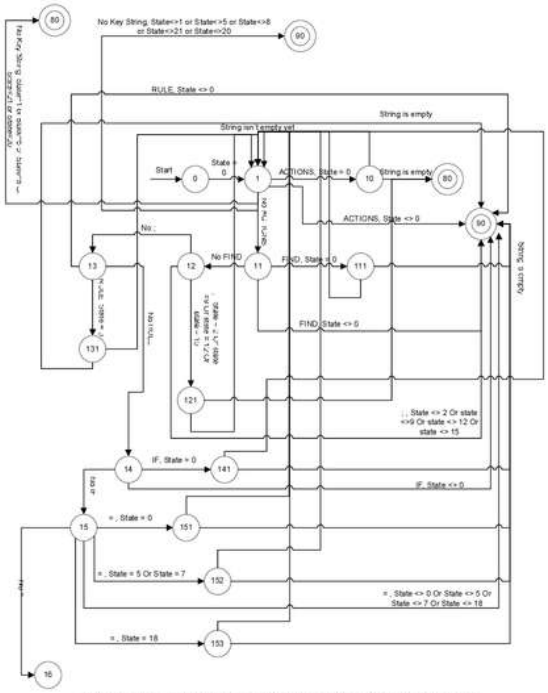

Figure 6. Diagram of error checking with cf

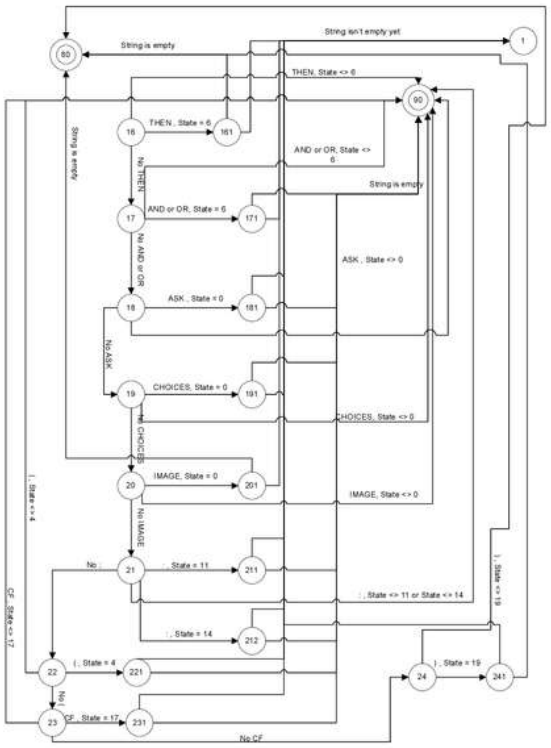

Figure 7. Diagram of error checking with cf (continue)

Table 2. State of error checking

State	Condition	Error Message
0	State = 0	-
1	Find key string	
10	State = 1	
11	Find 'FIND'	
111	State = 2	
12	Find ';'	
121	State = 20	
13	Find 'RULE'	
131	State = 4	
14	Find 'IF'	
141	State = 5	
15	Find '='	
151	State = 9	
152	State = 6	
16	Find 'THEN'	
161	State = 8	
17	Find 'AND' or 'OR'	
171	State = 7	
18	Find 'ASK'	
181	State = 11	
19	Find 'CHOICES'	
191	State = 14	
20	Find 'IMAGE'	
201	State = 21	
21	Find ':'	
211	State = 12	
212	State = 15	
22	Find '('	
221	State = 17	
23	Find 'CF'	
231	State = 18	
24	Find ')'	
241	State = 20	
80	Correct Syntax	
90	Wrong Syntax	Unknown Error

In RTESS, forward chaining is a method that take given set of rule then answer of given rule will be put into working memory. After that, each given rule will be checked, if rule premise produce true value then the result of the rule will be put into working memory. Then, the rule status becomes true so it does not need to be checked again. The rule checking process will be started from the beginning. This process repeats until the goal value has been reached or set of rule already answered and there is no finding goal. The algorithm in Figure 8 show the algorithm of forward chaining.

Algorithm: Forward Chaining

1. Initialization.
   Establish 3 empty tables, the Working Memory table, the Attribute-Queue table, and the Rule/Premise Status table.
2. Start inference.
   Assign a value to a specific premise attribute, where this attribute must not appear in any conclusion clause.
3. Rule scan and check for convergence.
   Examine the Rule/Premise Status table. If no rules are active, STOP. Otherwise, scan the active rule-set premise clauses for all occurrences of attribute on the top of the Attribute-Queue table, and record any changes in status of the premise clauses of active rule set.
   a. If the premise of any rule is false then mark the associated rule as being discarded. Repeat this for all rules having a false premise. When complete, proceed to step 3b.
   b. If the premise of any rule is true then mark the associated rule as being triggered and place its conclusion attribute and rule number at the bottom of the Attribute-Queue table. Repeat this for all rules having a true premise. When complete, proceed to step 3c.
   c. If no rules are presently in the triggered state, go to step 5. Otherwise, go to step 4.
4. Rule firing.
   Cross out the topmost attribute on the Attribute-Queue table. Change the status of the rule associated with the new topmost attribute from triggered to fired. Place the conclusion associated with the fired rule at the bottom of the Working Memory table. Return to step 3.
5. Queue status.
   Cross out the topmost attribute on the Attribute-Queue table and proceed to step 6.
6. Convergence check and rule marking.

Scan the active rule set for any unmarked, active rule. If no such rules can be found, STOP. Otherwise, mark the first such rule found and go to step 7.

7. Query
   For the most recently marked rule, query the user for the value of an attribute in any of the rule's free premise clauses. If the user has a response then goes to step 8. Otherwise, continue this step for all remaining free premise clauses of the marked rule. If all such clauses have been examined without a user response, return to step 6.
8. Rule unmarking
   Place the associated attribute and rule number on the top of the Attribute-Queue table. Unmark the most recently marked rule and return to step 3.

End

Figure 8. Forward Chaining Algorithm

Backward chaining is a method that finds goal position firstly. Figure 9 shows the algorithm of backward chaining. The procedure of backward chaining is as follows:

The goal firstly will be searched in working memory, if it is not found, do step b.

The goal will be searched in the rule which its variable related to goal. If rule has founded, go to step e, else search the goal into set of given data. If the goal is not found, then the value of goal is false. If found, put the answer into working memory.

Premise will be searched from rule which has variable related to premise variable. If it is not found, then go to step d, if it is found, premise will be changed as goal. The next process is step e.

Premise will be searched from set of given data. If it is not found, then the value of premise is false. Else, the answer will be put into working memory.

Rule will be checked its premise, if the premise is not found in working memory, then go to step c. If the value of all premises is true, the answer of the rule will be put into working memory.

Algorithm: Backward Chaining

1. Initialization.
   Establish 3 empty tables, the Working Memory table, the Goal table, and the Rule/Premise Status table.
2. Start inference.
   Specify a final goal. Place the associated goal attribute at the top of the Goal table.
3. Rule scan and check for convergence.
   Scan the conclusion clause of the active rule to find any concurrence of the goal attribute presently on the top of the Goal table.
   a. If the Goal table is empty, STOP.
   b. If only one such rule may be found, go to step 6. If several such rules may be found, and any of these are triggered, select any one of the triggered rules and proceed to step 6. Otherwise, arbitrarily select one rule among the rules found that contains the subject goal attribute in its conclusion clause set, and go to step 6.
   c. If no active rules are found that contain the subject goal attribute in their conclusion clause set, then go to step 4.
4. Query.
   For the goal attribute on top of the Goal table, find the associated query if one exists. If there is no query associated with this goal attribute, then STOP. Otherwise, query the user, record his or her response, remove the top goal attribute from the Goal table and place it in the Working Memory table. Go to step 5.
5. Rule/premise status update.
   Using the contents of the Working Memory table, update the Rule/Premise Status table. Specifically, if the premise of any rule is false, discard that rule, and if the premise is true, trigger that rule. Return to step 3.
6. Rule evaluation.
   a. If this rule is triggered, then remove the current topmost goal attribute from the Goal table and place it in the Working Memory table. Change the status of this rule from triggered to fired. Go to step 5. Otherwise proceed to step 6.
   b. If this rule is not triggered, then select the first unknown premise attribute of the rule and place it at the top of the Goal table. Return to step 3.

End

Figure 9. Forward Chaining Algorithm

Mixed chaining method is a combination of forward chaining method backward chaining method. In this method, user will be given set of data that need to be answered. These will be done in forward chaining. Then, user can select the implementation of backward chaining if the data that need to be answered satisfies user needs and goal is still searching. Figure 10 shows the steps of mixed chaining method.

Step	Rule (rule type)	Facts (goals)	Chaining (firing)
1	{}	AC(K)	
2	R1 (B)	AC(FH)	B
3	R3 (B)	AC(FEB)	B
4	R8 (F)	ACG(FEB)	F(fired)
5	R4 (F)	ACGB(FE)	F(fired)
6	R7 (F)	ACGBD(FE)	F(fired)
7	R5 (F)	ACGBDH(FE)	F(fired)
8	R6 (F)	ACGBDHE(F)	F(fired)
9	R2 (B)	ACGBDHEK(F)	F(fired)

Figure 10. Mixed Chaining Algorithm

The summarized of mixed-mode chaining algorithm is as follows. First, we are given A and C, and our goal is to determine K. This is summarized in step 1, the first line of table. Next, we proceed to the backward chaining rules and seek one having our desired goal (K) in the conclusion. The first such rule we come to is rule 1. From rule 1, we note that K is determined whenever F and H are true. Thus, in the third column of step 2 we replaced K with F and H. In the fourth column we simply denote the fact that we employed backward chaining for rule 1 and that it was not fired at this step.

Still using backward chaining, we now seek a rule that has either F or H in its conclusion. The only such rule is rule 3. Thus, we move to rule 3. Here, we see that H is determined by E and B and thus we replace the goal H by E and B. In the fourth column we note that backward chaining was used and that the rule was not fired. We move to the set of forward chaining rules and find that

rule 8 is triggered. We may then fire rule 8 which gives us the new fact, G, and G is added to our set of known facts in the third column of the table. In the fourth column we note that forward chaining was used and that rule was fired. Knowing A, C, and G, we may now fire rule 4, to derive B. And this action is listed in step 5. Knowing A, C, G and B, we may fire rule 7 to derive D. Knowing the facts indicated in step 6, we may fire rule 5 to derive H. Knowing the facts listed in step 7, we may next fire rule 8 to derive E, and knowing E, we may remove E from our list of desired goals.

At this point, we have fired all of our forward chaining rules and we still seek goal F. Returning to our set of backward chaining goals, we note that F does not appear as the conclusion of any rule. Consequently, our only move is to use forward chaining on the remaining rule set, and this may be done for rule 2. Specifically, and as summarized in step 9, we may fire rule 2 to derive K. And since K is the goal that was originally desired, we may terminate the process.

4. EXPERIMENTS

In this section, we present experimental result comparing the performance of new RTESS using several thousand of rules. This system was built in Microsoft Visual C++ on a PC with 2.4 GHz Pentium ® 4 CPU and 1 GB of RAM under MS Windows XP Pro. Figure 11 is a screenshot for a simulation using tourism rule data.

Figure 11. Interface of RTESS

It shows a result of the process where the answer and its certainty factor is displayed. This system shows the result not only in text format, but in image as well.

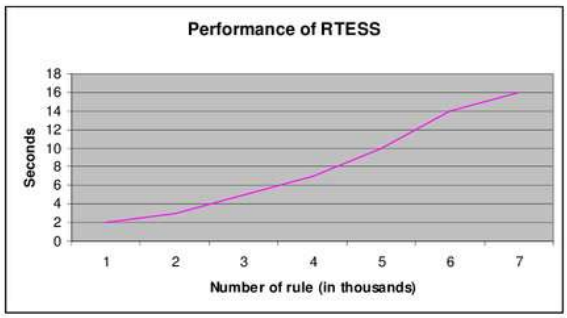

Figure 12. The Performance of RTESS

Figure 12 shows the performance of RTESS. This figure reports the execution times obtained by RTESS over rules with increasing number of rule. The curve shows an almost linear scalability. As can be seen from the graph, running times grow when the number of rule is increased.

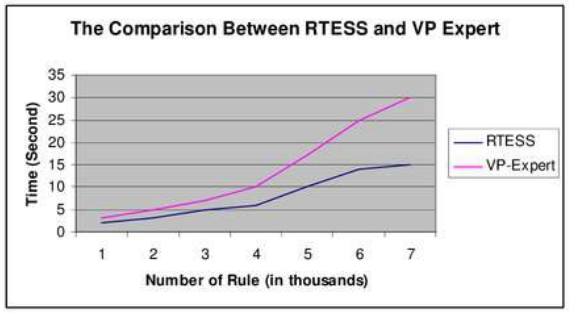

Figure 13. The Comparison Performance between RTESS and VP Expert

Figure 13 shows the performance comparison between RTESS and VP Expert. It can be seen that RTESS outperforms the VP Expert.

5. CONCLUSION

This paper deals with the implementation of balanced binary search tree and binary tree sort to support methods in inference engine – forward chaining, backward chaining and mixed chaining. The focus of this paper is to reduce running time and to display certainty factor of the result.

The emphasis of this paper was on feasibility – identification of possible approaches and development of methods to put them into practices.

We are currently working on the evaluation of the performance and the reliability of the methods proposed in this paper. Firstly, benchmarking for performance evaluation indicate which method is the most efficient and effective from response time point of view. Next, concerns is the quality of the result.

6. REFERENCES

[1] Angeli, C. 2000 Application of a real-time expert system for fault diagnosis. IEA/AIE 2000, LNAI 1821, pp 184-191. Springer-Verlag Berlin Heidelberg.

[2] Barnes, G. M., Noga, J., Smith, P. D. and Wiegley, J. 2005 Experiments with balanced-sample binary trees, ACM SIGCSE Bulletin, 37(1): 166-170.

[3] Dunning, B.B. and Switlik, J. 1988 A real-time expert system for computer network monitor and control. Database Summer pp 35-38.

[4] Giarratano, J.C. and Riley, G. 1994 Expert Systems: Principles and Programming, PWS Publishing Co., Boston, MA.

[5] Ignizio, J.P. 1991 Introduction to Expert Systems: The Development and Implementation of Rule-Based Expert Systems. Singapore:McGraw-Hill Book Co.

[6] Langsam, Y., Augenstein, M. J., and Tenenbaum, A. M. 1996 Data structure using C and C++, Prentice-Hall, New Jersey.

[7] Lee, K. C., Cho, H. R., and Kim, J. S. 2008 An expert system using an extended AND-OR graph. Knowledge-Based Systems, 21(1):38-51.

[8] Li, C. C. 2006 An immediate approach to balancing nodes in binary search trees, Journal of Computing Sciences in Colleges, 21(4): 238-245.

[9] Manthey, B. and Reischuk, R. 2007 Smoothed analysis of binary search trees, Theoretical Computer Science, 378(3):292-315.

[10] Negnevitsky, M. 2005 Artificial intelligence: a guide to intelligent systems, 2nd Ed., Addison Wesley, England.

[11] Rybin, V.M., Rybina, G.V., Ochinsky, V.V., and Stepankov, V.U. 1999 Real-time expert system for control of electrophysical complex. *In Proc. of International conference on accelerator and large experimental physics control systems* pp. 124-126.

[12] Sukuvaara, T., Koski, E.M., Makivirta, A., and Kari, A., 1993 A knowledge-based alarm system for monitoring cardiac operated patients – technical contruction and evaluation. Int J. Clin. Monit Comput., 10(2): 117-126.

RTESS: Real Time Expert System Shell

ORIGINALITY REPORT

6%	3%	3%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

MATCH ALL SOURCES (ONLY SELECTED SOURCE PRINTED)

1%

★ Leila Ooshaksaraie, Noor Ezlin Ahmad Basri, Azuraliza Abu Bakar, Khairul Nizam Abdul Maulud. "RP3CA: An expert system applied in stormwater management plan for construction sites in Malaysia", Expert Systems with Applications, 2012

Publication

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		