A FORWARD REASONING ALGORITHM

Bilić, Herman; Boljat, Ivica; Grbac, Zlatko; Jadrić, Ana; Marković, Dinko; Slapničar, Petar

Faculty of Electrical Engineering Mechanical Engineering and Naval Architecture
R. Boškovića str. bb, 58000 Split

When defining operating environment for experimentation in the domain of expert systems, one of basic modules is inference engine. On the basis of the predefined knowledge representation an inference engine algorithm is suggested. A programming module and the reasoning illustrated by examples have been implemented.

/knowledge base/forward reasoning/inference engine/

1. INTRODUCTION

The paper deals with expert systems having as basic components a knowledge base based on production rules and an inference engine /1/, /2/, /4/.
We start with the assumptions that experimentation is a possible and interesting starting point in research work dealing with expert systems, that it is not easy to obtain an adequate commercial shell of an expert system, and that it is useful and convenient to build one's own environment that would comply with a broad variety of experimentation demands.
A knowledge representation was earlier defined /3/,/5/,/7/ in a way that IF-THEN rules were stored in 5 relational data bases of a defined structure.
A variant of the next basic module of the expert system - an inference engine operating on the previously defined knowledge base is suggested here.

2. INFERENCE ENGINE

Research in AI consists of three inseparable components: theory, program implementation, and experimentation with finished programs. These three components are in constant interaction, which results in software implementation using the elements of "intelligent" behaviour. However, simulation of human reasoning and inferencing requires development of efficient scanning algorithms in a great number of variants, which would bring the potential possibilities to the level of practical feasibility. Therefore research work aiming at defining efficient task-solving procedures is of a particular importance.
The inference engine based on the given data base is described here. This inference engine is required to meet the following requirements:

- It must be user-oriented i.e. communication with software must be simple and efficient.
- It must result in overall reasoning flow with all the possible parallel branches.
- It must be valid for numeric and nonnumeric variables.

To implement such an engine 5 auxiliary temporary data bases are used:

INPUT_VAL base represents a temporary base of facts related to the particular reasoning. It contains all variable values given to the computer whether at the beginning of reasoning or during a particular cycle, as well as values of all variables obtained during reasoning, lest the computer might request several times the value of the same variable (in different cycles). Data base initiated at the beginning of each new reasoning process and during this process is constantly being given new records.

VAR_GAME data base is, like RUL_GAME base, a basic base needed for the functioning of the

algorithm. Each variable obtained by reasoning, (i.e. from THEN rules) is added at the end of the base. This is performed with the aim of examining all the paths in reasoning which continue with this variable (i.e. all further places in IF base where this variable appears are examined). A variable (always the last one) is deleted from VAR_GAME base only when it is found that in IF base there is no longer any rule containing this variable that has not been examined in reasoning.

RUL_GAME base. At the end of this base a designation of each IF rule reached during the reasoning process is written. This is done with the aim to examine all the possible further paths of reasoning if there is an operator in the THEN part. The rule (always the last one) is deleted from RUL_GAME base when all the records of TO_THEN base having the value of IF_RULE field equal to the designation of the observed rule are examined, i.e. when all reasoning paths connected with the operator in THEN rule are examined.

AUX base. A designation of each IF rule created in the reasoning process is written into the base. This is done to prevent repeated processing of the same reasoning flow from different branches.

OUT_VAL base. The base is added all the reasoning data so as to make the final output absolutely clear.

Configurations of these 5 data bases are shown in Table 1.

Table 1. Configuration of auxiliary data bases

a) INPUT_VAL data base

Field	Field name	Type	Length
1	VAR_NAME	Char	10
2	VAR_VALUE	Char	10
3	RELATION	Char	2

b) VAR_GAME data base

Field	Field name	Type	Length
1	VAR_NAME	Char	10
2	VAR_VALUE	Char	10
3	VAR_PTR	Num	5
4	RELATION	Char	2
5	SWITCH	Num	1

c) RUL_GAME data base

Field	Field name	Type	Length
1	IF_RULE	Char	10
2	ONTHEN_PTR	Num	5
3	GROUP_SGN	Num	5

d) AUX data base

Field	Field name	Type	Length
1	IF_RULE	Char	10

e) OUT_VAL data base

Field	Field name	Type	Length
1	VAR_NAME	Char	10
2	VAR_VALUE	Char	10
3	OPERATOR	Char	3
4	CYCLE	Char	1
5	GROUP_SGN	Num	5
6	RELATION	Char	2
7	VAR_VALUE2	Char	10
8	RELATION2	Char	2
9	CYC_NUM	Num	5

VAR_VALUE2 and RELATION2 pair of fields in OUT_VAL base refers to the pair of values found in the base and valid for the VAR_VALUE and RELATION pair. Validity criterion means that the whole validity domain of the VAR_VALUE and RELATION pair is within the validity domain of VAR_VALUE2 and RELATION2 pair. So e.g. if during reasoning we look in IF base for $\alpha < 7$ and find $\alpha < = 50$, the condition has been fulfilled. In this case VAR_VALUE equals 7, RELATION equals "<", VAR_VALUE equals 50, and RELATION2 equals "<=". CYCLE field records the existence of the cycle, CYC_NUM field denotes the cycle ordinal number, and GROUP_SGN field denotes the operator ordinal number. VAR_PTR field in VAR_GAME base records the current record ordinal number in VARIABLE base (in order to continue further search from that record on), and ONTHEN_PTR field in RUL_GAME base denotes the current record ordinal number in TO_THEN

base (in order to continue possible further search from this record on). SWITCH field in VAR_GAME base is used to distinguish the first variable from which the reasoning starts (so it is immediately entered into OUT_VAL base) from the other variables added to OUT_VAL base on the basis of reasoning.
The following is an inference engine algorithm pseudocode:

```
read NAME (name), VALUE (val) and RELATION (rel) for the beginning variable;
enter name, val, rel into tables INPUT_VAL, VAR_GAME, OUT_VAL;
PRNSW = 1; OP_NUM = C_NUM = 0;   /* PRNSW - print switch */
                                 /* OP_NUM - operator number */
                                 /* C_NUM  - cycle number */
while there are records do for the last VAR_GAME base record;
        while there are records search in VARIABLE base the record for which the following is valid:
                1. VAR_NAME = name and IF_THEN = I,
                2. rule Is not in AUX base,
                3. In the corresponding IF base record VAR_VALUE and RELATION in accordance with
                   val and rel,
                4. If the cycle is then
                        CYC = 1;
                        examine the whole cycle; /* search for the vaiues of the unknown variables
                        in INPUT_VAL base, and if there are no such variables, request the values from
                        the screen and enter into INPUT_VAL base */
                if there is a record then enter the record into the AUX and OUT_VAL bases;
                        PRNSW = 1;
                        if CYC = 1
                                C_NUM = C_NUM + 1;
                                into AUX and OUT_VAL base enter the cycle variables; cycle = C;
                        via TO_THEN base search the first IF rule and find variable in THEN base;
                                if there is an operator OP_NUM = OP_NUM + 1;
                                enter the rule at the end of the RUL_GAME base;
                                update bases OUT_VAL, INPUT_VAL and VAR_GAME;
                else
                        delete the last record from VAR_GAME base;
                        search on in TO_THEN base for the last rule from RUL_GAME base;
                        if there is a rule in TO_THEN base
                                PRNSW = 1;
                                update the last record in RUL_GAME base;
                                via TO_THEN base find a variable in THEN base and transfer it to
                                        OUT_VAL, INPUT_VAL and VAR_GAME bases;
                        else
                                delete the last rule from RUL_GAME base;
if PRNSW = 1  print OUT_VAL base;
```

The choice of programming language for coding the algorithm has not been paid a great attention to. Namely, we start from the hypothesis that soluble algorithms in any real software system can be translated into soluble algorithms in any other real software system. When a soluble problem is discovered, there is not a particular programming language for that problem. A reasonable design must be identified, but each general-purpose language can give a reasonable solution /6/. So on the basis of this algorithm, programming modules are implemented in CLIPPER (500 instructions of source code) and in C language (600 instructions of source code) and the functioning of both modules is successfully tested on numerous examples implemented with the example generator /7/ as well as with several other examples including the operator in the THEN part (which is not the case with graph).

3. EXAMPLES

Reasoning results are shown in two short examples. The first example uses numerical variables and is made only to exemplify, namely its aim is to show different reasoning variants as clearly as possible. Example 1 is as follows:

IF a=2 AND b>=3 AND c<=4	THEN d=10
IF a>=10	THEN x=3 AND y<5
IF a<50	THEN x=7 AND z=20
IF x=3	THEN z=18 OR u=90
IF z=18	THEN w<13

Example 2 refers to a nonnumerical variable and is typical in literature dealing with this domain:

IF INTEREST FALL	THEN STOCK RISE
IF INTERSET RISE	THEN STOCK FALL
IF DOLLAR FALL	THEN INTEREST RISE
IF DOLLAR RISE	THEN INTERSET FALL
IF FEDINT FALL AND FEDMON ADD	THEN INTEREST FALL

In Fig. 2 a), b), c), d), and e) it is shown how rules of Example 1 and Example 2 are stored in the knowledge base. There are no restrictions as for storing the rules of different examples into the common knowledge base.

IF_RULE	NEXT_IF	VAR_NAME	VAR_VALUE	RELATION	CERTAINTY
10	20	a	2	=	1.00
20	30	b	3	>=	1.00
30	10	c	4	<=	1.00
40	40	a	10	>=	1.00
50	50	a	50	<	1.00
60	60	interest	fall	=	1.00
70	70	interest	rise	=	1.00
80	80	dollar	fall	=	1.00
90	90	dollar	rise	=	1.00
91	92	fedint	fall	=	1.00
92	91	fedmon	add	=	1.00
93	93	x	3	=	1.00
94	94	z	18	=	1.00

Fig. 2 a) IF base for Example 1 and Example 2

THEN_RULE	VAR_NAME	VAR_VALUE	RELATION	CERTAINTY
100	d	10	=	1.00
200	x	3	=	1.00
300	y	5	<	1.00
400	x	7	=	1.00
500	z	20	=	1.00
600	stock	rise	=	1.00
700	stock	fall	=	1.00
800	interest	rise	=	1.00
900	interest	fall	=	1.00
901	z	18	=	1.00
902	u	90	=	1.00
903	w	13	<	1.00

Fig. 2 b) THEN base for Example 1 and Example 2

IF_RULE	THEN_RULE	OPERATOR
10	100	
20	100	
30	100	
40	200	AND
40	300	AND
50	400	AND
50	500	AND
60	600	
70	700	
80	800	
90	900	
91	900	
92	900	
93	901	OR
93	902	OR
94	903	

Fig.2 c) TO_THEN base for Example 1 and Example 2

THEN_RULE	IF_RULE
100	10
200	40
300	40
400	50
500	50
600	60
700	70
800	80
900	90
900	91
901	93
902	93
903	94

Fig.2 d) TO_IF base for Example 1 and Example 2

VAR_NAME	IF_THEN	RULE
a	I	10
b	I	20
c	I	30
d	T	100
a	I	40
a	I	50
x	T	200
y	T	300
x	T	400
z	T	500
interest	I	60
interest	I	70
stock	T	600
stock	T	700
dollar	I	80
dollar	I	90
interest	T	800
interest	T	900
fedint	I	91
fedmon	I	92
x	I	93
z	I	94

Fig.2 e) VARIABLE base for Example 1 and Example 2

If reasoning is actuated from variable a with the value of 2, the computer searches the value for variables b and c (cycle), and if b=3 and c=4 is entered, the reasoning flow is obtained on Table 2.

Table 2. Reasoning within Example 1 starting with a=2

VARIABLE	GIVEN VALUE	FOUND VALUE	OPERATOR	NUMB.OPER.	CYCLE	NUMB.CYCLE
a	= 2	= 2			C	1
b	= 3	>= 3			C	1
c	= 4	<= 4			C	1
d	=10					
a	= 2	< 50				
x	= 7		AND	1		
z	= 20		AND	1		

It is clear that the GIVEN VALUE in the line which does not belong to the cycle also represents the conclusion for the previous line (or lines if the cycle is dealt with). It can be seen that first four lines of the table represent a reasoning branch, and the last three lines a parallel branch. If reasoning is

initiated by e.g. a=15, there follows the flow on Table 3, and if, within Example 2, we start from fedint fall, and if for fedmon we enter the value add (requested by the computer), the flow shown in Table 4 is obtained.

Table 3. Reasoning within Example 1 starting with a=15

VARIABLE	GIVEN VALUE	FOUND VALUE	OPERATOR	NUMB.OPER.	CYCLE	NUMB.CYCLE
a	=15	>=10				
x	= 3	= 3	AND	1		
z	=18	= 18	OR	2		
w	<13					
u	=90		OR	2		
y	< 5		AND	1		
a	=15	< 50				
x	= 7		AND	3		
z	=20		AND	3		

Table 4. Reasoning within Example 2 starting with fedint fall.

VARIABLE	GIVEN VALUE	FOUND VALUE	OPERATOR	NUMB.OPER.	CYCLE	NUMB.CYCLE
fedint	= fall	= fall			C	1
fedmon	= add	= add			C	1
interest	= fall	= fall				
stock	= rise					

4. CONCLUSION

With intention to define the proper environment which opens broad possibilities of experimentation in the domain of expert systems, an inference engine module is set apart as one of basic modules. Reasoning must function within the previously defined knowledge base, which consists of production rules organized in 5 relational data bases. A forward reasoning algorithm, based on 5 auxiliary relational data bases has been worked out in detail. The following conditions are stipulated on this algorithm:

- Simple usage,
- It must result with overall reasoning flow,
- It must be valid both for numeric and nonnumeric variables.

According to this algorithm programming modules have been implemented in Clipper and C language, and they have been successfully tested on numerous examples.

References:

/1/ Hayes-Roth, F. (1984), "The Knowledge-Based Expert Systems", IEEE Computer, vol. 17, No. 9, Sept. 1984, pp. 11-28.

/2/ Steels, L. (1985), "Second Generation Expert Systems", Future Generation Computer Systems, vol.1, no.4, pp. 213-221, 1985.

/3/ Han-lin, Li. (1986), "To Use RDBMS as a Building Tool of Data-Knowledge Base Systems", The 6-th Int. Workshop on Expert Systems and Their Applications Avignon, April 1986, pp.1143-1163.

/4/ Neapolitan,.R. A. (1986), "Forward-chaining versus graph approach as the inference engine in expert systems", Proc. Applications of Artificial Intelligence III, SPIE vol. 635, pp. 62-69, apr. 1986.

/5/ Bilić, Herman; Boljat, Ivica; Grbac, Zlatko; Jadrić, Ana; Marković, Dinko; Slapničar, Petar; Saša, Slavko. An experiment with a knowledge base. Proceedings of the 11th International Symposium Computer at the University, Book II. Cavtat: SRCE-Zagreb, 1989, str. 8.10.1-8.10.8, lit. 7. sum.

/6/ Bilić, Herman; Boljat, Ivica; Grbac, Zlatko; Jadrić, Ana; Marković, Dinko; Slapničar, Petar. Generation of causal model for second generation expert systems. Proceedings of the 12th International Symposium Computer at the University. Cavtat: SRCE-Zagreb, 1990, str. 6, lit. 6. sum.

/7/ Bilić, Herman; Boljat, Ivica; Grbac, Zlatko; Jadrić, Ana; Marković, Dinko; Slapničar, Petar. Causal model in second generation expert systems. International Symposium and Exhibition Computational Intelligence 90. Milano: Universita degli studi di Milano, 1990, str. 149-150.

/8/ Bilić, Herman; Boljat, Ivica; Grbac, Zlatko; Jadrić, Ana; Marković, Dinko; Slapničar, Petar. Causal Model in Second Generation Expert Systems - Examination, Optimizing and Learning. Elektrotehnika, Zagreb, 1991, pp. 166-168, lit. 6. sum.

/9/ Bylander, Tom. Tractability and artificial intelligence. JETAI, vol. 3, No. 3, 1991, pp. 171-178, lit. 26, sum.

ITI '92

Proceedings of the
14th International Conference on

INFORMATION TECHNOLOGY INTERFACES

Pula 1992

Organized by

University Computing Centre, Zagreb

Cooperating institutions

Arizona State University
Athens University of Economics and Business
Croatian Society for Simulation Modelling
Department of Computer Science, North Carolina State University
Department of Management Science and Information Technology, University of Georgia
Department of Statistics and Operations Research, Polytechnical University of Catalonia, Barcelona
European Business Management School, Swansea
Faculty of Economics, University of Zagreb
Faculty of Electrical Engineering, University of Zagreb
Faculty of Organization and Informatics, University of Zagreb
"Jožef Stefan" Institute, Ljubljana
Working Group Simulation, Technical University, Vienna

Under the auspices of

Croatian Academy of Sciences and Arts
Ministry of Science, Technology and Informatics, Republic of Croatia
University of Zagreb

Published by: University Computing Centre
Engelsova bb, 41000 Zagreb, Croatia