ISSN 0254–2757

QI QUÆSTIONES INFORMATICÆ

Volume 5 • Number 1 April 1987

G.R. Finnie	**On Learning Styles and Novice Computer Use**	1
P.S. Kritzinger	**Local Area Networks in Perspective**	11
S. Berman	**Semantic Information Management**	19
P.C. Pirow	**Research Computeracy**	23
C.H. Hoogendoorn	**Experience with Teaching Software Engineering**	36
C. Levieux	**Education Rather than Training**	41
D. Podevyn	**Decision Tables as a Knowledge Representation Formalism**	46
J. Roos	**The Protocol Specification Language ESTELLE**	51
L.J. van der Vegte	**The Development of a Syntax Checker for LOTOS**	63
	BOOK REVIEWS	71

An official publication of the Computer Society of South Africa and of the South African Institute of Computer Scientists

'n Amptelike tydskrif van die Rekenaarvereeneging van Suid-Afrika en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

QUÆSTIONES INFORMATICÆ

An official publication of the Computer Society of South Africa and of the South African Institute of Computer Scientists

'n Amptelike tydskrif van die Rekenaarvereniging van Suid-Africa en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

Subscriptions

Annual subscription are as follows:

	SA	US	UK
Individuals	R10	$ 7	£ 5
Institutions	R15	$14	£10

Quæstiones Informaticæ is prepared by the Computer Science Department of the University of the Witwatersrand and printed by Printed Matter, for the Computer Society of South Africa and the South African Institute of Computer Scientists.

DECISION TABLES AS A KNOWLEDGE REPRESENTATION FORMALISM

D. Podevyn
Department of Computer Science and Information Systems
University of South Africa

ABSTRACT

A knowledge engineering tool called GESS (General Expert System Shell) is presented which is used for building different types of expert systems. Knowledge representation is based on direct extensions of the idea of decision tables. It was found that this formalism reduces the knowledge acquisition bottleneck by assisting the knowledge engineer in the structuring of knowledge because it enforces completeness and allows for automated checking of contradictions and redundancies. It was also found that this formalism is quite flexible in expressing knowledge and implementing different reasoning paradigms as is discussed in this paper. Complex systems have been built with Gess using its capability of creating networks of decision tables and the accessibilty of the external computer environment from within Gess.

1.INTRODUCTION

The formalism of decision tables (DT) was introduced to represent complex decision logic in a systematic manner [5, 7]. Theoretically decision tables can model an entire program [6] and yet they have been surprisingly neglected by many DP organisations [1].

However, the usefulness of the application of DT as a decision support technique was demonstrated by Maes et al [8, 11]. They have also formed the basis of control structures in user-oriented systems [4, 9]. We have taken this idea further by applying it in the field of knowledge based systems and expert systems.

To our knowledge, other related work in the field of knowledge based systems is limited to systems that allow the user to build decision trees [2, 12]. Decision trees have however the disadvantage of becoming unwieldy when there are a substantial number of conditions and actions. Another disadvantage is that they do not " force" the designer to look at every possible combination of conditions, and some might be overlooked. This is definitely not the case with decision tables. With Gess this completeness is checked by the system, together with the checking for contradictions and redundancies.

2. BASIC CONCEPTS IN GESS

2.1 Access to external data dictionary

Inputs and outputs for Gess decision tables are records which are described in an external data dictionary. This also allows Gess to access external operational or corporate databases which can then serve as " bulk" facts providers.

2.2 Network of decision tables

Gess uses the concept of subroutine decision tables invokable from conditions and/or actions.

It renders complex tables simpler by reducing them to several smaller ones. In fact this has proven to be so powerful that a test facility for running decision tables with test values was seldom used because the tables could easily be desk-checked.

It allows one to use the top-down refinement technique to construct the application logic.

It provides for modularity – a subroutine table can be called from any other decision table. This effectively allows one to create a network of decision tables, which is a powerful feature.

2.3 The four basic programming concepts

The subroutine concept also allowed us to introduce the three basic constructs of sequence, selection and iteration between decision tables. Together with the fact that calculations are supported, this provides the theoretical foundation that any algorithm can be implemented.

In figure 1 we see that depending on the condition, either table TAB3, TAB2 and TAB1 will be executed, and in that sequence, or TAB4 and TAB3 will be executed. Note that TAB1 calls itself recursively.

Through this important generalisation it is possible to implement different reasoning paradigms such as forward and backward chaining and generate-and-test as will be shown in section 3.

Record 1

A	B	C

TAB1

1.B < 10	Y	N
S.TAB1	3	
S.TAB2	2	
S.TAB3	1	2
S.TAB4		1

figure 1

2.4 Implementation

There are 2 basic parts in Gess. The first part is the compiler. It accesses the data dictionary to check record and field names used in the DT, checks for completeness for the combinations of conditions, checks for contradictions and redundancies and finally writes the compiled tables onto the rule base in a specialised internal format (not program code). This format allows to sort combinations and/or conditions according to frequency of use or other parameters.

The second part entails the use of the DT. This program consists also of 2 parts. The first one is the Gess executer which loads the necessary DT and interpretes the internal format. The second part is the master program.

This master program can be : replaced (user-rewritten),
simple/complex,
batch/online,
one/many.

This split between executer and master program gives Gess additional flexibility and also allows for a high degree of integration with the external computer environment. In fact this master program can range from a traditional application program (which consults the rule base via the executer) to a very simple master program that would call a first decision table to receive names of records to read and names of tables to call and would then have a loop to call all these tables sequentially. All the logic would thus be in the decision tables.

Gess has a standard set of master programs, but nothing prevents the user from replacing them with for instance a natural language interface program.

These are just a few basic examples of how the concept of the master program can be used.

3. EXPERIMENTS AND APPLICATIONS

3.1 Implementation of a backward chaining paradigm

Assume a set of rules like $A \rightarrow B$ (A implies B)
$B \rightarrow C$
$C,D \rightarrow E$ (C AND D imply E)

Backward chaining then starts from a goal, say E. In order to prove E the third rule is used so that the system now tries to prove C and D and so on.
A rule such as A implies B, becomes the following decision table.

A='N'	Y	N	N
A='Y'	-	Y	N
EXIT	*		
B = 'Y'		*	
SOLVE(A)			*

The backward chaining is provided by the SOLVE(A) action, which is executed when it is found that A is not evaluated yet. There are several implementations possible for SOLVE(A), one of them is to put A on top of a stack which keeps track of goals still to be proven, then leave this table and look for one which proves A.
It is important to remark that this basic table can easily be extended with more actions in order to include explanation and a trace. The explanation can be used to justify conclusions reached and the trace can be used to show the reasoning chain when asked for by the user.

A='N'	Y	N	N
A='Y'	-	Y	N
EXIT	*		
B = 'Y'		*	
TEXT= 'EXPLAIN'		*	
SOLVE(A)			*
TRACE='RULEi'			*

Note that everything that concerns the relationship between objects A and B is located together in 1 table, in an easy to comprehend format. We call this a decision unit.

3.2 Forward chaining paradigm

This is the natural execution mode of decision tables, whereby data triggers conditions which then select certain actions such as calling other tables, updating some records or reaching some conclusions.

3.3 Generate-and-test paradigm

To test the versatility of Gess a simplified version of the expert system GA1 [3] was emulated in Gess. GA1 is an expert system which infers complete molecular structures from measurements of molecular pieces. GA1 was chosen because it uses a generate-and-test paradigm.

A simplified version of GA1 has been successfully implemented in Gess using 7 tables.

3.4 Reasoning under uncertainty

We also found that different methods were possible to implement reasoning under uncertainty in Gess.

With the first method we structure one of the I/O records as in figure 2, where PROBi is probability (or certainty factor) number i and TABi is the name of the decision table which represents hypothesis i. Thus the system currently " believes" hypothesis TABi with probability PROBi.

Record 1	PROB1	TAB1	PROB2	TAB2	PROB3	TAB3	

figure 2

This RECORD 1 which partially plays the role of blackboard [3] can be updated by decision tables representing the hypothesis the system is trying to prove. Probabilities can be changed or hypothesis added or deleted from this " blackboard" record.

Another method is a combination of the above method and the earlier mentioned backward chaining implementation.

A = 0	Y	N	N	N	N
0 < A < 0.5	-	Y	N	N	N
0.5 ≤ A < 1	-	-	Y	N	N
A = 1	-	-	-	Y	N
EXIT	*				
B = B + 0.2		*			
B = B + 0.5			*		
B = 1				*	
SOLVE(A)					*

figure 3

Here A and B do not just use YES or No, but certainty or probability factors. In the example an arbitrary formula is used to update the certainty factor of B, but any formula can be used. Remark that we still have all the relevant information concerning the relationship between A and B located in one decision unit.

A third method is the application of fuzzy sets which are implemented in a similar way as in figure 3, but where we generalise the mapping function between A and B.

The implementation in Gess of these 3 methods is quite straigthforward and provides an elegant way to deal with reasoning under uncertainty, whilst at the same time allowing an easy integration of explanatory text for each decision or action as explained in section 3.1.

Using the methods mentioned above, several small expert systems have been or are being implemented in Gess (e.g. car-engine adviser, student curriculum adviser).

4. PERFORMANCE

Gess is written in PL/1 and runs on IBM mainframes. The compiler comprises about 500k bytes and the executer, about 80k bytes. If you disregard the loading time of a cluster of decision tables, which can be as small as 50ms if they are all in one physical I/O block, then the average execution time of a table with 10 conditions and 10 actions is about 5ms on an IBM 3033U. This increases linearly with the number of conditions and actions.

5. CONCLUSIONS

Gess, based on decision tables, has a different approach to expert systems. However, by extending this basic idea we have created a general purpose tool which allows us to build powerful systems.

The knowledge representation is straightforward and is quickly grasped, even by non-DP people. It has been our experience that one quickly finds new and unforeseen ways of applying Gess, helped by this format of extended decision tables which eases the knowledge acquisition and representation problem and forces one to be systematic.

Another important feature of Gess is its close integration with the external computer environment (databases, application programs). This allows Gess to access and handle large knowledge bases and actually run on its own (batch mode).

On the other hand, it has been pointed out that decision tables can have some ambiguity. Solutions have been proposed for that [10] in the form of adding weights to conditions and combinations. This is one of the extensions currently being investigated. Other extensions are implementing an automated trace facility and extending the learning capabilities which are now limited to updating the facts database (records) and not the rule database (decision tables).

REFERENCES

1. Chvalovsky V., [1979], Problems with decision tables, *CACM* **19**, 12, pp 705-707.
2. Expert Ease, [1984], *Which computer*, pp 68,70-71.
3. Hayes-Roth F., Waterman D.A., Lenat D.B., [1983], *Building expert systems*, **1**, Addison-Wesley, Reading, Massachusetts.
4. Higgs J.E., [1982], Decision tables in real time systems, *Proceedings of the first annual conf. on computers and communications*, pp131-136.
5. Kirk H.W., [1965], Use of decision tables in computer programming, *CACM*, **8**,1, pp 41-43.
6. Lew A., [1982], On the emulation of flowcharts by decision tables, *CACM*, **25**,12, pp 895-905.
7. London K.R., [1972], *Decision tables*, Auerbach publ., Princeton.
8. Maes R., Vanthienen J., Verhelst M., [1981], Procedural decision support through the use of PRODEMO, *Procedures 2nd int. conf. on information systems*, Cambridge, Mass., pp135-152.
9. Reinwald L.T. and Soland R.M., [1966], Conversion of limited-entry decision tables to optimal computer prgrams, 1. minimum average time processing, *Journal of the ACM*, **13**, 3, pp 339-358.
10. Schneider M.L., [1985], Weighted decision tables - an alternative solution for ambiguity, *The computer journal*, **28**, 4, pp 366-371.
11. Verhelst M., [1980], *De praktijk vam beslissingstabellen*, Kluwer, Amsterdam.
12. Warner E., [1985], TI software aids decisions, *Computerworld*, **9**, 14, pp 69-90

NOTES FOR CONTRIBUTORS

The purpose of the journal will be to publish original papers in any field of computing. Papers submitted may be research articles, review artilces and exploratory articles of general interest to readers of the journal. The preferred languages of the journal will be the congress languages of IFIP although papers in other languages will not be precluded.

Manuscripts should be submitted in triplicate to:

Prof. G. Wiechers
INFOPLAN
Private Bag 3002
Monument Park 0105
South Africa

Form of manuscript

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins. Manuscripts produced using the Apple Macintosh will be welcomed. Authors should write concisely.

The first page should include the article title (which should be brief), the author's name and affiliation and address. Each paper must be accompanied by an abstract less than 200 words which will be printed at the beginning of the paper, together with an appropriate key word list and a list of relevant Computing Review categories.

Tables and figures

Tables and figures should not be included in the text, although tables and figures should be referred to in the printed text. Tables should be typed on separate sheets and should be numbered consecutively and titled.

Figures should also be supplied on separate sheets, and each should be clearly identified on the back in pencil and the authors name and figure number. Original line drawings (not photocopies) should be submitted and should include all the relevant details. Drawings etc., should be submitted and should include all relevant details. Photographs as illustrations should be avoided if possible. If this cannot be avoided, glossy bromide prints are required.

Symbols

Mathematical and other symbols may be either handwritten or typewritten. Greek letters and unusual symbols should be identified in the margin. Distinction should be made between capital and lower case letters; between the letter O and zero; between the letter I, the number one and prime; between K and kappa.

References

References should be listed at the end of the manuscript in alphabetic order of the author's name, and cited in the text in square brackets. Journal references should be arranged thus:

1. Ashcroft E. and Manna Z., The Translation of 'GOTO' Programs to 'WHILE' programs., *Proceedings of IFIP Congress 71,* North-Holland, Amsterdam, 250-255, 1972.
2. Bohm C. and Jacopini G., Flow Diagrams, Turing Machines and Languages with only Two Formation Rules., *Comm. ACM,* **9**, 366-371, 1966.
3. Ginsburg S., Mathematical Theory of Context-free Languages, McGraw Hill, NewYork, 1966.

Proofs

Proofs will be sent to the author to ensure that the papers have been correctly typeset and *not* for the addition of new material or major amendment to the texts. Excessive alterations may be disallowed. Corrected proofs must be returned to the production manager within three days to minimize the risk of the author's contribution having to be held over to a later issue.

Only orginal papers will be accepted, and copyright in published papers will be vested in the publisher.

Letters

A section of "Letters to the Editor" (each limited to about 500 words) will provide a forum for discussion of recent problems.