Computers Math. Applic. Vol. 21, No. 11/12, pp. 111–116, 1991
Printed in Great Britain.

0097-4943/91 $3.00 + 0.00

FUZZY DECISION TABLES FOR EXPERT SYSTEMS

ART LEW
Department of Information and Computer Science
University of Hawaii

(Received July 1990)

Abstract—The use of fuzzy decision tables as a programming language for representing both the knowledge and the procedures in expert systems is discussed. Examples of their use for the generation of procedural code and for the generation of if-then rules are given.

1. INTRODUCTION

The main advantages of decision tables [1,2] are that:

(a) they provide a systematic way to design algorithms;
(b) they are amenable to certain forms of automated error-checking (such as for consistency [3]) and to formal verification (by correctness proofs [4]); and
(c) a processor to compile and execute them is relatively easy to implement [5].

Decision tables (DTs) are compatible with most programming languages, including general-purpose languages such as Ada and C as well as special-purpose languages such as LISP [6] and Prolog; the processor described in [5] can be adapted, with varying degrees of difficulty, to translate DTs into any of these languages. We previously discussed the use of DTs as a general-purpose programming language [7]; here, we discuss their use for the implementation of rule-based expert systems.

2. FUZZY ALGORITHMS

In an earlier paper [8], we demonstrated that any (nonfuzzy) algorithm can be implemented by a DT. In principle, any fuzzy algorithm [9,10], if it is to be realized (i.e., executed on an actual computer), can also be implemented using a DT, albeit possibly a nondeterministic one [11]. For example, a fuzzy version of the DT in [7] for finding the shortest path in a graph can be based on fuzzy optimization algorithms [12]; some other examples of fuzzy optimization algorithms appear in [13–15].

3. APPLICATION OF DTS TO EXPERT SYSTEMS

It is natural to use DTs to express the knowledge base of an expert system [16]. Each `IF-THEN` rule of the form

```
IF P1
...
AND Pn
THEN C1
...
AND  Cm
{with certainty factor cf=n}
```

can be represented by a rule (or column) of a DT

```
P1 : T
Pn : T
---+---
C1 : X
Cm : X
{cf = n}
```

where the premises and consequents of the productions are the conditions and the actions of the DT. This idea is quite old; numerous rule-based expert systems incorporating decision-table concepts in their design and analysis have been reported (see, for example, [17–19]). Much of this prior work concerned methods of detecting ambiguities, regarded as a problem in that other context; here, we regard ambiguities as a desirable way to express fuzziness.

We distinguish between "procedural" DTs, in which the consequents are general actions (which may include "call" statements), and "nonprocedural" DTs in which the consequents are simple actions (i.e., value assignments). Generally speaking, we liken procedural DTs to forward-chaining production systems such as OPS5 [20], and nonprocedural DTs to backward-chaining inference systems such as Mycin [16]. Decision table processors can be designed to perform both forward and backward chaining. In this paper, we focus our attention on procedural DTs.

4. FUZZY DTS

DTs are suitable for expressing fuzziness in expert systems in a number of ways. The fuzziness may be in multi-valueness of conditions, nondeterminism of rules, or a variety of forms of uncertainty in the entries of any of the four DT quadrants or with respect to any row or column. For example, in the above DT, `Pn` or `Cm` may be fuzzy expressions, their values may be fuzzy rather than truth-valued possibly with varying degrees of certainty, and rules themselves may have associated certainty factors and may or may not be mutually exclusive. These variations can all be accommodated within the framework of "ambiguous" DTs. Ambiguous procedural DTs can be implemented as nondeterministic algorithms.

One problem in the handling of fuzziness for which there is no clear-cut solution relates to how fuzzy certainty factors should be defined and blended. Even in nonfuzzy contexts, different expert systems have adopted different conventions, none of which may be appropriate for a given application [21]. Use of DTs permits users to program their own blending formulas; unlike the case in most systems, many different formulas may be used in the same application and these formulas may be dynamically defined.

5. EXAMPLES

Examples of the application of fuzzy procedural DTs to the design of expert systems appear in [12], in which the nonfuzzy DTs used to solve the stock market problem as given in [2] were modified in what are rather minor ways. Even so, there are several errors in the fuzzy DTs of [12], such as the one given here as Table 1.

Table 1.

	R1	R2	R3	R4
trading possible?	Y	Y	N	
stkavg (X1)\ } bndavg (X2)/	X1+X2	X1.X2		X1/X2
call(stocks)	<=0.2			>0
call(bonds)	<=0.2			=0
call(account)	<=0.2			X
process R4		<=0.4		
go again	<=0.2			X
stop		>0.4	X	

Table 2.

	R1	R2	R2'	R3 : R4
trading possible ? :	Y	Y	Y	N
stkavg (X1)\ :				
} :	X1+X2	X1.X2		X1/X2
bndavg (X2)/ :				
update averages :	X	X	X	
call(stocks) :	<=0.2			=0
call(bonds) :	<=0.2			>0
call(account) :	<=0.2			X
process R4 {goto} :		<=0.4		
go again {repeat} :	<=0.2			X
stop {exit} :		>0.4	X	X

(Here, +, ., and / are fuzzy max, min, and diff operators.) A corrected version is given in Table 2; note the reversal of the relations X1/X2 > 0 and X1/X2 = 0.

The adoption in these procedural tables of a left-to-right interpretation convention with implicit ELSE rules, as well as of conditional rules associated with "goto" statements (such as to rule R4), is, we believe, undesirable. Table 3 shows an equivalent table which utilizes different conventions in which the "logic" is expressed in the upper right quadrant.

Table 3.

trading possible? :	T	T	T	T	F
stkavg+bndavg :	<=0.2	>0.2	>0.2	>0.2	-
stkavg.bndavg :	-	<=0.4	<=0.4	>0.4	-
stkavg/bndavg :	-	>0	=0	-	-
update averages :	X	X	X	X	-
call(stocks) :	X	-	X	-	-
call(bonds) :	X	X	-	-	-
call(account) :	X	X	X	-	-
repeat :	X	X	X	-	-
exit :	-	-	-	X	X

One advantage of this format, in which the rules can be evaluated in any order, is that optimal conversion algorithms (such as in [22]) may then be applied. (To simplify lexical processing, we used a slightly different "syntax" in our implementation, as illustrated below.)

6. IMPLEMENTATION

We attribute the errors in Table 1 and the other fuzzy DTs of [12] to the unavailability of a programming system with which the tables could be tested. This provided us with the motivation to design such a programming system. Our system [which at this writing is not yet complete] is essentially a preprocessor which translates fuzzy DTs, expressed in a format similar to that used in Table 3, into a conventional (procedural) language in a manner like that described in [5]. The precise DT format we adopted is given in the Appendix 1; its translation into a procedural language (which can be executed using an ordinary compiler) is shown in Appendix 2. Using the slightly different syntax given in Appendix 3, the DT can be translated into the set of IF-THEN rules shown in Appendix 4, which in turn can be processed using a simple expert system such as in [23].

7. CONCLUSION

Fuzzy DTs can be used to express the knowledge base of an expert system. DTs can also be used as the programming language in which expert systems are implemented (i.e., programmed,

in a choice of environments, such as LISP and C). That use of DTs is an advantageous way for humans to interact with computers, especially as a means to represent production rules, is demonstrated by their frequent adoption in the AI literature (e.g., see [13,20,24]). Of course, since tables are commonly used for a variety of other kinds of information, such as relational databases, design of an integrated system employing tabular representations of knowledge and procedures is worth serious consideration.

Another advantage of the use of fuzzy DTs to implement expert systems is flexibility. DTs permit both forward and backward chaining, and allow the incorporation of different ways of handling fuzziness (such as blending certainty factors) within the same application. (Details of a design which is comparable to System Z-11 [25] will be discussed in a forthcoming paper.)

We conclude that the design of expert systems utilizing fuzzy DTs is worthwhile and warrants further research and development, and we are so proceeding. One open research problem is whether better expert systems can be implemented using fuzzy DT processors, i.e., by incorporating fuzziness in the processor rather than in the tables upon which they operate.

REFERENCES

1. CODASYL Task Group, *A Modern Appraisal of Decision Tables*, ACM, New York, (1982).
2. R.B. Hurley, *Decision Tables in Software Engineering*, Van Nostrand Reinhold, New York, (1983).
3. P.J.H. King, The interpretation of limited entry decision table format and relationships among conditions, *Computer Journal* **12**, 320–326 (1969).
4. A. Lew, Proof of correctness of decision table programs, *Computer Journal* **27**, 230–232 (1984).
5. A. Lew, A decision table processor, *University of Hawaii, technical report* (1988).
6. B.M. Schwartz, LISP 1.5 decision tables interpreted for a small computer and proposed for parallel computers, *SIGPLAN Notices* **6** (8), 93–103 (1971).
7. A. Lew, Decision tables for general-purpose scientific programming, *Software–Practice and Experience* **13**, 181–188 (1983).
8. A. Lew, On the emulation of flowcharts by decision tables, *Commun. ACM* **25**, 895–905 (1982).
9. L. Zadeh, Fuzzy algorithms, *Information and Control* **12**, 94–102 (1968).
10. L. Zadeh, The role of fuzzy logic in the management of uncertainty in expert systems, *Fuzzy Sets and Systems* **11**, 199–227 (1983).
11. A. Lew, Decision table programming–some new perspectives, *University of Hawaii, technical report* (1983).
12. A. Kandel, *Fuzzy Mathematical Techniques with Applications*, Addison-Wesley, Reading, Mass., (1986).
13. R. Bellman and L.A. Zadeh, Decision-making in a fuzzy environment, *Management Sci.* **17** (4), B141–164 (1970).
14. R.L.P. Chang, Fuzzy decision tree algorithms, *IEEE Trans. Sys. Man Cyb.* **7**, 28–35 (1977).
15. A.O. Esogbue, Dynamic programming, fuzzy sets, and the modeling of R & D management control systems, *IEEE Trans. Sys. Man Cyb.* **13**, 18–40 (1983).
16. B. Buchanan and E. Shortliffe, *Rule-Based Expert Systems*, Addison-Wesley, Reading, Mass., (1984).
17. M. Trigoboff and C.A. Kulikowski, IRIS: A system for the propagation of inferences in a semantic net, *Proc. 5th IJCAI*, 274–280 (1977).
18. B.J. Cragun and H.J. Steudel, A decision-table-based processor for checking completeness and consistency in rule-based expert systems, *Int. J. Man-Machine Studies* **26**, 633–648 (1987).
19. R. Maes and J.E.M. VanDijk, On the role of ambiguity and incompleteness in the design of decision tables and rule-based systems, *Computer Journal* **31**, 481–489 (1988).
20. T. Cooper and N. Wogrin, *Rule-Based Programming in OPS5*, Morgan Kaufmann, San Mateo, Calif., (1988).
21. D. Kopso, L. Pipino and W. Rybolt, A comparison of the manipulation of certainty factors by individuals and expert system shells, *J.MIS* **5**, 66–81 (1988).
22. A. Lew, Optimal conversion of extended-entry decision tables with general cost criteria, *Commun. ACM* **21**, 269–279 (1978).
23. B. Sawyer and D. Foster, *Programming Expert Systems in Pascal*, Wiley, New York, (1986).
24. M.A. Carrico, J.E. Girard and J.P. Jones, *Building Expert Systems*, McGraw-Hill, New York, (1989).
25. K.S. Leung and W. Lam, Fuzzy concepts in expert systems, *Computer* **21** (9), 43–53 (1988).

APPENDIX A

A Fuzzy Procedural DT.

```
dtbegin!
trading possible        :T      T       T       T       F
max(stkavg,bndavg)      !<=0.2  >0.2    >0.2    >0.2    -
min(stkavg,bndavg)      !-      <=0.4   <=0.4   >0.4    -
diff(stkavg,bndavg)     !-      >0      =0      -       -
------------------------+-......-.......-.......-.......-
update averages         :X      X       X       X       -
call(stocks)            :X      -       X       -       -
call(bonds)             :X      X       -       -       -
call(account)           :X      X       X       -       -
repeat!                 :X      X       X       -       -
exit!                   :-      -       -       X       X
dtend!
```

APPENDIX B

Translation of the DT of Appendix A.

```
while not(lambda<0) do begin
if (trading possible)
   and (max(stkavg,bndavg)<=0.2)
  then begin
    update averages;
    call (stocks);
    call (bonds);
    call (account);
    lambda:=1;
  end else
if (trading possible)
   and (max(stkavg,bndavg)>0.2)
   and (min(stkavg,bndavg)<=0.4)
   and (diff(stkavg,bndavg)>0)
  then begin
    update averages;
    call (bonds);
    call (account);
    lambda:=1;
  end else
if (trading possible)
   and (max(stkavg,bndavg)>0.2)
   and (min(stkavg,bndavg)<=0.4)
   and (diff(stkavg,bndavg)=0)
  then begin
    update averages;
    call (stocks);
    call (account);
    lambda:=1;
  end else
if (trading possible)
   and (max(stkavg,bndavg)>0.2)
   and (min(stkavg,bndavg)>0.4)
  then begin
    update averages;
    lambda:=-1;
  end else
if not(trading possible)
  then begin
    lambda:=-1;
  end else
begin lambda:=-2; end;
end;
```

APPENDIX C

A Fuzzy Expert System (in a DT Format).

```
dtbegin!
trading possible=      !T      T       T       T       F
max(stkavg,bndavg)     !<=0.2  >0.2    >0.2    >0.2
min(stkavg,bndavg)     !       <=0.4   <=0.4   >0.4
diff(stkavg,bndavg)    !       >0      =0
-----------------------+-------+-------+-------+-------
update averages        !,      ,       ,       ,
call(stocks)           !,              ,
call(bonds)            !,      ,
call(account)          !,      ,       ,
exit                                           X       X
dtend!
```

APPENDIX D

Rules Associated with the Expert System of Appendix C.

```
Rule 1.1: if
    trading possible=T   and
    max(stkavg,bndavg)<=0.2
then
    update averages, and
    call(stocks), and
    call(bonds), and
    call(account).

Rule 1.2: if
    trading possible=T   and
    max(stkavg,bndavg)>0.2   and
    min(stkavg,bndavg)<=0.4   and
    diff(stkavg,bndavg)>0
then
    update averages, and
    call(bonds), and
    call(account).

Rule 1.3: if
    trading possible=T   and
    max(stkavg,bndavg)>0.2   and
    min(stkavg,bndavg)>=0.4   and
    diff(stkavg,bndavg)=0
then
    update averages, and
    call(stocks), and
    call(account).

Rule 1.4: if
    trading possible=T   and
    max(stkavg,bndavg)>0.2   and
    min(stkavg,bndavg)>0.4
then
    update averages, and
    exit.

Rule 1.5: if
    trading possible=F
then
 and
    exit.
```