On the Role of Ambiguity and Incompleteness in the Design of Decision Tables and Rule-Based Systems

R. MAES AND J. E. M. VAN DIJK*
University of Amsterdam, Vakgroep BIA, Jodenbreestraat 23, NL 1011 NH Amsterdam, The Netherlands

Reasoning about the practical meaning of ambiguity and incompleteness in decision tables leads to the notion of 'the life cycle of a decision table'. It is shown how both concepts can be applied in a positive sense during that life cycle. Also, a comparison is made of decision tables and rule-based knowledge systems. A number of problems connected with the occurrence of ambiguities and incompleteness in the design of decision tables and rule base systems are discussed.

Received January 1988

1. INTRODUCTION

The discussion on 'ambiguity', 'incompleteness' and the related concepts 'redundancy' and 'contradiction' in the context of decision tables is almost as old as the decision table technique itself (see, for example, Refs 24, 7, 8 and 9). In a more recent article, Schneider[30] argues in favour of the expulsion of ambiguities by the introduction of a rather complex set of weighting schemes.

A decision table is said to be **ambiguous** if 'there is a logically permissible piece of data which satisfies the condition entries of two (or more) rules having different action sets'.[7] Rules 2, 3 and 4 of Table 1 (and hence Table 1 itself) are ambiguous since they contain the combined condition state (A = Y, B = Y and C = N), and the action sets I (rule 3), II (rule 2) and III (rule 4) are different.

If the implied action sets are proven to be mutually exclusive, which means that they cannot be executed for the same piece of data, then we call the decision rules in question **contradictory**. If, on the other hand, the action sets of the overlapping rules are equal or equivalent, then the decision rules are called **redundant**. Note that in the (most common) case of not completely overlapping rules, the redundancy is partial. Rules 4 and 5 of Table 1, for example, are (partly) redundant by the fact that both rules specify that the common action set III has to be executed for the case (A = Y, B = N, C = N). These definitions are summarised in Table 2.

A decision table is defined to be **incomplete** if there are some logically possible combinations of condition states that are not included in the table. Table 1, for example, is incomplete by the fact that the combination of condition states (A = N, B = Y, C = Y) is not contained in this table (supposing the latter combination is logically feasible).

Two common features of the above-mentioned and many other publications on decision tables are (1) their purely theoretical point of view and (2) their exclusively negative attitude towards both ambiguity and incompleteness. By their theoretical nature, these publications almost completely disregard the pragmatic context in which, and hence the reasons why, decision tables are used. This attitude is stressed by the self-evidence by which any problem-oriented extended-entry decision table is replaced by the logically equivalent yet pragmatically inconvenient limited-entry decision table.

The occurrence of **ambiguities** is said (as, for example, in Ref. 30) to be a degeneration of decision tables, only existing in badly formed tables and posing basic constraints on the applicability of the decision-table technique. In many cases, **incomplete** decision tables are complemented by the introduction of an ELSE-rule, capturing any combination of condition states not explicitly mentioned in the other, so-called 'normal' decision rules. A common practice to exclude any confusion about ambiguities and at the same time to facilitate completeness checking is to confine the decision-table format to mutually exclusive, non-overlapping decision rules.

It is our belief that the discussion on ambiguities, incompleteness and the value of the restricted decision-table format should be carried on in the light of the **pragmatic use** of the decision table at issue. One rather traditional classification of the use of decision tables is by application area. In actual practice, decision tables are used for representing program logic (as is advocated in Refs 12 and 13) and for determining information requirements as they are found in managerial decision processes, prescriptive texts, etc. (as is argued in Refs 3

Table 1

	Rule no.				
	1	2	3	4	5
Cond. A	Y	Y	—	Y	—
Cond. B	N	Y	Y	—	N
Cond. C	—	—	N	N	N
Act I	X	—	X	—	—
Act II	—	X	—	—	—
Act III	—	—	—	X	X

Table 2

Rules are overlapping?	Y		N	
Action sets are different?	Y		N	—
Action sets are mutually exclusive?	Y	N	—	—
Rules are:				
ambiguous	X	X	—	—
contradictory	X	—	—	—
redundant	—	—	X	—

* To whom correspondence should be addressed.

and 19). The former use was criticised in Ref. 15, while the latter use is very near to the application of formalisms for the rule-based representation of knowledge. A vigorous argument against the application of these rule-based formalisms for representing law was given by Leith.[10]

Another, less conventional classification of decision-table pragmatics is by the object pursued by the person using and/or manipulating the table. In general, one can distinguish between the use of decision tables as (active) tools for **modelling** decision processes and as (passive) expedients for **making** decisions. It will be shown in the following sections that both ambiguities and incompleteness play a positive role in the modelling phase (where the decision table is gradually built up), whereas they have to be excluded from the execution phase (where the decision table is merely interpreted). Their positive contribution is mainly exploited when dealing with the analysis of complex decision logic, where the introduction of computer assistance is almost obligatory (see also Ref. 31).

It will be argued in Section 2 that one and the same decision situation has successively to be represented by different formats, if one wants to fully exploit the advantages of decision tables. This observation leads to the notion of a 'decision table life cycle'.

A further application of our observations is made by comparing decision tables with rule-based representation formalisms. It can be observed, for example, that both terms are defined in a very similar manner when dealing with rule-based systems (see, for example, Refs 14 and 22). Detecting ambiguities, contradictions and incompleteness is a research area of growing importance in designing rule-based systems.[5, 14, 22, 23, 26, 33]

In Section 3 we compare rule-based systems with decision tables. This enables us to discuss the use of ambiguities and incompleteness in the design of decision tables and rule-based systems in Section 4.

2. THE LIFE CYCLE OF A DECISION TABLE

The conventional way of discussing the merits of decision tables is by testing them against the demands made by users (humans and computers) of the *final* version of the table. These demands are dictated by the general demand for quick and correct decision making (i.e. interpretation of the decision table). In what follows we call this ready-to-use version the 'interpretation time' version of a decision table; its features are discussed in Section 2.4.

Quite different demands are made by the persons in charge of the *decision modelling process* and hence of the construction of the 'interpretation time' version of the decision table. Their demands are dictated by their needs for tools that systematically and thoroughly support the problem analysis process. Similar observations are made by Levesque with regard to the design and use of knowledge-based systems: 'It is shown how different the use of the language must be, depending on whether the interaction involves querying or defining the knowledge base'.[11] The features of the successive 'construction time' and 'test time' versions of a decision table are discussed in Sections 2.2 and 2.3 respectively. In Section 2.1 we first discuss the form of the original problem description.

2.1 Stage 0: the unanalysed problem description

Before starting the decision-modelling process, one has to carefully examine whether decision tables are the most appropriate tools for representing and analysing a given problem. The answer to this question, however, is beyond the scope of the present paper; in what follows, we assume that this preliminary consideration has been made.

The original description (that has to be converted into its decision-table equivalent) can exist, as has been stated in Ref. 3, in two different forms; often these forms will be intermixed.

(*a*) In many cases, there exists a more or less explicit description of the decision logic. This description can either be available in written form (as is regularly the case when dealing with prescriptions, regulations, procedures, etc.) or it can exist in a (quasi)structured manner in the mind of the expert(s). In these cases, the purpose of the use of decision tables is to analyse and eventually correct the existing description and to standardise and speed up the ultimate decision-making process. In essence, the decision table is applied as a vehicle for knowledge debugging.

(*b*) In other cases, there merely exists an implicit description of the decision process to be modelled. This situation is sometimes called an 'a priori unstructured' situation. By the introduction of decision tables (for example in an interview session), one tries not only to pursue the objectives mentioned in (*a*), but also to make the decision process explicit: the decision table is used as a vehicle for knowledge explication.

The first step in decision modelling is to make the description as transparent and formalised as possible. This can be done by translating the original perception of the decision situation into an intermediate decision-modelling language, an example of which was presented in Ref. 20. In this language the given situation is split up into a number of conditions (constraints), actions (outcomes) and decision logic. Further, each chunk of decision logic is converted into a conditional expression (a rule). This intermediate description is a (partly) formalised, yet unanalysed model of the given decision situation. In many situations this model will be incomplete, inconsistent and/or partially redundant.

Example

In the following fictitious example, we are restricting ourselves to introducing the decision logic of an English-like variant of the decision-modelling language proposed in Ref. 20. We assume that there exist 3 conditions (with respectively 3, 4, and 3 values each) and 4 actions. Conditions are indicated by the prefix 'CON', condition values by the lower-case letters 'a', 'b', 'c' and 'd' (where 'ab' means 'a OR b') and actions by the prefix 'ACT'.

<1> ACT_1 ALWAYS IF CON_2a
<2> ACT_2 IF (CON_1a OR CON_2b) AND CON_3ab
<3> NOT ACT_2 IF CON_2b AND CON_1b
<4> CON_2c IMPLIES CON_3c
<5> ACT_1 IMPLIES ACT_3 OR ACT_4
<6> CON_1ab AND CON_3b AND CON_2d IS IMPOSSIBLE

Table 3

	Conditional expression no.										
	1	2	2	2	2	3	7	7	8	8	8
CON_1	—	a	a	—	—	b	a	c	—	—	—
CON_2	a	—	—	b	b	b	—	—	—	—	—
CON_3	—	a	b	a	b	—	—	—	c	c	c
ACT_	!1	2	2	2	2	—	—	—	—	—	—
NOT ACT_	—	—	—	—	—	2	4	4	1	2	4
Rule no.	1	2	3	4	5	6	7	8	9	10	11

Note. !1 = EXECUTE ALWAYS ACT_1 – no exception allowed.

<7> ACT_4 ONLY IF CON_1b
<8> ONLY ACT_3 IF CON_3c
This list of conditional expressions is thought to be a literal translation of (part of) the original problem statement.
End example

2.2 Stage 1: the 'construction time' decision table

The unanalysed description of stage 0 is first transformed into a 'construction time' decision-table format. The successive 'construction time' versions of a decision table are primarily used for analysing a given problem situation. In the optimal case, the decision modeller is assisted by a decision-table generating program, examples of which are given in Refs 4, 17, 19 and 34. These generators not only manipulate and ultimately transform the table into its 'test time' and 'interpretation time' versions, yet also give the modeller substantial support and feedback by exploiting the decision-table format: the decision table is used as a discipline of thought. Examples can be found in Refs 3, 17 and 19, while the specific role of both ambiguities and incompleteness in the design of decision tables and other rule-based systems will be discussed in Section 4.

Both Maes and Van Dijk[20] and Puuronen[25] argue in favour of the use of the extended decision grid chart in this stage of the modelling process. A decision grid chart is nothing but a multiple-hit decision table (i.e. a decision table with mutually non-exclusive rules); the relations between the conditions as well as between the actions are added as extensions in separate tables.

Example

The logical expressions given in Section 2.1 are transformed into a 'construction time' decision table (Table 3).

The impossible combinations and the implications of conditions (conditional expressions <4> and <6>) and of actions (conditional expression <5>) are put in two separate 'IMPOSSIBLE'-tables:

Table 4*a*. Impossible combinations of conditions

	Conditional expression no.			
	4	4	6	6
CON_1	—	—	a	b
CON_2	c	c	d	d
CON_3	a	b	b	b

Table 4*b*. Impossible combinations of actions

Conditional expression no.	5
ACT_1	Exe
ACT_2	—
ACT_3	NExe
ACT_4	NExe

An analysis (preferably by means of a decision-table generating program such as the ones presented in Refs 4 and 17) reveals that there exists, among other things, a contradiction between rules no. 1 and no. 9 of table 3.
End example

In general, 'construction time' decision tables are *analytical* in that they help the decision modeller to analyse, to complete and to correct the decision description being gradually built up. It will be shown in Section 4 that both ambiguity and incompleteness play a vital role in this analysis and debugging process. This general nature can be translated into the following features.

(1) Problem- and action-orientated

At each moment of the modelling process, a 'construction time' decision table is the translation of the instantaneous problem logic into its disjunctive normal form. The conditions of a single rule have to be 'AND'-ed, whereas the rules themselves are interlinked by an implicit 'OR'-operator. This means that an expression in the decision-modelling language containing one or more 'OR'-operators results in more than one rule in the 'construction time' decision table. Negations in the conditional part of the expressions are replaced by the complementary values of the conditions.

Because of this, each rule of the table is an elementary conditional expression, linking an action to (some of) its controlling conditions, see also Ref. 27. An action can be governed by more than one rule, whereas two or more rules can be mutually contradictory. Basically, a 'construction time' decision table shows under which conditions the individual actions have/have not to be executed. The same remark applies to the decision-modelling language[20] and to many texts describing procedures or regulations.[16]

The problem- and action-orientated nature of the 'construction time' decision table also means that the traditional way of specifying actions ('execute'/'don't execute') has been extended into the more subtle set of five values:

- execute (no exception allowed) ('!' in the above example);
- execute unless otherwise specified in another rule;
- undetermined for the moment;
- don't execute unless otherwise specified in another rule;
- don't execute (no exception allowed).

This extension of the set of action values is comparable to the distinction made by Sridharan:[32] 'rules can be used to communicate not only **permissions**, but also to specify **obligations** as well as **prohibitions**'.

Table 5

Rule no.	1	2	3	4	5	6	7	8	9	10	11	12
Con-1	—	a	a	—	—	a	c	—	—	b	a	c
Con-2	a	—	—	b	b	—	—	—	—	—	a	a
Con-3	—	a	b	a	b	—	—	c	c	—	—	—
Act	!1	2	2	2	2	—	—	—	—	3	3	3
Not Act	—	—	—	—	—	4	4	2	4	—	—	—

(2) Incomplete

By its instantaneous nature, each version of the 'construction time' decision table is in itself an incomplete description of the decision situation. The decision-modelling process is transformed into the process of building the complete decision table step by step.

(3) Multiple hit

In fact, the 'construction time' decision table is an intermediate form between the original insights as they are uttered by the decision modeller and added to the instantaneous description at one side, and the ultimate decision model to be used by the decision maker at the other side. By opting for a multiple-hit representation format (i.e. a decision table with mutually overlapping rules), one allows for the manipulation of the original utterances. A typical manipulation is the compaction of rules, for example by applying the iterative consensus algorithm developed in Ref. 18. This manipulation helps the decision modeller in clarifying his/her insight into the problem situation (see, for example, Ref. 25 for a discussion of this aspect).

2.3 Stage 2: the 'test time' decision table

The successive versions of a 'construction time' decision table ultimately lead to a problem description in the form of a huge collection of decision rules. The relation between these rules (and between the rules of the main decision table and those of the 'impossible' tables) are implicit: there exists no synthetic view of the set of actions to be executed for a given combination of condition values.

A 'test time' decision table is the translation of a 'construction time' table together with its accompanying 'impossible' tables into its condition-orientated equivalent. This means that, for the first time, a decision modeller is confronted with a synthetic view of the decision in which a rule no longer tells under what conditions an action has/has not to be executed, yet which set of actions has to be executed for a given combination of condition values.

In actual practice, 'test time' decision tables are mainly constructed at the end of the analysis phase. In that case, they give a synthetic overview of the actual implications of the *combined* conditional expressions that are entered in the previous modelling stage and hence they can lead to a further tuning of the decision model under development.

Example

Suppose that during Stage 1 the 'construction time' Table 3 was ultimately altered into the version seen in Table 5 (we deliberately omit the references to the conditional expressions).

All obviously detectable contradictions have been removed from Table 5 and the 'impossible' tables 4*a* and 4*b* remain the same.

The corresponding 'test time' decision table then looks like Table 6.

End example

Table 6 immediately reveals the inherent drawback of 'test time' decision tables, namely their comprehensiveness. 'Test time' decision tables, however, appear to be a valuable intermediate format for finishing the decision-analysis process; for the first time, the decision modeller is confronted with a visual, condition-orientated overview of the complete set of individual situations that can occur. Typical examples of interventions to be undertaken by the modeller include the specification of empty rules (where no action was specified during the previous stage) and checking for impossible combinations of actions (a decision-table generating program can deal with this problem based on the 'construction time' representation and the impossible action table, for example Table 4*b*). On many occasions the confrontation with a test-time decision table seems to be an eye-opening exercise, even for the experienced decision modeller (see also Ref. 2).

A 'test time' decision table is built by taking the Cartesian product of all the previously specified condition values. Each rule indicates the combined set of actions to be taken for one particular combination of condition values. Hence, 'test time' decision tables are (1) complete;

Table 6

Rule no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Con-1	a	a	a	a	a	a	a	a	a	a	a	a	b	b	b	b	b	b	b	b	b	b	b	b	c	c	c	c	c	c	c	c	c	c	c	c
Con-2	a	a	a	b	b	b	c	c	c	d	d	d	a	a	a	b	b	b	c	c	c	d	d	d	a	a	a	b	b	b	c	c	c	d	d	d
Con-3	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c
Act-1	x	x	x	.	.	.	.	.	.	.	.	.	x	x	x	.	.	.	.	.	.	.	.	.	x	x	x	.	.	.	.	.	.	.	.	.
Act-2	x	x	.	x	x	.	x	x	.	x	x	.	.	.	.	x	x	.	.	.	.	.	.	.	.	.	.	x	x	.	.	.	.	.	.	.
Act-3	x	x	x	.	.	.	.	.	.	.	.	.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	.	.	.	.	.	.	.	.	.
Act-4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Impos	.	.	.	.	.	.	x	x	.	.	x	.	.	.	.	.	.	.	x	x	.	.	x	.	.	.	.	.	.	.	x	x	.	.	.	.

Table 7

Rule no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Con-1	a	a	a	a	a	a	a	a	a	a	a	a	b	b	b	b	b	b	b	b	b	b	b	b	c	c	c	c	c	c	c	c	c	c	c	c
Con-2	a	a	a	b	b	b	c	c	c	d	d	d	a	a	a	b	b	b	c	c	c	d	d	d	a	a	a	b	b	b	c	c	c	d	d	d
Con-3	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c
Act-1	x	x	x	.	.	.	.	.	.	.	.	.	x	x	x	.	.	.	.	.	.	.	.	.	x	x	x	.	x	.	.	x	.	.	x	.
Act-2	x	x	.	x	x	.	x	x	x	.	.	.	.	.	.	x	x	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.
Act-3	x	x	x	.	.	x	.	.	.	x	.	x	x	x	x	x	x	x	.	.	x	x	.	x	x	x	x	.	x	x	.	x	.	x	x	x
Impos	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	x	x	.	.	x	.	.	.	.	.	.	.	x	.	x	.	.	.

Table 8

Rule no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Con-1	a	a	a	a	a	a	a	a	b	b	b	b	b	b	c	c	c	c	c	c	c
Con-2	a	a	a	b	b	b	c	d	a	b	b	b	c	d	a	—	a	b	b	d	d
Con-3	a	b	c	a	b	c	—	—	—	a	b	c	—	—	a	b	c	a	c	a	c
Act-1	x	x	x	.	.	.	.	.	x	.	.	.	.	.	x	x	x	.	.	.	.
Act-2	x	x	.	x	x	.	x	.	.	x	x	.	.	.	.	.	.	x	.	.	.
Act-3	x	x	x	.	.	x	.	x	x	x	x	x	x	x	x	x	x	.	x	x	x

(2) single hit; (3) condition orientated; (4) expanded. Expanded 'test time' decision tables that are too comprehensive can be split up into a hierarchy of smaller decision tables.

2.4 Stage 3: the 'interpretation time' decision table

The final version of a decision table is merely interpreted by its users. Both humans and computers can make use of decision tables in order to execute a given decision process. It has been noted elsewhere (for example, in Refs 3 and 17) that the applicability of decision tables for human decision making has thus far been under-exposed.

The general nature of 'interpretation time' decision tables is synthetic, as opposed to the analytical nature of 'construction time' tables. The following characteristics are commonly mentioned when dealing with the former kind of decision tables.

(1) *Complete.* All logically feasible combinations of data a user can be confronted with should be anticipated in the decision table.

(2) *Solution-orientated.* The decision table should contribute to correct decision making. This means that, in the case of a human user, extended-entry decision tables are preferable to limited-entry tables for the simple reason that they are much more user-orientated and hence lead to fewer interpretation errors.

(3) *Optimally compressed.* The use of decision tables should also lead to fast decision making (in the case of human users) or to minimal interpretation/execution time (in the case of computer programs). It should be observed that, in general, algorithms for compressing decision tables only lead to optimal results in the case of a decision table that is not too complex. These algorithms take care of the interpretation time of the individual conditions and of the rule frequencies: condition tests demanding a great amount of time and occurring frequently are removed by preference from as many as possible rules of the 'interpretation time' version of the table. A detailed description of compressing decision tables is beyond the scope of the present paper. This subject will be dealt with in a forthcoming paper, see Ref. 21.

(4) *Unequivocal.* In the case of human users, both fast and correct decision making is highly stimulated by the use of single-hit decision tables with mutually exclusive decision rules. It is rather unclear from the literature whether the compression of decision tables into multiple hit tables (with mutually overlapping rules) can lead to decision tables that can be interpreted by computers in a faster way than their single-hit equivalents. In the case of multiple-hit tables, a supplementary convention should be made on the interpretation one has to give to mutually overlapping rules in order to avoid ambiguities. In Ref. 8 King gives five alternative conventions, four of which represent cases where no more than one action set is executed on each occasion the decision table is interpreted.

Example

Suppose Table 6 is converted into Table 7 during the test stage.

The 'test time' decision Table 7 can be converted into the 'interpretation time' decision Table 8.

The rules have been maximally contracted. Condition 3, for example, is irrelevant for Rules 7–9 of Table 7 (the same action set Act-2 applies). These three rules have been contracted in Rule 7 of Table 8. We further suppose that a combination of condition values that is logically impossible does not in fact occur during the interpretation of the table. By this means, rules with an impossible action can be deleted or can be contracted with other rules.

End example

The attention paid to the decision-table format in the

Table 9

	Construction time	Test time	Interpretation time
Main orientation	Action	Condition	Condition
Combinatorially complete	No	Yes	Yes
Single or multiple hit	Multiple	Single	Single
Compressed	—	No	Yes

past was almost exclusively devoted to this type of decision table. It can be concluded from the foregoing discussion that we fully agree with the common opinion that both ambiguity and incompleteness should be excluded from 'interpretation time' decision tables.

2.5 Conclusion

Each stage of the decision modelling process requires a dedicated decision table format. The main features of these formats are summarised in Table 9.

Each stage in the modelling process may include some major or minor revisions of the decision table in the actual or in a previous format. In a further stage, the maintenance of the decision model results in an update of one or more of the stages described in Sections 2.1–2.4.

3. A COMPARISON OF DECISION TABLES AND RULE-BASED SYSTEMS

The similarity of decision tables and rule-based systems is mentioned by several authors (for example Hart[6]), and more thoroughly compared in Refs 5, 26 and 34. The most interesting points of comparison, i.e. the reasoning mechanism, the knowledge-representation structure and the induction process, are dealt with in the following subsections. The motivation behind this comparison is our belief that advances in both research fields can be mutually fructifying. An example of this mutual interest is given in Section 4, where the role of ambiguity and incompleteness in the design of decision tables and rule-based systems is discussed.

3.1 Reasoning in decision tables and rule bases

Two methods of reasoning frequently used in rule-based systems are forward reasoning (data driven) and backward reasoning (goal driven). Forward reasoning consists of two cycles; first the known facts are looked up, then these known facts are used to deduce new facts from the rules. Backward reasoning consists of three cycles; first a goal or a subgoal is chosen, secondly the rules that have that (sub)goal as a conclusion are looked up, thirdly it is determined whether these rules apply, and if not a new subgoal is chosen.

Reasoning in 'interpretation time' decision tables is always data driven: during the interpretation process, no hypotheses are made concerning the actions to be executed. We further suppose that during that stage, all necessary condition values can be obtained from the user (the decision maker) or can be searched for in a database. The interpretation process is straightforward and unidirectional: the number of known condition values successively increases, and the number of rules that satisfy that particular combination of condition values steadily decreases until only one rule is still applicable, at which moment the set of actions to be executed is determined. If the system consists of a hierarchy of decision tables, one or more of the actions can activate a decision table of the next level in the hierarchy.

Further, the reasoning process in interpreting decision tables is generally interrupted when encountering a condition value that cannot be determined. The easy, static way around is to try to contract the decision table during the design of the 'interpretation time' version in such a way that as many tests as possible of that particular condition value are removed from the decision table, see Refs 21 and 25. A more advanced, dynamic way would be to recontract the decision table during the interpretation stage and hence to avoid the test of that condition value in the column on hand. In a lot of cases, both approaches are unsuccessful and hence the interpretation of the decision table does not lead to the execution of a set of actions.

The reasoning mechanism of rule-based systems, on the contrary, is more flexible in that encountering a condition value that cannot be determined does not necessarily lead to the interruption of the reasoning process. The inference engine then tries to reach conclusions via another path by chaining rules that do not contain undeterminable condition values.

3.2 Rule structure in decision tables and rule bases

Suppose we have a decision situation that is described by the following seven rules: if B then F, if A & C & D then G, if B & G then J, if F & H then X, if J & K then X, if G & E then K, if E then H.

In this example A, B, C, D, E are 'basic' conditions (they only turn up at the left-hand side of the rules). We further suppose that only the values of the basic conditions can be obtained from the user or can be looked up in a database. All other conditions and actions must be determined by applying the rules. Terms F, G, J, H and K are 'intermediate' terms (they turn up at the left-hand side of some rules and at the right-hand side of other rules), and X is the only final goal.

This decision situation can be implemented in one decision table (see Table 10). This decision table is similar to a 'construction time' decision table (as discussed in 2.2), with many terms appearing both in the condition part and in the action part. Therefore the execution of this ambiguous decision table is iterative and hence rather laborious.

Converting this decision logic to a 'test time' decision table containing only the 'basic' conditions A, B, C, D and E and the final conclusion X reveals that in fact only conditions B and E are relevant in order to determine whether action X has to be executed. These relations are shown in Table 11.

Table 10

A	—	Y	—	—	—	—	—
B	Y	—	Y	—	—	—	—
C	—	Y	—	—	—	—	—
D	—	Y	—	—	—	—	—
E	—	—	—	—	—	Y	Y
F	—	—	—	Y	—	—	—
G	—	—	Y	—	—	Y	—
H	—	—	—	Y	—	—	—
J	—	—	—	—	Y	—	—
K	—	—	—	—	Y	—	—
	F	G	J	X	X	K	H

Table 11

B	Y	Y	N
E	Y	N	—
X = Yes	x	—	—
X = undetermined	—	x	x

This example demonstrates the advantages of the life-cycle approach. An inference engine would need to chain rules to reach the final conclusion if the original rules were implemented without further analysis. The analysis results in a single-hit table that does not contain any redundancy and therefore can be efficiently executed. The life-cycle approach gives the decision modeller a deeper understanding of the decision situation and supports better decision making.

3.3 Induction and the design of decision tables and rule bases

Several authors (see, for example, Arbab,[1] Hart[6] and Quinlan[28,29]) propose induction as an alternative method for the design of rule bases. The advantages of their approach are that the human expert can restrict himself to the specification of a number of examples. These examples are picked up by an induction mechanism that automatically generates the more general rules of the rule base.

The examples given by these authors, however, are very near the concepts of decision tables and decision trees: there is a strong separation of conditions and actions, the rules are contracted to avoid unnecessary condition testing and uncertainty is not allowed in the rules. This might mean that induction is the appropriate way of knowledge elicitation in those cases where decision tables are well-suited alternative representations. In Ref. 20 it is indicated that knowledge generalisation can be achieved by applying common decision-table techniques to rule-based systems. The application of these techniques to the induction process seems to be a promising research area.

The induction process frequently involves the use of probabilistic rule bases. Decision tables, on the contrary, are purely deterministic forms. It should however be remarked that the impossibility of using uncertainties and probabilities in rules is less severe than is often stated. The rules of probabilistic rule bases are mostly derived from examples taken from large databases (for example patient data) and are not provided by any expert. Both the use and the maintenance of this type of rule base are restricted by the inadequacy of human experts matching their knowledge to the generated knowledge.

4. AMBIGUITY AND INCOMPLETENESS IN DECISION TABLES AND RULE-BASED SYSTEMS

Both ambiguity and incompleteness are recognised as major problems in the design of decision tables and (more recently) of rule-based systems. A rising number of publications on knowledge engineering is explicitly devoted to this topic (see, for example, Refs 5, 14, 22, 23, 26 and 33). We firmly believe that the explicit recognition of the occurrence of ambiguity and incompleteness during the successive stages of the design process adds to the value of the final model (whether a decision table or a rule base). One can further infer that we argue in favour of the application of the 'life cycle' concept, as explained in Section 2, to the design of rule-based systems.

In what follows, we discuss a number of design problems related to different appearances of redundancy, ambiguity and incompleteness. For each problem, the stage of the life cycle in which it is detected and a solution are given. Note that both the problem and the solution are valid, independent of the question whether the 'interpretation time' model will be implemented as a decision table or as a rule base. For ease of survey, we refer to the stages of the life cycle as follows: 0 means the unanalysed problem description, 1 the 'construction time' stage, 2 the 'test time' stage, 3 the 'interpretation time' stage.

Problem 1. No action is specified for one or more combinations of condition values.

Occurrence. Stage 2.

Solution. Specify which action set has to be executed for that combination of condition values, or establish that that combination is logically impossible. One of these two options has to be entered in the 'construction time' decision table.

Problem 2. One of the actions specified is never used as one of the executable actions in the decision table.

Occurrence. Stage 1 (or 2).

Solution. Remove the action from the set of actions or add the appropriate action entries by entering (a) new rule(s) (Stage 1) or (an) action entr(y)(ies) (Stage 2).

Problem 3. One of the conditions appears to be irrelevant in the decision situation (see example in Table 13, con-2 is irrelevant).

Occurrence. Stage (2 or) 3, during the contraction of the decision table.

Solution. Remove the irrelevant condition from the set of conditions.

Problem 4. A decision situation is specified in a conflicting way in two or more overlapping rules. The

Table 13

CON-1	a	a	a	b	b	b
CON-2	a	b	c	a	b	c
Act	1	1	1	2	2	2

following ambiguities are possible (ACT-1 and ACT-2 are supposed to be mutually exclusive; we refer to the 5-valued logic as presented in 2.2).

(*a*) ACT-1 ! and ACT-2 ! Serious contradiction that has to be removed from the specification.

(*b*) ACT-1 ! and ACT-2. ACT-1 overrules ACT-2 because 'ACT-2' means 'execute ACT-2 unless otherwise specified'. This occurrence may be an indication of an unclear understanding of the decision situation and has in general to be further examined.

(*c*) ACT-1 and ACT-2. This ambiguity has to be resolved by the decision modeller: there is no indication of which action overrules the other one.

(*d*) ACT-1 (or ACT-1 !) and IMPOSSIBLE. See (*a*), unless one adopts the convention that 'IMPOSSIBLE' always overrules any other specification.

Occurrence. Stage 1 or 2. Remark that the same problems occur when ACT-2 in the above examples is replaced by the negation of ACT-1.

Problem 5. Not all the relevant conditions or condition values are specified.

Occurrence. Stage 1, 2 or 3.

Solution. Add the condition or condition values to the condition list and extend the decision tables in the different stages in accordance. It is clear that remedying this omission is a time-consuming activity.

Problem 6. An action that is important in the decision situation is not specified.

Occurrence. All stages.

Solution. Add the action to the action list and to the appropriate rules.

Problem 7. Redundancy.

Occurrence. Stages 1 and 2.

Solution. The problem of redundancy is 'automatically' solved when the decision table is compressed. Optimally compressed decision tables that are single hit do not contain redundant information.

The systematic elimination of these and similar problems during Stages 1, 2 and 3 of the life cycle is very helpful in producing a consistent and complete model of the decision situation. Further research on the typology of the problems related to ambiguity, incompleteness and redundancy could lead to the construction of an expert system-based decision support environment, a skeleton of which was presented in Ref. 20.

5. CONCLUSION

At any stage of the decision-modelling process, the modeller's knowledge and insight may be incomplete, inconsistent or at least crumbled and unclear. This results in a redundant, contradictory and/or incomplete decision model. We have shown that the systematic elimination of these irregularities via the 'life cycle' concept can be a thought-provoking exercise, leading to a consistent and complete description of the decision situation.

The current approaches to the design and use of decision tables and rule bases are different. Research on decision tables is concentrated on the 'interpretation time' stage of the life cycle (optimal contraction, conversion to computer programs, etc.), while research on rule bases is mainly dealing with the first stages of the life cycle (knowledge elicitation, knowledge acquisition, etc.) and only recently paid attention to the efficient use (for example minimal number of questions put to the user and avoidance of ambiguity, redundancy and incompleteness) of the rule base.

The successive decision-table formats as they are described in this article are effective tools for debugging decision logic. This effectiveness is independent of the decision whether the 'interpretation time' decision logic is implemented in a decision table or in a rule base.

REFERENCES

1. B. Arbab and D. Michie, Generating rules from examples. *Proceedings IJCAI 1985*, pp. 631–633 (1985).
2. W. J. Clancey, Heuristic classification. *Artificial Intelligence* **27**, 289–350 (1985).
3. *Codasyl, A Modern Appraisal of Decision Tables*. Report of the Decision Table Task Group, ACM, New York (1982).
4. L. H. Geesink and J. E. M. van Dijk, The construction of decision tables in PROLOG. Forthcoming.
5. B. J. Gracun and H. J. Steudel, Completeness and consistency in rule-based expert systems. *International Journal of Man–Machine Studies* **26**, 633–648 (1987).
6. A. Hart, The role of induction in knowledge elicitation. *Expert Systems* **2** (1), 24–28 (1985).
7. P. J. H. King, Ambiguity in limited entry decision tables. *Communications of the ACM* **11** (10), 680–684 (1968).
8. P. J. H. King, The interpretation of limited entry decision table format and relationships among conditions. *The Computer Journal* **12** (4), 320–326 (1969).
9. P. J. H. King and R. G. Johnson, Some comments on the use of ambiguous decision tables and their conversion to computer programs. *Communications of the ACM* **16** (5), 287–290 (1973).
10. P. Leith, Fundamental errors in legal logic programming. *The Computer Journal* **29** (6), 545–552 (1986).
11. H. J. Levesque, The logic of incomplete knowledge bases. In *On Conceptual Modelling*, edited M. L. Brodie *et al.*, pp. 165–189. Springer-Verlag, New York (1984).
12. A. Lew, Proof of correctness of decision table programs. *The Computer Journal* **27** (3), 230–232 (1984).
13. A. Lew, *Computer Science: a Mathematical Introduction*. Prentice-Hall, Englewood Cliffs, N.J. (1985).
14. D. W. Loveland and M. Valtorta, Detecting ambiguity: an example in knowledge evaluation. *Proceedings IJCAI 1983*, pp. 182–184 (1983).
15. R. Maes, On the representation of program structures by decision tables: a critical assessment. *The Computer Journal* **21** (4), 290–295 (1978).
16. R. Maes, An algorithmic approach to the conversion of decision grid charts into compressed decision tables. *Communications of the ACM* **23** (5), 286–293 (1980).
17. R. Maes, J. Vanthienen and M. Verhelst, Procedural decision support through the use of PRODEMO. *Proc. 2nd Int. Conference on Information Systems, Boston*, pp. 135–153 (1981).
18. R. Maes, On minimizing decision grid charts. *Angewandte Informatik* **24** (9), 451–455 (1982).
19. R. Maes, J. Vanthienen and M. Verhelst, Practical experiences with the PROcedural DEcision MOdelling system. *Proc. IFIP Working Conference on Processes and Tools for Decision Support*. North-Holland, Amsterdam, pp. 139–154 (1983).

20. R. Maes and J. E. M. van Dijk, A user-friendly propositional formalism for a managerial DSS-generator. In *Artificial Intelligence in Economics and Management*, edited L. F. Pau, pp. 65–76. North-Holland, Amsterdam (1986).
21. R. Maes and J. E. M. van Dijk, Criteria and algorithms for contracting decision tables. (Forthcoming.)
22. W. Marek, Completeness and consistency in knowledge base systems. *Proc. 1st Int. Conf. on Expert Database Systems*, pp. 75–82 (1986).
23. T. A. Nguyen, W. A. Perkins, T. J. Laffey and D. Pecora, Checking an expert systems knowledge base for consistency and completeness. *Proceedings IJCAI 1985*, pp. 375–378 (1985).
24. S. L. Pollack, *Analysis of Decision Rules in Decision Tables*. Memo RM-3669-PR, Rand Corporation, Santa Monica (1963).
25. S. Puuronen, An interactive rule-compacting method. *Proc. 2nd Int. Symposium on Knowledge Engineering, Madrid*, 15 pp. (1987).
26. S. Puuronen, A tabular rule-checking method. *Proc. 7th Int. Workshop on Expert Systems and Their Applications, Avignon*, pp. 252–268 (1987).
27. S. Puuronen, Direct execution of an extended decision grid chart. *Angewandte Informatik* **8/9**, 351–357 (1987).
28. J. R. Quinlan, Induction of decision trees. *Machine Learning* **1** (1), 81–106 (1986).
29. J. R. Quinlan, Generating production rules from decision trees. *Proceedings IJCAI 1987*, pp. 304–307 (1987).
30. M. L. Schneider, Weighted decision tables – an alternative solution for ambiguity. *The Computer Journal* **28** (4), 366–371 (1985).
31. J. F. Sowa, *Conceptual Structures*, Addison Wesley, New York, 479 pp. (1984).
32. N. S. Sridharan, Evolving systems of knowledge. *The AI Magazine*, Fall, 108–120 (1985).
33. M. Suwa, A. C. Scott and E. H. Shortliffe, An approach to verifying completeness and consistency in a rule-based system. In *Rule-based Expert Systems*, edited B. Buchanan and E. H. Shortliffe, pp. 159–170. Reading, Massachusetts (1984).
34. J. Vanthienen, Automatiseringsaspekten van de specificatie, constructie en manipulatie van beslissingstabellen. *Doctoral thesis*, K. U. Leuven, 378 pp. (1986).