DEPARTEMENT TOEGEPASTE ECONOMISCHE WETENSCHAPPEN

ONDERZOEKSRAPPORT NR 9739

A TOOL-SUPPORTED APPROACH TO INTER-TABULAR VERIFICATION

by

Jan VANTHIENEN

Christophe MUES

Geert WETS

Kristof DELAERE

Katholieke Universiteit Leuven

Naamsestraat 69, B-3000 Leuven

ONDERZOEKSRAPPORT NR 9739

A TOOL-SUPPORTED APPROACH TO INTER-TABULAR VERIFICATION

by

Jan VANTHIENEN

Christophe MUES

Geert WETS

Kristof DELAERE

D/1997/2376/41

A TOOL-SUPPORTED APPROACH TO INTER-TABULAR VERIFICATION

J. VANTHIENEN, C. MUES, G. WETS & K. DELAERE

Katholieke Universiteit Leuven
Department of Applied Economic Sciences
Naamsestraat 69, 3000 Leuven, Belgium
E-mail: {Jan.Vanthienen, Christophe.Mues,
Geert.Wets}@econ.kuleuven.ac.be

Abstract

The use of decision tables to verify KBS has been advocated several times in the V&V literature. However, one of the main drawbacks of those systems is that they fail to detect anomalies which occur over rule chains. In a decision table based context this means that anomalies which occur due to interactions between tables are neglected. These anomalies are called inter-tabular anomalies.

In this paper we investigate an approach that deals with inter-tabular anomalies. One of the prerequisites for the approach was that it could be used by the knowledge engineer during the development of the KBS. This requires that the anomaly check can be performed on-line. As a result, the approach partly uses heuristics where exhaustive checks would be too inefficient. All detection facilities that will be described, have been implemented in a table-based development tool called PROLOGA. The use of this tool will be briefly illustrated. In addition, some experiences in verifying large knowledge bases are discussed.

Keywords

Decision tables, V&V, Modular KBS

1. Introduction

That decision tables can be checked rather easily on anomalies has always been a reason to propagate the use of decision tables. Along the years, these properties have been mentioned by several authors in their work about decision tables, among them [4, 11]. More recently, the use of decision tables to verify KBS has been advocated [1, 3, 6, 7, 8]. Further developments advocate the integration of decision tables in the modeling environment to reduce V&V problems [9].

The most important drawback most of the proposed systems suffer from is that they fail to detect anomalies over inference chains. In a table-based context, this means that these systems fail to detect anomalies that may arise from inter-tabular interactions. Therefore, in this paper it will be investigated how such anomalies can be detected. To this end, we will look at the PROLOGA system [10], which has been developed at K.U.Leuven. One of the cornerstones of the PROLOGA system is that the anomaly detection component should be integrated in the modeling phase of the KBS, thus aiding the process of knowledge acquisition and representation, and preventing errors that would be more expensive to fix at a later time in the development process. In order to realize this kind of incremental verification, PROLOGA :

- has a strong emphasis on knowledge modularization (e.g. also in [12]);
- operates as much as possible on the decision table format as is, without having to retranslate it into some other operational form, such as Petri nets, first order logic (e.g. [2, 13] among many others);
- partly uses heuristics where exhaustive checks would be too inefficient for on-line use.

The organization of this paper is as follows. First, decision tables and table-based systems are defined. Second, a classification of possible anomalies in table-based systems is described. Subsequently, the problem of inter-tabular verification is tackled. In that section, the different types of inter-tabular anomalies that can occur, are described and illustrated. Furthermore, a practical checking procedure for each of these anomalies is explained. Next, it is illustrated how the explained checks are implemented into the PROLOGA system, and some experiences in verifying large knowledge bases are discussed. Finally, some concluding remarks and directions for further research are given.

2. Decision Tables

In this section, the decision table formalism is defined, and it is discussed how modular knowledge bases can be built using different (sub)tables being linked to each other.

2.1 Decision Tables ; definitions and concepts

A decision table can be defined as follows :

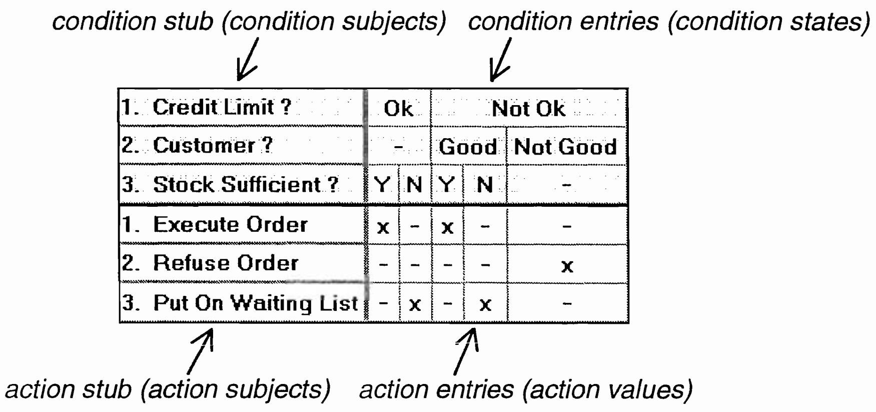
- CS = $\{CS_i\}$ (i=1..cnum) is the set of condition subjects ;
- CD = $\{CD_i\}$ (i=1..cnum) is the set of condition domains,
  with CD_i : the domain of condition subject i, i.e. the set of all possible values of condition subject CS_i ;
- CT = $\{CT_i\}$ (i=1..cnum) is the set of condition state sets,
  with $CT_i = \{S_{ik}\}$ (k=1..n_i) : an ordered set of n_i condition states S_{ik}. Each condition state S_{ik} is a logical expression concerning the elements of CD_i, that determines a subset of CD_i, such that the set of all these subsets constitutes a partition of CD_i;
- AS = $\{AS_j\}$ (j=1..anum) is the set of action subjects ;
- AV = $\{AV_j\}$ (j=1..anum) is the set of action subject value sets,
  with AV_j = {true (x), false (-), null (.)} : the set of all possible values of action subject AS_i , which is, in first instance, null for every action subject.

The decision table DT is a function from the Cartesian product of the condition states to the Cartesian product of the action values, by which every condition combination is mapped into one (completeness criterion) and only one (exclusivity criterion) action configuration :

$$DT: CT_1 \times CT_2 \times \ldots \times CT_{cnum} \rightarrow AV_1 \times AV_2 \times \ldots \times AV_{anum}$$

Note that in this definition, the decision table concept is deliberately restricted to the so-called ‘single-hit’ table, where columns are mutually exclusive. Only this type of table allows for a more easy verification and optimally supports modeling.

A decision table is represented graphically as a table which is split both horizontally and vertically, resulting in four quadrants (cf. Figure 1). The horizontal line divides the table in a condition part (above) and an action part (below). The vertical line divides subjects and entries in the stub (left) and the entry part (right).


condition stub (condition subjects)
condition entries (condition states)
action stub (action subjects)
action entries (action values)

1. Credit Limit ?	Ok		Not Ok		
2. Customer ?	-		Good		Not Good
3. Stock Sufficient ?	Y	N	Y	N	-
1. Execute Order	x	-	x	-	-
2. Refuse Order	-	-	-	-	x
3. Put On Waiting List	-	x	-	x	-

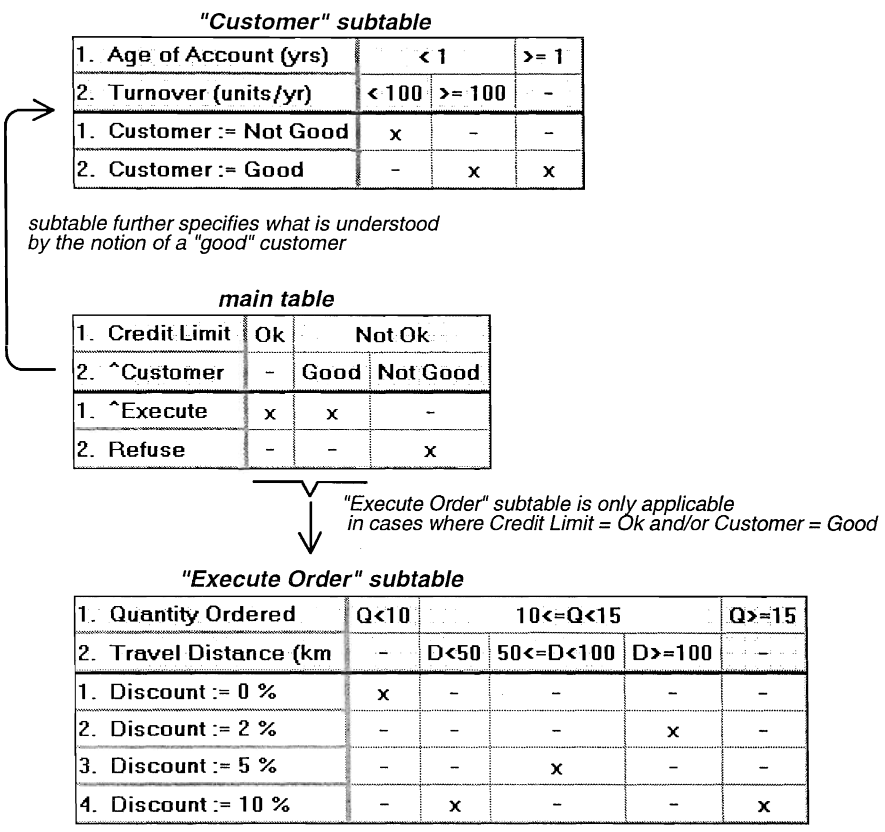
Figure 1 : Representation of a decision table

If each column only contains simple states (no contractions or irrelevant conditions, which are indicated by a " - " entry), the table is called an *expanded* decision table *(canonical form)*, in the other case, the table is called a *contracted* decision table *(consolidated form)*. Figure 1 shows an example of a contracted decision table. Notice that, for ease of legibility, the columns are ordered such that the states of the lower condition subjects vary first. Human decision makers can easily read such tree-structured tables in a top-down fashion by continuing to choose from the relevant condition states until a specific column is reached.

2.2 Multiple-table systems ; condition and action subtables

In order to modularize the tabular knowledge base, one should be able to work with a set of interrelated tabular representations. To this end, a subtable can be linked to the condition part (condition subtable) or to the action part (action subtable) of a particular table, cf. Figure 2. Note that link conditions and actions start with '^'.

Each (sub)table can again be linked to other condition or action subtables, such that more complex subtable structures can be formed.


"Customer" subtable
subtable further specifies what is understood by the notion of a "good" customer
main table
"Execute Order" subtable is only applicable in cases where Credit Limit = Ok and/or Customer = Good
"Execute Order" subtable

"Customer" subtable

1. Age of Account (yrs)	< 1		>= 1
2. Turnover (units/yr)	< 100	>= 100	-
1. Customer := Not Good	x	-	-
2. Customer := Good	-	x	x

main table

1. Credit Limit	Ok	Not Ok	
2. ^Customer	-	Good	Not Good
1. ^Execute	x	x	-
2. Refuse	-	-	x

"Execute Order" subtable

1. Quantity Ordered	Q<10	10<=Q<15			Q>=15
2. Travel Distance (km	-	D<50	50<=D<100	D>=100	-
1. Discount := 0 %	x	-	-	-	-
2. Discount := 2 %	-	-	-	x	-
3. Discount := 5 %	-	-	x	-	-
4. Discount := 10 %	-	x	-	-	x

Figure 2 : Multiple-table systems ; example

3. Classification of anomalies in a table-based system

In [9], starting from the classification by Preece and Shingal [5], it was investigated how these kinds of anomalies can be detected in a table-based system. In the classification proposed in this paper, a distinction has been made between intra-tabular anomalies (which occur in a single decision table) and inter-tabular anomalies (which originate from interactions between several decision tables). Nearly the same classification will be used in this paper. Some additions have been made. In the inter-tabular part of the original classification, anomaly categories that are concerned with the detection of redundancy and ambivalence over inference chains, have been made explicit. These types of anomalies are important, because most table-based anomaly detection systems cannot detect them. Furthermore, inter-tabular checking is partly tackled differently than in [9]. In the next section, anomalies that can occur in tabular systems will be investigated in detail.

4. Inter-tabular verification

Anomaly detection systems based on decision tables or related formalisms, mainly suffer from the drawback that they fail to find anomalies in the rule base that occur over inference chains. Most of those systems can only detect anomalies between simple pairs of rules. Also most of them do not check for circularity. Therefore, in order to overcome these limitations, a different perspective has been taken in developing PROLOGA, as will be explained next.

To check a system of decision tables for anomalies, it is not sufficient to check each of the decision tables separately. It should also be checked, whether there exist inter-tabular anomalies. These anomalies can typically be detected by comparing different parts of the system against each other, because these anomalies result from the interactions between two or more tables. In Figure 3, a classification of inter-tabular anomalies is depicted.

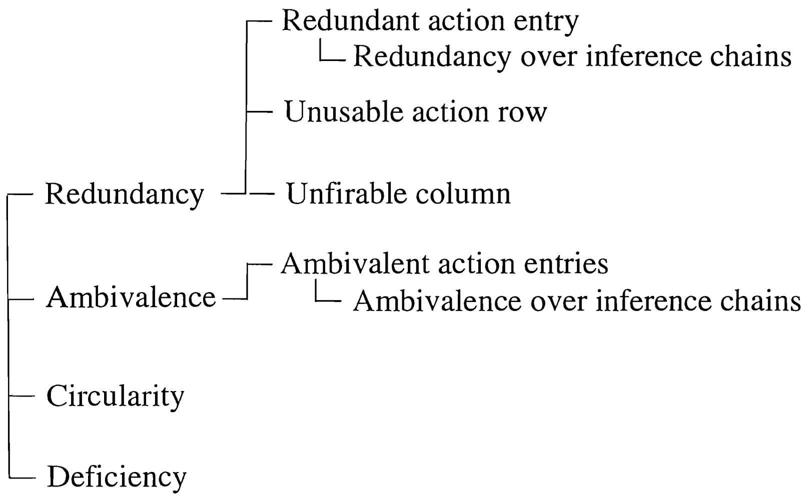


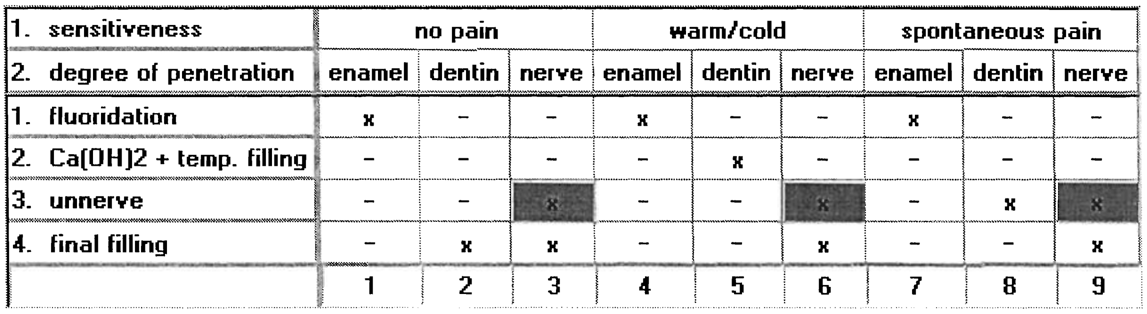
Figure 3: Classification of inter-tabular anomalies

In this section, we will make the assumption that an intra-tabular check has already preceded the inter-tabular check, thus each individual decision table is free of anomalies. For a thorough discussion on intra-tabular anomalies, we refer to [7]. Next, we will investigate, which inter-tabular anomalies can occur.

4.1 Redundancy

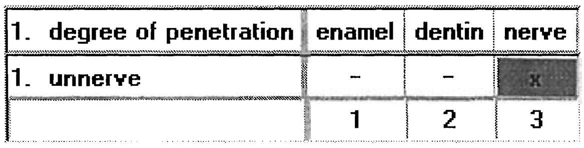
A first type of inter-tabular redundancy is *redundant action entries.* Redundant entries can exist, if some of the actions that are specified, occur in more than one action stub. An example of such a type of inter-tabular redundancy is given in Figure 4.

Table 'caries lesion'



1. sensitiveness	no pain			warm/cold			spontaneous pain		
2. degree of penetration	enamel	dentin	nerve	enamel	dentin	nerve	enamel	dentin	nerve
1. fluoridation	x	-	-	x	-	-	x	-	-
2. Ca(OH)2 + temp. filling	-	-	-	-	x	-	-	-	-
3. unnerve	-	-	x	-	-	x	-	x	x
4. final filling	-	x	x	-	-	x	-	-	x
	1	2	3	4	5	6	7	8	9

Table 'unnerve'



1. degree of penetration	enamel	dentin	nerve
1. unnerve	-	-	x
	1	2	3

Figure 4 : Redundant action entries

In this example system of two tables, it can be seen that there are redundant action entries, e.g. the entries for 'unnerve' in columns 3, 6 and 9 of the table 'caries lesion' are redundant given the entry for 'unnerve' in column 3 of the table 'unnerve', and vice versa.

PROLOGA's checking mechanism deals with redundant action entries on a two-fold level, that is, the verification facilities of the system include a fast heuristic test, as well as a complete test.

First of all, to detect the anomaly in this simple example, the system has to check, for the tables involved, whether one column would specify the same value for an action that also appears in another column, while one of these columns applies to a set of input environments that is a subset of the set associated with the other column. Remark, that the system thus has to check for subsumption between the condition parts of two columns in different tables.

This kind of pairwise check is of course only a heuristic procedure that will not reveal all redundant action entries, neither will it guarantee that all detected cases do in fact refer to redundant knowledge. On the one hand, suppose that the tables in Figure 4 are actually part of a larger hierarchy of tables, in such a way that they are to be applied to different cases. In this case, the detected subsumption does not refer to redundancy from a dynamic perspective, since the columns involved will fire on different occasions. On the other hand, redundant action entries can be caused not only by subsumption between columns of different tables, they can also be the tabular occurrence of what is called in rule-based systems *redundancy over inference chains*. In Figure 5, an example of this type of redundancy is depicted. For example, the decision rules which are used for column 5 of the head table and column 2 of the action subtable are :

- (sensitiveness=warm/cold) and (degree of penetration=dentin)
  $\rightarrow$ (not unnerve) $\wedge$ (after 6 weeks=Y) $\wedge$ ...
- (after 6 weeks=Y) $\wedge$ (sensitiveness after 6 weeks=no pain)
  and (degree of penetration =dentin) $\rightarrow$ (not unnerve) $\wedge$ ...

Condition subtable 'after 6 weeks'

1. sensitiveness	no pain			warm/cold			spontaneous pain	
2. degree of penetration	enamel	dentin	nerve	enamel	dentin	nerve	enamel	dentin or nerve
1. fluoridation	x	-	-	x	-	-	x	-
2. Ca(OH)2 + temp. filling	-	-	-	-	x	-	-	-
3. unnerve	-	-	x	-	▨ -	x	-	x
4. final filling	-	x	x	-	-	x	-	x
5. after 6 weeks:=Y	-	-	-	-	x	-	-	-
6. after 6 weeks:=N	x	x	x	x	-	x	x	x
	1	2	3	4	5	6	7	8

Head table

1. ^after 6 weeks	Y									N
2. sensitiveness after 6 weeks	no pain			warm/cold			spontaneous pain			-
3. degree of penetration	enamel	dentin	nerve	enamel	dentin	nerve	enamel	dentin	nerve	-
1. unnerve	-	▨	-	-	▨	x	-	x	x	.
2. final filling	-	x	x	x	x	x	x	x	x	.
	1	2	3	4	5	6	7	8	9	10

Figure 5: Example of redundancy over chains of rules in tabular systems

In this example, the head table has two redundant entries for the action 'unnerve' in columns 2 and 5, because these entries apply to subsets of cases where the condition 'after 6 weeks' = Y, and degree of penetration = 'dentin', while the condition subtable shows that 'after 6 weeks' can only take the value 'Y' if degree of penetration = 'dentin', and, at the same time, already specifies the same entry for 'unnerve' for these cases. In this example, the redundant action entry anomaly would not be detected by solely looking for subsumed columns across tables.

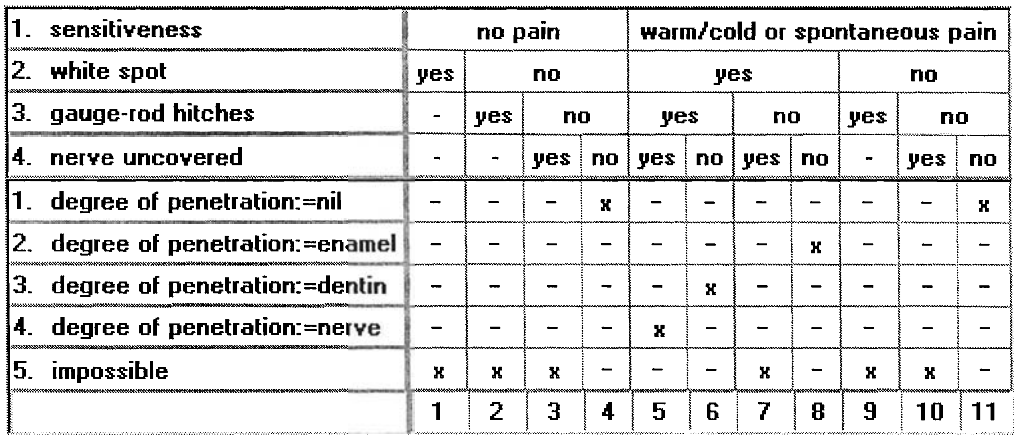
Consequently, to make sure that the detected anomaly is in fact a real case of redundancy from a dynamic perspective, and to also detect specific cases of redundancy across inference chains, the system has to perform a complete check and look for redundancy between multiple columns, and specifically take into account the (inference) paths between them. In other words, PROLOGA has to follow the inference paths through the table structure, and check whether it is possible for at least one combination that the examined action receives the same truth value in more than one table. Note that, as was explained before, earlier tabular systems for the verification of rule-based systems were not equipped to detect this kind of redundancy.

A complete check for redundant action entries has been proposed in [9], but is subject to the same computational limitations as the rule extension checks proposed in rule-based systems verification, since the system has to trace all possible paths, that may lead to redundancy. However, it should be noted that, as pointed out in [5], inference paths in rule-based systems are often relatively short, and that paths in tabular systems can typically be expected to be even shorter than in rule-based systems, because more decision logic is packed into one node. This would seem to indicate that, in practice, complete checks might be feasible for some systems (cf. infra).

The presence of redundant action entries in the knowledge base usually does not indicate that there are errors in the represented knowledge, but can increase maintenance problems, i.e. when changing parts of the knowledge base. In the given example, the detected anomaly should help the knowledge engineer realize that the two occurrences of 'unnerve' should actually be different actions, that is, 'unnerve immediately' and 'unnerve after six weeks'.

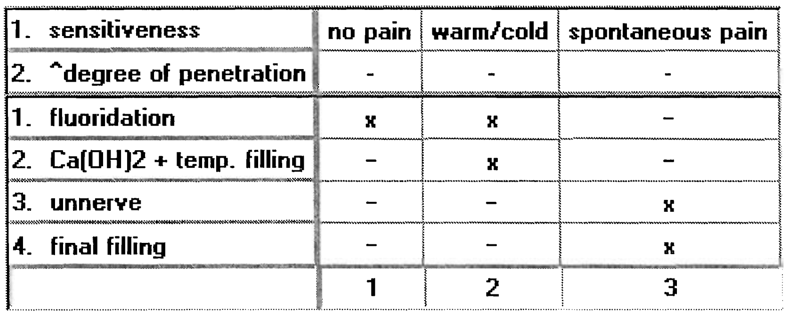
A second type of inter-tabular redundancy is an *unusable action row*. In condition subtable structures, this anomaly can occur, either if an action is not part of a goal state, nor is used in the condition stub of an other table, or if the linking condition in a condition subtable link is irrelevant. As a consequence, the actions in the condition subtable will become unusable. Notice that by only removing the intra-tabular anomaly (irrelevant condition subject) in the head table, the inter-tabular anomaly (unusable action row) in the condition subtable remains. In Figure 6, an example of this type of inter-tabular redundancy is depicted.

Condition subtable 'degree of penetration'



1. sensitiveness	no pain				warm/cold or spontaneous pain						
2. white spot	yes	no			yes				no		
3. gauge-rod hitches	-	yes	no		yes		no		yes	no	
4. nerve uncovered	-	-	yes	no	yes	no	yes	no	-	yes	no
1. degree of penetration:=nil	-	-	-	x	-	-	-	-	-	-	x
2. degree of penetration:=enamel	-	-	-	-	-	-	-	x	-	-	-
3. degree of penetration:=dentin	-	-	-	-	-	x	-	-	-	-	-
4. degree of penetration:=nerve	-	-	-	-	x	-	-	-	-	-	-
5. impossible	x	x	x	-	-	-	x	-	x	x	-
	1	2	3	4	5	6	7	8	9	10	11

Head table 'caries lesion'



1. sensitiveness	no pain	warm/cold	spontaneous pain
2. ^degree of penetration	-	-	-
1. fluoridation	x	x	-
2. Ca(OH)2 + temp. filling	-	x	-
3. unnerve	-	-	x
4. final filling	-	-	x
	1	2	3

Figure 6: Unusable action rows

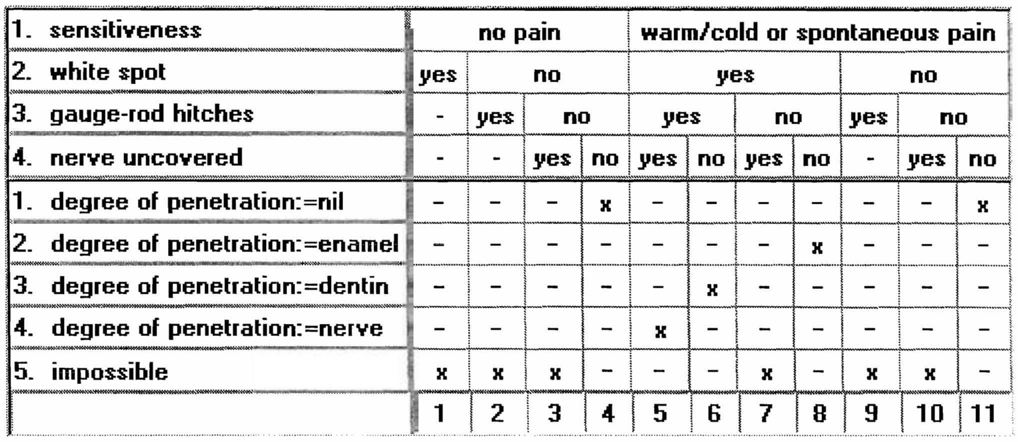
Because, in the example, condition degree of penetration in the head table is irrelevant, all the action rows in the condition subtable are unusable. Note that this type of anomaly can also occur with action subtable links, if the link action either refers to a non-existing table or never specifies the action subtable to be applicable. In the latter case, the whole action subtable can become redundant.

Note that we consider unused action row anomalies as a form of redundancy. One could also argue that it could be an indication of incompleteness, since the decision logic necessary to arrive at the unusable part might be missing.

In order to detect unusable action rows, the system has to check, whether in an condition (action) subtable structure, there is a linking condition (action) in the head table, that is irrelevant (has no x-entries). Then, if this condition (action) subtable cannot be reached from within an other table, there are unusable actions in the condition (action) subtable.

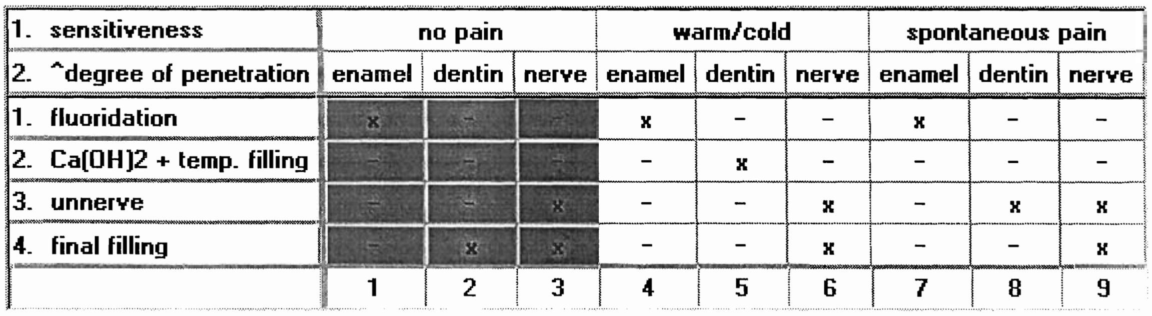
A third type of inter-tabular redundancy that may occur is that of an *unfirable column* anomaly. This type of anomaly may occur in a condition subtable structure or an action subtable structure. It typically occurs when the same (combination of) condition(s) is used in two or more tables. In Figure 7, an example of an unfirable column in a condition subtable structure is depicted.

Condition subtable 'degree of penetration'



1. sensitiveness	no pain				warm/cold or spontaneous pain						
2. white spot	yes	no			yes				no		
3. gauge-rod hitches	-	yes	no		yes		no		yes	no	
4. nerve uncovered	-	-	yes	no	yes	no	yes	no	-	yes	no
1. degree of penetration:=nil	-	-	-	x	-	-	-	-	-	-	x
2. degree of penetration:=enamel	-	-	-	-	-	-	-	x	-	-	-
3. degree of penetration:=dentin	-	-	-	-	-	x	-	-	-	-	-
4. degree of penetration:=nerve	-	-	-	-	x	-	-	-	-	-	-
5. impossible	x	x	x	-	-	-	x	-	x	x	-
	1	2	3	4	5	6	7	8	9	10	11

Head table 'caries lesion'

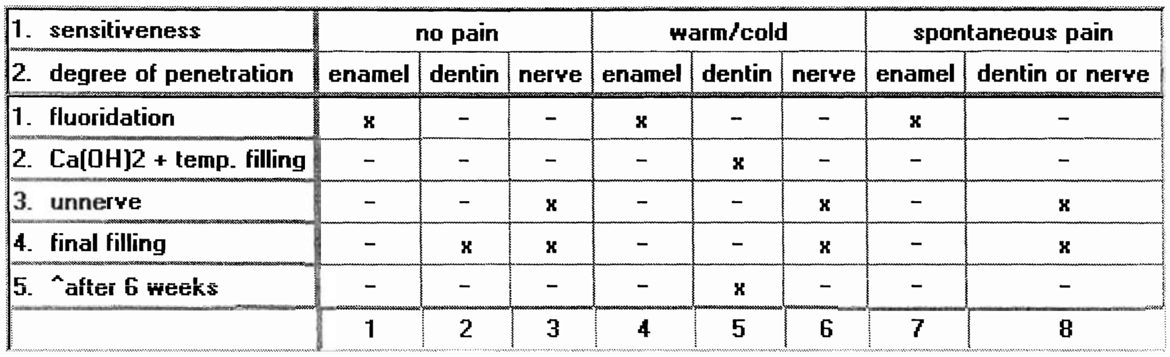


1. sensitiveness	no pain			warm/cold			spontaneous pain		
2. ^degree of penetration	enamel	dentin	nerve	enamel	dentin	nerve	enamel	dentin	nerve
1. fluoridation	x	-	-	x	-	-	x	-	-
2. Ca(OH)2 + temp. filling	-	-	-	-	x	-	-	-	-
3. unnerve	-	-	x	-	-	x	-	x	x
4. final filling	-	x	x	-	-	x	-	-	x
	1	2	3	4	5	6	7	8	9

Figure 7: Unfirable column anomaly in a condition subtable structure

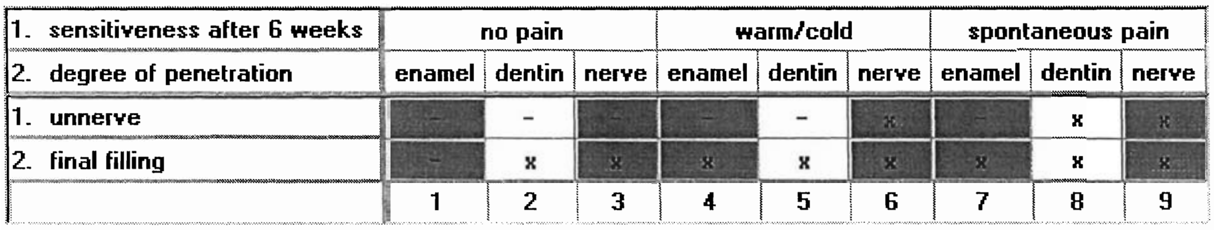
In this example, columns 1, 2 and 3 of the head table are unfirable, since the condition 'degree of penetration' does not obtain a value from the condition subtable if the state of condition sensitiveness is 'no pain'. In the same way, an action subtable structure can occur in which some columns are unfirable. Such an example is given in Figure 8. In the action subtable, the columns 1, 3, 4, 6, 7 and 9 are unfirable, because this subtable will never be applied if the condition 'degree of penetration' gets the values 'enamel' or 'nerve'.

Head table ‘caries lesion’



1. sensitiveness	no pain			warm/cold			spontaneous pain	
2. degree of penetration	enamel	dentin	nerve	enamel	dentin	nerve	enamel	dentin or nerve
1. fluoridation	x	-	-	x	-	-	x	-
2. Ca(OH)2 + temp. filling	-	-	-	-	x	-	-	-
3. unnerve	-	-	x	-	-	x	-	x
4. final filling	-	x	x	-	-	x	-	x
5. ^after 6 weeks	-	-	-	-	x	-	-	-
	1	2	3	4	5	6	7	8

Action subtable ‘after 6 weeks’



1. sensitiveness after 6 weeks	no pain			warm/cold			spontaneous pain		
2. degree of penetration	enamel	dentin	nerve	enamel	dentin	nerve	enamel	dentin	nerve
1. unnerve	-	-	-	-	-	x	-	x	x
2. final filling	-	x	x	x	x	x	x	x	x
	1	2	3	4	5	6	7	8	9

Figure 8: Unfirable column anomaly in an action table structure

In order to detect unfirable columns, the PROLOGA system will basically adopt the following strategy. In order to check a given table against its condition subtables, check for each (combination of) linking condition(s) in this head table, whether there is a possible state (combination), which does not get a value of the corresponding action(s) in the condition subtable(s). If this is the case, columns in which these states occur in the head table, will be unfirable. Furthermore, if the table being examined is an action subtable of one or more other tables, the system has to walk through its ancestors in the action subtable hierarchy, and check for each (combination of) condition(s) in these tables, that also occurs in the condition stub of the table being examined, whether the table can be reached at least from one of the paths that lead to it.

Unfirability may be an indication of action entries being erroneously set to false in the table that is linked to the table that contains the unfirable columns, or may suggest that the action entries in the unfirable columns should be left unspecified as they do not apply to possible input combinations.

4.2 Ambivalence

Ambivalent action entries can occur in tabular systems, if some actions are specified in more than one action stub. Figure 4, that was used in the previous section on redundancy, also contains an example of ambivalent entries. There, column 8 in the table 'caries lesion' specifies that if degree of penetration = 'dentin' and sensitiveness = 'spontaneous pain', then unnerve = 'x'. However, column 2 in the table 'unnerve' specifies that if degree of penetration = 'dentin', regardless of sensitiveness, action unnerve = '-'. This result is clearly ambivalent.

One particular subtype of ambivalence that might occur in tabular systems, is *ambivalence over inference chains*. Again, this type of anomaly is quite similar to its counterpart, redundancy over inference chains. An example of the occurrence of ambivalence over inference chains can be found in Figure 5. One case of ambivalence that can be found there is as follows. If sensitiveness takes the value 'warm/cold' and degree of penetration takes the value 'dentin', the condition subtable specifies 'unnerve' as false, and at the same time, specifies that 'after 6 weeks' = 'Y'. In the head table however, in column 8, the action 'unnerve' is specified true for a subset of cases where degree of penetration = 'dentin', this subset overlapping with the earlier set of cases. Therefore, these are ambivalent entries.

The tests used here are similar to the ones used for checking redundant action entries. The checking can again be done in a complete manner, or heuristically.

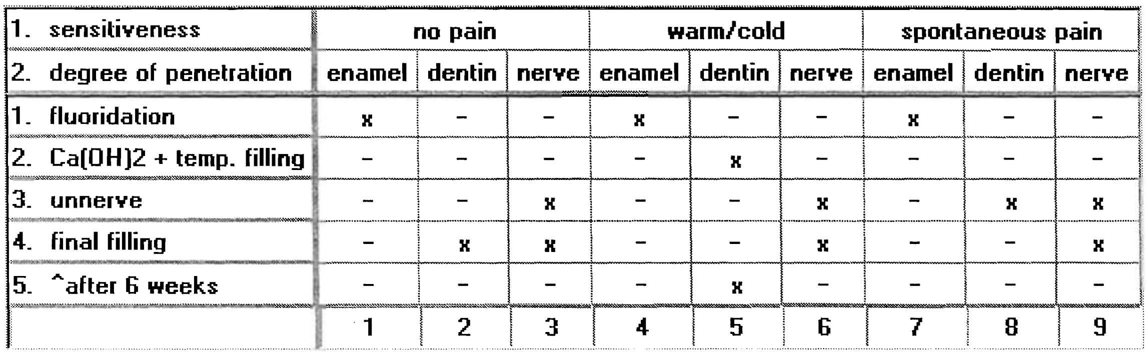
4.3 Deficiency

If we define deficiency as in [5], inter-tabular deficiency would refer to a situation where the system as a whole would not be able to infer anything for a particular set of input environments, whereas each decision table separately would show no deficiency. However, if each decision table satisfies the completeness criterion (condition part) and is non-deficient in its own light (action part), then there cannot be any inter-tabular deficiency. In other words, the problem of ensuring completeness is effectively delegated to each separate table.

4.4 Circularity

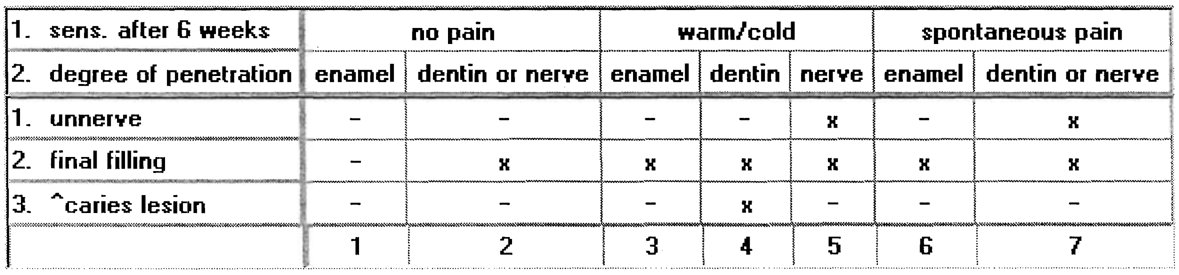
Whereas intra-tabular circularity is not possible because of the strict distinction within one decision table between conditions and actions, inter-tabular circularity is an anomaly the developer should be aware of. In Figure 9, an example of inter-tabular circularity is given.

Action subtable 'caries lesion'



1. sensitiveness	no pain			warm/cold			spontaneous pain		
2. degree of penetration	enamel	dentin	nerve	enamel	dentin	nerve	enamel	dentin	nerve
1. fluoridation	x	-	-	x	-	-	x	-	-
2. Ca(OH)2 + temp. filling	-	-	-	-	x	-	-	-	-
3. unnerve	-	-	x	-	-	x	-	x	x
4. final filling	-	x	x	-	-	x	-	-	x
5. ^after 6 weeks	-	-	-	-	x	-	-	-	-
	1	2	3	4	5	6	7	8	9

Action subtable 'after 6 weeks'



1. sens. after 6 weeks	no pain		warm/cold			spontaneous pain	
2. degree of penetration	enamel	dentin or nerve	enamel	dentin	nerve	enamel	dentin or nerve
1. unnerve	-	-	-	-	x	-	x
2. final filling	-	x	x	x	x	x	x
3. ^caries lesion	-	-	-	x	-	-	-
	1	2	3	4	5	6	7

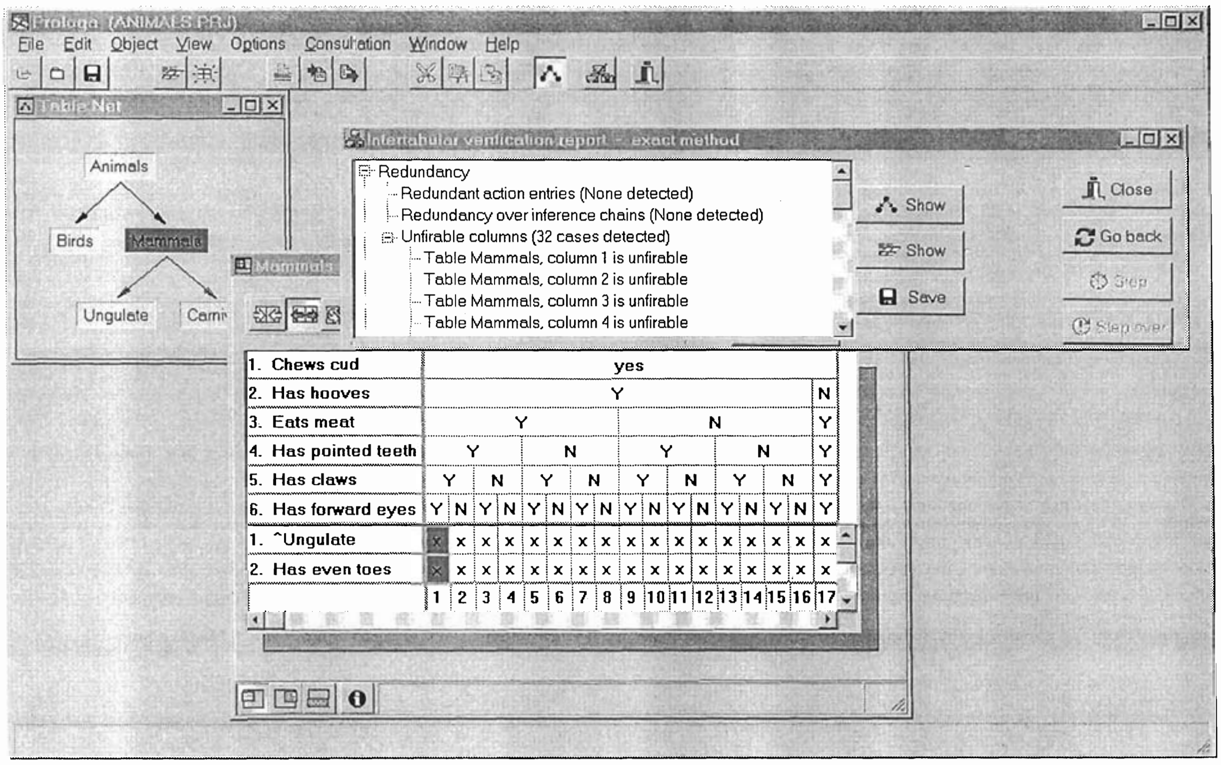
Figure 9: Inter-tabular circularity

In this example, inter-tabular circularity is present for input environments where sensitiveness = 'warm/cold', sensitiveness after 6 weeks = 'warm/cold' and degree of penetration = 'dentin'.

PROLOGA adopts a preventive strategy with respect to inter-tabular circularity. From all condition and action subjects in the system, PROLOGA extracts an inter-tabular dependency graph. As soon as cycles are detected in the updated graph, the developer is warned about it. Since, in a tabular context, cycles in subtable structures are generally undesirable.

5. Implementation within PROLOGA

PROLOGA's verification engine is fully integrated within the development environment itself. It consists of an intra-tabular and an inter-tabular checking module. The intra-tabular verification component signals anomalies, as soon as they appear. The inter-tabular checks are performed on demand. The user can select detected cases from the verification report, such that PROLOGA automatically displays and highlights the anomalous decision logic (cf. Figure 10).



- Redundancy
  - Redundant action entries (None detected)
  - Redundancy over inference chains (None detected)
  - Unfirable columns (32 cases detected)
    - Table Mammals, column 1 is unfirable
    - Table Mammals, column 2 is unfirable
    - Table Mammals, column 3 is unfirable
    - Table Mammals, column 4 is unfirable

1. Chews cud	yes																
2. Has hooves	Y																N
3. Eats meat	Y								N								Y
4. Has pointed teeth	Y				N				Y				N				Y
5. Has claws	Y		N		Y		N		Y		N		Y		N		Y
6. Has forward eyes	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y
1. ^Ungulate	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
2. Has even toes	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Figure 10: PROLOGA screen example

6. PROLOGA's V&V component in practice

We have used decision tables in a large number of applications and environments. Two large systems, in which PROLOGA has been a key factor during development, are :

- HANDIPAK : a KBS that contains legal information on financial benefits for the disabled in Belgium, and uses it to support first line social workers with the introduction of applications for benefits. The analysis of the proposed regulation by means of the decision table technique enabled the authors to eliminate a considerable number of ambiguities in the bill before it was published. The legislation has been formally specified into a structure of 45 tables.

- VLAREM : a KBS that contains a subset of the environmental legislation on permits, etc., and that is designed to provide users with information and advice on this matter. The knowledge is modeled into 61 interrelated tables.

At the time that these systems were developed, PROLOGA only supported intra-tabular checking ; the checks proposed in this paper were not yet available. We have recently carried out the newly implemented procedures on both systems. Using the heuristic approach, the HANDIPAK knowledge base was checked for inter-tabular anomalies in about 45 seconds (for the test, a Pentium-200Mhz was used). For the VLAREM legislation, the check took about 15 minutes. The complete checks, on the other hand, took several hours to complete. Of course, only part of the anomalies reported by the heuristic checking procedures were in fact anomalous from a dynamic perspective, that is, if we also take the inference paths through the structure into account. This success factor was considerably lower for redundant or ambivalent action entries than for the other anomaly types.

7. Conclusion and future research

In this paper, a practical approach has been presented that deals effectively with inter-tabular anomalies. The traditional limitations of anomaly checking systems based on decision tables, are solved in PROLOGA. These include: the possibility to check for anomalies over chains of rules and the check for circularity. However, there remain some limitations. The approach presented allows for exhaustive checks, but some of these can be very time-consuming for large systems. Therefore, alternative heuristic tests are also available. It would be interesting to examine in more detail the relationship between the feasibility of an exhaustive detection approach and certain characteristics of the knowledge, such as path lengths, morphology (e.g. trees vs. graphs), the successfulness of the modularization (in terms of conditions and actions concentrated in one table vs. being spread across several tables), etc. Another limitation is that the intra-tabular check for missing rules remains very computer intensive. This should create an incentive for the developer to modularize the decision tables if possible. Apart from being computationally inefficient, large tables also are undesirable because of their reduced visualization qualities.

8. References

[1] Cragun B. and Steudel H., A decision-table based processor for checking completeness and consistency in rule-based expert systems, *Man-Machine Studies 26*, 1987, 633-648.

[2] Larsen H. and Nonfjall H., Modeling in the Design of a KBS Validation System, *Intl. Journal of Intelligent Systems* 6, 1991, 759-775.

[3] Maes R. and Van Dijk J. E. M., On the role of ambiguity and incompleteness in the design of decision tables and rule-based systems, *The Computer Journal 31*, 1988, 481-489.

[4] Metzner J. and Barnes B., *Decision table languages and systems*, Academic Press Inc., New York, 1977.

[5] Preece A. and Shinghal R., Foundation and application of knowledge base verification, *Intelligent Systems 9*, 1994, 683-701.

[6] Puuronen S., A tabular rule checking method, *Proc. Avignon87, Vol. 1*, 1987, 257-268.

[7] Suwa M., Scott A. and Shortliffe E., An approach to verifying completeness and consistency in a rule-based expert system, *AI Magazine 3*, 1982, 16-21.

[8] Vanthienen J., *Automatiseringsaspecten van de specificatie, constructie en manipulatie van beslissingstabellen*, PhD dissertation, K.U.Leuven, Dept. Applied Economics, 1986.

[9] Vanthienen J., Aerts A., Mues C. and Wets G., A modelling approach to KBS verification, *Proc. of the European Symposium on the Validation and Verification of Knowledge Based Systems*, 1995, 155-171.

[10] Vanthienen J. and Dries E., Illustration of a Decision Table Tool for specifying and implementing Knowledge Based Systems, *Proc. of The Fifth Int. Conf. on Tools with Artificial*, 1993, 198-205.

[11] Verhelst M., *De praktijk van beslissingstabellen*, Kluwer, Deventer/Antwerpen, 1980.

[12] Wendler B. and Ayel M., Verifying Dynamic Coherence in Modular Knowledge Bases, *Proc. of the European Symposium on the Validation and Verification of Knowledge Based Systems*, 1995, 173-187.

[13] Zhang D. and Nguyen D., A Tool for Knowledge Base Verification, *IEEE Transactions on Knowledge and Data Engineering 6*, 1994, 983-989.

[14] Robben F., *HANDIPAK: computeradviessysteem m.b.t. de financiële tegemoetkomingen aan gehandicapten*, in VAN BUGGENHOUT, B., ROBBEN, F., LEUS, I., CASTELEYN, H., HERTECANT, G. en DEMEESTER, W., Het nieuw gehandicaptenrecht. Commentaar bij de nieuwe wetgeving en recente evoluties in het beleid, Recht en Sociale Hulpverlening, Brugge, Die Keure, 1988, 21-26.