

An Illustration of Verification and Validation in the Modelling Phase of KBS Development

J. Vanthienen, C. Mues, A. Aerts

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

Abstract

Reliability has become a key factor in KBS development. For this reason, it has been suggested that verification and validation (V&V) should become an integrated part of activities throughout the whole KBS development cycle. In this paper, it will be illustrated how the PROLOGA workbench integrates V&V aspects into its modelling environment, such that these techniques can be of assistance in the process of knowledge acquisition and representation. To this end, verification has to be performed incrementally, and can no longer be delayed until after the system has been completed. It will be shown how this objective can be realised through an approach that uses the decision table formalism as a modelling instrument.

Key Words: Verification & Validation; Decision Tables; KBS Building Tools;
Knowledge Modelling

1. Introduction

V&V has become an important part of the KBS development cycle. The built systems have to be reliable, so the knowledge on which these systems are based, has to be consistent as a whole. In this paper, the verification component of PROLOGA will be illustrated. PROLOGA is a table-based knowledge modelling tool into which a verification component is integrated. As a result of this integration, verification can already be done in the modelling phase instead of be performed afterwards on the implemented system. We believe that performing V&V during an early development phase strongly improves the process of knowledge acquisition and representation, and prevents errors that would be more expensive to fix at a later time.

In our opinion, using the decision table formalism as a modelling instrument offers significant advantages in verification, because its structured nature eliminates, for a large number of anomaly types, the need for a translation into some other operational form, such as Petri nets, first order logic, etc. [2],[3],[4],[5],[16],[17], in order to detect them. The usefulness of this table-based approach, as adopted by PROLOGA, will be demonstrated using an example of Zhang and Nguyen [16].

The paper is structured as follows. Firstly, the initial rule base will be given. Secondly, the decision table formalism, and our tabular modelling approach, will be introduced. Then, a classification of anomaly types will be given. Next, the proposed verification process will be explained and applied to the example. We conclude with a summary of the obtained results, some real-world applications, and, finally, a brief overview of current and future research.

2. The example rule base: Car

In their article, Du Zhang and Doan Nguyen [16], describe a tool, named PREPARE, that is used for knowledge base verification. PREPARE is designed to verify a knowledge base using a predicate/transition net representation. The different anomaly types are defined as patterns of the predicate/transition net model, and are detected through a syntactic pattern recognition method. The rule base that is verified needs to be translated into a Petri net format in order to detect the anomalies.

The purpose of using this example, is not to evaluate the predicate/transition net based tool, but to have a small example knowledge base, not designed by ourselves, that enables us to explore the verification features of our own tool-supported approach. The following figure shows the example rule base, that contains information for classifying cars into three base types, namely Sport_cars, Luxury_cars, and Economical_cars :

Facts:

C1: Price_less_than(HONDA-CIVIC,7000)
C2: Price_greater_than(CORVETTE,7000)
C3: Price_greater_than
(CARDILLAC_SERVILLE,7000)
C4: Air_cond(CARDILLAC_SERVILLE)
C5: Air_cond(CORVETTE)
C6: Power_window(CARDILLAC_SERVILLE)
C7: Power_window(CORVETTE)
C8: Sun_roof(CARDILLAC_SERVILLE)
C9: Sun_roof(CORVETTE)
C10: Cassette(CARDILLAC_SERVILLE)
C11: Cassette(CORVETTE)
C12: Color_choices(HONDA-CIVIC)
C13: Color_choices(CARDILLAC_SERVILLE)
C14: Color_choices(CORVETTE)
C15: Aerodynamic(CORVETTE)
C16: Aerodynamic(HONDA-CIVIC)
C17: Spacious(CARDILLAC_SERVILLE)
C18: Spacious(HONDA-CIVIC)
C19: Elegant(CARDILLAC_SERVILLE)
C20: Elegant(CORVETTE)
C21: High_mileage(HONDA-CIVIC)
C22: Speed(CORVETTE)

Rules:

C23: Body_shape(x) <--- Aerodynamic(x)
C24: Look(x) <--- Color_choices(x)
C25: Extra_opt(x) <--- Luxury_car(x)
C26: Low_cost(x) <--- Price_less_than(x,7000)
C27: High_cost(x) <--- Price_greater_than(x,7000)
C28: High_cost(x) <--- Spacious(x)
C29: Extra_opt(x) <--- Air_cond(x), Power_window(x),
Sun_roof(x), Cassette(x)
C30: Look(x) <--- Elegant(x), Body_shape(x), Color_choices(x)
C31: Comfortable(x) <--- Spacious(x), Extra_Opt(x)
C32: Import_tax(x) <--- High_cost(x)
C33: Import_car(x) <--- Import_tax(x)
C34: Export_car(x) <--- Import_tax(x)
C35: Extra_opt(x) <--- Import_car(x)
C36: Extra_opt(x) <--- Export_car(x)
C37: Body_shape(x) <--- Aerodynamic(x)
C38: Luxury-car(x) <--- Look(x), Comfortable(x), High_cost(x)
C39: Economical_car(x) <--- High_mileage(x), Low_cost(x)
C40: Sport_car(x) <--- Look(x), High_cost(x), Speed(x),
Performance(x)
C41: Four_seats(x) <--- Spacious(x)
C42: High_insurance(x) <--- Expensive_car(x)
C43: Car(x) <--- Luxury_car(x)
C44: Car(x) <--- Sport_car(x)
C45: Car(x) <--- Economical_car(x)

3. A tabular modelling approach

In this section, the decision table concept, and the use of multiple-table structures, will be discussed briefly. For a more extensive overview we refer to [10], [11], [12].

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

Figure 1 : Decision table, example

A decision table consists of four parts (cf. Figure 1). *The condition subjects* are the criteria that are relevant to the decision making process. They represent the items about which information is needed to take the right decision. Condition subjects are found in the upper-left part of the table. *The condition states* are logical expressions determining the relevant sets of values for a given condition. Condition states are found in the upper-right part of the table. *The action subjects* describe the possible outputs of the decision making process. They are found in the lower-left part of the table. *The action values* are the possible values a given action can take. They are found in the lower-right part of the table.

Different variations of the decision table concept exist, many of them looking similar at first sight [8]. However, one main difference is whether these variations demand that table columns should be mutually exclusive (single-hit versus multiple-hit). In a single-hit table, each possible combination of condition states can be found in one and only one column. This exclusivity criterion is a key factor in verification, since it prevents most kinds of redundancy and ambivalence. Therefore, PROLOGA tables are all single-hit.

For realistic problems, using only one table to hold all the knowledge will not be sufficient. One should be able to structure the decision logic into a set of inter-related tables. To this end, PROLOGA distinguishes between two kinds of subtables :

- condition subtables, which determine the state of a condition, and
- action subtables, which give further details on a certain action.

The inter-tabular relationships can be visualised by means of a directed graph (cf. Figure 2). Each node of this structure represents a decision table. The arcs correspond to links between parent and child table, pointing away from condition subtables, or towards action subtables.

The process of deciding on how to modularize the knowledge can either be done intuitively, or automatically [15]. The car example can easily be modularised in an intuitive manner. Firstly, we look at the final goal literals, viz. those concerning the different types of cars. These literals will be the actions of the main table. The conditions of the main table can be obtained from the premises of the rules that decide on the type of car. One of these conditions, ‘Look of car’, is also the conclusion of rules C30 and C24. We can then decide to use a condition subtable to represent the knowledge about ‘Look of car’. This process can be repeated, until we finally arrive at the structure depicted in Figure 2. The resulting structure consists of 9 decision tables. Five of these tables, CAR(X), LOOK(X), BODY(X), COST(X) and INSUR(X) will be discussed in the following sections.

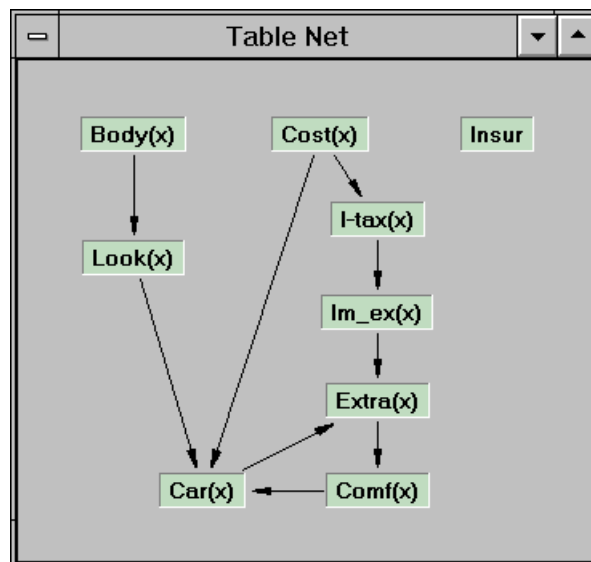


Figure 2 : Decision table structure

4. Classification of table anomalies

Before working out the example with the PROLOGA workbench, we will first indicate in this section what different kinds of anomalies can occur in a table-based system. Because of the differences between table and rule representation formalisms, the corresponding anomalies will indeed take on a different form. When classifying possible table anomalies, one can first distinguish between cases of redundancy, ambivalence, circularity and deficiency [1], [5]. Secondly, one can further break down each of these anomaly types into intra- and inter-tabular subtypes. Whereas intra-tabular anomalies can be fully explained in terms of the components of one single table, inter-tabular anomalies instead arise through the interaction between (components of) different tables. Consequently, both individual tables, and relations between different tables, should be verified. For a more detailed explanation of these anomalies we refer to [8].

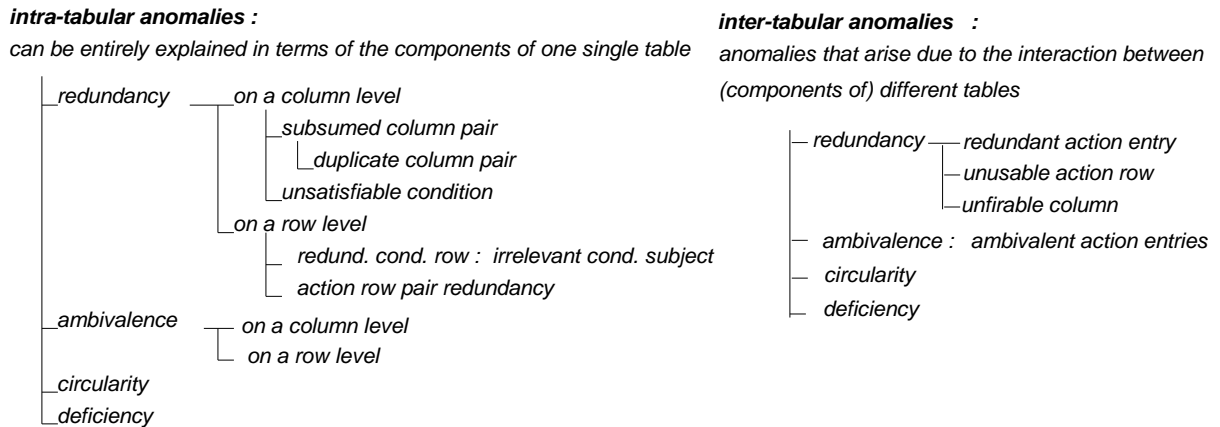


Figure 3 : Classification of table anomalies

5. The verification of tabular knowledge bases; illustration

In this section, the different anomalies that were found in the example knowledge base will be discussed. A first subsection will cover the inter-tabular anomalies, a second one will focus on intra-tabular anomalies.

5.1. Inter-tabular anomaly-types

5.1.1. Anomalies on a table-to-table level

Anomalies that relate to interactions on an aggregate table-to-table level can be detected and visualised by extracting an inter-tabular structure graph from the given specification.

If we take a closer look at the structure graph of our example (cf. Figure 4), we can see that, although we intuitively consider CAR(X) as the main table, an arrow is pointing back to EXTRA(X). Since the argument 'Luxury-car(x)' is referred to in a condition of table EXTRA(X), while in CAR(X) the same argument is assigned a value, CAR(X) is in fact a condition subtable of EXTRA(X). Given that EXTRA(X) is in turn a condition subtable of COMF(X), and COMF(X) is a condition subtable of CAR(X) itself, a *circular chain* is found.

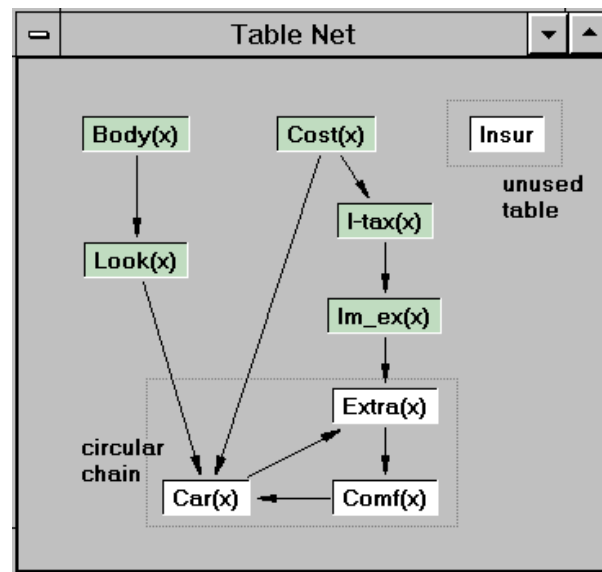


Figure 4 : Dealing with anomalies on a table-to-level

Furthermore, the structure graph also indicates that the knowledge base contains one isolated decision table, viz. INSUR(X), that is not linked to the other tables. Consequently, since its actions are not concerned with deciding on car types, the knowledge represented in this table is of no actual importance in arriving at a possible goal state. This so-called *unused table* anomaly is a kind of inter-tabular redundancy, and might indicate a missing link.

5.1.2. Interactions between components of different tables

When looking for inter-tabular anomalies, one should take into account possible interactions, not only on an aggregate table-to-table level, as in the preceding section, but also between overlapping components of different tables. For example, the sharing of the same conditions or actions by different tables can cause inter-tabular redundancies or ambivalence. However, because in this example, no decision elements are spread across several tables in a manner that could cause this kind of anomalies, this section does not have to cover the actual detection methods in detail. In general, a well-chosen modularisation of the decision problem will in fact strongly reduce the presence of this class of anomalies. For a more detailed discussion on this matter, we refer to [8], [9].

5.2. Intra-tabular anomaly types

This section will illustrate how PROLOGA's verification component deals with intra-tabular anomaly types. It will be shown how the knowledge base is incrementally checked, as it is being built, and how warnings and hints are generated, as soon as the knowledge engineer wants to perform certain actions on the knowledge base that are relevant from a V&V perspective. Only CAR(X), LOOK(X), BODY(X) and COST(X) will be discussed ; the other tables can be handled in a similar way.

5.2.1. CAR(X)

CAR(X) contains knowledge that directly concerns the different types of car, viz. luxury, economical or sport. The original rules that correspond to this knowledge are C38, C39 and C40. If we have a constraint stating that a car can only be of one type, that is, a car is either a luxury, an economical or a sport car, then there is a conflict between the following rules :

- C38: $\text{Luxury_car}(x) \leftarrow \text{Look}(x), \text{Comfortable}(x), \text{High_cost}(x)$
- C40: $\text{Sport_car}(x) \leftarrow \text{Look}(x), \text{High_cost}(x), \text{Speed}(x), \text{Performance}(x)$

PROLOGA will immediately detect the addition of *conflicting rules*. The constraint itself is defined by means of an ‘only’ operator. This operator specifies that for the given condition combinations, only the designated conclusions should be deduced. As soon as the second rule is added, the contradictory rule pair is detected, and a warning is given to the knowledge engineer (cf. Figure 5).

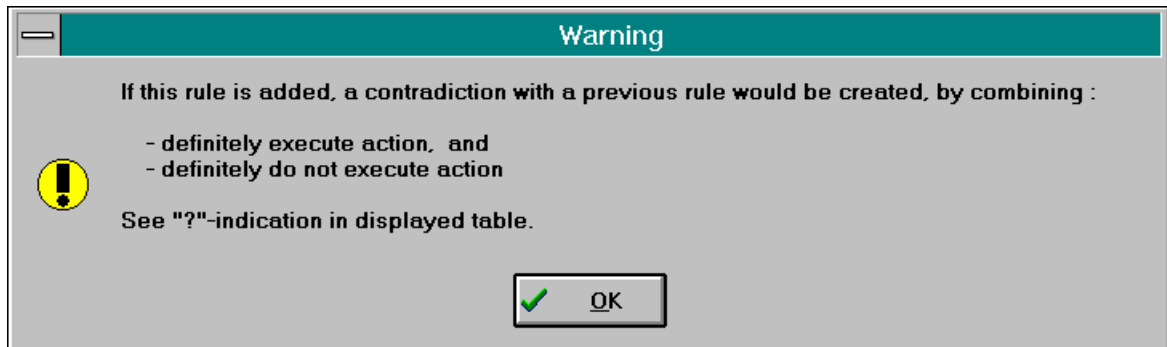


Figure 5 : CAR(X), contradiction message

If we analyse the anomalous table (cf. Figure 6), and have a closer look at those columns that have '?' entries in their action parts, we can easily trace for which subset of cases a contradiction has been found. This clearly illustrates how the decision table format can be an intuitive and powerful visualisation technique. By providing the knowledge engineer with an easier way of communicating with the expert, it can be of substantial aid during knowledge acquisition, and improve the quality of the knowledge base.

1. $\hat{\text{Cost}}(x)$	High				Low				
2. $\hat{\text{Look}}(x)$	Y				N		-		
3. $\hat{\text{Comfort}}(x)$	Y		N		-		-		
4. High-mileage(x)	-		-		-		Y	N	
5. Speed(x)	Y	N	Y	N	-		-		
6. Performance(x)	Y	N	-	Y	N	-		-	
1. Car(x) := Luxury Car	?	x	x	-	.	.	.	-	.
2. Car(x) := Economical Car	-	-	-	-	.	.	.	x	.
3. Car(x) := Sport Car	?	-	-	x	.	.	.	-	.

Figure 6 : CAR(X), contracted table

Further information can be obtained from an intra-tabular verification report. This report can be consulted at any moment in the construction phase of the table. Figure 7 contains the report on table CAR(X). Besides indicating the earlier discussed contradiction, the report also signals that there are 4 empty columns, which means that there are 4 different subsets of cases in which no decisions about the action subjects can be made. For example, if Cost(x) is low and High-mileage(x) is false (cf. last column), then no knowledge can be deduced, so the knowledge base is *incomplete*.

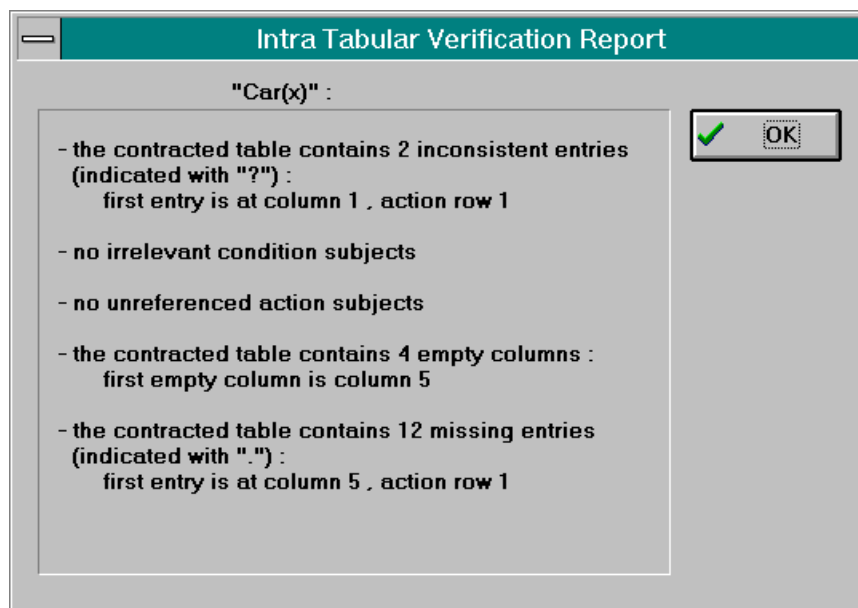


Figure 7 : CAR(X), Intra-tabular verification report

5.2.2. LOOK(X)

Table LOOK(X) is a condition subtable of CAR(X) and is used to determine the truth value of the literal Look(x). Figure 8 displays PROLOGA's table editor for LOOK(X).

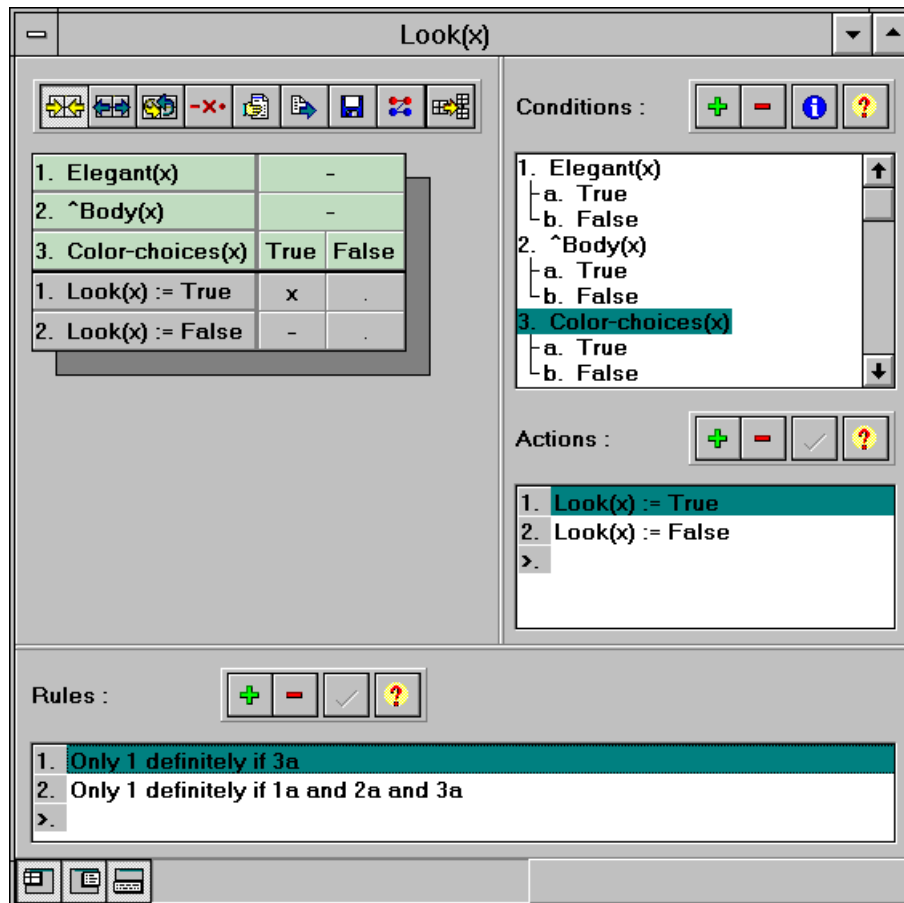


Figure 8 : LOOK(X), table editor

Conditions and actions are indicated by numbers, states by letters. Action entries can either be filled in directly on the table itself by mouse-clicks, or one can choose to type the input rules into the list at the bottom of the window. For example, rule 1 ('only 1 definitely if 3a') can be interpreted as 'Look(x) is true, and the other action literals are false, if Color_choices(x) is true'. Changes in the input rules immediately result in them being remapped onto a new single-hit decision table. The decision rules that have been entered correspond to rules C24 and C30 of the original rule base.

- C24: Look(x) <--- Color_choices(x)
- C30: Look(x) <--- Elegant(x), Body_shape(x), Color_choices(x)

In the original rule base, rule C24 is said to subsume rule C30. If we would translate this notion of a subsumed rule pair anomaly to a table-based context, the equivalent of it would be a subsumed column pair anomaly. However, since PROLOGA directly maps these rules onto a single-hit table, in which all columns are mutually exclusive, the anomaly does not persist in the final decision table. PROLOGA ensures column exclusivity by automatically constructing a tree-structured table, based on an ordered combination of relevant condition states such that the states of the lower condition subjects vary first (cf. Figure 9). This is another example of how verification can often be embedded into the modelling framework itself.

multiple-hit table :

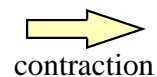
1. Elegant(x)	-	True	-
2. ^Body(x)	-	True	-
3. Color-choices(x)	True	True	False
1. Look(x) := True	x	x	.
2. Look(x) := False	-	-	.



subsumes

single-hit table :

1. Elegant(x)	True				False			
	True		False		True		False	
2. ^Body(x)	True	False	True	False	True	False	True	False
3. Color-choices(x)	True	False	True	False	True	False	True	False
1. Look(x) := True	x	.	x	.	x	.	x	.
2. Look(x) := False	-	.	-	.	-	.	-	.



contraction

1. Elegant(x)	-	-
2. ^Body(x)	-	-
3. Color-choices(x)	True	False
1. Look(x) := True	x	.
2. Look(x) := False	-	.

Figure 9 : Subsumption, single-hit vs. multiple-hit

Note that the subsumption in the input rules themselves is not removed, because we consider the decision table format, and not the input rules, as our main representation format. An other reason for doing so is that PROLOGA allows the knowledge engineer to re-extract an optimized rule representation from the table format, and let these rules replace the old ones.

Now, we have a look at the intra-tabular verification report for LOOK(X) (cf. Figure 10).

Here, it is indicated that two conditions subjects are *irrelevant* : Elegant(x) and Body(x).

A condition subject is considered irrelevant, if its state does not influence which actions are to be undertaken. If so, the entire condition row becomes redundant. Often, this kind of anomaly suggests overgeneralisation : the expert or knowledge engineer might have overlooked some of the more detailed decision rules. In our example, the first input rule (cf. rule C24 in the original rule base) might e.g. be too general.

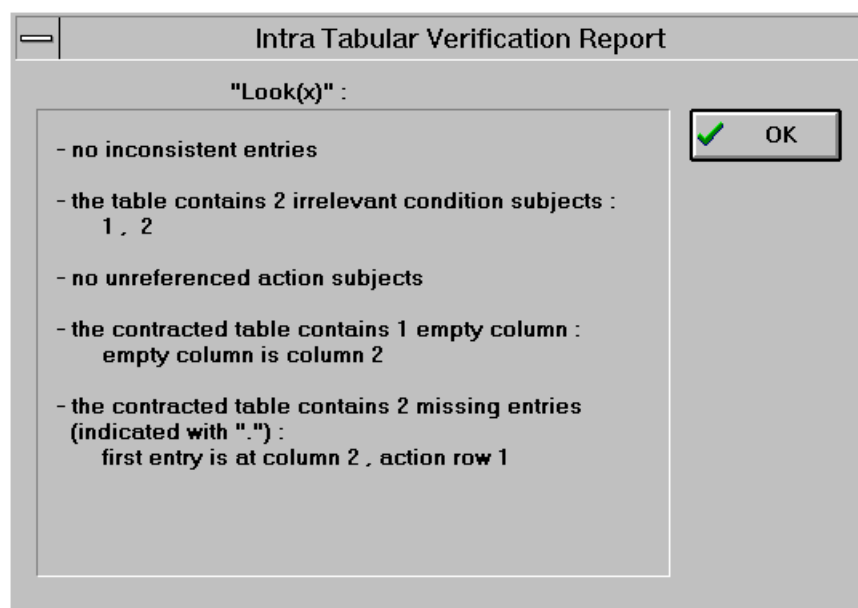


Figure 10 : LOOK(X), intra-tabular verification report

5.2.3. BODY(X)

BODY(X) is a condition subtable of LOOK(X). As explained in the preceding section, condition subject Body(x) of table LOOK(X) is irrelevant, so we can conclude that the associated condition subtable BODY(X) also is irrelevant or *unused*.

The knowledge expressed by this table comes from rules C23 and C37. Because the condition and action sets of both rules are identical, these rules constitute a so-called duplicate rule pair, which is a special case of subsumption.

- C23: Body_shape(x) <--- Aerodynamic(x)
- C37: Body_shape(x) <--- Aerodynamic(x)

For a decision table, this would boil down to the presence of duplicate columns, but again, by enforcing the exclusivity criterion in all decision tables being constructed, PROLOGA excludes any such situation.

The intra-tabular verification report also indicates that Body(x) will never get the value false (*unreferenced action*). Consequently, one might conclude that the knowledge base is incomplete.

5.2.4. COST(X)

COST(X) is a condition subtable of CAR(X). The knowledge represented by this table (cf. Figure 11) corresponds with rules C26, C27 and C28. The found contradiction is indicated in the table.

1. Price(x)	< 7000		≥ 7000
2. Spacious(x)	True	False	-
1. Cost(x) := Low	?	x	-
2. Cost(x) := High	?	-	x

- C26: Low_cost(x) <--- Price_less_than(x,7000)
- C27: High_cost(x) <--- Price_greater_than(x,7000)
- C28: High_cost(x) <--- Spacious(x)

Figure 11 : COST(X), contracted table

In [5], a pair of rules R and R' is defined ambivalent if the antecedent of R' subsumes the antecedent of R, and their consequents violate a constraint. Based on this definition, rules C26 and C28, which, if fired together, would violate the constraint NOT (Cost(x) = Low_cost(x) AND Cost(x) = High_cost(x)), do not constitute an ambivalent rule pair, because there is no subsumption in their condition parts. However, the condition parts of both rules are applicable for a common subset of inputs. A car can be spacious and have a price < 7000 at the same time, e.g. a Honda_Civic. PROLOGA treats this kind of contradiction in the same way as it treats the ambivalent column anomaly discussed in section 5.2.1, namely by imposing column exclusivity.

6. Summary

In this section, a summary of the anomalies detected or prevented by PROLOGA is given.

6.1. Redundancy

- *duplicate column pair* : In the original rule base, rules C23 and C37 constitute a duplicate rule pair. In our definition of a decision table, the exclusivity criterion prevents this type of redundancy from occurring. In PROLOGA, this criterion is operationalized by means of a particular table construction method.
- *subsumed column pair* : Rule C24 subsumes rule C30. Again, ensuring that table columns are mutually exclusive makes that no subsumed column pairs can be introduced.
- *redundant condition row* : In decision table LOOK(X), the presence of an irrelevant condition row was reported (cf. rules C24 and C30). Also in table EXTRA(X), an irrelevant condition row was found (cf. rules C35 and C36). This table was not presented in the text.
- *redundant action row* : This kind of redundancy, which was not illustrated in the given example, can easily be ruled out at construction time, by making sure that no duplicate action subjects can be introduced in the table.

When we take a look at the classification proposed in section 4, we can now conclude that all types of intra-tabular redundancy are dealt with in our approach.

6.2. Ambivalence

In section 4, two kinds of ambivalence were indicated.

Ambivalence on a column level would imply that there are columns in the table, that are all applicable to a particular subset of input environments, but result in contradictory conclusions.

However, this anomaly type will never occur, if the exclusivity principle is sustained.

Consequently, contradictory input rules are prevented from being entered into the table. Instead, an error message is generated. This is the case in tables CAR(X) and IM_EX(X), where conflicting input rules exist that correspond to rule pairs {C38, C40} and {C33, C34} of the original rule base. Another case of ambivalent rules was found in COST(X), and could be traced to rules C26 and C28.

A second kind of *ambivalence* is located *on a row level*. This type of ambivalence occurs when there are two or more rows with contradicting entries in the same column.

6.3. Circularity

In the section on inter-tabular anomaly types, it was shown how circular chains, e.g. the one originating from C25, C31 and C38, if not explicitly needed in our application, are prevented from being introduced, by checking for cycles in the inter-tabular structure graph.

6.4. Deficiency

The first kind of deficiency or incompleteness is caused by isolated rules, such as C42. This rule corresponds to the *unused table* 'INSUR'.

The second type of deficiencies indicated by PROLOGA are *unreferenced actions*, in other words, goal or sub-goal states that can never be reached. In the example, Body(x) will never receive the value false.

PROLOGA also indicates the presence of several empty columns, corresponding to *missing rules*. The completeness criterion makes sure that, on an intra-tabular level, all possible states of the environment are included in the condition part, thus revealing any missing entries in the action part.

6.5. Comparison

As mentioned in the beginning of this paper, the example that we used was designed by Du Zhang and Doan Nguyen in order to test a verification tool named PREPARE. After an evaluation of our own tool PROLOGA, it seems reasonable to compare both tools and concentrate on anomaly types that are not indicated in one system, but are detected by the other system, or that are handled differently.

First of all, subsumption is treated in an entirely different manner by both systems. Because of the construction method used, a PROLOGA table can never contain a subsumed column pair at any point in development (cf. section 5.2.2). In a typical rule based system however, subsumed rule pairs can occur, so PREPARE must check for them. Analogously, with PROLOGA contradictions are already ruled out during the construction of the decision table, so no (perhaps more costly) corrections must be made in a later stage of development. Verification tools that are not tightly integrated into the modelling environment itself, such as PREPARE, will typically only detect the errors until after the knowledge base is almost completed, and ready for processing.

Secondly, there are two classes of anomalies that are not accounted for as such by PROLOGA, but that are indicated by PREPARE.

The first one are unsatisfiable conditions, such as $\text{Performance}(x)$ in rule C40. An unsatisfiable condition is a predicate in the condition part of a rule, that does not appear as the conclusion of another rule, nor is substantiated by a fact. The mere reason for the absence of this type of checks in PROLOGA, is that our modelling environment, at this point, does not feature the inclusion of (static) facts into the knowledge base itself. One of our research topics however is to further improve on the expressiveness of our modelling language.

Another anomaly type not checked for by PROLOGA, are useless conclusions, e.g.

Four-seats(x) in rule C41. Because PROLOGA does not allow the developer to indicate which conclusions are actual goals, and which are only intended as intermediate states of the inferencing process, PROLOGA cannot differentiate between desirable and useless action subjects at the end of a reasoning chain. Again, this element is, in our opinion, of less importance, and is not inherent to our approach itself, but a result of our current tool implementation.

Finally, we discuss the anomaly types that PREPARE fails to detect, whereas PROLOGA does support their detection.

First of all, PROLOGA does not restrict its notion of intra-tabular ambivalence to cases of condition subsumption, but also compares rules with overlapping (though not necessarily subsumed) condition parts (cf. section 5.2.4). That is, decision rules are considered ambivalent if they are applicable to a common set of inputs, whereas their consequents violate a constraint. The more general case of overlapping rules is ignored in PREPARE.

A second important difference lies in the scope of completeness checking. PROLOGA indicates for each table which situations will not result in a solution due to incomplete knowledge. For instance, in the intra tabular verification report for table BODY(X), it is indicated that there is one unreferenced action, viz. Body(x) will never get the value false. PREPARE, on the other hand, does not check for incompleteness at this level of detail. In most rule-based modelling frameworks, this kind of incompleteness is resolved using the principle of negation as failure. This principle states that, if there are no rules which conclude that Body(x) is true, it must be false. Though negation as failure is a simple way to deal with the problem of unspecified knowledge, a multi-value approach (true, false, unknown), such as the one used in PROLOGA, provides a richer interpretative context.

7. Applications

We have used decision tables in a large number of applications and environments. Some examples of the more common areas of experience include :

- modeling and verification of complex managerial decision situations in general ;
- calculation of rates and premiums in banks and insurance companies ;
- verification and visualization of legal procedures ;
- help desk applications.

Two actual systems, in which PROLOGA has been a key factor during development, are :

- HANDIPAK [7] : 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 45 mutually related decision tables (cf. Figure 12).
- 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 modelled into 61 interrelated tables.

In both cases, the experiences with regard to our approach were very positive. Most real-world problems do seem to lend itself for this kind of modelling. The emphasis on modularization also makes sure that the V&V mechanisms embedded in PROLOGA, remain manageable even for larger knowledge bases.

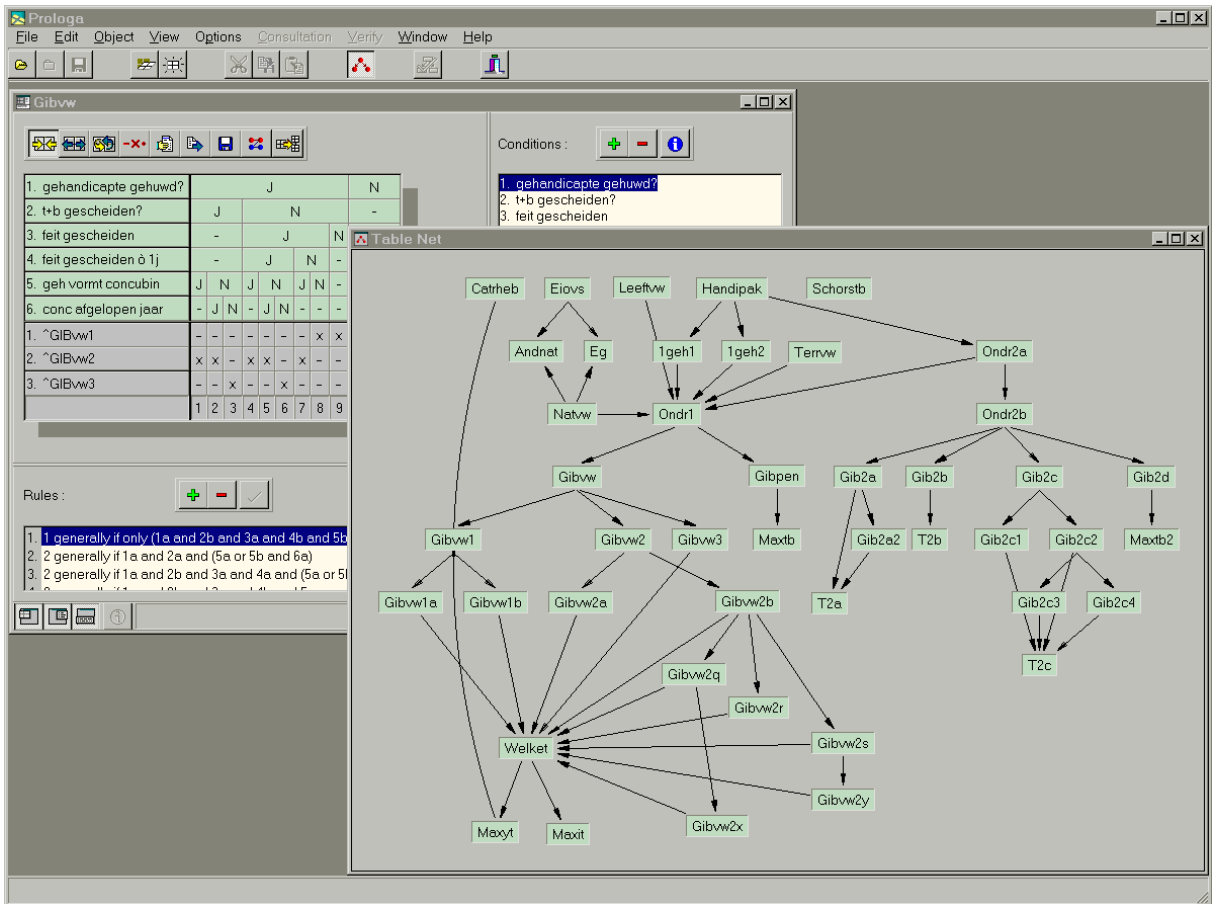


Figure 12 : A look inside the HANDIPAK knowledge base

8. Current & future research

Current research primarily focuses on expanding the expressive power of our model,

- by extending the decision table formalism with fuzzy logic features [13], and
- by integrating it with an object-oriented formal specification model for representing domain concepts and relations.

9. Conclusion

In this paper, it was argued that a proper integration of V&V aspects into the modelling process has the advantage that a lot of anomaly types can easily be prevented, or detected and consequently also be corrected, in an early stage of development. It has also been shown how the decision table formalism can be applied to both model knowledge, and verify it, in an incremental way.

Acknowledgements

The authors wish to thank the referees for their comments and suggestions. Furthermore, we would also like to thank the research fund K.U.Leuven (OT/94/2).

References

- [1] R.M. O’Keefe and D.E. O’Leary, Expert system verification and validation : a survey and tutorial, *Artificial Intelligence Review* 7 (1993) 3-42.
- [2] A. Lalo, TIBRE : Un Système Expert Qui Teste Les Incoherences Dans les Bases de Règles, *Proc. Avignon88 Vol. 3* (1988) 63-84.
- [3] H. Larsen & H. Nonfjall, Modeling in the Design of a KBS Validation System, *Int. Journal of Intelligent Systems* 6 (1991) 759-775.
- [4] N. Liu, T. Dillon, ..., Detecting of Consistency and Completeness in Expert Systems using Numerical Petri Nets, in: J. Gero and R. Stanton, ed., *Artificial Intelligence Developments and Applications* (North-Holland, 1988) 119-134.
- [5] A. Preece & R. Shinghal, Foundation and Application of Knowledge Base Verification, *Int. Journal of Intelligent Systems* 9 (1994) 683-701.
- [6] S. Puuronen, A Tabular Rule Checking Method, *Proc. Avignon87 Vol. 1* (1987) 257-268.
- [7] Robben F., HANDIPAK: computeradviesstelsysteem m.b.t. de financiële tegemoetkomingen aan gehandicapten, in: B. Van Buggenhout, F. Robben, I. Leus, H. Casteleyn, G. Hertecant, W. Demeester, ed., *Het nieuw gehandicaptenrecht. Commentaar bij de nieuwe wetgeving en recente evoluties in het beleid, Recht en Sociale Hulpverlening* (Brugge, Die Keure, 1988) 21-26.
- [8] J. Vanthienen, A. Aerts and C. Mues, A Modelling Approach to KBS Verification, *European Symposium on the Validation and Verification of Knowledge Based Systems (EUROVAV 95)* (Chambéry, France, 1995) 155-171.
- [9] J. Vanthienen, C. Mues and G. Wets, Inter-tabular Verification in an Interactive Environment, *European Symposium on the Validation and Verification of Knowledge Based Systems (EUROVAV 97)* (Leuven, Belgium, 1997) 155-165.
- [10] J. Vanthienen and E. Dries, Decision Tables: Refining the Concept and a Proposed Standard, accepted for publication in: *Communications of the ACM*.
- [11] J. Vanthienen and E. Dries, Illustration of a Decision Table Tool for Specifying and Implementing Knowledge Based Systems, *International Journal on Artificial Intelligence Tools*, Vol. 3(2) (1994) 267-288.
- [12] J. Vanthienen and G. Wets, From Decision Tables to expert system shells, *Data & Knowledge Engineering* 13 (1994) 265-282.
- [13] J. Vanthienen, G. Wets and G. Chen, Incorporating fuzziness in the classical decision table formalism, *Int. Journal of Intelligent Systems* 11 (1996), 879-891.
- [14] J. Vanthienen, Knowledge acquisition and validation using a decision table engineering workbench, *The World Congress on Expert Systems* (Orlando, 1991) 1861-1868.
- [15] J. Vanthienen & J. Wijzen, On the Decomposition of Tabular Knowledge Systems, *New Review of Applied Expert Systems* (1996), pp. 77-89.
- [16] D. Zhang & D. Nguyen, A Tool for Knowledge Base Verification, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 6, No. 6 (1994) 983-989.
- [17] N. Zlatareva , A Framework for Verification, Validation, and Refinement of Knowledge Bases: The VVR System, *Int. Journal of Intelligent Systems* 9 (1994) 703-737.