Developing Legal Knowledge Based Systems Using Decision Tables

J. VANTHIENEN
Katholieke Universiteit Leuven
Department of Applied Economic Sciences
Dekenstraat 2, B-3000 Leuven (Belgium)
Phone: (0)16-28.58.09, Fax: (0)16-28.57.99, E-mail: fdban06@blekul11

F. ROBBEN
Katholieke Universiteit Leuven
Interdisciplinair Centrum voor Recht en Informatica
Tiensestraat 41, B-3000 Leuven (Belgium)
Phone: (0)2-741.84.02, Fax: (0)2-741.83.00

Abstract

Knowledge based systems can be of great use to lawyers in many different areas. Within the framework of the INFOSOC project implemented at the Katholieke Universiteit Leuven, legal knowledge based systems were developed to achieve some of these purposes, using a methodology based on the decision table technique, and a decision table engineering workbench, PROLOGA.

This paper describes the adopted methodology, which is still being refined, the tools used and the experiences in developing a concrete system: HANDIPAK, a knowledge based system with regard to financial benefits for the disabled in Belgium.

Developing HANDIPAK showed that the decision table technique was not only very useful for testing the consistency of legal knowledge, but also for supporting the acquisition and the representation process of that knowledge.

Keywords

Legal knowledge based systems, decision tables, knowledge acquisition, verification & validation

1. Introduction

Knowledge based systems can be of great use to lawyers in many different areas (see OSKAMP [16] and SCHAUSS [22, p. 25-36]).

First and foremost, they can offer substantial support when issuing legal regulations. Knowledge based systems can help simulate the effects of changes introduced to the regulations or help examine the regulations for their internal and external coherence thereby ascertaining the need for new regulations.

Once the regulations have been issued, legal knowledge based systems can be used to sustain legal counselling or legal practice. Most operational legal knowledge based systems are to be situated in this area. Generally, they offer specific advice on the application of law with regard to a real case. They can also help the user to consistently fill in the allotted discretionary decision margin, by presenting the results of a multi criterion analysis of the parameters deduced from jurisdiction. When the legal knowledge based systems are used by the legal or executive bodies that are really entrusted with the application of the law, they usually not only contain the basic regulation but also rules regarding evidence and the procedures to be adopted; they are also often integrated in an administrative processing environment in charge of file management and required document creation.

Legal knowledge based systems can also be used to support legal education. Handed on information is broader and much more accurate and a clearer insight into the legal argumentation patterns is promoted.

Finally, legal knowledge based systems offer support to functional comparative law (see DEBROCK et al. [6] for the description of such a system). Here, the systems concerned are used to compare miscellaneous legal systems with each other in order to detect differences and their causes, and to ascertain

whether it is advisable or necessary to propose amendments or harmonizing initiatives.

When using an appropriate user interface and inference mechanism, it is even possible to use one knowledge base for several of the above mentioned purposes.

Within the framework of the INFOSOC project implemented at the Katholieke Universiteit Leuven, legal knowledge based systems were developed to achieve some of the above mentioned purposes on the basis of a decision table technique based methodology.

This text describes the adopted methodology, which is still being refined, the tools used and the experiences in developing a concrete knowledge based system in this way.

2. The Proposed Approach: From Decision Tables to Consultation Systems

Decision tables and knowledge based systems show some striking similarities, although both approaches put strongly different emphases. While decision tables traditionally stress the representation facilities (with the resulting additional checking capabilities for completeness, consistency and correctness), knowledge based systems are mainly dealing with knowledge formulation (modularity, flexibility) and inference (performance, user friendliness).

Moreover, a lot of knowledge based systems built nowadays are propositional expert systems, which are equivalent to decision tables (COLOMB [4]). It is, therefore, worthwhile to examine in more detail how both approaches can be combined in real problem situations. This does not mean that all areas of law can be reduced to propositional logic, but in a lot of cases (and in most of the current legal knowledge based applications) the representation is appropriate.

Some previous efforts, however, to link knowledge based systems to decision tables, focus only on the transformation from expert systems to decision tables, for reasons of validation & verification (e.g. CRAGUN & STEUDEL [5]) or execution efficiency (e.g. COLOMB [4]). In our point of view, these are important issues, but there is no need to restrict the use of decision tables to this transformation, and it is clearly superior to model the knowledge by means of decision tables from the start, to validate and to optimize the tables and then to implement the system, e.g. using an expert system shell. In other words, the path from decision tables to knowledge based systems is at least as relevant as the reverse direction.

Decision tables provide formal procedures for checking completeness, consistency and redundancy, but table construction may be a difficult task when performed manually. It is therefore our conviction that decision tables, when computer supported construction is available, largely outperform other conditional logic techniques (see also SANTOS-GOMEZ & DARNELL [21]).

The usefulness of the decision table concept in law, e.g. to check or apply legal procedures, has been indicated before (e.g. BERGMANN [1, 2], REISINGER [18]). Also when issuing legal regulations, the decision table technique can be beneficial (see e.g. OVERHOFF & MOLENAAR [17], which is based on the PROLOGA (PROcedural LOGic Analyzer) system for computer-supported construction and manipulation of decision tables, as described further).

This paper, however, reports on the first full scale application of decision tables (using a decision table engineering workbench) in knowledge acquisition and verification & validation for designing and implementing a legal knowledge based system.

2.1. DECISION TABLES: CONCEPTS AND NOTATION SCHEME

A decision table is a tabular representation used to describe and analyze procedural decision situations, where the state of a number of conditions determines the execution of a set of actions. Not just any representation, however, but one in which all distinct situations are shown as columns in a table, such that every possible case is included in one and only one column (completeness and exclusivity).

"A decision table is a table, representing the exhaustive set of mutual exclusive conditional expressions, within a predefined problem area." (VERHELST [28])

The decision table uniquely relates each possible combination of condition states to a combination of action values (CODASYL [3]).

The tabular representation of the decision situation is characterized by the separation between conditions and actions, on one hand, and between subjects and conditional expressions (states), on the other hand. Every table column (decision column) indicates which actions should (or should not) be executed for a specific combination of condition states. The decision table is represented as a table which is split by a double line, both horizontally and vertically, resulting in four *quadrants*. 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) respectively. The resulting quadrants are: *condition stub*, *action stub*, *condition entries* and *action entries*.

The entries part consists of columns (with condition states and action values) separated by a vertical line from the first different condition state. A column then contains a state for each condition or a contraction of states which yield the same result (possibly "irrelevant" ("-") if this is the case for all states), followed by the resulting value for each action.

Because all possible combinations of condition states are present in the decision table (and only once), the table provides a good overview of the combined influence of all conditions. 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 easy checking for consistency and completeness.

Figure 1 shows an example of a decision table. Condition subjects are found in the upper left part of the table, action subjects in the lower left part. Condition states and action values are found at the right hand side. Note that the example

table is not in its most compact form, as repeating patterns could be factored into subtables, but this is beyond the scope of this example.

Every column in the decision table contains a state for each condition subject or a contraction of states that yield the same result (possibly irrelevant (-) if this is the case for all states of the condition), followed by the resulting value for each action subject.

A table column represents a decision rule of the form:

IF CS_1 is S_{1k} AND CS_2 is S_{2m} AND ...
THEN action AS_j AND ...

If each column only contains simple states (no contractions or irrelevant conditions), the table is called an *expanded* decision table *(canonical form)*, in the other case the table is called a *contracted* decision table *(consolidated form)*. The translation from one form to the other is defined as *expansion (rule expansion)* and *contraction (consolidation)* respectively (CODASYL [3]).

The condition subjects and action subjects can refer to other tables (*subtables*). The replacement of these references by the tables themselves, the *junction* of tables, is called (*table*) *expansion*. The reverse process, the *division* into subtables, is defined as *factoring*. Two types of subtables are possible: the action subtable, i.e. a further specification of a certain action, and the condition subtable, determining the value of a condition. All subtables are of the closed type, this means that after ending a subtable, the calling table regains control.

Some combinations of conditions may be *impossible*, in other words, they *cannot* occur. Such combinations may be deleted from the table (see further). Keep in mind that only real impossibilities are to be deleted, combinations that *should not* occur must stay in the table, since they will occur at some point in time (according to Murphy's Law).

1. Married	Yes					No			
2. Separated	Yes				No	-			
3. Concubinage	Yes	No			-	Yes	No		
4. Child at charge	-	Yes	No		-	-	Yes	No	
5. Living alone	-	-	Yes	No	-	-	-	Yes	No
1. Category I	x	x	-	-	x	x	x	-	-
2. Category II	-	-	x	-	-	-	-	x	-
3. Category III	-	-	-	x	-	-	-	-	x

Figure 1: decision table representation

2.2. DECISION TABLES AND KNOWLEDGE BASED SYSTEMS

When building legal knowledge based systems, several objectives are really pursued:

- formalization of the knowledge (the acquisition process)
- checking completeness and consistency
- consultation of the knowledge based system.

The advantage of traditional rule based systems is that when the knowledge is formalized in terms of rules, the consultation is possible through the inference engine. The difficult issue of validation and verification, however, already indicates a major problem with this approach. The real problem is that each of these objectives has its specific way of looking at the problem domain:

- In *knowledge acquisition* we want to stay close to the text. Most texts, procedures, laws, etc. are described in **action oriented**, partial or modular decision specifications: condition combinations for one action are enumerated and the text is organized in an action by action fashion, not suited for fast and correct decision making. This is basically because the objective is different: a legal text often indicates for a specific action (e.g. allowance, tax, subsidy, ...) what the required conditions are.

- As a decision maker (in a consultation environment) we want to know for a specific configuration of conditions what the resulting actions are. This is an opposite point of view. The *decision making* procedure is essentially **condition oriented**: given a specific combination of condition values (one case), what are the applicable actions. Here we are looking for a relevant and efficient (optimized) answer, but this orientation is not present in the text.

- In *validation and verification*, the approach is also **condition oriented**, except now we are not interested in just one answer, but in an **overview** picture of possible combinations and conclusions, which enables us to verify completeness,

correctness and consistency. Traditional rule based systems do not offer these representational advantages.

The decision table approach, however, unifies these three complementary aspects of a decision situation (figure 2): *specification*, *representation* and *execution*.

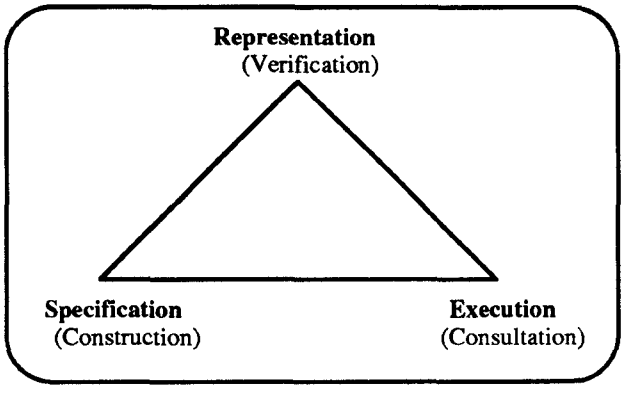

Figure 2: three aspects of a decision situation

The decision table is **condition oriented** and therefore effective in representing and displaying procedural knowledge, with such advantages as: overview, readability, consistency, completeness and correctness. This structured enumeration of decision columns, however, is not the way in which procedural knowledge is acquired, specified or described. To this end, the decision table construction process allows to transform the **action oriented** specification into a condition oriented representation. The decision table being a representation mechanism, (execution) *optimization* is not the main concern. But it may be, once the table has to be converted to an operational system or a manual decision making procedure, e.g. in order to minimize the total test time or the number of questions. The advantage of the decision table approach, however, is that implementation aspects can be separated from the representation, through transformation.

2.3. PROLOGA: DECISION TABLE CONSTRUCTION AND VALIDATION

The PROLOGA (PROcedural LOGic Analyzer) system is an interactive rule-based design tool for computer-supported construction and manipulation of decision tables (VANTHIENEN [24, 26]). This decision table engineering workbench incorporates powerful rule based knowledge acquisition and representation, table based verification, and adequate consultation interfaces to common shells and languages. The system not only supports the manual design techniques, but also offers additional features to enhance construction, manipulation, validation and optimization of decision tables.

A major drawback of the use of decision tables (and many other condition oriented representations) is the complexity of the manual building process. A lot of redrawing work results from small changes like adding a condition, a condition state or an action. Some manipulations like the reordering of conditions are quite impossible to perform manually. It is a major aim of PROLOGA to free the decision table developer from this cumbersome **drawing** job (VANTHIENEN [24]).

In addition, however, PROLOGA was designed to offer some fundamental **modelling** issues :

- A powerful specification language allows the designer to formulate the decision specification in a straightforward way.
- A number of routine jobs like filling action entries from decision rules, generating all condition combinations (without any missing combinations) can be done in a faster and more correct way by the computer.
- The modelling process can be simplified considerably by the use of interactive possibilities such as automatic checking for consistency, correctness and completeness.
- The system can be used for optimization purposes, such as optimal contraction, layout, decomposition into subtables or conversion into efficient program code.

Basic features of the automated construction process

Depending on the characteristics of the problem domain, different *manual* methods for constructing decision tables can be distinguished (see VERHELST [28]). In PROLOGA the construction process largely follows the same steps as in the manual construction methods.

When building a decision table, the designer essentially provides the system with the following information : a list of conditions with their states, a list of actions and a list of relations between condition states and actions (in the form of logical expressions or rules). This will enable the system to construct, display and optimize the corresponding decision table.

Throughout the modelling effort, the following features are available in order to enable a flexible construction and manipulation of the decision table(s) :

- inserting, deleting, changing, reordering the elements of the decision table. The updates are immediately reflected in the decision table(s).
- reference to condition or action subtables (indicated with '^') and display of the hierarchy of decision tables.
- changing the table layout : expanded versus contracted, minimal column width, state repetition, column numbers, etc.
- display and consultation of the equivalent decision tree.
- transformation of the table to other formalisms : code generation, optimal decision tree, consultation monitor, expert system shells.

Validation and verification

Gathering the knowledge is one of the main problems in building knowledge based systems, and usually, after the knowledge acquisition process is finished, a lot of contradictions and insufficiencies remain to be detected and solved. Also, maintaining the knowledge base is not a trivial task which often introduces unnoticed inconsistencies or contradictions. A lot of current knowledge based systems, however, offer little or no guarantees to support validation, change and complexity control.

Verification and validation of knowledge based systems are receiving increased attention (HAMILTON [7], O'LEARY [15], LARSEN [9]).

The emerging problems of validation and verification have led to the occasional use of schemes, tables or similar techniques in knowledge representation and validation. It has been reported earlier (e.g. VANTHIENEN [24], CRAGUN & STEUDEL [5]) that, in a vast majority of cases, the decision table technique is able to provide for extensive validation and verification assistance. It easily enables the designer to check for contradictions, inconsistencies, incompleteness, redundancy, etc. in the problem specification. Most of the common validation problems (NGUYEN [14]) can easily be solved using decision tables (see e.g. VANTHIENEN [26]):

- **Consistency and Correctness of Knowledge:** Dividing knowledge over a large number of rules, designed independently, may lead to problems of inconsistency, such as: *Conflict, Cyclical rules, Invalid attribute values, Unreachable conditions.*

- **Non-redundancy of Knowledge:** Redundancy usually does not lead to errors during consultation of the system, but it may considerably harm efficiency. The main problem with redundancy, however, is not inefficiency, but maintenance and the risk of creating inconsistencies when changing the knowledge base. Common problems are: *Subsumption, Redundant premises, Redundant rules.*

- **Completeness of Knowledge:** No current system is able to incorporate all possible knowledge, but within the specific problem area, the following omissions often occur: *Missing knowledge, Unused attribute values or combinations, Unreachable conclusions.*

One of the major advantages of the decision table approach is that checking for completeness and consistency can already be performed during the design of the knowledge based system.

2.4. FROM DECISION TABLES TO THE CONSULTATION ENVIRONMENT

This transformation framework consists of two parts. First the transformation of (optimized) decision tables to available knowledge representation formalisms (rules, statements, ...) is performed. Finally the consultation environment is added.

Conversion of the decision tables

When designing or generating an application for an existing knowledge based tool, an object structure has to be developed: conditions, condition states, conclusions and table references have to be transformed to the available modelling facilities.
When transforming the decision tables, the following information is available:

- Names of conditions, condition states and actions. References to subtables are indicated in the condition or action names.
- The (contracted) decision tables on a column by column basis.
- Every table can also be provided with an explanation file containing prompts, help texts or other information specific to the consultation. The explanation file is basically independent from the decision logic, such that the decision table can be altered without having to update the help information.

Then the decision logic knowledge which uses this object structure has to be implemented. The knowledge has been modelled in the form of decision tables (with proper verification and validation) and the table logic will be converted to the knowledge base. The implementation of the decision logic can be realized in two different ways.

- When the reasoning process is constant, the table might be transformed to a nested if-then-else structure, where the outcome of the decision is obtained by choosing the appropriate branch in the selection.

- In a lot of cases, conversion to one decision statement is not flexible enough to deal with the knowledge in the knowledge base. One might prefer to transform the decision table into a set of rules. Several alternatives are available, e.g. trying to minimize the number of rules.

Each decision table is converted to a class and condition and action variables are converted to slots, which may be linked to other conditions or actions. The decision table is converted to if-then-else code as a function in a class. All subtables in a hierarchy are generated and linked automatically.

The consultation environment

The consultation environment offers the necessary user and system interfaces to produce a working application. Its implementation will depend on the tool used, but is not problem-specific.

The hierarchy of decision tables is translated into a question and answer interface. The user, however, need not be aware of the existence of the decision tables or any relations between them. In translating the hierarchy, attention must be paid that no circular inferencing arises, that recursive table calls remain possible and that condition subtable results are properly assigned to the calling conditions.

The following demands will have to be met by the consultation environment:

- The consultation environment has to be as independent as possible from the decision tables. Therefore the environment can be built in a generic way and according to specific user interface standards.
- The user interface must allow easy communication with the user, e.g. using multiple windows, mouse support, etc.
- Prompt and help texts should be available for conditions, condition states and actions, to explain the meaning of questions and conclusions.
- At any moment a list of questions and supplied answers can be shown with the ability to change previous answers (WHAT IF simulations).
- At any moment it must be possible to leave the application (saving the current contents of the consultation) and restart

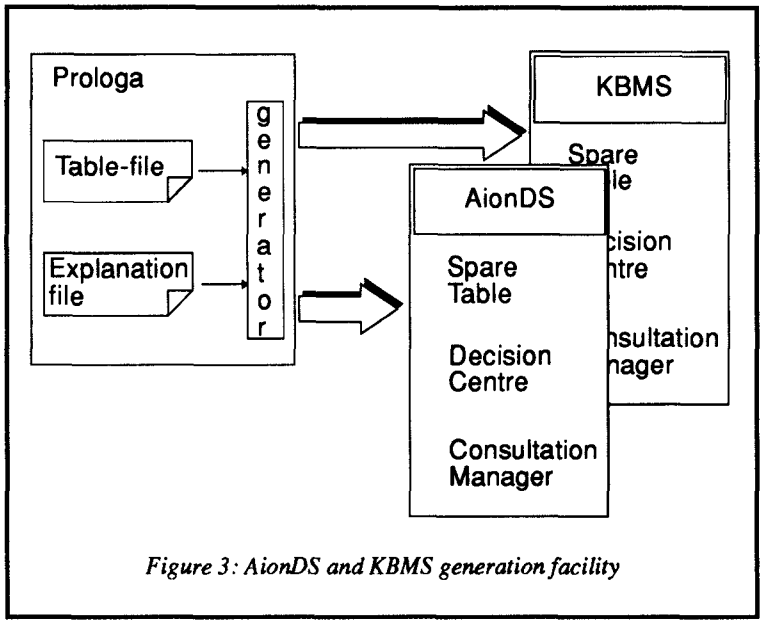

Figure 3: AionDS and KBMS generation facility

it later. This also enables to store a list of prototype cases which can be adapted using the selective change facility.

- Maintaining the knowledge base must be easy. It should for instance be possible to plug in an updated decision table without having to generate the complete application again.

The above transformation process has been implemented in the decision table engineering workbench PROLOGA (PROcedural LOGic Analyzer, VANTHIENEN [26]). An interface was built between PROLOGA and AionDS (Aion Development System, TRINZIC) (JANS [8]). This enables the knowledge engineer to model and validate the knowledge in the form of decision tables and generate a complete consultation application in AionDS (figure 3). It is important to notice that after this transformation process, there will always remain a mapping between the decision tables and the expert system. In this way maintaining the knowledge base becomes easier. The generation process has also been implemented for another tool, KBMS.

3. An Example System: Handipak

3.1. CONTENTS AND PURPOSE OF THE SYSTEM

HANDIPAK is a knowledge based system with regard to financial benefits for the disabled in Belgium (for more details, see ROBBEN [19]).

The first version of this system was developed in 1987 by order of the former State Secretary for the Disabled, at a time when Belgium was drawing up a new regulation with regard to the financial benefits for the disabled. 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 development of the system thus increased the internal coherence of the proposed regulation.

HANDIPAK's main goal however is to support first line social workers when helping the disabled with the introduction of their applications for benefits. In 1991 a new and more comprehensive version was elaborated.

HANDIPAK helps to accurately complete the applications by guiding the user through the relevant regulations; it also gives an idea of what probably will be the outcome of the demand. Concretely, HANDIPAK is a kind of question-answer game that gives a personalized and motivated conclusion based on the answers given. HANDIPAK comprises an extensive on-line textbook and context-sensitive help screens. Answers can be altered at any time, which permits "what-if"-simulations.

3.2. THE CONSTRUCTION PROCESS

The knowledge base of the HANDIPAK advisory system was developed by using the PROLOGA package. The relevant law was couched in 45 mutually related decision tables.

First the regulation to be represented was split up into a number of regulatory subunits dealing with the same subject. The regulation with regard to financial benefits for the disabled was divided into subunits that respectively concern:

- the validity of the application for benefits
- the applicability of the regulation, in which a further distinction is made between rules with regard to
  - the nationality conditions
  - the age conditions
  - the residence conditions
- the granting conditions, in which a further distinction is made between rules with regard to
  - the determination of the category to which the disabled belongs
  - the check on the exhaustion of other social benefits
- the calculation of the maximal height of the benefit
- the calculation of the revenues to be deduced
- the modalities of the payment.

For each of the regulatory subunits, one or more mutually related decision tables were developed by using PROLOGA. The relations between the different tables were specified in a number of control tables.

The decision to create condition or action subtables was, depending on the case, based on one or more of the following criteria:

- preserving the overview; 6 to 7 conditions per table seemed to be the maximum in this respect;
- avoiding redundancy; combinations of condition values repeated in one or more tables were placed in a separate table, to be called by the thus discharged tables;
- showing well-defined actions as intermediate conclusions.

Each of these decision tables was constructed by using PROLOGA according to the steps described below. Some of these steps are illustrated with examples concerning the table in figure 1.

Step 1 : Define the conditions, condition states and actions.

When conditions are defined, a list of condition states is expected for each condition (not restricted to the traditional limited entry type).

Actions have a standard set of action values, only the action subjects must be defined. The possible action values are :

x : this action should be executed.
\- : this action must not be executed
. : undefined
? : contradiction (in the resulting decision table)

First a list is made of all action statements that are mentioned in the text. What is to be retained as an action, depends on the purpose of the table. The purpose of the example table is to determine the category to which the disabled belongs. Therefore, the actions are categories.

Once the actions are defined, a list of condition statements is drawn up from the text. All conditional expressions influencing the execution of an action are taken into account.

Next, the restatements and the complements of other condition statements are deleted from the list, and from the remaining statements, the actual conditions are formulated. In the example table, these are

- the fact to be married;
- the fact to be separated;
- the fact to live in concubinage;
- the fact to have a child at charge;
- the fact to live alone.

Finally, for each condition the exhaustive set of mutual disjoint states relevant to the problem domain is determined. In the example table this happens to be the mere affirmation and negation of the fact.

Step 2 : Define the decision rules.

During this phase, the problem is described as a series of logical IF ... THEN ... relations where the connection is made between a combination of condition states and some actions that must be executed. Therefore, the decision rules are derived from the regulatory text to be represented and are described in terms of a relation between a combination of condition states on the one hand and one or more actions on the other hand. The nature of the relation is indicated by the applied elements of PROLOGA's specification language.

To resemble closely the decision situation that has to be modelled, the specification language offers more and also more powerful facilities than simple IF-THEN rules with AND/OR operators. Designing the procedural decision situation requires a specification language which closely matches natural language and its nuances. A decision situation usually does not consist of a collection of independent descriptions, but contains several levels of structure, e.g. general rules, exceptions, ... Logical expressions can therefore be assigned different levels of significance, in the sense that certain expressions (general rules) can be overruled by later specifications (exceptions), or that a (definite) expression can not be neutralized. Basically two levels are distinguished : **definite** and **preliminary** consequence. A definite rule can not be overwritten (overwriting it creates a contradiction). A preliminary rule on the other hand can be overwritten by subsequent rules, thereby creating a general rule - exception mechanism. In combination with the ONLY operator, a **possible** consequence is also provided: stating that some action is only possible in certain cases means that it is impossible in all other cases.

A decision rule then consists of three parts : an action part, an IF part and a condition part. Some example skeletons of decision rules :

*Actions **[generally]** if condition combinations*
***Not** action **definitely** if condition combinations*
*Action **only possible** if condition combinations*
*Action **definitely if and only if** condition combination*

Condition combinations are expressed using several logical operators : not, and, nand, minus, or, xor, nor, ... For a more detailed discussion of the decision rules, see VANTHIENEN [24].

Because every decision rule is immediately and automatically reflected in the decision table, verification and validation can be performed in an interactive, incremental way. Inconsistent rules are detected automatically. Moreover, knowledge elicitation is ameliorated through the expressive power of the decision rules.

Step 3 : The decision table is filled out

Every decision rule, once defined, is automatically added to the decision table. The decision table is available for display at any time, in expanded or contracted form (and also in the form of a horizontal or vertical decision tree).

Step 4 : Check for Completeness, correctness and consistency

PROLOGA automatically generates a warning message when conditions or actions occur which are not referred to by at least one decision rule. This either indicates that the condition or action is truely irrelevant or that the

decision process has not been represented correctly or completely.

The package also supports checking the table for

- empty columns: when one or more columns show an empty action part, those cases will be evaluated without result;
- contradictions: a column pointing to several actions which are mutually exclusive contains a contradiction;
- incorrectness: one only has to check column by column whether the correct actions are marked.

A more detailed description of the knowledge validation possibilities PROLOGA offers, was given in section 2.3.

The interactivity of PROLOGA allows to correct any incompleteness, inconsistency or incorrectness with immediate view of the result.

Step 5 : Simplify the decision table

From the expanded decision table, adjacent columns or groups of columns leading to identical action configurations are contracted into combined columns, thereby minimizing the number of columns for the given condition order. When all the states of a condition can be combined, this will lead to a condition state which is irrelevant, but partial contraction is also provided (VANTHIENEN [24]).

Step 6 : Optimize the decision table

When the decision table has been constructed and verified, it can be subject to different optimizations, e.g. :

- *Row order optimization*: This determines the condition order which results in the minimum number of (contracted) columns. The condition order is the same for all columns of the decision table. For a table with n conditions, this implies a choice between $n!$ alternative condition orders, some of which might be infeasible because of precedence constraints. It would for instance be senseless in the example table to ask whether someone is separated before having determined that the person in question is married, even though it would reduce the number of columns in the table. For only married people could be considered as separated.
- *Execution time optimization*: Depending on the purpose of the decision table, it might be transformed into optimal test sequences, taking into account condition test times and column frequencies (if available). In the resulting execution tree, conditions are not always tested in the same order anymore. For a table with n limited entry conditions, this implies a choice between $f(n)$ decision trees, where :

$$f(n) = n.[f(n-1)]^2, \text{ with } f(1) = 1$$

$$= \prod_{j=1}^{n} j^{2^{(n-j)}}$$

In decision table research, a lot of effort has been spent on generating this optimal tree (see e.g. MARTELLI [11], LEW [10]).

The set of optimally contracted decision tables was finallly converted into IF-THEN-ELSE structures in the first version of HANDIPAK, and into a set of AionDS statements in the latest version.

3.3. EXPERIENCES FROM DEVELOPING THE SYSTEM

Developing HANDIPAK showed that the decision table technique was not only very useful for testing the consistency of legal knowledge, but also for supporting the acquisition and the representation process of that knowledge. The techniques used in this respect have to be easily manageable by domain experts and must preferably fit in with their "natural" knowledge representation method. The contacts with domain experts, established when developing knowledge based systems, as well as practical seminars given to final-year law students showed that decision tables suit lawyers.

The PROLOGA package offers effective logistic support when drawing up the tables, optimizing them and converting the result towards different technical environments, ranging from IF-THEN-ELSE structures used within traditional computer languages to different available shells such as AionDS or KBMS. Thus the facilities offered by these consultation environments can be used when developing the end product.

Nevertheless, in some areas the methodology can still be refined:

First and foremost, modelling the time dependence of decisions within decision tables is rather difficult. Decision tables are very appropriate to represent argumentations which lead to punctual decisions, but are less suited to represent decision making processes regarding evolutive data. However, this kind of processes frequently occur in law. Legal decisions often take into account the values of certain decision variables at a given moment or during a given period in the past. Moreover, law has a dynamic nature. More than any other knowledge area, legislation continuously undergoes changes likely to strongly influence both its structure and its coherence. The law which was applicable in the past frequently has a more or less substantial influence on future legislation.

Moreover, decision tables are most suited for the representation of decision making processes using selections. It is more difficult (and not appropriate) to represent iterative and, in some way, sequential structures in this manner. This implies that decision tables are particularly useful for legal domains where most of the knowledge takes the form of conditional expressions.

It would also be useful to support the decision of spreading the knowledge to be represented over different decision tables. Presently, the decision to create subtables is based upon certain rules of thumb or on the experience of the knowledge engineer. It is obvious that this practical and implicit knowledge can form the basis for a more systematical approach.

With regard to the optimization of the decision making process, the existing methodology can also be improved. An optimization of the table on the representation level by means of a maximal contraction does not necessarily imply the optimization of the execution of the underlying decision

process, because the optimal evaluation order of the decision variables depends on the specific case. The relative level of difficulty of the evaluation of certain parameters and case frequencies have to be taken into account then.

And finally it would be useful to see the PROLOGA development tool within the framework of a dictionary of conditions and actions e.g. to determine the equality when used in different tables.

4. Conclusion

This paper describes the development of HANDIPAK, a legal knowledge based system with regard to financial benefits for the disabled in Belgium.

The adopted methodology is based on the decision table technique, and a decision table engineering workbench, PROLOGA.

Experiences in developing the concrete system showed that the decision table technique was not only very useful for testing the consistency of legal knowledge, but also for supporting the acquisition and the representation process of that knowledge.

Acknowledgement

The authors greatly acknowledge the invaluable contribution of M. Verhelst.

References

[1] Bergmann W., Entscheidungstabellen - ein fachliches Werkzeug in der Praxis des Steuerberaters (I), DSWR, March 1984, 59-66.

[2] Bergmann W., Entscheidungstabellen - ein fachliches Werkzeug in der Praxis des Steuerberaters (II), DSWR, May 1984, 111-117.

[3] Codasyl, *A Modern Appraisal of Decision Tables*, Report of the Decision Table Task Group, ACM, New York, 1982, 230-232.

[4] Colomb R. and Chung C., *Very Fast Decision Table Execution of propositional Expert Systems* (Proc. AAAI90 1990) 671-676.

[5] Cragun B. and Steudel H., *A Decision-Table Based processor for Checking Completeness and Consistency in Rule-Based Expert Systems*, Int. Journal of Man-Machine Studies 5 (1987) 633-648.

[6] Debrock K., Lemmens V., Robben F. and Van Buggenhout B., *The development of a knowledge based system for the comparison of national social security systems of member states of the European Communities in a dynamic perspective*, in THE FOUNDATION FOR LEGAL KNOWLEDGE SYSTEMS, Legal knowledge based systems. Aims for research and development, Lelystad, Koninklijke Vermande, 1991, 36-47.

[7] Hamilton D., Kelley K. and Culbert C., *State-of-the-Practice in Knowledge-Based System Verification and Validation*, Expert Systems with Applications 4 (1991) 403-410.

[8] Jans S., *Een interface tussen Prologa en AionDS, Toepassing van de beslissingstabellenbenadering bij het ontwerp en de implementatie van een kennissysteem*, K.U.Leuven TEW dissertation, 1992.

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

[10] Lew A., *Optimal Conversion of Extended-entry Decision Tables with General Cost Criteria*, Comm. of the ACM 4 (1978) 269-279.

[11] Martelli A. and Montanari U., *Optimizing Decision Trees through Heuristically Guided Search*, Comm. of the ACM 12 (1978) 1025-1039.

[12] Merlevede P. and Vanthienen J., *A Structured Approach to Formalization and Validation of Knowledge*, in Proc IEEE/ACM International Conference on Developing and Managing Expert System Programs, Washington, DC, 1991, 149-158.

[13] Moulin B. and Rousseau D., *Automated Knowledge Acquisition from Regulatory Texts*, IEEE Expert, October 1992, 27-35.

[14] Nguyen T., Perkins W., Laffey T. and Pecora D., *Knowledge Base Verification*, AI Magazine 2 (1987) 69-75.

[15] O'Leary T. and Goul M. et al., *Validating Expert Systems*, IEEE Expert 3 (1990) 51-58.

[16] Oskamp A., *Het ontwikkelen van juridische expertsystemen*, Reeks Informatica en Recht nr. 11, Antwerpen/Deventer, Kluwer, 1990.

[17] Overhoff R., and Molenaar L., *In de Regel Beslist, Een beschouwing over regelgeving met behulp van beslissingstabellen*, Doctoraal Proefschrift, Rijksuniversiteit Leiden, SDU Uitgeverij Plantijnstraat, 's-Gravenhage, 1991, 382 pp.

[18] Reisinger L., *Rechtsinformatik*, Walter De Gruyter, Berlin/New York, 1977, 240-245.

[19] 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.

[20] Robben F., Herbosch E., Van Buggenhout B. and Van Bulck K., *The computer supported development of juridical advice systems based on the decision table technique*, in THE FOUNDATION FOR LEGAL KNOWLEDGE SYSTEMS, Legal knowledge based systems. An overview of criteria for validation and practical use, Lelystad, Koninklijke Vermande, 1990, 50-56.

[21] Santos-Gomez L. and Darnell M., *Empirical evaluation of decision tables for constructing and comprehending expert system rules*, Knowledge Acquisition 4 (1992), 427-444.

[22] Schauss M., (ed.), *Systèmes experts et droit*, Bruxelles, Story-Scientia, 1989.

[23] Susskind R., *Expert systems in law, Oxford*, Clarendon Press, 1987.

[24] Vanthienen J., *Automatiseringsaspecten van de specificatie, constructie en manipulatie van beslissingstabellen*, K.U.Leuven Dept. Applied Econ. Doctoral Dissertation, 1986.

[25] Vanthienen J., *Een moderne kijk op beslissingstabellen*, Informatie, December 1988, 912-937.

[26] Vanthienen J., *Knowledge Acquisition and Validation Using a Decision Table Engineering Workbench*, The World Congress on Expert Systems, Pergamon Press, Orlando, 16-19/12/91, 1861-1868.

[27] Vanthienen J. and Wets G., *Mapping Decision Tables to Expert System Shells: An Implementation in AionDS*, Research Paper 9228, K.U.Leuven, Dept. Applied Econ., 1992.

[28] Verhelst M., *De Praktijk van Beslissingstabellen*, Kluwer, Deventer/Antwerpen, 1980.