v. 3, n. 36, 2023

Journal of Engineering Research

ON DISCOVERING QUALITATIVE KNOWLEDGE IN RULE BASED KNOWLEDGE BASES. AN INTELLIGENT APPROACH

Gabriel Fiol Roig

Abstract. Production rules, also called antecedent-consequent rules, are a com- mon way to express the knowledge base (KB) for reactive agents. Only one of the KB's rules can be triggered at any specific moment in a reactive KB. This means that an order for evaluating the rules must be stablished. A top-down order is typically considered, where rules at the top of the KB have a higher priority. Let n the number of attributes of the domain of a KB and m the amount of rules of KB. The KB's computational efficiency to determine the action to be performed is $O(n'm)$ time units (where $O(k)$ refers to the asymptotic big O nota- tion, widely used to express the computational cost of algorithms). However, other ways that are more efficient to describe actions in terms of attribute-value pairs can be considered. In this way, decision trees constitute a simpler alternative decision structure, whose computational efficiency is $O(n)$. A decision tree fits a KB if it strictly respect the description of the actions in terms of the attributes just like the rules of the KB do.

This paper describes a method to discover qualitative knowledge from a KB by generating a fitting and optimal decision tree from it. Whereas the considera- tion of the fitting factor for a decision tree depends exclusively on the KB from which it is induced, the optimality factor depends on the nature of the problem in hand. Thus, the resulting tree strictly respects all the properties and priorities of the KB's rules as well as the optimality criterion.

Keywords: Knowledge Acquisition, Qualitative Knowledge, Data Mining, Rule Based Knowledge Bases, Decision Trees, Inductive Learning.

INTRODUCTION

Data mining seeks to take advantage of huge amounts of data, known as input data or examples, in order to acquire useful knowledge (i.e. qualitative knowledge) [5] about some concepts. Such input data are expressed extensively in terms of conjunctions of attribute-value pairs, so that each example is associated with a given concept or action. The set of all input data is known as *data source*. Knowledge acquisition is based on inductive processes applied to the data source. The results are expressed intensively, in terms of some abstract structure (for example, production rules, decision trees, graphs, statistical expressions, etc.) wich constitutes a qualitative descripcion of the conceps in terms of the attribute values.

There is a category of poorly studied problems whose aim is to improve the quality and efficiency of data sources expressed in an intensive format (for example, produc- tion rules, graphs, etc). A rigorous improvement in the quality and efficiency of an intensive knowledge base requires an ingenious and complex process, since there are several determining factors affecting the process. The three main issues are mentioned below.

The first issue is to consider what specific portions of knowledge covered by the original data source have been omitted or simply are not part of the problem domain (it is a common practice in rule-based systems that a rule covers specific portions of knowledge that are not part of the problem domain). Since it is impossible to determine a solution for this problem, then the only possibility is to assume that any improvement of the original data source must strictly respect its data cover.

The second issue is to determine how to treat the individual specific portions of knowledge coming from different covers. For example, how to treat specific portions of knowledge covered by two or more different

conflicting rules. The results of this stage are made up of a source of specific portions of knowledge (i.e. maximally specific data).

The third issue is to determine the criteria that should govern the improvement of efficiency and quality of the final results. They will be applied, by way of inductive procedures, to the specific data source obtained in the previous issue. Regarding the efficiency improvement, one must decide what kind of structure will be used (i.e. whether a structure of the same type as the initial (abstract) data source, or another type of structure considered more efficient). As for the quality of the final results, it is im- portant to remark that it depends exclusively on the nature of the problem in hand.

This paper is focussed on this category of problems, particularly in those where the initial data source is expressed in terms of a rule-based knowledge base. The problem in hand is stated as follows: given a data source expressed as a rule base (KB hence- forth) for a reactive agent, generate an optimal decision tree that fits strictly with the KB, that is, that strictly respect the description of the actions in terms of the attributes just like the rules of the KB do. The optimal factor depends on the nature of the problem in hand.

RELATED WORKS

Only a few works to do with building decision trees from a set of production rules can be found in literature. However, all of them are designed to deal with declarative rule bases (i.e. rule bases with absence of constraints about the order of evaluating the rules) without conflicting rules. In contrast to this apprach, the method proposed in this paper is capable of dealing with both, declarative and procedural rule bases with con- flicting rules. Declarative rule bases are much simpler than procedural ones, since they are just focused the on structural description of the conceps/ actions in terms of attribute- value pairs. Thus, a declarative rule base is much easier to modify and adapt to different situations than a procedural one.

In the following, a summary of the few published methods on creating decision trees from a declarative rule base is presented.

The AQDT-1 system [6] is the first published work found in literature. Taking a declarative rule base as data source, the method build a decision tree just from the rules.

Decision rules used in this method are learned by the AQ15 [2] or AQ17 [3] inductive learning programs. The main issue of the method is creating the minimum cost decision tree (i.e., a tree that minimizes the overall cost of making classification decisions). Thus, the method is guided by some attribute selection criteria in order to determine the attribute to be placed at each node of the tree. The method favor the attributes with the best cost. For example, attributes appearing in the rules describing the most frequent concepts (i.e. equivalence classes). Three criteria based on the rule properties are used to measure and choose the attributes: *disjointness*, *dominance* and *extent*.

The AQDT-2 system [8] constitutes a later version of AQDT-1. It generates a deci- sion tree from decision rules, which have also been learned by either rule learning sys- tems, AQ15 or system AQ17. AQDT-2 includes some new features, some of them are: a method to make use of new attributes; a method for controlling the degree of gener- alization; two new attribute selection criteria; a new method for combining different attribute selection criteria and the ability to generate unknown nodes in situations when there is insufficient information for generating a complete decision structure.

The RBDT-1 method [14] generates a decision tree from a declarative rule base by using three different criteria to determine the

best fitting attribute for each node of the tree. These criteria are the attribute effectiveness (AE), the attribute autonomy (AA), and the minimum value distribution (MVD).

MOTIVATION

The above methods were designed to generate decision trees from declarative rule bases without conflicting rules. They have been conceived to be simple, allowing a fast (on- demand) generation of adequate decision trees.

Some important limitations of all the previous methods are: 1) They don't strictly respect the description of the actions in terms of the attributes just like the rules of the KB do. This issue reveals the ability of the generated decision trees to cover unknown or missing input data that are not covered by the decision rules. Under this situation, these methods work correctly only for a certain (small) number of simple problems, in particular, for the so called problems with a closed domain. That is, problems whose domains include all the possible input data (i.e. no missing or unknown input data are possible). 2) These methods are only designed to process declarative rules bases; there- fore, they do not allow inducing decision trees from rules bases with conflicting rules.

3) The methods cannot be applied to improve the quality of the initial knowledge base, since they do not consider criteria that depend on the nature of the problem in hand.

Three issues strongly related to rule bases are: *accuracy*, *cost* and *efficiency*. Accu- racy refers to the possibility of the rule base to produce wrong decisions. As the rule base is supposed to be created by an external entity (a learning program or a human expert) then it is assumed to be guaranteed for any situation. The cost of a rule base concerns that issues demanded by the user (for example, a rule base with a minimum cost, where the cost refers to the sum of the explicit cost of the individual attributes used to describe the actions; a rule base with a minimum number of attributes; a rule base guaranteing the fastest decision making; etc.). Cost issues need to be improved sometimes, since the user sometimes does not know precisely what really is needed, or perhaps the initial cost requirements are no longer valid at the current moment. Efficiency has to do with the computational resources required to make use of the rule base and it can often be improved.

The motivation of this work lies in the improvement of the above issues. Therefore, given an initial rule base, a more efficient structure in decision tree format will be generated. This tree is optimal according to some criteria that depend on the nature of the problem; In addition, it strictly respects the properties of the rule base.

ON RULE BASES

The most common way of expressing the knowledge base of a reactive agent [12] is through a set of production rules, also called antecedent-consequent rules. The antecedent of the rules is made up of conjunctions of attribute-value pairs, whereas the consequent consists of a single action (also called a concept), which can be triggered under the assumption of the truth of its antecedent.

A rule base is a set of rules. For example:

$\{ (A=x) \wedge (B=1) \rightarrow Ai, (C=3) \wedge (B=0) \rightarrow Aj, (A=x) \rightarrow Ak, (A=y) \wedge (B=0) \rightarrow Aj \}$

The rule base includes four rules, three attributes, *A*, *B*, *C* and three actions, *Ai*, *Aj* and *Ak*. The domain of attribute A is {*x*, *y*}, that of B is {0, 1} and that of C is {3, Ø3}, where Ø3 means any value other than 3.

When the antecedent of a rule is true, then the rule is said to be triggerable. In the case of rule bases for reactive agents, only one rule can be triggered. If the antecedent of all the rules is false. Then the agent must not trigger

any action, which is usually represented by the empty action, *skip*. Thus, the previous rule base can be extended with a new rule, as follows:

{*(A=x) ^ (B=1) → Ai, (C=3) ^ (B=0) → Aj, (A=x) → Ak, (A=y) ^ (B=0) → Aj,*

Ø[(A=x) ^ (B=1)] ^ Ø[(C=3) ^ (B=0)] ^ Ø[(A=x)] ^ Ø[(A=y) ^ (B=0)] → skip} In order to preserve the initial format of the rule base, the last rule (also called a *skip rule*) must first be turned into a disjunctive normal form (DNF format) expression. Then it is easy to write the resultant DNF expression as a set of skip rules whose antecedents

are made up of conjunctions of attribute-value pairs.

Now, the question focuses on «which of the rules with a true antecedent should be triggered».

DOMAIN OF A RULE BASE

The domain of a rule base, also know as Object Attribute Table (OAT from now on), is the set of input data covered by the rules. It constitutes an extensional representation of knowledge. Formally, an OAT [4], [9] is a two-dimensional structure whose rows represent input data or examples described in terms of the values of a set of attributes, so that each column, except the last one, refers to an attribute. The last column of a row represents the action (concept) associated with the corresponding example. The inter- section of a row and a column represents the value of an example for the corresponding attribute (action in the case of the last column).

The process of obtaining the domain of a KB is based only on the KB's characteris- tics and not on the initial data domain from which the KB was induced. The basic ele- ments of the domain of a KB are: the set of attributes appearing in the rules, the domain of these attributes (i.e. values that take part of some rule) and the actions involved in the rules. Once these elements have been setted then the domain/OAT can be obtained by proceeding as follows:

First, the KB must be extended with all the skip rules. These rules cover the input data not covered by the rule base. Thus, the resultant rule base covers all of possible input data. This rule base is known as the *complete knowledge base* (CKB).

Next example will facilitate the discussion. Let us consider the following KB:

r1: *(A=x)* → α

r2: *(B=1)* → β

The basic elements of the domain are: the attributes A and B, with domains, {*x, Øx*} for attribute A and {1, Ø1} for attribute B and the involved actions, α and β.

There is only one possible skip rule, which can be represented as: r3: $(A^1\ x) \wedge (B^1\ 1) \rightarrow skip$

Thus, the CKB contains three rules: r1, r2 and r3.

Second, the condition part of each rule of the CKB must be extended so that it contains all the attributes involved in the KB. The question that arises at this point is how to interpret the values of those attributes that do not take part in the conditions of the rules of the CKB.

For example, rule r1: *(A=x)* → α, only contains attribute A. How should the value of attribute B be interpreted for this rule? Obviously, attribute *B* is not necessary to decide the action to trigger. In other words, if the value of *A* is *x*, then it does not matter what the value of *B* is to decide that α must be the action to take. In these cases, the value of *B* will be represented by an asterisk, *, whose interpretation is « Any value of the domain, DB, of attribute B». Thus, rule r1 can be rewritten as: *(A=x) ^ (B=*)* → α. This process is known as rule extending process.

The set of rules resulting of applying the two previous steps constitutes the domain of the rule base. Figure 1 illustrates the domain/

OAT of the rule base of the example.

An OAT containing unespecified values (i.e. values represented by asterisk symbol) for some input data is known as an incompletely specified OAT [10], [11].

	A	B	action
r_1	x	*	α
r_2	*	1	β
r_3	$\neg$x	$\neg$1	skip

Fig. 1. OAT

INDUCING DECISION TREES FROM DECISION RULES

In order to facilitate the discussion, let us show first a simple example on how to create intuitively a decision tree that strictly respect the accuracy properties of a declarative rule base, that is, a rule base with absence of constraints (i.e. priorities) on the order of evaluating the rules.

INDUCTION FROM DECLARATIVE RULE BASES

Example. Consider the example of the rule base of section 4, without priorities established on the evaluation order of rules:

r1: *(A=x)* → α

r2: *(B=1)* → b

Three steps must be taken to induce a decision tree from the considered KB: First, extend the initial rule base by adding the skip rules (i.e. obtain the CKB). Second, obtain the OAT from the CKB. Figure 1 shows such an OAT.

Third, apply an inductive procedure to the OAT in order to acquire qualitative knowledge in order to create a decision tree.

It is very important to note that the inductive procedure of step 3 must be capable of inducing qualitative knowledge from incompletely specified examples (i.e. examples including the asterisk value for some attribute) and generating only decision trees that strictly respects the accuracy properties of the knowledge base. The only method found in literature that guarantees such properties is described in [10], [11].

Figure 2 shows all the possible decision trees that can be generated from the men- tioned inductive method. Any of both decision trees can be selected to replace the de- clarative rule base.

Note that decision trees are procedural structures that determine a well defined order on the attribute testing.

Now, we are in position to describe formally the accuracy constraints that a decision tree must satisfy to guarantee the accuracy properties of the rule base from which it was induced.

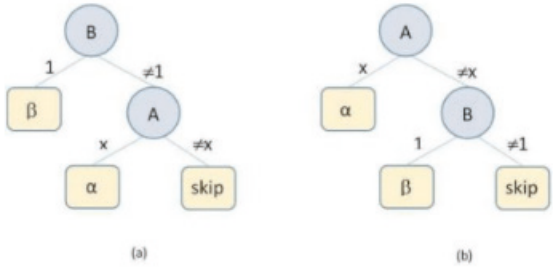

Fig. 2. Decision Trees

Definition 1. Let KB be a rule base, A is a decision tree whose attributes, attribute domains and actions correspond to the basic elements of the domain of KB, and KBA the rule base of tree A. Note that each branch of the tree constitutes a rule, whose antecedent is made up of the conjuction of the attribute-value pairs of the inner nodes in the branch, and the consequent term is the action represented by the leaf node. Tree A is said to strictly respect the accuracy properties of KB if, and only if, KB is a logical consequence of KBA, that is, KBA|=KB.

By applying definition 1 we can prove formally that both trees of figure 2 strictly respect the accuracy properties of the rule base. Let EKB(a), EKB(b) be the sets of input

data (i.e. states) covered by trees (a) and (b) respectively. Note that EKB(a)={*(A=x, B=1,* b*), (A*^{1} *x, B=1,* b*), (A=x, B*^{1}*1,* α*), (A*^{1} *x, B*^{1}*1, skip)*} and EKB(b)= {*(A=x, B=1,*

α), *(A*^{1} *x, B=1,* b*), (A=x, B*^{1}*1,* α), *(A*^{1} *x, B*^{1}*1, skip)*}. On the other hand, the set of input data, EKB, covered by KB, is: {*(A=x, B=1, α), (A=x, B=1,* b*), (A*^{1} *x, B=1,* b*), (A=x, B*^{1}*1,* α*), (A*^{1} *x, B*^{1}*1, skip)*}. As EKB(a), EKB(b) Í EKB, then KB(a) |= KB and KB(b) |= KB.

Any decision tree generated by the inductive procedure described in [10], [11] sat- isfies definition 1.

Continuing with the example, we have only considered the accuracy properties of the decision tree until now; however, nothing has been said about the cost constraints that decision trees must satisfy. It is the case, for example, of a user that demands a minimum cost classification of the concepts/actions, where the cost depend on the on an explicit cost defined for each individual attribute. Let us consider for example that the cost of attribute A is 3 units and the cost of B is 1 unit. Thus, the cost of tree (a) of figure 2 is 1 unit to classify/trigger the concept/action beta, 4 units to classify alfa and 4 units for the skip action. The total cost of three (a) is 9 units. In case of tree (b), its total cost is 11 units. Therefore, tree (a) is the best of all the possible decision trees and so, it must be selected.

As the cost constraints of a decision tree depend on the nature of the problem (i.e. the user demands) then there exist a wide range of criteria based on cost requeriments. Designing an inductive system capable of covering all the possible cost constraints is not a conceivable task. However, designing a parametrized system, that is, a system that implements some general cost constraints from which other more particular constraints can be derived, is a feasible task. This is the case of the UIB-IK system [9], which has been applied to a wide range of practical problems coming from different domains. Next are some related applications. In [7] a medical application is presented. Paper [16] describes an application in the field of finance and business. Work [13] pre- sents a practical study in the field of home assistance for disabled people. [17] is a contribution in the domain of web page classification. Papers [10], [11] describe an extension of the UIB-IK system [9] to deal with incompletely specified OATs.

INDUCTION FROM PROCEDURAL RULE BASES

When constraints about the order of evaluating the rules are stablished, the resulting decision tree must respect them. Consider the KB of the previous example with a top- down priority (i.e. rules are evaluated from the top to the bottom). Tree (a) of figure 2 is not correct, since rule r1 of the KB should always have higher priority on the rest of rules. However, tree (b) is correct since it respects the priority order of KB.

The priorities may cause that a state (i.e. an input data) is covered by several rules with different actions associated to their consequents. Such states are known as *conflicting states* and rules covering conflicting states are called *conflicting rules*. There- fore, the problem is to determine the order which the rules covering conflicting states must be evaluated in. This work considers the most common order of evaluation of the rules, known as the top-down order.

The problem is now stated as follows: given a KB with a top-down evaluation order stablished, create a decision tree that strictly respect the accuracy properties of the KB (i.e. definition 1) and the cost constraints. Next example will facilitates the discussion. Example. Consider the following set of four rules, r1, r2, r3 and r4 covering all the possible states (i.e. there is no need for adding

any skip rule), whose attributes, *A* and *B*, are binary and a top-down evaluation order of priorities is considered:

{ r1: *(A=0)* → α; r2: *(A=1)* ^ *(B=0)* → α; r3: *(B=1)* → b; r4: *(B=0)* → b }

The conflicting states are: *(A=0)* ^ *(B=0)*, *(A=0)* ^ *(B=1)* y *(A=1)* ^ *(B=0)*. The actions associated with each of these states are α and b. However, the system of prior- ities establishes that these states have only the α action associated. There is only one non-conflicting state, *(A=1)* ^ *(B=1)*, whose associated action is b.

The above can easily be seen through a Karnaugh map, as illustrated in figure 3.

A\B	0	1
0	α, β	α, β
1	α, β	β

Fig. 3. Karnaugh map

Cells of the Karnaught map have been filled with the actions associated to each state covered by the rules.

A cell containing more than one different action represents a conflicting state. Actions in the cells are listed in left-right order, according to their decreasing order of priorities. Thus, the first action of the list (i.e. the left action) has a hightest priority (i.e. is a first order action), the second action is a second order action, and so on If two or more

actions have the same priority, then the relative order among them is not relevant.

From figure 3, we have that state *(A=0)* ^ *(B=1)* is covered by rules r1 and r3. This state has associated the actions α and b. Since r1 has higher priority than r3, then the corresponding actions have been sequenced by first placing α and then b in the figure. The same happens with states *(A=0)* ^ *(B=0)* and *(A=1)* ^ *(B=0)*, which are covered by rules (r1, r4) and (r2, r4) respectively.

HOW TO SOLVE CONFLICTS AMONG RULES?

Conflicts among rules can be removed by associating each state with one single ac- tion, the highest priority action. Figure 4 shows the results for the previous example.

Once all the conflicts have been removed, then the resulting rule base can be interpreted as a declarative rule base. The OAT obtained from the rule base of figure 4 is illustrated in figure 5.

A\B	0	1
0	α	α
1	α	β

Fig. 4. Priority actions associated with each state

	A	B	action
r_1	0	0	α
r_2	0	1	α
r_3	1	0	α
r_4	1	1	β

Fig. 5. OAT of the rule base of figure 4

Now, we can create, from the OAT, a decision tree that strictly respects the accuracy properties and the priorities of the KB, by applying the mentioned inductive procedures described in [10], [11]. Figure 6 shows two of such decision trees.

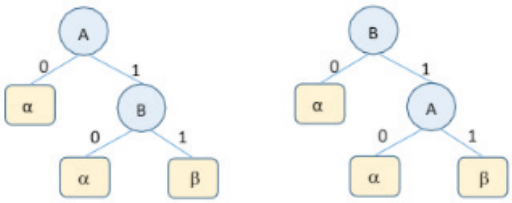

Fig. 6. Decision trees resulting from applying the inductive process

Decision trees in Figure 6 are *functionally equivalent* to the rule base (i.e. they describe actions with strict accuracy, just as the rule base does) which they were induced from.

Any of such trees can be used as a decision structure to replace the rule base. Note that in this example we have not considered issues related to the cost of the attributes.

FORMAL CONSIDERATIONS ABOUT DECISION TREES FUNCTIONALLY EQUIVALENT TO A KB

The following definitions specify the formal properties of a decision tree functionally equivalent to a KB.

Definition 2. *Intersection between two rules*. Let r_i: $c_i \rightarrow A_i$ y r_k: $c_k \rightarrow Ak$, be two rules. If $c_i \wedge c_k \neq F$, being F the symbol representing the proposition *False*, then rules ri and rk are said to intersect. Such rules are known as *intersection rules*. Note that two rules intersect if they cover common states. Two rules can be intersection rules but not necessarily conflicting rules.

Definition 3. *Highest priority rules*. Let KB be a rule base and r_i: $c_i \rightarrow A_i$ a rule whose antecedent (c_i) and consequent action (A_i) belongs to the attribute and action domains, respectively, of KB. Moreover, rule r_i don't necessarily belong to the rule base KB (i.e. r_i is not necessarily one of the rules of KB). Let KB_{ri} be the set of rules of KB intersecting with r_i. Rule r_i is said to be a *highest priority rule*, if the consequent action of r_i matchs with that of the higest priority rule of KB_{ri}.

Definition 4. *Decision tree functionally equivalent to a rule base KB*. Let KB be a rule base, A a decision tree, and KB_A the rule base of tree A. Tree A is said to be *functionally equivalent to rule base KB* if, and only if, KB is a logical consequence of KB_A, that is, $KB_A \models KB$, and any rule of KB_A is a highest priority rule.

GENERATING A DECISION TREE FUNCTIONALLY EQUIVALENT TO A KB

The general procedure to turn a KB (with or without priorities) into a functionally equivalent decision tree is as follows.

Step 1. Complete the KB with all possible skip rules.

Step 2. Remove all the conflicts among the rules of the KB through determining all the conflicting states and assigning the corresponding relevant action to each of them (i.e. the highest priority action).

Once this step is performed, no state will have more than one action associated. Consequently, the resultant rule base is really a declarative rule base. Now, we are ready to build the OAT.

Step 3. Build the OAT from the KB resulting from step 2.

Step 4. Apply the corresponding inductive procedure to the OAT of step 3. Next example will clarify the above four steps.

Example. Generate a decision tree for the following binary rule base with top-down priorities:

{r_1: *(A=1)* ∧ *(B=1)* → α, r_2: *(A=1)* ∧ *(C=0)*→ b, r_3: *(B=0)* ∧ *(C=1)* → α}.

Step 1. Complete the KB with all possible skip rules.

In order to facilitate the discussion, we will make use of the Karnaught map of figure 7, which illustrates all the possible states of the KB.

A\BC	00	01	11	10
0	skip	α	skip	skip
1	β	α	α	α, β

Fig. 7. Karnaugh map of the KB

Two simplified skip rules, r_4, r_5, can be derived from figure 7. Obviously, also three simpler skip rules could be derived. It does

not matter which of these sets you choose since it will not affect the final results.

r4: *(A=0) ^ (C=0) → skip*

r5: *(A=0) ^ (B=1) → skip*

Step 1 is now completed. The new KB contains five rules.

Step 2. Remove conflicts. Determine the states involved in each conflict among rules and assign the highest priority action to each of them.

A\BC	00	01	11	10
0	skip	α	skip	skip
1	β	α	α	α

Fig. 8. Resulting states free of conflict

It can be seen from figure 7 that there is only one conflicting state: *(A=1) ^ (B=1) ^ (C=0)*. According to the priorities of the KB, action α must be assigned to this state. The resulting assignment of actions to individual states is illustrated in figure 8.

Once conflicts have been removed, then it is easy to obtain the corresponding KB without conflicts. As in step 1, more than one set of rules may possibly be generated. It does not matter which of these sets is choosen. We will choose the next set of rules:

{r_1: (A=1) ^ (B=1) → α, r_2: (B=0) ^ (C=1) → α, r_3: (A=1) ^ (B=0) ^ (C=0) → b, r_4: (A=0) ^ (C=0) → skip, r_5: (A=0) ^ (B=1) → skip}

Step 3. Building the OAT from the KB resulting from step 2. Figure 9 illustrates the resultint OAT.

	A	B	C	action
r_1	1	1	*	α
r_2	*	0	1	α
r_3	1	0	0	β
r_4	0	*	0	skip
r_5	0	1	*	skip

Fig. 9. Conflict-free OAT

Step 4. Apply the corresponding inductive procedure to the OAT of step 3. As indicated in section 5.1, the inductive procedure considered is that described in [10], [11]. Regarding the cost of the decision tree, we will consider, as an example, a decision tree with a minimum number of attributes (i.e. with a minimum number of inner nodes). This tree can be created by considering a same explicit cost for all the attributes, for example the unit cost. Figure 10 shows such a tree with four nodes.

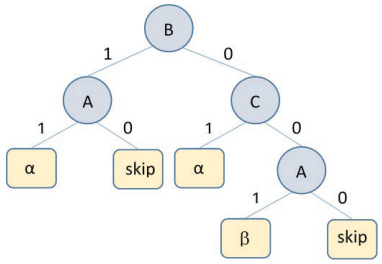

Fig. 10. Decision tree with a minimum number of nodes

CONCLUSIONS

Decision trees constitute a simple and highly efficient procedural decision structure. A formal four-stage method to turn declarative and procedural rule bases into a decision tree has been presented. The method is simple and provides some relevant advantages in relation to the methods found in literature. Among these benefits we stand out the following:

- It allows turning both, declarative and procedural rule bases into a decision tree.
- There is no need to distinguish between the declarative or procedural character of rule bases when applying the method.
- It strictly preserves the accuracy properties of the rule base.
- It strictly respects the priorities of evaluating the rules.
- It allows to consider the nature of the problem in hand (i.e. take into account

the cost restrictions on the tree demanded by the user).

Other important properties of the method are:

- It just depends on the properties of the rules and not on the set of training data from which the rules were induced.
- Its computational efficiency to generate a decision tree depends mostly on the cost criterion used. If no cost constraints are considered, then the method's efficiency is significantly reduced and it could be used to create on-demand (i.e. online) a decision tree.

It makes no sense to compare the proposed method with any of the methods found in literature, since these are designed only to deal with declarative rules bases; in addition, decision trees generated by these methods do not strictly respects the properties of the original rule base.

We have tested the described method by developing some examples of significant size. The main results are: all generated decision trees strictly respect the properties of the original rule-based knowledge bases; decision trees are also optimal regarding the cost criteria demanded by the user; in addition, the application of inductive procedures based on binarization techniques of multivalued attributes has allowed decision trees whose descriptive accuracy far exceeds that of the original rule base. Such results have not been described due to the extension of this work. Anyway, some of them will be in- cluded in the presentation.

REFERENCES

1. E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms,* Computer Science Press, USA, 1978.

2. R. S. Michalski et al, The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains, *Proceedings of AAAI-86* (1986), 1041-1045.

3. E. Bloedorn and R. S. Michalski, Data Driven Constructive Induction in AQ17-PRE: A Method and Experiments, *Proceedings of the Third International Conference on Tools for AI* (1991), 9-14.

4. G. Fiol et al, A New Perspective in the Inductive Acquisition of Knowledge from Examples,

Lecture Notes in Computer Science **682** (1993), 219–228.

5. G. Fiol, On Qualitative Knowledge in a Rule Based Knowledge Base, *Proceedings of the IJCAI-93 workshop on Validation, Verification and Test of KBSs* (1993), 27-36.

6. I.F. Iman and R.S. Michalski, Learning Decision Trees from Decision Rules: A Method and Initial Results from a Comparative Study, *Journal of Intelligent Information Systems,* **2** (1993), 279–304.

7. G. Fiol et al, Computer-Aided Causal Diagnosis of Ascites. Analysis of a Prototipe, *Infor- mation, Intelligence and Systems vol. 2*, (1996), 1102-1107.

8. R.S. Michalski and I.F. Iman, Learning problem-oriented Decision Structures from Decision Rules: the AQDT-2 System, *Lecture Notes in Artificial Intelligence* **869** (1994), 416-426.

9. G. Fiol, UIB-IK: A Computer System for Decision Trees Induction, *Lecture Notes in Artifi- cial Intelligence* **1609** (1999), 601-611.

10. G. Fiol, Inductive Learning from Incompletely Specified Examples, *Frontiers in Artificial Intelligence and Applications* **100** (2003), 286–295.

11. G. Fiol, Learning from Incompletely Specified Object Attribute Tables with Continuous At- tributes, *Frontiers in Artificial Intelligence and Applications* **113** (2004), 145–152.

12. S. Russell and P. Norvik, *Inteligencia Artificial. Un enfoque moderno, 2ª Edición*, Pearson Educación S.A., España, 2004.

13. G. Fiol et al, The Intelligent Butler: A Virtual Agent for Disabled and Elderly People Assis- tance, *Advances in Soft Computing vol. 50/2009* (2008), 375-384.

14. A. Abdelhalim and I. Traore, A New Method for Learning Decision Trees from Rules, *Pro- ceedings of the Eighth International Conference on Machine Learning and Applications* (2009), 693-698.

15. D. Poole and A. Mackworth, *Artificial Intelligence: Foundations of Computational Agents,* Cambridge University Press, Canada, 2010.

16. G. Fiol and M. Miró, Stock Market Analysis using Data Mining Techniques: a Practical Application, *International Journal of Artificial Intelligence vol. 6, num. S11* (2011), 129-143.

17. G. Fiol et al, Data Mining Techniques for Web Page Classification, *Advances in Intelligent and Soft Computing vol. 89* (2011), 61-68.