A Basic Proof Method for the Verification, Validation and Evaluation of Expert Systems

[1] Armin Ghasem Azar*
a.ghasemazar@iasbs.ac.ir
[2] Zohreh Mohammad Alizadeh
[1,2] Department of Computer and Information Sciences Institute for Advanced Studies in Basic Sciences (IASBS)
z.alizadeh@iasbs.ac.ir

Received: 06/Oct/2012 Accepted: 23/Feb/2013

Abstract
In the present paper, a basic proof method is provided for representing the verification, Validation and evaluation of expert systems. The result provides an overview of the basic method for formal proof such as: partition larger systems into small systems prove correctness on small systems by non-recursive means, prove that the correctness of all subsystems implies the correctness of the entire system.

Keywords: Expert System, Partition, Non-Recursive.

1. Introduction

An expert system is correct when it is complete, consistent, and satisfies the requirements that express expert knowledge about how the system should behave.

For real-world knowledge bases containing hundreds of rules, however, these aspects of correctness are hard to establish. There may be millions of distinct computational paths through an expert system, and each must be dealt with through testing or formal proof to establish correctness.

To reduce the size of the tests and proofs, one useful approach for some knowledge bases is to partition them into two or more interrelated knowledge bases. In this way the VV&E problem can be minimized [1].

2. Overview the Proofs Using Partitions

The basic method of proving each of these aspects of correctness is basically the same. If the system is small, a technique designed for proving correctness of small systems should be used. If the system is large, a technique for partitioning the expert system must be applied and the required conditions for applying the partition to the system as a whole should be proven. In addition the correctness of any subsystem required by the partition must be ensured. Once this has been accomplished this basic proof method should be applied recursively to the sub-expert systems. Once the top level structure of the Knowledge base has been validated, to show the correctness of the expert system, the following criteria must be accomplished [6]:

- Show that the Knowledge base and inference engine implement the top level structure;
- Prove any required relationships among sub-expert systems or parts of the top level Knowledge representation;
- Prove any required properties of the sub-Knowledge bases.

2.1 A Simple Example

To illustrate the basic proof method, *Knowledge Base* 1 will be proved correct in Table 1 and although this Knowledge base is small enough to verify by inspection.

2.1.1 Illustrations of Knowledge Base 1

The *Knowledge Base* 1 (KB1) has six rules. There are seven variables which can take two possible values. It is, therefore a seven dimensional, binary problem [5]. Let's focus on Rule 3 to understand the illustrations of KB1.

It has two hypotheses, and one conclusion. The hypotheses are “Do you buy lottery tickets?”=”yes”, and “Do you currently own

* Corresponding Author

stock"="yes". They are associated with the logical operator "or". The consequent is Risk Tolerance"="high". This is illustrated in Figure 1. For the two variables of the hypotheses in Rule 3, there are two possible values: "yes" or "no". The number of possible combinations of values for the variables is four. These four combinations appear in Figure 1 as four square regions defined by the closed boundary (defining the domain or the variables) and the line boundaries separating the possible values for each variable. Each square is a Hoffman region.

Rule 1	If "Risk tolerance" = high AND "Discretionary income exists"= yes then investment = stocks.
Rule 2	If "Risk tolerance" = low OR "Discretionary income exists" = no then investment = "bank account".
Rule 3	If "Do you buy lottery tickets" = yes OR "Do you currently own stocks" = yes then "Risk tolerance" = high.
Rule 4	If "Do you buy lottery tickets" = no AND "Do you currently own stocks" = no then "Risk tolerance"= low.
Rule 5	If "Do you own a boa" = yes OR "Do you own a luxury car" = yes then "Discretionary income exists" = yes.
Rule 6	If "Do you own a boat" = no AND "Do you own a luxury car" = no then "Discretionary income exists" = no.

Table 1: Knowledge Base 1 [7]

If variable "Do you buy lottery tickets" is assigned a value "yes", then two of the four regions are relevant. In Figure 1.a, they are shown with a hatch. The two regions corresponding to hypotheses "Do you currently own stock?"="yes" are hatched in Figure 1.b.

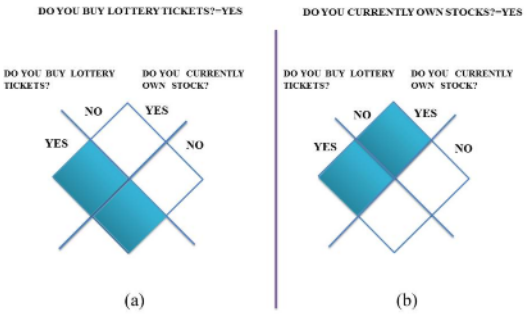

Fig. 1: Knowledge Base 1 [7]

In two dimensions, a Hoffman region is a surface as shown in this example. In three dimensions, it would be a volume.

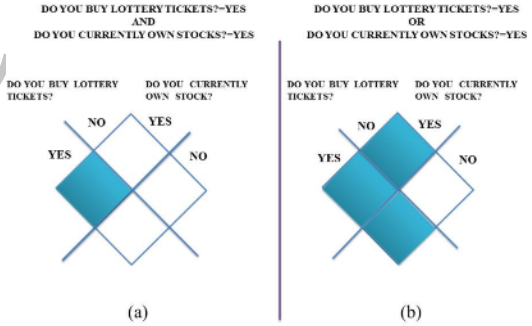

Fig. 2: Knowledge Base 1 [7]

The logical operators are "and", "or" and "not". In Figure 1.a and 1.b, the Hoffman regions corresponding to hypothesis of Rule 3 are hatched. When combined with an "and" logical operator, intersection of the two sets of Hoffman regions. This is shown in Figure 2.a.

The intersection in this case is a unique Hoffman region. In Rule 3, an "or" operator connects the two hypotheses.

In this case, the union two sets of Hoffman regions are taken, as shown in Figure 2.b.

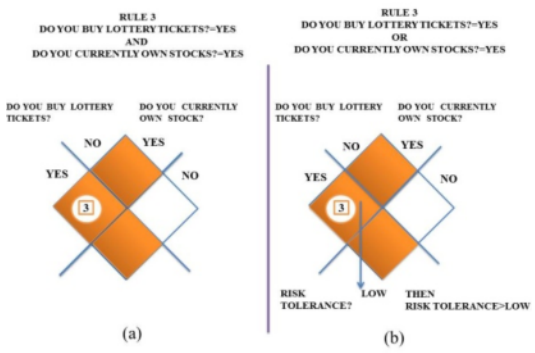

Fig. 3: Knowledge Base 1 [7]

Next, the region by the logical expression of the hypotheses is labeled with its rule. For Rule 3, the three Hoffman regions are labeled with a circled 3 as shown in Figure 3.a. Consequence for the Rule is linked to the label of the region of the hypotheses. In Figure 3.b, an arrow starts at the circled 3 and ends at the value "low" of the variable "Risk".

2.2 Step 1-Determine Knowledge Base Structure

To prove the correctness of Knowledge Base 1 (KB1), the expert Knowledge can determine that the system represents a 2-step process [3]:

- Find the values of some important intermediate variables, such as risk tolerance and discretionary income;
- Use these values to assign a type of investment.

KB1 was built using this Knowledge; therefore, it can be partitioned into the following pieces:

- A subsystem to find risk tolerance (Part of Step 1);
- A subsystem to find discretionary income (Part of Step 1);
- A subsystem to find type of investment given this Information (Part of Step 2).

2.3 Step 2-Find Knowledge Base Partition

To find each of the three subsystems of KB1, an iterative procedure can be followed:

- Start with the variables that are goals for the subsystem, e.g., risk tolerance for the risk tolerance subsystem;
- Include all the rules that set subsystem variables in their conclusions. For the risk tolerance subsystem, Rules 3 and 4 are included;
- Include all variables that appeared in rules already in the subsystem and are not goals of another subsystem;
- For the risk tolerance subsystem, include "Do you buy lottery tickets" and "Do you currently own stocks";
- Quit if all rules setting subsystem variables are in the subsystem, or else go to Step 2. For the risk tolerance subsystem, there are no more rules to be added.

Figure 4 below shows the partitioning of KB1 using this method.

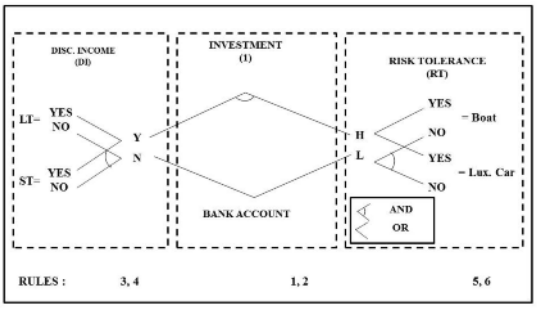

Fig. 4: Knowledge Base 1 [3]

2.4 Step 3-Completeness of expert systems

2.4.1 Completeness Step 1-Completeness of Subsystems

The first step in proving the completeness of the entire expert system is to prove the completeness of each subsystem. To this end it must be shown that for all possible inputs there is an output, i.e., the goal variables of the subsystem are set. This can be done by showing that the OR of the hypotheses of the rules that assign to a goal variable is true [7].

2.4.2 Completeness Step 2-Completeness of the entire system

The results of subsystem completeness are used to establish the completeness of the entire system. The basic argument is to use results on subsystems to prove that successively larger subsystems are complete. At each stage of the proof there are some subsystems known to be complete; initially the subsystem that concludes overall goals of the expert system will be complete. At each stage of the proof, a subsystem that concludes some of the input

variables of the currently-proved-complete subsystem is added to the currently complete subsystem. After a number of steps equal to the number of subsystems, the entire system can be shown to be complete.

2.5 Step 4-Consistency of the entire system

The first step in proving the consistency of the entire expert system is to prove the consistency of each sub- system. To do this, the user must show that for all possible inputs, the outputs are consistent, i.e., that the AND of the conclusions can be satisfied.

For example, if an expert system concludes:

"temperature >0" and "temperature <100"

The AND of these conclusions can be satisfied. However, if the system concludes:

"temperature <0" and "temperature>100"

The AND of these two conclusions has to be false. It is clear that based on the input that produced these two conclusions, it is not possible for all of the system's conclusions to be true at the same time and thus the system producing these conclusions is inconsistent.

2.5.1 Consistency Step 1-Find the mutually inconsistent conclusions

The first step in proving consistency is to identify those sets of mutually inconsistent conclusions for each of the subsystems identified in the "Find partitions" step above. Some sets of conclusions are mathematically inconsistent [2]. For example, if a system describes temperature, the set: "temperature <0", "temperature >100" is mathematically inconsistent.

Because some sets of conclusions are inconsistent because of domain expertise, finding all sets of inconsistent conclusions generally requires expert Knowledge.

Note that if there are no mutually inconsistent conclusions in the expert system as a whole, then consistency is true by default, and no further consistency proof is necessary.

2.5.2 Consistency Step 2-Prove consistency of subsystems

If there are inconsistent conclusions in the Knowledge base as a whole, then the next step in proving consistency is to prove the subsystems consistent. This can be done by showing that no set of inputs to a subsystem can result in any of the sets of inconsistent conclusions.

2.5.3 Consistency Step 3-Consistency of entire system

The results of subsystem consistency are used to establish the consistency of the entire system. The basic argument is to use results on subsystems to prove that successively larger subsystems are consistent. At each stage of the proof, there are some subsystem known to be consistent; initially, this is the subsystem that concludes goals of the expert system as a whole. At each stage of the proof, a subsystem that concludes some of the input variables of the currently-proved-consistent subsystem is added to the currently consistent subsystem. After a number of steps equal to the number of subsystems, the entire system can be shown to be consistent [2].

2.6 Step 5-Specification satisfaction

In order to prove that KB1 satisfies its specifications, the user must actually know what its specifications are. This is a special case of the general truth that in order to verify and validate, the user must know what a system is supposed to do. Specifications should be defined in the planning stage of an expert system project [4].

To illustrate the proof of specifications it will be assumed that KB1 is supposed to satisfy:

A financial advisor should only recommend investments that an investor can afford.

As with many other aspects of verification and validation, expert Knowledge must be brought to bear on the proof process. For KB1, an expert might say that anyone can afford a savings account. Therefore, the user only has to look at the conditions under which stocks are recommended. However, that same expert would probably say that just having discretionary income does not mean that the user can afford stocks; that judgment should be made on more than one variable. Therefore, it would be reasonable to conclude that KB1 does not satisfy the above specification.

3. Conclusion

This paper has argued that V&V techniques are an essential part of the Knowledge engineering process, because they offer the only way to judge the success (or otherwise) of a KBS development project. This is equally true in the context of Knowledge management, where V&V techniques tell us whether or not the KBS can be

relied upon to accurately embody the Knowledge of the human experts that supplied it.

However, examination of known studies on the effectiveness of existing KBS VV&E techniques has shown that the state of Knowledge in this area is sparse. The way to improve this situation would be by systematically gathering data from a representative set of KBS projects and V&V techniques. Without such a study, Knowledge engineering will remain very much an art and, by extension, so will the use of KBS technology in Knowledge management.

It is difficult to generalize our results to all Knowledge based systems and, of course, further evaluations of other applications are necessary to confirm (or challenge) our conclusions. However, since the method we have used minimizes the need for experts' interpretation of the faults, we can reasonably conclude that if we use an application of similar size and complexity to GIBUS, we would expect to obtain similar results. Consequently, since our application has a size and a complexity which is representative of actual practice, we would expect that consistency and completeness checking, in addition to testing, would be an effective combination of methods to validate many of the Knowledge based systems actually under development.

References

[1] Ayel M and Laurent J-P, two different ways of verifying Knowledge-based systems, Validation, Verification and Test Of Knowledge-Based Systems, Wiley, New York, Year. 1991, pp. 63-76.

[2] Bendou A, A constraint-based test data generator, EUROVAV-95, Saint Badolph, France, Year. 1995, pp. 19-29.

[3] Ginsberg A, Knowledge-based reduction: A new approach to checking Knowledge bases for inconsistency & redundancy, AAAI Vol. 88, No. 2, Year. 1988, pp. 585-589.

[4] Kirani S, Zualkernan I.A, and Tsai W.T., Comparative Evaluation of Expert System Testing Methods, Computer Science Department, University of Minnesota, Minneapolis Vol. 2, Year. 1992, pp. 92-30.

[5] Laurent J-P, Proposals for a valid terminology in KBS validation, ECAI-92, Wiley, New York, Vol. 2, Year. 1992, pp. 829-834.

[6] Lounis R and Ayel M, Completeness of KBS, EUROVAV-95, Saint Badolph, France, Vol. 2, Year. 1995, pp. 31-46.

[7] O'Leary D, Design, development and validation of expert systems: A survey of developers, Vol. 2, Year. 1991.