Identifying and Handling Structural Incompleteness for Validation of Probabilistic Knowledge-Bases

Eugene Santos Jr.
Dept. of Comp. Sci. & Eng.
University of Connecticut
Storrs, CT 06269-3155
eugene@cse.uconn.edu

Sheila B. Banks
Calculated Insight
Orlando, FL 32878-0214
sbanks@calculated-insight.com

Scott M. Brown **David J. Bawcom**
Dept. of Elec. & Comp. Eng.
Air Force Institute of Technology
WPAFB, OH 45433

Abstract

The PESKI (Probabilities, Expert Systems, Knowledge, and Inference) system attempts to address some of the problems in expert system design through the use of the Bayesian Knowledge Base (BKB) representation. Knowledge gathered from a domain expert is placed into this framework and inferencing under uncertainty is performed over it. However, by the nature of BKBs, not all knowledge is incorporated, i.e. the representation need not be a complete representation of all combinations and possibilities of the knowledge, as this would be impractical in many real-world systems. Therefore, inherent in such a system is the problem of incomplete knowledge, or gaps within the knowledge base where areas of lacking knowledge preclude or hinder arrival at a solution. Some of this knowledge is intentionally omitted because its not needed for inferencing, while other knowledge is erroneously omitted but necessary for valid results. Intentional omission, a strength of the BKB representation, allows for capturing only the relevant portions of knowledge critical to modeling an expert's knowledge within a domain. This research proposes a method for handling the latter form of incompleteness administered through a graphical interface. The goal is to detect incompleteness and be corrected by a knowledge engineer in an intuitive fashion.

Introduction

Many issues need to be considered when constructing a complete knowledge based system. First, in the knowledge acquisition phase of a system, knowledge engineers must thoroughly extract knowledge from an expert. Methods of extracting this knowledge are numerous. A representation scheme for this knowledge must then be chosen. The knowledge engineer must carefully build this knowledge representation into an expert system for which it can be inferenced over. There are many opportunities for inputting incorrect, incomplete, or inconcise information while building a new system. Often many modifications to the knowledge base are necessary in existing systems as well, which can often adversely affect other areas unintentionally. For these reasons among others, verification and validation (V & V) of these knowledge based systems is an increasingly important part of today's sophisticated knowledge based systems.

A great amount of research has been performed in the area of V & V over the past few years (Ram & Ram 1996; Preece 1995; Bass, Ernst-Fortin, & Small 1997; O'Leary 1997). However, with the large number of knowledge representations, inferencing techniques, and knowledge acquisition techniques, there is no common consensus on the best method of V & V. This research will center on one particular representation scheme known as Bayesian Knowledge Bases (BKBs) which captures uncertainty and its associated issues. This BKB representation scheme is part of an overall expert system shell known as Probabilities, Expert System, Knowledge, and Inference (PESKI), which is an integrated framework for expert system development (Santos, Banks, & Banks 1997; Santos, Gleason, & Banks 1997).

This research focuses on developing a methodology to correct one problematic area of V & V, namely unintentional incompleteness that may be present in the knowledge base. The incompleteness is recognized in the validation phase, and a tool for correcting this lack of knowledge is introduced. The results of this work are currently integrated into the PESKI system. Test cases are the instrument used for validating BKBs in PESKI. These test cases are submitted to the system and its results are compared to expected results. Incompleteness occurs when the inferencing cannot reach an expected solution as defined by a test case. This incompleteness can come from several different sources and are investigated in this paper. After identifying that incompleteness does exist in the BKB, the graphical incompleteness tool assists the knowledge engineer in locating the area of incompleteness and then extracts the missing information from him/her for insertion into the knowledge base.

Verification and Validation

The difficulties in the development of knowledge based systems, particularly with knowledge representation and knowledge acquisition, often leads to errors in several forms. One of these types of errors is incompleteness. This research stems from a need to handle incompleteness during the validation stage of development.

While the main objectives of V & V are closely knitted together, it is important to understand the distinct differences between them. Verification is best defined as making sure the system is built correctly. Critical to this step is ensuring all information deemed necessary is included and that this information is interpreted and applied correctly by the system being inspected. If specifications exist for a particular system, verification will check for compliance with these specifications. It also oversees the correct software syntax from which it was built. Verification is often referred to as clear-box testing.

Validation, on the other hand, is used to ensure the output of the system is correct. It is also used to check the system developed is what the users requested. It must assume the knowledge base was built satisfactorily. Typically, expert system validation consists of running a sequence of test cases through the system and comparing system results against known results or expert opinions (O'Keefe, Balci, & Smith 1987). This is a time-consuming process and never guarantees finding all errors, especially in larger systems. O'Keefe et al. stated "Validation can be considered the cornerstone of evaluation (of an expert system), since highly efficient implementations of invalid systems are useless" (O'Keefe, Balci, & Smith 1987). Validation is often referred to as black-box testing. Concern is placed not upon what is inside the system, but what the results are coming out of the system. Despite the importance of validation, the majority of V & V literature is solely concerned with verification, specifically automatic rule-based error checking. This aspect of V & V has now become reasonably mature and many such automated tools exist (Meseguer & Verdaguer 1993; Prakash & Mahabala 1993). This automation is often built into the system so that verification is continually addressed throughout knowledge base construction to ensure a quality final product.

Testing, including validation, is best done throughout the entire development of the knowledge base. Incremental testing can aid in finding inaccuracies or incompleteness early in the development of the system rather than later when corrections can be much more difficult to detect, locate, and correct. In determining the overall validity of a system, it is often beneficial to determine how well human experts do in the problem area and to create reasonable expectations of the systems performance. Typically, expert systems and their knowledge bases' performance can change drastically from initial release to later stages of use. Some systems can be field tested and validated in its early use without harm. In critical applications where lives may be at risk, field testing is not always possible. The expert whose knowledge was modeled should maintain involvement throughout development of the system whenever possible. This can often assist in identifying errors early on in the development cycle that may not have been detected until later stages of validation.

Validating after modifications or enhancements have been implemented is just as important as earlier testing. Testing needs to ensure that the original system was not degraded as well as that the modifications made were correctly implemented. Comparison of previous test case results and their performance after the modifications is an effective way of testing the updated system remains validated in areas both inside and outside of the modified areas. This type of testing can be particularly important in probabilistic representations, since chains of inference can be unintentionally altered.

Bayesian Knowledge-Bases

BKBs are a new, powerful, and highly flexible knowledge representation (Santos & Santos 1999). BKBs are closely related to Bayesian networks and in fact subsume them. BKBs, just as Bayesian networks, are strongly based upon probability theory. This foundation allows a framework for enabling inferencing over incomplete knowledge. In contrast to BKBs, Bayesian networks demand for a complete specification of probability distributions can make knowledge acquisition, knowledge base creation, and inferencing quite difficult and cumbersome. BKBs avoid an over-defined system easing maintainability, verification, and validation. They are more powerful from the fact that they are specifically designed for allowing incompleteness. However, when desired conclusions are unable to be drawn from the knowledge base given the appropriate evidence, this incompleteness needs to be corrected through incorporation into the knowledge base.

In the BKB representation, as in Bayesian networks, random variables (RVs) are used to represent objects and/or events in the world. These RVs are then instantiated with state values and are used in combination with one another to model the current state of the world. Inferencing over this knowledge representation then involves computing the joint probabilities of these RVs. This type of inferencing is known as belief revision. Belief revision is useful in diagnostic domains where an explanation of the most probably state of the world is needed.

BKBs are built through the combination of instantiation nodes, support nodes, and arcs. An example BKB is shown in Figure 0.1. Instantiation nodes, or I-nodes for short, are represented by an oval. An I-node represents one instance of an RV. The arcs represent the relationships between these I-nodes. Support nodes, or S-nodes, are represented by smaller rectangles or circles. S-nodes are assigned probabilities that are associated with one or more I-nodes. In Figure 0.1, I-node **Clouds = Heavy** is supported by a single S-node with a probability of 0.1500. I-node **Sidewalk = Wet** is supported by the single I-node **Clouds = Heavy** through an S-node probability of 0.8500. In order for the S-node to be active, the supporting I-node, in this case **Clouds = Heavy**, must be active.

Some constraints of this BKB representation include the following:

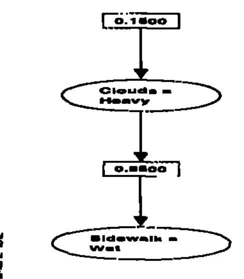

FIG. 0.1. Example BKB.

- Support conditions for an RV instantiation must be mutually exclusive. Only one S-node may be active at any one time in the support of an I-node.
- Cyclic knowledge is not allowed. A node is not allowed to support itself.
- Probabilities from the same RV may not sum to values greater than 1.

For a complete discussion of these constraints see Banks (Santos, Banks, & Banks 1997; Santos, Gleason, & Banks 1997).

Incorrectness and Incompleteness

Imperfect information is ubiquitous - almost all the information that we have about the real world is not certain, complete, or precise (Parsons 1996). It is critical for the knowledge engineer to understand that these conditions exist during creation of a knowledge base. Three concepts that are essential for V & V of BKBs are inconsistency, incompleteness, and incorrectness (Santos, Banks, & Banks 1997; Santos, Gleason, & Banks 1997). Inconsistency in a BKB is primarily related to probabilistic values. For example, conditional probabilities summing to greater than one. These types of errors are often discovered and corrected within the knowledge acquisition process. When this form of inconsistent knowledge is introduced into the PESKI system, the knowledge engineer is immediately informed through a continuously updated status window. This process assures that knowledge is consistent throughout the entire knowledge building process.

Incorrectness occurs when a query to the system results in an incorrect solution. Finding the location of the error can be difficult, and correcting it even harder. This aspect of validation will continue to be addressed through a variety of approaches such as sensitivity analysis and neural network reinforcement learning techniques (Santos, Gleason, & Banks 1997).

Incompleteness exists when a set of input values is passed to the system and fails to arrive at a conclusion. This type of omission can be very difficult to detect and locate as well. Knowledge base incompleteness can be both intentional, particularly in the case of BKBs, or unintentional as in an oversight. Incompleteness can come from several different sources. Human error is often the major source for this type of error. Experts often have difficulties in conveying complete heuristic knowledge to the knowledge engineer. This lack of information often leads to incompleteness in the knowledge base. Often information is missing during development of the knowledge base and is left out for future modifications. Other types of knowledge are yet to be discovered, particularly in some areas such as medicine, where new types of drugs and medications are constantly under development. For these and other reasons, the ability to handle incompleteness is critical in the validation of these systems. This ability is even more critical in a representation like BKBs, in which the ability to incorporate incompleteness is an advantage and a normally desired quality of the representation.

Incompleteness can be identified in a BKB through the direct dependency region (formally defined in next section) of an evidence item from a test case. This region must be modified through the addition of a link or links to the corresponding answer item or it's direct dependency region in the BKB. This addition of a link must be done in a manner that places the answer item in the direct dependency region of the evidence. Only then is the incompleteness dissipated. The methodology developed through this research graphically presents the BKB in a manner suitable for a knowledge engineer to locate this area of incompleteness. The incompleteness link can then be added to the BKB for future inferencing. This link is added by the tool in a manner that forces maintaining the rules of the BKB representation.

Methodology

We now describe a methodology to handle incompleteness caused by missing links in a Bayesian knowledge base (BKB). The incompleteness links are identified and located in the BKB using test cases. When test cases lack a direct dependency connection from the evidence and answer items, we can correct the incompleteness through a graph-based approach. This graphical presentation of the BKB gives the knowledge engineer a means of locating and correcting the incompleteness found in the test case. Some other desired traits of this methodology include the following:

- Maintain the structure of the previous information contained in the BKB. An assumption is made that information not addressed by the current test case in the BKB is correct.
- Disallow the knowledge engineer from inputting information into the BKB that violates the knowledge representation rules (e.g. circularity, mutual exclusion, etc) (Santos, Banks, & Banks 1997).
- Inherent with any large knowledge base is the problem of finding where the incompleteness exists in order to fix the problem. The tool should avoid presenting too much information that overwhelms the knowledge engineer, particularly in larger knowledge bases.

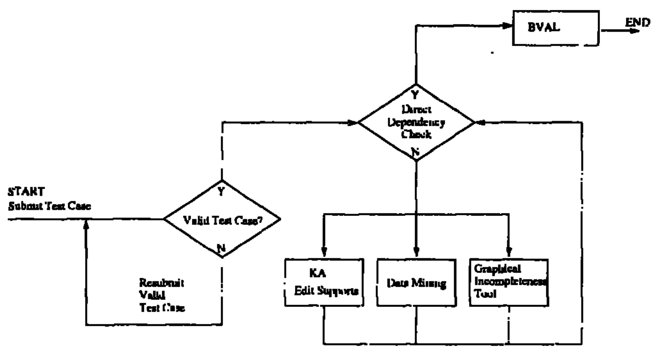

FIG. 0.2. Test case processing.

A test-case consists of two set of distinct RV assignments, A and E. The first set, A, consists of the answers while E consists of the evidences. These test cases are submitted to the system and the results are compared to expected results. The test cases are used to validate the BKB through a number of steps using the PESKI system. Figure 0.2 depicts the flow of a test case through PESKI. The PESKI system validation tools place some assumptions upon the supplied test cases. The test cases used in knowledge base validation are constrained in that they are assumed correct in their entirety. The knowledge engineer's burden is to ensure that each test case is completely valid. In addition, it is important to understand that each evidence item is directly related to each answer item, and vice versa. If this is not so, the non-contributing evidence or answer item should not be a part of the "valid" test case. Since all answers are in each evidence item's dependency region, the intersection of all the evidence dependency regions should be non-empty and contain, at a minimum, all answer instances.

Direct dependency regions are the key to validation efforts in the PESKI environment and are formally defined below. RV instances directly dependent upon one another are connected by a sequence of parent or child relationships. Therefore, I-node A is directly dependent upon I-node B if there is a sequence of parent nodes, or a sequence of child nodes, between A and B that connect the two nodes. Figure 0.3 shows the direct dependency region for the evidence item **D** = **1**. Formally,

DEFINITION 0.1. *A random variable instance A is* **directly dependent** *on a random variable instance B, if and only if there exists a sequence of n random variable instances $\{X_1 \equiv A, X_2, ..., X_{n-1}, X_n \equiv B\}$, where n is positive integer, and*

1. *Each element X_i in the sequence of random variable instances is an element of a support condition of X_{i-1}, for all i, $2 \leq i \leq n$, or*
2. *Each element X_i in the sequence of random variable instances is an element of a support condition of X_{i+1}, for all i, $1 \leq i \leq n-1$.*

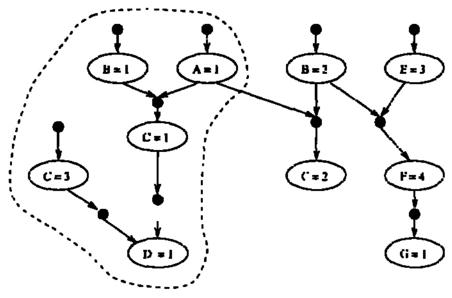

FIG. 0.3. Direct dependency region.

After validating a test case, the next step is to check for direct dependency region connections between the evidence and answer items. If the evidence and answer(s) are found to be both contained in the same direct dependency region, in the case that no incompleteness exists, the BKB is then passed to a tool for probabilistic validation (Santos, Gleason, & Banks 1997).

If the direct dependency region check fails, incompleteness exists in the test case and needs to be corrected in the knowledge base structure.

Graphical Incompleteness Tool (GIT)

A visual interactive approach allows the knowledge engineer to examine the knowledge base for completeness as well as accuracy. The tool actively assists in the generation of possible solutions from which the engineer must select a preferred choice. This type of correction also avoids any non-sensical modifications and/or assumptions that an automatic validation tool may make. The ability to view the relevant portions of the knowledge base where problems are occurring is key. The incompleteness methodology extends the graphical presentation of the BKB into a display for allowing incompleteness correction in an intuitive fashion.

We have defined incompleteness in three ways: *missing links*, *missing RVs*, and *missing states*. Incompleteness caused by missing RVs or states can be corrected through data mining or through the normal knowledge acquisition modes of PESKI. Of concern in the remainder of this research is unintentional incompleteness caused by missing links or relationships between I-nodes. After the test case direct dependency region check has discovered this type of incompleteness, the methodology begins by presenting the BKB in a graphical format that allows for the location of the incompleteness. This is further aided by distinctly displaying the direct dependency regions of the evidence and answer items from the incomplete test case in the BKB. The ability to traverse anywhere in the BKB is also allowed to assist the knowledge engineer by allowing him/her to search for pertinent information. The incompleteness link should be allowed irregardless of the location of the nodes within the BKB, as long as the BKB representation constraints are not violated. The addition of a link can cause constraints to be violated if care is not taken to avoid these situations. Some links between nodes cannot be allowed at all, while others must be specially handled.

This methodology contains two modes of incompleteness correction - an add (link) mode as well as an insert (S-node) mode. These modes allow for correction of incompleteness in several forms.

After locating the source node(s) of incompleteness, the knowledge engineer is allowed to select an I-node for correction of the incompleteness. The I-node must be either the evidence or answer node or a direct dependency descendant of either the evidence or answer node. The location of the target S-node is dependent upon the currently active mode. In add mode, the target node must be a direct dependency ancestor of the corresponding evidence or answer node. In insert mode, the target node may be any node, either descendant or ancestor, in the direct dependency region of the corresponding evidence or answer node. These restrictions ensure that the directed link will be placed in the correct cause/effect direction and will also resolve the current incompleteness problem. Selection of the source I-node and target S-node in any other manner will not correct the incompleteness due to the direct dependency relationships between the nodes.

After extending the I-node, these allowed target S-nodes are clearly identified to the knowledge engineer. These allowed target S-nodes will guarantee the correction of the incompleteness and avoid any BKB constraint violations.

Conclusion

With the understanding of the functionality of the graphical incompleteness tool, the question arises whether or not this functionality allows for any addition of an incompleteness link between two nodes' direct dependency regions. For preliminary testing purposes, the functionality of the add and insert modes together have been used to recreate a highly connected BKB starting with only non-connected I-nodes. All links in this BKB were removed so that only the required single S-node existed above each of the I-nodes. The graphical incompleteness tool was then able to recreate the BKB links after receiving the necessary number of test cases. The order the test cases were given to PESKI determined the amount of insertions versus additions that were necessary. Since this BKB was able to be totally constructed using only the graphical incompleteness tool, this demonstrates that any one incompleteness link may be handled by the graphical incompleteness tool for even a highly connected BKB.

This work was supported in part by AFOSR Project #940006.

References

Bass, E. J.; Ernst-Fortin, S. T.; and Small, R. L. 1997. Knowledge base development tool requirements for an intelligent monitoring aid. In *Proceedings of the Tenth International Florida Artificial Intelligence Research Symposium*, 412 416.

Meseguer, P., and Verdaguer, A. 1993. Verification of multi-level rule-based expert systems: Theory and practice. *International Journal of Expert Systems* 6:163 192.

O'Keefe, R. M.; Balci, O.; and Smith, E. P. 1987. Validating expert system performance. *IEEE Expert* 2(4):81 90.

O'Leary, D. 1997. Validation of blackboard systems: On the order of knowledge sources. In *Proceedings of the AAAI Workshop on Verification & Validation of Knowledge-Based Systems*, 47–52.

Parsons, S. 1996. Current approaches to handling imperfect information in data and knowledge bases. *IEEE Transactions on Knowledge and Data Engineering* 8:353–372.

Prakash, G. R., and Mahabala, H. 1993. Svepoa: A tool to aid verification and validation of ops5-based ai applications. *International Journal of Expert Systems* 6:193–236.

Preece, A. D. 1995. Validation of knowledge-based systems: Current trends and issues. *The Knowledge Engineering Review* 10(1):69–71.

Ram, S., and Ram, S. 1996. Design and validation of a knowledge-based system for screening product innovations. *IEEE Transactions on Systems, Man, and Cybernetics (Part A)* 26(2):213 221.

Santos, Jr., E., and Santos, E. S. 1999. A framework for building knowledge-bases under uncertainty. *Journal of Experimental and Theoretical Artificial Intelligence*.

Santos, Jr., E.; Banks, D. O.; and Banks, S. B. 1997. Mack: A tool for acquiring consistent knowledge under uncertainty. In *Proceedings of the AAAI Workshop on Verification and Validation of Knowledge-Based Systems*, 23 32.

Santos, Jr., E.; Gleason, H. T.; and Banks, S. B. 1997. Bval: Probabilistic knowledge-base validation. In *Proceedings of the AAAI Workshop on Verification and Validation of Knowledge-Based Systems*, 13 22.