Generalizing Knowledge Representation Rules for Acquiring and Validating Uncertain Knowledge

Gregory Johnson Jr. and Eugene Santos Jr.
Department of Computer Science and Engineering
University of Connecticut
Storrs, CT 06040-3155
gjohnson@engr.uconn.edu
eugene@engr.uconn.edu

Abstract
In response to the difficulties traditionally experienced in knowledge acquisition for knowledge-based systems, we wish to allow domain experts to work directly with the knowledge-base in a more intuitive manner. However, we also wish to represent uncertainty in our knowledge. Unfortunately, knowledge representations which handle uncertainty are counterintuitive and difficult to use. To achieve these dual goals, we introduce a layer of abstraction above the underlying knowledge representation. Probabilistic formalisms are preferred because they provide a means for modeling uncertainty in the rules by weighing our confidence in a conclusion with a probability value. The abstraction allows the expert user to specify knowledge in familiar if-then style rules in forms which appear often and are difficult to construct with the existing knowledge representation rules. For example, we may desire to weigh our confidence in a conclusion based on the number of pieces of evidence found in agreement with that conclusion. Given only the evidence and a conclusion from the user, we wish for the system to construct the corresponding rule in the knowledge representation automatically, leaving the implementation details hidden from the domain expert. With these rules built automatically by the system, the user can concentrate on expressing the domain knowledge rather than the structure and consistency of the knowledge-base.

Introduction
One of the most difficult, time consuming and expensive tasks in constructing knowledge-based systems remains in the knowledge engineering phase of system development (Santos, Banks, & Banks 1997). For example, the knowledge-base of the Guardian medical diagnostic system took the work of four to five people over approximately three years to build. It is still admittedly incomplete (Larsson & Hayes-Roth 1998). One approach to addressing this problem recognizes that the domain expert has a clearer picture of the knowledge to be represented and allows the expert to work directly with the knowledge base. Removing the knowledge engineer from the process reduces the time and cost of knowledge acquisition while eliminating errors in translation between domain expert and knowledge engineer. This approach has been implemented successfully in MACK, the knowledge acquisition tool of the PESKI expert system shell (Santos, Banks, & Banks 1997; Santos et al. 1999), as well as the Protégé and EXCEPT tools of other systems (Swartout 1996). PESKI uses the Bayesian Knowledge Base (BKB) knowledge representation which extends the familiar Bayesian Network (Pearl 1988) representation. The BKB representation allows domain experts to interact with the knowledge-base in familiar if-then style rules (Santos, Banks, & Banks 1997). However, considering the uncertainty prevalent in real-world domains and exceptions to the rule still require building complicated knowledge structures in which the number of rules required quickly explodes and the intuition gets lost (Santos, Gleason, & Banks 1997). In addition, domain experts have traditionally been responsible for validation of the knowledge base with little assistance offered by any tools within the system. This task is usually done by inspection of the knowledge-base and becomes increasingly difficult as the knowledge being validated becomes more complex (Santos, Gleason, & Banks 1997). By offering more intuitive rules in the knowledge representation and hiding implementation details, our goal is to allow the expert user to interact with the knowledge-base more independently. We wish to offer several different ‘generalized’ rules to the user which can express common relationships among evidence for a conclusion and can be constructed algorithmically. Implementing the rules by algorithms allows the system to hide much of the structure necessary to guarantee the internal consistency of the knowledge base. Hiding the implementation details from the expert user allows the expert to concentrate on knowledge acquisition and validation. These generalized rules are introduced as an extension to the BKB representation and represent knowledge in a probabilistic, consistent and efficient manner that remains intuitive to the expert user. This approach facilitates knowledge acquisition as well as validation while limiting the complexity of the knowledge base, thereby keeping reasoning tasks computationally tractable. The most novel and interesting of these generalized rules implements a rule of the form ‘If at least 3 of A, B, C or D are true then Z is true’ which is surprisingly difficult to represent in BKBs. Not only does this type of rule allow the expert user to rate his confidence in a conclusion based on the number of random variable instantiations found to be true during inferencing, the construct subsumes additional knowledge rules. By implementing such a construct, domain experts can also express rules in the
BKB of the form 'If A = true OR B = true then C = true' as well as XOR, NAND, and NOR rules in the familiar 'if-then' format. Such a construct lends flexibility to the BKB representation which allows the expert user to express his knowledge in the form which he understands. Enabling the user to present knowledge in generalized rules which are more intuitive to use for the domain expert continues the philosophy embodied in the MACK tool of enabling the domain expert to interact directly with the knowledge-base.

**Bayesian Knowledge Base**

Bayesian Knowledge Bases (BKBs) are an extension of Bayesian Networks (BNs) (Pearl 1988; Jensen 1996) which accommodate incompleteness, certain types of cyclicity in the represented knowledge as well as non-causal information (Santos & Santos 1999). BKBs, like BNs, offer a directed graph representation of knowledge. BKB graphs, however, consist of two distinct types of nodes. Instantiation nodes (I-nodes) represent specific instantiations of a random variable (r.v.). Support nodes (S-nodes) represent conditional dependence relationships (rules) between I-nodes. S-nodes in a BKB have at most one head (sink) condition, the conclusion. S-nodes may have several tail conditions representing the states supporting a conclusion. S-nodes with more than one tail condition specify a logical AND relationship among the evidence included in the tail condition. Figure (0.1) shows a sample BKB excerpt representing the rule 'If hives are present and itching is present then it an allergic reaction is present with probability 0.75.' In contrast to BNs, by representing conditional dependence relationships individually, we avoid having to specify conditions which may be unknown, irrelevant or impossible. By satisfying the following eight specific constraints, the knowledge in a BKB is guaranteed to be probabilistic and internally consistent (Santos, Gleason, & Banks 1997):

- **Constraint 1** - All instantiation nodes must be the head of at least one support node.
- **Constraint 2** - The sources and sinks of a node must be well-defined.
- **Constraint 3** - Each instantiation node represents a unique instantiation of a random variable.
- **Constraint 4** - Any support nodes that share a head instantiation must be mutually exclusive.
- **Constraint 5** - Given any inference chain, a support node's head must not occur in the tail of any of its predecessors in that chain.
- **Constraint 6** - The instantiation nodes for a given r.v. must not simultaneously appear in the head and tail of a support node.
- **Constraint 7** - At most one instantiation node for a given r.v. can occur in the tail of a support node.
- **Constraint 8** - Given any set of support nodes such that no two support nodes are mutually exclusive but whose heads each denote a distinct instantiation of a single r.v., the probabilities must sum to less than or equal to one.

The Bayesian Knowledge Base (BKB) representation is an excellent choice to implement these generalized rules because it's formal semantics guarantees the internal consistency of the knowledge-base yet it's flexibility lends itself toward this type of abstraction.

**N of K**

Consider Figure (0.3), we would like to include a rule in our representation of the form 'If three of Chest Pain = Present, Difficulty Breathing = Present, Nausea = Present or Sweating = Present are true then Heart Attack = Present is true with probability 0.65.' Such a rule is useful in complex and uncertain domains where several pieces of evidence support a particular conclusion but not all possible evidence is expected or found and reasoning must occur with incomplete information. This example illustrates the real-world case in the medical domain in which a patient experiencing a heart attack may have chest pain, difficulty breathing, nausea or be perspiring for no apparent reason. Often however, a patient may have a heart attack without one or more of these symptoms such as chest pain. Situations such as these are numerous in real-world domains. Handling this type of uncertainty led researchers to conclude it was absolutely necessary for a cardiac diagnosis application to deal directly with causal probabilities (Long & Naimi 1997). Our goal is to be able to represent and reason with non-causal and incomplete information in such complex and uncertain domains.

**Algorithm**

Leveraging the strengths of the BKB knowledge representation we can build our desired knowledge structure from the existing BKB I-nodes and S-nodes. That is, we can build our general rule using the basic BKB I-nodes and S-nodes and this construction can be carried out by an algorithm. With each new r.v. added as support for a rule, we insert additional I-nodes into the knowledge-base whose values indicate the number of the first N evidence I-nodes which are found to agree with the conclusion. When we have added all of the K evidence I-nodes in the rule, each I-node of the last new r.v. supports the conclusion with the probability indicated by the user. Additionally, since we have worked within the constraints outlined above, our structure is guaranteed to be probabilistic and internally consistent. Therefore, we would like to provide an abstraction of the BKB representation which includes rules of this form as well as other rules which can be expressed by this construct. Figure (0.4) shows how the Heart Attack example would be expressed in a BKB by the algorithm in Figure (0.2). For simplicity, we assume all r.v.s are binary adding that r.v.s of n states are a simple extension. Figure (0.3) shows the corresponding rule in generalized form. Note that in both figures, the probabilities 0.00001 . . . 0.85 are specified by the user when adding the rule to the knowledge-base. This abstraction will allow an expert user to specify domain knowledge in an intuitive manner while hiding the im-
1. Let $A_i = a_i$ denote the I-node corresponding to the $i^{th}$ r.v. in the state which is selected by the user as evidence and $A_i \neq a_i$ denote the I-node corresponding to the $i^{th}$ r.v. in the state which does not support the conclusion.

2. We introduce additional r.v.s named $N$ of i for each i from 2 to K the total number evidence r.v.s

3. For each new r.v. we add a new state with value j for each j from 0 to i by adding new I-nodes to the BKB named $N$ of $i = j$.

4. We allow $N$ of 1 = 0 to be represented by $A_1 \neq a_1$ and $N$ of 1 = 1 by $A_1 = a_1$ to avoid duplication.

5. Where $p_i$ represents the probability specified by the user for i of the K evidence r.v.s found:

   for $i = 2$ to $K$
   for $j = 0$ to $i$
     create a new I-node named $N$ of $i = j$
     if ($j \neq 0$) then
       $A_j = a_j$ AND $N$ of $(i-1) = (j-1)$ supports $N$ of $i = j$
       with probability 1.0
     if ($j \neq i$) then
       $A_i \neq a_j$ AND $N$ of $(i) = (j-1)$ supports $N$ of $i = j$
       with probability 1.0
   for $i = 0$ to $K$
   $N$ of $K$ = 1 supports conclusion with probability $p_i$

Fig. 0.2. The algorithm for generating the NoK generalized S-node.

Implementation details. In addition, this abstraction can allow the expert user to weigh confidence in a conclusion based on the number of evidence random variables which support the conclusion.

Analysis

By inspection of Figure (0.4), one can see that representing $N$ of $K$ for two I-nodes ($K = 2$) requires four (2K) S-nodes and an additional r.v. with three states which adds three (K+1) I-nodes. In fact, the addition of each new I-node as evidence adds (2K) S-nodes and (K+1) I-nodes where $K$ is the total number of evidence I-nodes after adding the new evidence. So, adding the $K^{th}$ r.v. adds 3K+1 nodes to the BKB graph. Also note that in adding the $K^{th}$ r.v., we need K+1 S-nodes for each of the K+1 I-nodes which directly support the conclusion. By Equation (1) we see that the number of nodes added to the BKB is $O(K^2)$.

$$\sum_{n=3}^{K} [3n + 1] + K + 1 < 3 \sum_{n=1}^{K} [n + 1] + K + 1$$
$$< 3K(K + 1) + 4K + 1$$
$$< \frac{3K^2 + 11K}{2} + 1$$
$$= O(K^2)$$

Given that BNs are known to grow exponentially in the number of supporting r.v.s (Pearl 1988), for binary r.v.s this produces a probability table with $2^n$ entries. Together the number of I-nodes and S-nodes in our representation are still $O(n^2)$. This is a significant decrease in the complexity of the represented knowledge when compared to BNs. This decreases the burden computationally because representing these conditional (in)dependence relationships in a BN would require us to store $2^n$ probabilities. It also removes the task of assigning probabilities to all of the $2^n$ possible combinations of r.v. instantiations, many of which are irrelevant or impossible, from the user entering knowledge.

Reasoning With $N$ of $K$

By inspection of our example one can see that all combinations of r.v. instantiations are included in the knowledge structure. That is, there is one unique path through the graph to the conclusion for all possible combinations of evidence r.v.s. Observe that all conditional probabilities in the network are set to 1.0 except
those directly supporting the conclusion which are set to the probabilities defined by the user. By the chain rule, we compute the probability of the conclusion as
\[ \prod_{i=1}^{\infty} (1 \cdot p) = p \]
where \( n \) is the number of I-nodes supporting the conclusion and \( p \) is the appropriate user-defined probability based on the number of evidence r.v.s whose state supports the conclusion during reasoning. Consider Figure (0.4) once again and consider reasoning when 'Difficulty Breathing = True' and 'Nausea = Present' are set as evidence. In this case, 'Difficulty Breathing = True' and 'Chest Pain = Absent' so 'N of 2 = 1' with probability 1.0. Also, 'Nausea = Present' and 'N of 2 = 1' so 'N of 3 = 2' with probability 1.0. Since 'Sweating = Absent' and 'N of 3 = 2' then 'N of 4 = 2' with probability 1.0 and therefore 'Heart Attack = Present' with probability 0.35.

**Thrashing**

Due to the incompleteness allowed in the BKB representation, the knowledge-base may oscillate, or thrash, between two or more states of a random variable during validation (Santos, Gleason, & Banks 1997). The implementation of the N of K support node only builds support for one particular state of a r.v. By offering the generalized N of K support node this effect can be minimized or eliminated since the r.v. states of the last I-node added all support the same state of the conclusion r.v. The probabilities of two or more states of the r.v. can not oscillate since only one r.v. is supported by the evidence and only those probability values are adjusted during validation of the knowledge base.

**Conclusions**

The N of K support node is a useful addition to the BKB representation. It permits the simple expression of non-causal uncertain knowledge in an efficient and consistent manner. Additional rules are also useful including an OR-node, XOR-Node, NAND-node and NOR-node. These rules can be expressed via the same construct by selecting the appropriate I-Nodes of the last r.v. added to the construct to support the conclusion. Consider Figure (0.5), showing the implementation of a generalized OR node using the same construct as the N of K node. Since the same construct is used, a generalized OR node implemented in this way is also guaranteed to be internally consistent and the number of nodes added to the graph is \( O(n^2) \). All of the other generalized nodes mentioned above can also be represented by this construct. Together, these generic rules provide a toolbox from which the expert user can select the most appropriate rule instead of becoming buried in the details of the knowledge representation in trying to express the knowledge in terms of only one type of rule. Building knowledge algorithmically can also keep the complexity of the knowledge-base to a minimum, rather than allowing the domain expert to build complicated knowledge structures which may be inefficient or worse, incorrect. Also, by taking advantage of the incompleteness accommodated by the BKB knowledge representation, we are able to significantly reduce the complexity of the knowledge represented by the N of K rule when compared to BNs.

**Fig. 0.5.** Implementation of a Generalized OR node.

**Future Work**

The generalized rules presented here are currently being added to a new version of the PESKI system, introducing a Generalized Bayesian Knowledge Base. The underlying structure of the N of K support node will be investigated as a possible solution for reliably and consistently combining information from several, possibly conflicting, sources.

**Acknowledgements**

This work was supported by AFOSR Grant No. F49620-99-1-0059 and a grant from the Connecticut Space Grant Consortium.

**References**


Santos, Jr., E.; Banks, S. B.; Brown, S. M.; and Bawcom, D. J. 1999. Identifying and handling structural incompleteness for validation of probabilistic