MACK: A Tool for Acquiring Consistent Knowledge Under Uncertainty

Eugene Santos Jr.[1] **and Darwyn O. Banks and Sheila B. Banks**
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433-7765
esantos@afit.af.mil

Abstract

We develop a new methodology and tool for knowledge acquisition under uncertainty. A new knowledge representation called Bayesian Knowledge Bases provides a powerful key to our approach and is well-grounded in probability theory. In this paper, we demonstrate the ease and flexibility with which knowledge acquisition can be accomplished while ensuring the *consistency* of the knowledge base as data is both acquired and subsequently maintained. We present the MACK tool and apply it to NASA's Post-Test Diagnostics System which locates anomalies aboard the Space Shuttles' Main Engines.

Introduction

Knowledge engineering new domains remains a daunting task for both the expert and the engineer involved. The knowledge must be elicited from the expert and then converted into a form according to the internal knowledge representation of the target expert system. Furthermore, the knowledge itself must then be validated and verified to ensure the system's reliability. Figure 1 lays out the standard three phase knowledge acquisition process.

The ease with which we perform a given phase is intricately tied to each of the other phases. For example, the interview should ideally be as painless as possible avoiding problems such as redundant questioning, overly rigid response templates — requiring the expert to answer using an inflexible and often unrealistic format, etc. Only the most intuitive and flexible of knowledge organization should be required of the expert. Unfortunately, if the internal knowledge representation is significantly different from the organization of the interview, then the knowledge engineer is faced with the onus of properly encoding the information. This typically entails a radical re-structuring of the given information while simultaneously attempting to preserve its original content.

[1]This research was supported in part by AFOSR Project #940006.

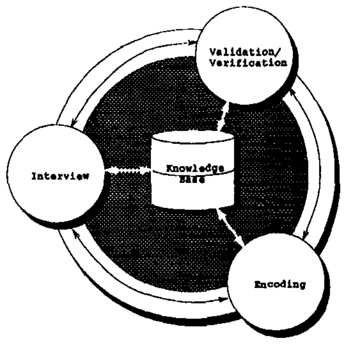

FIGURE 1. Knowledge acquisition process.

For the moment, let's assume that our knowledge representation is also relatively simple and mirrors the interview organization. While this simplifies the job for the knowledge engineer, we end up impacting our last phase. Clearly, there is a tradeoff between the amount of organization and flexibility inherent in our knowledge representation versus our ability to perform verification and validation over it. For example, the problem of consistency is especially sensitive. Without much organization, it is nearly impossible to detect when and where an incosistency has occurred.

There are many other factors that determine the success or failure of a knowledge acquisition tool. This includes many pragmatic concerns such as learnability of the tool, input requirements — user interface, and knowledge induction (Gaines & Shaw 1993; Gonzalez & Dankel 1993). All of the above factors serve towards building an ideal tool for assisting knowledge engineers.

The major difficulty faced by all knowledge engineers is in dealing with uncertainty: uncertainty in the expert's themselves about their knowledge and uncertainty in the engineer trying to translate the knowledge. Although it seems that all knowledge can be encoded in logical "if-then" style rules — indeed, most agree that it is the simplest and most intuitive approach to organization (Buchanan & Shortliffe 1984), exceptions to the rule quickly explode the number of rules necessary. Unfortunately, this further leads to questions on completeness. Unless all the exceptions have been identified, the original rule being as general as it is will produce an incorrect result in the situations where the yet to be identified exceptions occurs.

A wide variety of models for uncertainty are available. However, we must be careful in choosing an appropriate one for our task. Managing the uncertainty is also a task by itself and the most successful approaches ground themselves with strong semantics such as probability theory. This grounding helps to reduce general ad-hocness and reasoning anomalies which arise from weakly defined semantics. Furthermore, completeness and consistency are easier to guarantee. The trade-off, of course, again lies in the flexibility of the representation and most importantly in how intuitive it is for the expert and knowledge engineer to work with.

In this paper, we present a new knowledge acquisition tool called MACK. Its knowledge representation is semantically well-grounded yet remains extremely flexible and intuitive. Furthermore, MACK automatically guarantees that our knowledge-base is always consistent and assists the engineer in avoiding or correcting inconsistencies as knowledge is acquired. Finally, we demonstrate our system by applying it to the real world problem of Post-Test Diagnosis on the Space Shuttle Engines for NASA Lewis Research Center.

Managing Uncertainty

There is an inverse relationship between the efficiency of an automated reasoning algorithm and the flexibility of its associated knowledge representation. However, the former is secondary to the overall capabilities of the final product when weighed against the latter. Without an appropriate representation we cannot properly store our knowledge. Thus, system designers often give preference to flexible representations (Feigenbaum 1980). Nevertheless, real-world applications for intelligent or expert systems, such as those in space operations domains, require a balance of these two capabilities. As we shall see, Bayesian Knowledge Bases (abbrev. BKBs) (Santos & Santos 1996) are just such a representation.

Inference mechanisms such as those in BKBs are noteworthy for their ability to represent uncertainty precisely because they marry the strong models of probability theory with a simple "if-then" rule structure. Reliance on the well-established laws of probability helps guarantee that BKBs are a sound and consistent representation of knowledge, and therefore, that the results they generate will not be inconsistent.

Probabilistic reasoning in intelligent systems exploits the fact that probability theory is and has been an accepted language both for the description of uncertainty and for making inferences from incomplete knowledge (Keshavan *et al.* 1993). Using the semantics of probability theory, we designate random variables to represent the discrete objects or events in question. We then assign a joint probability distribution to each possible state of the world, *i.e.*, a specific value assignment for each random variable. This assignment allows us to reason over the set of probabilities.

Unfortunately, the size of this joint distribution can grow exponentially in the number of random variables making inferencing and knowledge acquisition computationally intractable. One way to address this complexity is by assuming many, if not all, of the random variables to be independent. However, while such independence assumptions significantly facilitate knowledge acquisition and ultimate resolution, if applied carelessly, they can oversimplify the problem statement such that the final answer loses considerable validity.

Bayesian approaches avoid oversimplification by couching their independence assumptions in terms of conditional dependencies. Let D, E and F be random variables. The conditional probability, $P(D|E,F)$, identifies the belief in D's truth given that E and F are both known to be true. If $P(D|E,F) = P(D|E)$, we say that random variables D and F are conditionally independent given E. In other words, once we know E is true, we can establish D's truth with or without any knowledge of F's. We call D the *head* of $P(D|E,F)$ and $\{E,F\}$ the *tail*.

Bayesian philosophy holds that such conditional relationships — *e.g.*, $P(D|E)$ — are more in keeping with the way humans tend to organize knowledge (Shafer 1987; Shachter & Heckerman 1987). The following equation below shows Bayes' Formula for computing these probabilities: $P(D|E) = \frac{P(E,D)}{P(E)}$. We can view random variable D as a possible hypothesis (or set of hypotheses) held in advance and E as the actual evidence that was or will be generated. This formula shows how previous hypotheses should be modified in light of that new evidence.

By manipulating Bayes' Formula we can compute the joint distribution for n variables as shown in Equation (1):

$$P(D,E,F,G,H) = P(D|E,F,G,H)P(E|F,G,H)* \\ P(G|F,H)P(H|F)P(F) \quad (1)$$

More importantly to inferencing, the subsequent incorporation of the known independence conditions

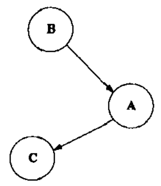

FIGURE 2. Example graph with random variables as nodes.

further reduces the amount of information we must actually store to be able to compute the required joint probability (Pearl 1988). Thus, let us assume that random variable E is known to be conditionally independent from both F and H when the value of G is known, and D is likewise conditionally independent with knowledge of both E and G. Then, we can simplify Equation (1) further still: $P(D,E,F,G,H) = P(D|E,G)P(E|G)P(G|F,H)$ $P(H|F)P(F)$

These conditional dependencies can also be represented pictorially with a directed graph. Let A, B and C be random variables representing a traffic light, its associated vehicle detector and pedestrian signal, respectively. Figure 2 graphically depicts this network over these variables. Since the signal depends upon the light, we say that A is the parent of C. Similarly, B is the parent of A.

Now, assume we want to expand this set with a probability for the detector stating the likelihood of its being tripped during rush hour. Such an inclusion would introduce a cycle into our graph since the detector and traffic light cannot both depend upon the other. It becomes synonymous to the classic circular reasoning example: "If Smoke, then Fire" coupled with "If Fire, then Smoke."

As we can see, conditional independence at the random variable level is overly restrictive. We must proceed to an even finer level of distinction. This is the basis of the BKB representation. Assuming the same trio of random variables and the partial set of values below:

$$P(C = \text{"Don'tWalk"}|A = \text{red}) = x_1 \tag{2}$$

$$P(C = \text{"Walk"}|A = \text{green}) = x_2 \tag{3}$$

$$P(A = \text{green}|B = \text{On}) = x_3 \tag{4}$$

$$P(A = \text{red}|B = \text{Off}) = x_4 \tag{5}$$

We can quite legally add the new constraint:

$$P(B = \text{On}|A = \text{red}, D = \text{rushhour}) = x_5 \tag{6}$$

without creating a directed cycle. Figure 3 shows the graphical representation of this example.

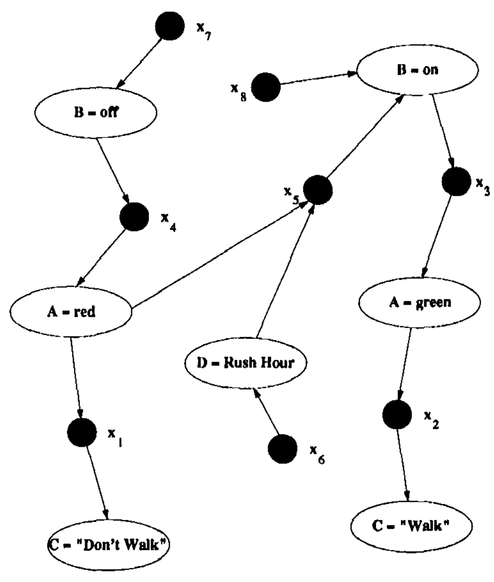

FIGURE 3. Example Bayesian Knowledge Base.

Notice how the graph of a BKB is simple, bipartite and directed. It has two distinct types of nodes. The first, shown as lettered ovals, corresponds to individual random variable instantiations. However, now these nodes are particular not simply to a random variable, but to a specific instantiation thereof. Hence Figure 2's single node for variable A, the traffic light, becomes two distinct instantiation nodes for "A = Red" and "A = Green" in Figure 3. These are called *instantiation nodes* or *I-nodes* for short.

The second type of node, depicted as a blackened circle, is called a *support node.* When drawn, these nodes, which represent the numeric probability value itself, have exactly one outbound arrow to the instantiation node representing the probability's head. Support nodes also provide a graphical terminus for zero or more inbound dependency or conditioning arrows from each of the parent instantiations in the probability's tail. Conversely, all instantiations nodes must have one or more inbound arrows to establish their probabilities and sets of supporting conditions, but need not contain any outbound arrows.

BKBs subsume many existing probabilistic models such as Bayesian networks (Pearl 1988). They are more powerful from the fact that cyclicity can be represented in BKBs and that there is no requirement that all conditional probabilities must be specified unlike Bayesian networks. Furthermore, conditional independence conditions are much more explicit in the BKBs providing a clearer organization to the knowledge engineer.

Guaranteeing Consistency

As we have seen, BKBs allow cyclical construction, *i.e.*, we can have $P(B = b_2 | A = a_2)$ and $P(A = a_1 | B = b_2, C = c_1)$ in our database simultaneously. However, this additional constructive capacity changes the concept of consistency for a knowledge base in ways unique to BKBs.

BKB inference algorithms operate straightforwardly by computing a joint probability for a particular instantiation to all the random variables set — henceforth we will refer to such an assignment as a "state of the world." In an inconsistent knowledge base, there will be multiple ways to compute this value for any particular state of the world with no guarantee that the results will be equal as they must.

To illustrate, let X, Y and Z be boolean random variables. Clearly, there are eight possible states of the world. Given the following probabilities as the entire population of the BKB's database:

$$P(X = \text{true} | Y = \text{false}) = .40 \tag{7}$$
$$P(X = \text{true} | Z = \text{true}) = .80 \tag{8}$$
$$P(Y = \text{false} | X = \text{true}) = .70 \tag{9}$$
$$P(Y = \text{false} | Z = \text{true}) = .45 \tag{10}$$
$$P(Z = \text{true}) = .75 \tag{11}$$

The inference engine may compute $P(X = \text{true}, Y = \text{false}, Z = \text{true})$[2] by multiplying equations (7), (10) and (11) or by multiplying equations (9), (8) and (11). However, the joint probability on the first of these logical paths is .135, while along the second path it is .42, more than three times greater!

We eliminate this inconsistency by requiring any two tail-compatible[3] probabilities which share a head to have exactly equal probability values. In other words, we guarantee that the probabilistic value of (8) is the same as the value of (7) and that (10) is also equal to (9) as shown below.[4]

$$P(X = \text{true} | Y = \text{false}) = .40 \tag{7a}$$
$$P(X = \text{true} | Z = \text{true}) = \cancel{.80}.40 \tag{8a}$$
$$P(Y = \text{false} | X = \text{true}) = .70 \tag{9a}$$
$$P(Y = \text{false} | Z = \text{true}) = \cancel{.45}.70 \tag{10a}$$
$$P(Z = \text{true}) = .75 \tag{11a}$$

While this equality requirement clearly forbids inconsistencies, it does little either to explain or to assist the knowledge engineer in his or her efforts to build the system. The engineer's problem, then, is to determine the equating formula which will be used initially to create a viable knowledge base. Such formulae are countless—*e.g.*, minima, maxima, weighted or unweighted averages, any real function over the values, etc. Moreover, in the absence of other information, all can be equally valid. This makes an algorithm to construct BKBs all the more elusive.

Before we can develop a knowledge acquisition methodology, we must be aware of those areas of a BKB with the potential to violate its construction constraints or to harbor inconsistent bits of knowledge. Once identified, we can ensure both the validity and consistency of a knowledge base by induction as it is being built. However, unlike inductive systems such as those predicated on the ID3 algorithm (Quinlan 1986), BKBs have no requirement for the complete specifications of all attributes and values which make those systems less tenable for large data sets.

Following are descriptions of the eight construction constraints we determined, as well as the manner in which our implementation satisfies them. Taken together, they guarantee both the structure of the BKB and, more importantly, its probabilistic consistency. The fact that there are only eight underscores the flexibility of the BKB representation and its ability to obey the laws of probability theory while still being general enough directly to interface with the expert.

- **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 single random variable.*
- **Constraint 4** - *Any support nodes which share a head instantiation must be mutually exclusive.*
  Given any state of the world, all but one of an instantiation node's support conditions must conflict with that particular assignment of global values. In the parlance of logic, we could say that this constraint requires the truth or falsity of any instantiation node to be established via an exclusive-or condition among its attendant support nodes.

$$P(X = \text{true} | Y = \text{true}) = y_1 \tag{12}$$

[2]Note that there is insufficient data here to compute any other joint probability.

[3]We define probabilities to be *tail-incompatible* or *mutually exclusive* only if there exists a random variable in the tail of both which takes on a different value in each.

$$P(Y = \text{false} | Z = \text{true}) = .45$$
$$P(Y = \text{false} | Z = \text{false}) = .83$$

For example, these probability equations are tail-incompatible since both place conditions upon random variable Y's being false and the variable Z assumes a different value in their tails. Similarly conditioned probabilities with non-identical heads are considered *mutually exclusive*.

[4]In this case we have arbitrarily set the value of the second member of each pair equal to the first.

$$P(X = \text{true}|Y = \text{true}, Z = \text{false}) = y_2 \quad (13)$$
$$P(X = \text{true}|Y = \text{false}, Z = \text{false}) = y_3 \quad (14)$$
$$P(X = \text{true}|Y = \text{false}, Z = \text{true}) = y_4 \quad (15)$$

For example, the equations above show a set of support conditions using boolean random variables: X, Y and Z. Clearly, each of these support conditions modifies the same head instantiation: setting X to "true." However, probabilities (12) and (13) are not mutually exclusive, since the former, which does not depend upon random variable Z, will always be valid any time that the latter is. In this case when Y is true, either probability affords a valid inference path to substantiate X's truth, thus y_1 must equal y_2 for the database to be consistent.

Constraint 1 guarantees there will be one or more support nodes for each instantiation. This fourth constraint provides the necessary distinctions between those support nodes such that one, and only one, may be active.

- **Constraint 5** - *Given any inference chain, a support node's head must not occur in the tail of any of its successors in that chain.*
The fifth constraint closes the door on logical cycles. For example, (16) through (19) below form a loop in that all four can be simultaneously active,

$$P(A = a_1|D = d_2) \quad (16)$$
$$P(D = d_2|B = b_3, C = c_1) \quad (17)$$
$$P(B = b_3|E = e_2) \quad (18)$$
$$P(E = e_2|A = a_1, C = c_1) \quad (19)$$

i.e., no mutual exclusivities exist within the set, and (19) depends in part upon the head instantiation of a predecessor, in this case "$A = a_1$." Failure to preclude this cycle would allow the inference engine potentially to enter an infinite loop since (16) can clearly be re-investigated as a successor to (19).
As with many search problems, discovering these cycles can quickly become combinatorial. However, we can conduct this search *as the expert identifies support conditions* which ensures the consistency of the knowledge base. This also assists the expert in correcting inconsistencies by flagging them sooner, *i.e.*, upon introduction to the database, rather than later. We accomplish this search cheaply and efficiently using a depth-first algorithm which begins at the head of the new support node and branches throughout the BKB structure, as necessary.
- **Constraint 6** - *The instantiation nodes for a given random variable must not simultaneously appear in the head and tail of a support node.*
- **Constraint 7** - *At most one instantiation node for a given random variable can occur in the tail of a support node.*
- **Constraint 8** - *Given any set of support nodes whose tails are mutually tail-compatible but whose heads each denote a distinct instantiation of a single random variable, the probabilities must sum to less than or equal to 1.*
Because we are using a probabilistic reasoning scheme, we need this last constraint to disallow any simultaneously valid sets of probabilities for the same item from ever summing to values greater than 1. We use the fact that the head instantiations of the sets' elements share the same random variable to identify each set.
In the example below, we have collected all support conditions for random variable A (regardless of instantiated value) which depend upon random variable B's first value.

$$\begin{aligned} P(A = a_1|B = b_1, C = c_1) &= v \quad (20)\\ P(A = a_1|B = b_1, C = c_2, D = d_1) &= w \quad (21)\\ P(A = a_2|B = b_1) &= x \quad (22)\\ P(A = a_3|B = b_1, D = d_1) &= y \quad (23)\\ P(A = a_3|B = b_1, D = d_2) &= z \quad (24)\end{aligned}$$

Equations (20) and (21) can never be active at the same time since they depend on different instantiations of C. Similarly, (24) is mutually exclusive both with (21) and (23) due to random variable D. Under this constraint, these dependencies on different states of the world divide this set of equations such that the following inequalities must all be true:

$$x + z \leq 1 \quad (25)$$
$$v + x + z \leq 1 \quad (26)$$
$$w + x + y \leq 1 \quad (27)$$
$$v + x + y \leq 1 \quad (28)$$

MACK automatically normalizes the values of any probabilities which violate this constraint simply by dividing each element of the set by the total.[5] Notice that (22)'s probability, x, is a factor in all these subsets since it does not depend either on C or D and can thus be simultaneously true with any of the other four. We also note that in this example (26) effectively overrides (25) because if the former holds, then the latter must also *a fortiori*.[6]

[5]The pre-normalized value is maintained in the system for use in subsequent normalizations over different sets of probabilities.

[6]v, w, x, y, z are all non-negative real-valued variables between 0 and 1.

A complete consistency check essentially involves verification of each of the eight aforementioned constraints. Obviously, such an approach may become computationally expensive, especially as the size of the network grows. However, we have implemented our BKB knowledge acquisition routine such that complete consistency checking is rarely necessary. Specifically, guarantors for Constraints 2, 3, 6 and 7 are built into the object creation routines. Thus, as the expert defines the items of the BKB, their associated values and dependencies, the software objects themselves prohibit duplicate instantiations and ensure the validity of all references between the instantiation node and support node classes. Constraints 1, 4, 5 and 8, then, are the only constraints explicitly tested during a review. In addition, our BKB implementation conducts such reviews incrementally. We check each new support node as it is entered into the database to ensure it introduces no new inconsistencies to the existing consistent BKB, *e.g.*, by engendering a logical cycle.

Post-Test Diagnosis

Development, maintenance and improvement of any large system, especially one with human lives at stake, usually involves extensive testing. NASA's Space Transportation System, or Space Shuttle, is no exception. Marshall Space Flight Center in Huntsville, Alabama routinely conducts ground tests and collects actual flight data on the shuttle's boosters to better assess the health, status and current capabilities of the reusable engines and their many components. Presently, these assessments involve large teams of engineers who review remote data received from hundreds of on-board sensors called PIDs. Officials then use these manual reviews to determine the fitness of the engine for another test or flight.

The Post-Test Diagnostic System is an on-going cooperative project to automate the Space Shuttle Main Engine (SSME) review process using intelligent systems. Its stated goals are:

- To aid in the detection and diagnosis of engine anomalies.
- To increase accuracy and repeatability of rocket engine data analysis.
- To reduce analysis time.

When complete, its components will validate engine sensors, reliably extract salient features from telemetry data, and analyze SSME performance systems, combustion devices, turbomachinery and dynamics.

As of this writing, two different versions of one component—the High Pressure Oxidizer Turbopump (HPOTP)—have been built and validated by government contractors under the auspices of researchers at NASA Lewis Research Center.[7] These systems provided us the opportunity to test MACK's applicability to a real-world domain and a set of known parameters against which to corroborate the utility of MACK-acquired knowledge for BKB reasoning.

The HPOTP is an engine component designed initially to raise, then to maintain the pressure of the liquid oxygen flowing into the engine at the varying levels of thrust during the shuttle's flight profile (ric 1988). Using a turbine powered by the oxidizer preburner's hydrogen-rich hot gas, this centrifugal pump manages the flow of liquid oxygen into the engine's main and preburner injectors. Beside the pumps and turbines, the HPOTP's third major group of subcomponents contains the extensive shaft seals which separate pumps, turbines and the fluids they regulate (ric 1988).

Being an automated tool, MACK is designed to be operated directly by the domain expert. In fact, it is a key component of the PESKI Knowledge Organization and Validation subsystem currently under development for Air Force Office of Scientific Research (AFOSR). As a result, we, the knowledge engineers, simulated the expert's involvement. We note, however, that much of the previous knowledge engineering accomplished for HPOTP has involved collating and sorting the information gathered in numerous interviews with the Alabamian crew of rocket scientists.[8] (Banks 1995) correlates selected text from knowledge acquisition interviews with anomaly definitions from the second version of HPOTP and with conditional probabilities in the HPOTP BKB. These correlations show that it is, in fact, plausible partially or completely to remove the middleman and allow the expert to be his/her own knowledge engineer—*i.e.*, if the expert is so inclined, s/he can with minimal instruction create a BKB from scratch.

The MACK Tool

Having described both BKBs and the HPOTP application, we now explore some of the processes by which MACK acquires knowledge. The PESKI architecture within which MACK resides assumes the knowledge engineer to be optional. As a result, the MACK tool, like those discussed by Sandahl (Sandahl 1994), can potentially be the primary interface with the expert. This role places a premium on user-friendliness as much

[7]The first was developed by Science Applications International Corporation, San Diego, California (sai 1994). The second by personnel from Aerojet Propulsion Division, Sacramento, California (Bickmore 1994).

[8]NASA Lewis researchers have been conducting interviews with Marshall Space Flight Center engineers since Spring 1992.

as adherence to BKB constructs and probabilistic formalisms.

MACK is a menu-driven system. These menus allow us to handle the simpler BKB constraints — Constraints #2, 3, 6 & 7 —by simple manipulation of the menu options presented to the user. Other illegal choices simply trap program control until a valid selection is entered. The examples excerpted below are taken from the HPOTP application.[9] It includes data entry of the conditions governing the shift anomaly noted via sensor, PID 990, here called "Anomaly 990 Shift." This anomaly depends upon the sensor's peak and equilibrium values which are represented by the random variables, "PID 990 Peak" and "PID 990 Equilibrium," respectively. Here the expert is creating the first support condition for the instantiation of the random variable "Anomaly 990 Shift" to value "Found."[10]

```
At present, [Anomaly 990 Shift]'s being [Found] depends
upon the following sets of conditions:
   No support conditions!
Enter 0 to add new support conditions for [Anomaly 990
Shift]'s being [Found].

Otherwise, enter 1 to quit
 0
```

Recall that Constraint 2 requires the instantiations and supports connected in a BKB to be well-defined. Thus the system when creating a support condition only presents a choice among the previously instantiated random variables. MACK additionally restricts the options to those variables which can actually be used in the nascent support condition. We can see this in the absence of Anomaly 990 Shift itself from the subsequent menu shown below which is in keeping with Constraint 6.

```
[Anomaly 990 Shift]'s being [Found] can depend upon
which of the following components:
   2 - PID 990 Equilibrium
   3 - PID 990 Peak
   0 - None of the Above Components
Choice: 2
   1 - Nominal
   2 - Out of Family
   0 - None of the Above; Abort
Choice: 2
```

Having already selected a value of PID 990 Equilibrium, the expert is now queried for continuance. In this abbreviated example we see that the only remaining random variable option available to the expert is PID 990 Peak. Anomaly 990 Shift has been previously removed under Constraint 6 and PID 990 Equilibrium, as a new addition to the condition, is now illegal in accordance with Constraint 7.

```
Presently, this condition holds that [Anomaly 990 Shift]'s
being [Found] can depend upon the following:
   PID 990 Equilibrium = Out of Family
Do you wish to extend this condition? Y / N
y

[Anomaly 990 Shift]'s being [Found] can depend upon
which of the following components:
   3 - PID 990 Peak
Choice: 3
   1 - Nominal
   2 - Out of Family - High
   3 - Out of Family - Low
   0 - None of the Above; Abort
Choice: 2

Presently, this condition holds that [Anomaly 990 Shift]'s
being [Found] can depend upon the following:
   PID 990 Equilibrium = Out of Family
   PID 990 Peak = Out of Family - High
Do you wish to extend this condition? Y / N
n

Please complete the sentence below from the following
list of choices:
   0 - inconceivable
   1 - not likely
   2 - possible
   3 - probable
   4 - almost certain
It is ______________ that the [Anomaly 990 Shift] is
[Found] depending upon ...
   PID 990 Equilibrium = Out of Family
   PID 990 Peak = Out of Family - High
Choice: 3
```

MACK then presents the expert with a menu of choices from which it will internally derive the support condition's probability. Since a BKB's probabilistic nature is masked from the expert, we use only the qualitative and linguistic terms shown below with their current value ranges.

inconceivable:	0.00 - 0.10
not likely:	0.10 - 0.35
possible:	0.35 - 0.65
probable:	0.65 - 0.90
almost certain:	0.90 - 1.00

It is important to note here that during knowledge acquisition for a BKB, the actual numeric value assigned to any given probabilities is not significant. Refinement of these values through belief revision and belief updating is the province of the reasoning and explanation facilities in PESKI. The values associated with each node only attain meaning after the inference engine reasons over them during belief updating.

[9] See (Banks 1995) for complete transcripts.

[10] Anomaly variables are basically boolean: "Found" or "Not Found."

It should be obvious, however, the inference engine's propagation of probabilities must begin somewhere. In his discussion of the validity of such values to probabilistic reasoning schemes, Pearl (Pearl 1988) writes:

> [p. 148, The] conditional probabilities characterizing the links in the network do not seem to impose definitive constraints on the probabilities that can be assigned to the nodes. ... The result is that any arbitrary assignment of beliefs to the propositions a and b can be consistent with the value of $P(a|b)$ that was initially assigned to the link connecting them ....

Thus, the decision to use a random number generator in the initial stages of database development neither adds to nor detracts from the BKB. More germane to the topic at hand, it certainly has no impact upon the consistency of the data's logical organization within that BKB.

These menu restrictions only account for the simple constraints. The more involved BKB formalisms are found in a separate consistency checking routine. MACK initiates this larger routine itself after any change to the set of support conditions, removal of an instantiation or upon receipt of up to five new, unsupported instantiations.

```
Welcome to M.A.C.K. — The BKB Module for the
        Acquisition of Consistent Knowledge!!
  0 – Generate new BKB
  1 – Edit existing BKB
  2 – Display current BKB
  3 – Load BKB from file
  4 – Save BKB to file
  5 – Check Knowledge Base Consistency
  6 – Run BKB Belief Revision Program
  7 – Delete the current BKB
  8 – Exit BKB program
```

This consistency checking routine sampled below covers the four remaining BKB constraints and, as a courtesy, also notifies the user of any conditions with zero probability. Initially, we see below that the knowledge base has failed Constraint 1 since the system has no condition defining a probability value for the absence of Anomaly 990 Shift. In these cases, MACK prompts the expert appropriately. Were the expert to answer any of these negatively, the consistency routine aborts there and returns the expert to the edit menu.

```
This BKB is currently inconsistent.

Is it correct that
  [Anomaly 990 Shift] being [Not Found]
does not depend on anything else? Y/N
y

Please complete the sentence below from the following
list of choices:
  0 – inconceivable
  1 – not likely
  2 – possible
  3 – probable
  4 – almost certain
It is ____________ that the [Anomaly 990 Shift] is
[Not Found] depending upon ...
  Nothing!
Choice: 3

This BKB is currently inconsistent.

Is it correct that
  [PID 990 Equilibrium] being [Nominal]
does not depend on anything else? Y/N
n

Please edit the conditions for
  [PID 990 Equilibrium] being [Nominal]
accordingly.

Growing the graph for your BKB.
Instantiations:
  0 – Add new instantiation
  1 – Delete instantiation
Support Conditions:
  2 – Add new support condition
  3 – Edit existing support condition
  4 – Delete support condition

  5 – Return to main menu
```

Constraint 4 is an important one which identifies support conditions that are not mutually exclusive. With insufficient information to make any automatic resolution assumptions here, MACK again queries the expert.

```
ERROR: Support conditions below are not mutually ex-
clusive.

At present, [Anomaly 990 Shift]'s being [Found] depends
upon the following sets of conditions:
  Support Node #1:
    PID 990 Equilibrium = Out of Family
    PID 990 Peak = Out of Family – High
  Support Node #2:
    PID 990 Peak = Out of Family – Low
    PID 990 Equilibrium = Nominal
  Support Node #3:
    PID 990 Equilibrium = Nominal
This BKB is currently inconsistent.

The following pair of conditions for [Anomaly 990 Shift]
being [Found] are not mutually exclusive.
  First Set:
    PID 990 Peak = Out of Family – Low
    PID 990 Equilibrium = Nominal
```

```
Second Set:
  PID 990 Equilibrium = Nominal
Does [Anomaly 990 Shift]'s being [Found] really depend
  upon both sets of conditions? [Enter 0]
   or
  upon each set separately? [Enter 1]
Choice: 1

Which of these conditions may we add to eliminate the
overlap?
  1 - [PID 990 Peak] can be [Nominal]
  2 - [PID 990 Peak] can be [Out of Family - High]
  0 - None of the Above
Choice: 2
```

The expert is given the option either of merging the two conditions into one or of distinguishing them in some way. While the first option is straightforward,[11] the second could conceivably draw upon any component in the BKB except the one in question, *i.e.*, the head of the two support conditions. To assist the expert in this area, MACK makes an initial simplifying assumption that excludes all random variables that are not already present. Since the only way to establish mutual exclusion is for both of the support conditions to contain in their tails a different instantiation of one or more variables. The basis of the assumption is that at least one of the current random variables can be expanded to meet this requirement, thus allowing the tool automatically to select and present options as it does in other areas. These options will be all the values of the existing variables which are not already represented. MACK can easily determine which of the two support nodes should obtain the adjustment since, of course, Constraint 7 which proscribes against multiple values remains in effect.

Verifications of Constraint 8, shown below, occur somewhat innocuously. Since the expert is not aware of the actual probabilistic values anyway, MACK can simply normalize the pertinent sums[12] and reports any adjustments of the support conditions' qualitative variables — *e.g.*, inconceivable, not likely, possible, probable, or almost certain — to the expert. These normalizations always use the original probabilistic range the expert assigned in order to avoid a new, high-value addition from overwhelming predecessors whose values may have already been reduced.

[11]This merger cannot be illegal, *i.e.*, violate either Constraint 6 or 7. If the support conditions in question reference instantiations which will conflict when merged, then they are mutually exclusive and, therefore, not inconsistent.

[12]It is worth noting that although the normalization itself may be a trivial operation, the determination of the support node sets which are eligible to be normalized is not.

```
This BKB is inconsistent.

Currently, support ranges overlap. Adjusting ranges for
consistency ...
Conditions were:
  It is probable that the [Anomaly 990 Shift] is
  [Found] depending upon ...
    PID 990 Equilibrium = Out of Family
    PID 990 Peak = Out of Family - High
  It is probable that the [Anomaly 990 Shift] is [Not
  Found] depending upon ...
    Nothing!
New conditions are:
  It is possible that the [Anomaly 990 Shift] is [Found]
  depending upon ...
    PID 990 Equilibrium = Out of Family
    PID 990 Peak = Out of Family - High
  It is possible that the [Anomaly 990 Shift] is [Not
  Found] depending upon ...
    Nothing!
```

The HPOTP application turned out to be a rather flat knowledge base. By that we mean that the sensor readings which represent the bulk of the random variables are unconditioned, and most anomaly determinations depend directly on the sensors rather than on some intermediate calculations. As a result, the application did not violate Constraint 5 which searches for logical cycles in the knowledge base.

Conclusions

This research develops a viable knowledge acquisition and maintenance tool and its associated methodology which together implement the new BKB knowledge model. This new tool, MACK, guarantees the consistency of the data stored in a BKB's knowledge base as it is both acquired and later maintained. Moreover, this tool has been applied to a real-world domain—NASA's Post-Test Diagnostic System—which supports Space Shuttle main engine analysis.

MACK, which is implemented on an explicit object-oriented analytical foundation, contains routines designed automatically and incrementally to confirm the consistency of the knowledge being received from the expert and provide him/her with natural assistance in the transfer of knowledge. Regular incremental checks preserve both probabilistic validity and logical consistency by flagging the inconsistent data points to the expert as they are entered and presumably under his/her current consideration. Such checking guards against expert oversight—*e.g.*, the "Whoops! I forgot to run the consistency checker again!" phenomenon—and helps prevent information overload since there can be at most five adjustments required of the expert immediately after any given run of the consistency checking module.[13]

The tool is also able to accept and manipulate time-dependent data which is both common and required not only in the PTDS domain modelled herein, but in many other real-world applications as well (See (Santos & Banks 1996) for more details). Moreover, this capability will prove crucial to any eventual efforts to operate a BKB inferencing mechanism in real- time or near real-time.

In order to implement the MACK tool properly to guarantee consistency of the knowledge, we had to formalize the notion of consistency for BKBs and then determine the necessary conditions and constraints. The constraints ensure the proper relationships between the instantiation and support nodes are in force at all times. This includes algorithms both to detect constraint violations and to facilitate corrections.

We have availed ourselves of the BKB's computational efficiencies without sacrificing the increased structural flexibility they afford both for reasoning with uncertainty and when compared to other Bayesian inferencing methods. We evidence this by the construction of a BKB for the Post-Test Diagnostics System. Continued work in the application domain by PESKI and BKB researchers promises to facilitate ongoing PTDS knowledge acquisition within NASA as well as to provide the agency with novel, probabilistic reasoning alternatives.

With MACK, by allowing the expert to enter his/her data, conduct verification test runs, and receive from those tests an explanation detailed enough to allow the expert to refine and adjust the knowledge base appropriately, this will establish BKBs as a new methodology for reasoning under uncertainty.

References

Banks, D. 1995. Acquiring consistent knowledge for bayesian forests. Master's thesis, Graduate School of Engineering, Air Force Institute of Technology.

Bickmore, T. W. 1994. Personal communication. NASA Aerojet Propulsion Division, Sacramento, CA.

Buchanan, B. G., and Shortliffe, E. H. 1984. *Rule-Based Expert Systems.* Addison Wesley.

Feigenbaum, E. A. 1980. *Knowledge Engineering: The Applied Side of Artificial Intelligence.* Edward A. Feigenbaum.

[13]The consistency checking routine runs after each addition, removal or edit of *any* support node, following the removal an instantiation node, and after receipt of the fifth consecutive instantiation.

Gaines, B. R., and Shaw, M. L. G. 1993. Eliciting knowledge and transferring it effectively to a knowledge-based system. *IEEE Transactions on Knowledge and Data Engineering* 5(1):4–13.

Gonzalez, A. J., and Dankel, D. D. 1993. *The Engineering of Knowledge-Based Systems: Theory and Practice.* Prentice-Hall, Inc.

Keshavan, H. R.; Barnett, J.; Geiger, D.; and Verma, T. 1993. Introduction to the special section on probabilistic reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15:193–195.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, CA: Morgan Kaufmann.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning* 1(1):81–106.

1988. Space transportation system training data, SSME orientation (part a - engine). Rocketdyne Division, Rockwell International Corporation.

1994. Reusable rocket engine turbopump health management system. Technical Report Final Report, Contract NAS3-25882, Science Applications International Corporation.

Sandahl, K. 1994. Transferring knowledge from active expert to end-user environment. *Knowledge Acquisition* 6(1):1–22.

Santos, Jr., E., and Banks, D. O. 1996. Acquiring consistent knowledge. Technical Report AFIT/EN/TR96-01, Department of Electrical and Computer Engineering, Air Force Institute of Technology.

Santos, Jr., E., and Santos, E. S. 1996. Bayesian knowledge-bases. Technical Report AFIT/EN/TR96-05, Department of Electrical and Computer Engineering, Air Force Institute of Technology.

Shachter, R. D., and Heckerman, D. E. 1987. Thinking backward for knowledge acquisition. *AI Magazine* 55–61.

Shafer, G. 1987. Probability judgement in artificial intelligence and expert systems. *Statistical Science* 2:3–44.