12-1996

A Test-Case Based Approach to Bayesian Knowledge Base Incompleteness Detection and Correction

Louise J. Lyle

A Test-Case Based Approach to Bayesian Knowledge Base Incompleteness Detection and Correction

THESIS
Louise J. Lyle
Captain

AFIT/GCS/ENG/96D-17

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/96D-17

A Test-Case Based Approach to Bayesian Knowledge Base Incompleteness Detection and Correction

THESIS

Louise J. Lyle
Captain

AFIT/GCS/ENG/96D-17

AFIT/GCS/ENG/96D-17

A Test-Case Based Approach to Bayesian Knowledge Base Incompleteness Detection and Correction

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Louise J. Lyle, B.S., M.A.

Captain

December, 1996

Acknowledgements

This has been an interesting, no doubt character building experience, that I'll probably look back on with humor.

Thanks to all the individuals who contributed to my time here at AFIT – the teachers who filled my life with stress and knowledge, and the classmates who did their best to help me remember the knowledge and forget the stress.

I particularly want to thank my advisor Dr. Santos for the direction and motivation for producing this research effort, and to the rest of my committee, Maj. Banks and Dr. Hartrum, for ploughing through it and giving me suggestions on improvements.

As to the PESKI group ... good wonder twin Dan, GUI (gooey?) Bob, and QuickBKBrett ... gee guys, it was fun while it lasted. Thanks for the support and encouragement – from now on, every time I get a seg fault, this wonder twin'll think of you. :)

Louise J. Lyle

Table of Contents

List of Figures

List of Tables

AFIT/GCS/ENG/96D-17

Abstract

This work develops tools and techniques to identify particular Bayesian Knowledge Base (BKB) incompletenesses, and to modify the existing knowledge-base (KB) structure to correct these problems. The methodology performs manually or automatically, informing the user of either problems causing the incompleteness, or of details resulting from the automatic knowledge-base correction. The proposed methodology is designed for integration with BVAL[9], to augment BVAL's validation techniques.

A Test-Case Based Approach to Bayesian Knowledge Base Incompleteness Detection and Correction

I. Introduction

An important focus within artificial intelligence (AI) is the idea of knowledge-based systems (KBSs). Such systems could assist in decision making and reasoning, freeing human counterparts to focus attention on other areas. In addition, such systems could potentially offer new insights into heretofore unsolved, possibly intractable, problems. The origin of KBSs can be traced back to the 1960s, with the creation of DENDRAL in 1965. DENDRAL was the first AI program to emphasize the power of specialized knowledge over generalized problem- solving methods[3]. The development of DENDRAL opened the door for subsequent KBSs to build on its vision. Possibly the best known of these successors were MYCIN[3] (a blood disease diagnosis tool), PROSPECTOR[7] (a geologic mineral exploration tool), and XCON/R1[15] (a computer system configuration tool).

While the goal of KBSs may be easy to enunciate, problems in design and implementation continue. Two such problems are the areas of uncertainty and knowledge acquisition (KA). Both seem inextricably linked; even if we could solve the problem of uncertainty, the problem of getting the desired knowledge into the system would be no easier to attain[21]. With disagreements among experts, as well as the existence of unknown knowledge [1], it is possible that KA would be no easier to attain with consistency methods in place. Assuming, however, knowledge could be elicited, the knowledge itself must then be validated and verified to ensure the reliability of the system.

Much work has been done to address some of the issues of KA and knowledge-base (KB) verification and validation (V & V) within the AI community. However, there seems to be no consensus on the "best" way to proceed, since each method, whether in the realm of KA, V & V,

[1]Neither human nor machine can really claim to know everything since nothing can know of something that will be learned/created at a future time. Hence, unknown knowledge may exist.

representation, etc., appears to have benefits and limitations in its use. This research focuses on a single aspect of the entire KBS process, V & V, with direct applications to the probabilistic representation scheme of Bayesian Knowledge Bases (BKBs). Within V & V, the main concern is ensuring acceptable system performance–specifically, that the system will provide acceptable responses to every query. Thus the system must be able to verify the KB, ensuring it is collected and structured correctly; and the system should be equipped to validate the KB, to ensure the knowledge produces correct results. While the validation tool BVAL[9] was developed to address both these issues, it does not further tools to handle incompleteness discovered during operation. This research addresses the incompleteness problem.

Chapter II presents background on KA elicitation, KA representation, and V & V. Incompleteness methods developed are the focus of Chapter III, the application of these methods to specific test-case examples form Chapter IV, and Chapter V offers conclusions drawn from this research. Suggestions for future research in this field are offered in Chapter VI.

II. Background

Central to the development of any KBS is the idea of knowledge acquisition (KA), a two-part procedure consisting of eliciting knowledge from an expert source (knowledge elicitation), and representing the knowledge in a formal way (knowledge representation). Because of problems in the area of knowledge elicitation (KE) and knowledge representation (KR), KA has often been termed the "bottleneck" of KBS or expert system development[8].

2.1 Knowledge Acquisition – Elicitation

Automated techniques have been developed, such as the use of inductive tools and data-mining (automatic knowledge extraction from databases), to formalize the process of KE as the construction of a more concrete model of problem solving behavior[6].The intelligent tool MORE[13], for example, looks for inconsistencies in user input and successfully uses models to structure the KE process.

Although such automated techniques are effective, and may be used in conjunction with more established procedures of direct interaction between human knowledge engineers and human experts, problems with the bottleneck remain. Many problems continue to be shared by machine-based and human-based approaches, and these contribute to the incompleteness factor (IF) of the knowledge base. Some of these problems are as follows:

1. Missing data.
2. Unavailable data.
3. Unreliable data.
4. Invalid data.

2.1.1 Missing Data. While automated techniques are designed to be thorough in their elicitation from human domain experts, it is still possible that the expert will "forget" crucial factors for which the system will not prompt. Oftentimes, human experts have difficulty expressing implicit knowledge[1], leading to missing facts in the KB. In addition, even apparent non-human sources (like large databases) were often initially created from human expertise and thus may also contain knowledge gaps of this type that are then passed on to the developing KBS.

2.1.2 Unavailable Data. Very few domains are completely enumerated. The medical field, for example, is in a constant state of flux. New drugs are discovered, new techniques perfected, all of which affect the state of knowledge in the field; medical expertise is constantly being expanded with *new* expertise. Therefore, to build an expert system for medical use, regardless of methods used to elicit knowledge, information will be missing. In addition, some cures, diseases, and side effects are simply not yet known and thus their incorporation into a KB not possible. But these cures, diseases, and side effects, if they were known, would have a place in the KB. Since this knowledge cannot be incorporated, the KB is incomplete, since KB interaction with the missing fragments of knowledge is not known.

2.1.3 Unreliable Data. Experts disagree. Given a collection of "experts" in any domain, it is unlikely they will come to concrete agreement on all topics of concern to the knowledge engineer[5, 10]. Thus decisions must be made as to what is included in the KB, and as to the reliability of the included data. If the data is represented, an incorrect determination of the effects of the data on the KB in general could make the data unreliable. If the KB is unreliable, it is incomplete – it is missing the "right" meaning. Consider the simple example where we want to determine if it is day or night based on whether the sun is shining. We have two represented rules:

1. Sun is shining $\Longrightarrow$ It is day (with probability 0.0)
2. Sun is shining $\Longrightarrow$ It is night (with probability 1.0)

The relationships of the sun shining and whether it is day or night is represented in Rules 1 and 2; however the probabilities are incorrect. Thus, given the sun is shining, the correct implication

[1]Knowledge known at a subconscious level, "second nature" type information that is often difficult to enunciate[5].

should be that it is day; however, the KB responds that it is night. Thus the data stored is unreliable and incomplete.

2.1.4 Invalid Data. Invalid data can result from input error, specification error, or a knowledge error[2]. If KB data is invalid, the KB is incomplete, since the valid form of the data is missing from the KB. For example, if the KB should contain the information "THE SKY IS BLUE", but actually contains the information "THE SKY IS RED", the KB is invalid since the information "THE SKY IS RED" is wrong. In addition, the correct instantiation of "THE SKY IS BLUE" is not represented so the KB does not contain a complete specification of SKY.

2.2 Knowledge Acquisition – Representation

Another aspect of the KA process is knowledge representation (KR), the mechanism whereby expert knowledge is stored in the target system. The choice of a KR is the responsibility of the knowledge engineer, one of whose major considerations is the handling of uncertainty – uncertainty about the knowledge represented, about the validity of the knowledge, and about the translation of the knowledge[23]. Most things cannot be categorized as either true or false. In many cases it is more a degree of correctness that must be represented. While there is often a notion of cause and effect between events, our understanding of the exact nature of what is going on is incomplete[4]. Hence uncertainty leads to incompleteness and the argument for a method to attach semantic meanings of likelihood to the represented knowledge. Popular methods use probability, making use of the strong theoretical foundation of probability theory. Bayesian networks[4, 19, 17] are a method of KR where probabilistic information is incorporated into the knowledge represented. Specifically, an intuitive "if-then" type rule structure, linked to the reasoning model employed by experts[3], contains attached probabilistic data related to the likelihood that specific events occur.

[2]It is possible for knowledge to be thought correct, and yet be proved false at a later time. For example, the drug thalidomide was once thought a safe sedative, only to be later discovered to have serious side-effects for the unborn fetus of pregnant women[14].

2.2.1 Bayesian Knowledge Base (BKB).

An extension of the Bayesian Network representation, used in the development of PESKI (see Appendix A), is the Bayesian Knowledge Base (BKB)[2, 23, 9, 22]. Components or random variables (rvs) represent events and/or objects, and each of these may assume one or more states or instances. Thus, if the object "SKY" is an rv, its instantiations could include "SKY = clear blue", "SKY = grey", or "SKY = black". It is also important to note certain objects may be independent of others (e.g. "SKY = grey" has nothing to do with "CAR = fiat"). Independence,however, cannot be universally applied else meaning is lost ("SKY = grey" is not independent of "RAINING = true"). Bayesian approaches avoid oversimplification by couching their independence assumptions in terms of conditional dependencies[3][22].

The BKB representation is a directed graph consisting of s-nodes, i-nodes, and arcs that connect the two node types[4]. I-nodes, or instance nodes, correspond to individual rv instantiations (rv states). I-nodes must have at least one inbound arc from an s-node to establish their probability, and may have zero or more outbound arcs[22]. S-nodes, or support nodes, contain probabilistic values. They have exactly one outbound connector, leading to an i-node, and zero or more inbound arcs. Intuitively, each i-node has one or more parent s-nodes, and one or more child s-nodes. S-nodes have exactly one child i-node, and zero or more parent i-nodes (Figure 2.1).

I-nodes with an outbound connection to a specific s-node form a support condition for the i-node at the terminus of the s-node's outbound arc. Each i-node so attached to the s-node is referred to as a tail condition for the support. A support condition contains those rv-instances (tails) that must be active for the supported i-node (the child of the s-node) to be active. Thus if "Cough" and "Well" are rvs, and if "Cough = true" implies "Well = false", "Cough = true" forms a support condition with one tail (namely "Cough = true") for "Well = false". The probability for

[3]For a discussion of Bayes theorem, $P(A \mid B) = \frac{P(B|A)P(A)}{P(B)}$, see [4, 19, 17, 1].

[4]Arcs may not connect nodes of the same type. Thus, arcs only connect individual s-nodes to individual i-nodes, not s-node to s-node or i-node to i-node.

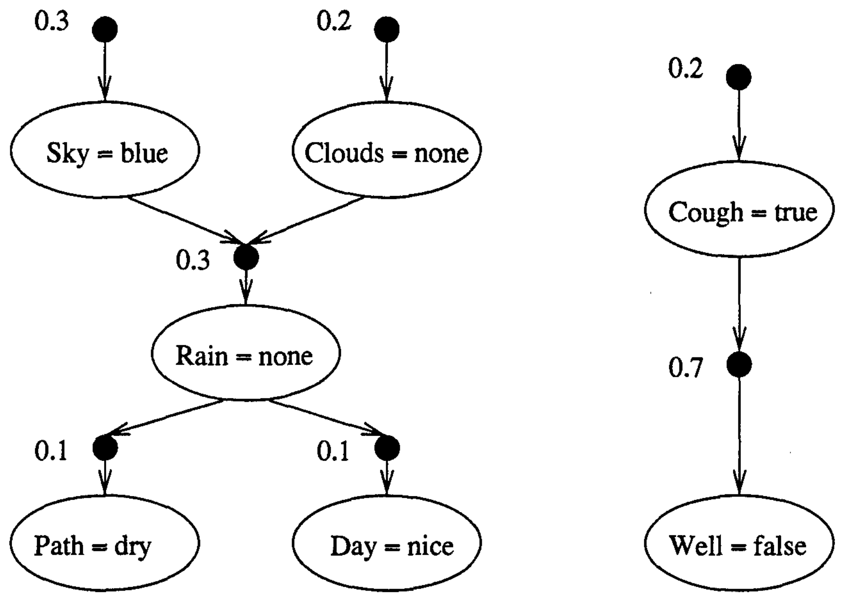

Figure 2.1 **Simple Bayesian Knowledge base.** The solid circles are **support nodes (s-nodes)** and ovals are **instance nodes (i-nodes)**. Interior text represents instance information, and is of the form "*rv name = state name*"

the event "Well = false" given "Cough = true" is stored in the s-node linking the two instantiations (see Figure 2.1).

Certain constraints of the representation are of particular interest:

1. Only one instance of an rv may be active at any one time.
2. Support conditions for a particular rv instantiation must be mutually exclusive to ensure BKB consistency.
3. Cyclic knowledge is not acceptable.

(See Appendix B for discussion of items 2 and 3. For a complete discussion on all BKB constraints, see Banks[2].)

2.3 Verification & Validation

An important stage in the development process of a KBS is knowledge-base verification and validation (V & V). This procedure ensures system reliability for specific tasks, and that minimal levels of performance are guaranteed. These latter measures of performance must be assured before widespread system use[10]. Generally verification is building the system right whereas validation refers to building the right system[18]. This research addresses the area of validation. Expert system validation, testing systems to ascertain whether they achieve acceptable performance levels, has (with few exceptions) been ad hoc, informal, and of dubious value[18]. In fact, much of the focus in V & V has been on verification, and the identification and correction of certain rule inconsistencies(redundant rules, conflicting rules, missing rules, etc.[16]). In terms of validation, and the need to check for correct system performance, a host of strategies have been devised; but there are many problems associated with validation, including what, when, and how to validate a system. However, typically, expert system validation consists of running a sequence of test cases through the system and comparing system results[18, 20].

The determination of what constitutes a good test case, how many test cases are needed to validate the system, and how to measure acceptable performance continues to be of concern[18]. However, given an acceptable suite of test cases, through the validation process, inconsistencies and errors in the KB can be identified and repaired.

2.3.1 BVAL. BVAL[9] is designed to directly compare the KB against its requirements. The requirements of the KB are a set of test cases, where each test case has evidence and an expected answer[9]. BVAL assumes the KB over which validation is performed is complete[5]. During operation, BVAL performs negative and positive reinforcement learning[6] of probabilities for a given

[5]If BVAL encounters an incompleteness problem, it will either make assumptions or disregard information and continue.

[6]If the correct solution does not return as the highest probability, all s-nodes in the connected region[9], not in the correct solution path, are multiplied by a reduction factor. This process ensures the best probability, after reduction (negative reinforcement) belongs to the solution path. Afterwards upscaling is performed (positive reinforcement), which maintains probabilistic relationships while maximizing probability values.

evidence and answer set. This method ensures that if the system is later queried with the same evidence, the highest solution probability contains the answer.

One of the limitations of BVAL is incompleteness. When BVAL is invoked with a test-case, it has no mechanism to automatically handle incompleteness problems. Central to BVAL's methods are dependency regions, which are similar to the direct dependency regions discussed in Chapter III. If problems are encountered with the test-case, in some instances BVAL arbitrarily removes test-case data (either evidence or answer items), in others it makes assumptions about certain test-case items. Specifically, if BVAL determines that an evidence element has no direct bearing on an answer element, it simply removes the troublesome item from the test-case and continues. This "modification" of the test-case is a severe limitation to BVAL's performance in that the noted problems between evidence and answer items illustrate incompleteness problems within the BKB that should be corrected. Similarly problematic is BVAL's tendancy to circumvent incompleteness problems by making assumptions. In this case, if BVAL cannot find causal links between evidence and answer, it will often "assume" the answer, despite the fact that the current state of the knowledge base and the full test-case specified does not back up the assumption. Of more concern in BVAL's performance is the actual view of the test-case itself. Key regions are computed, taking evidence items as individual elements as opposed to a unified group within a test-case. The idea of test-case dependency is discussed fully in the next section, but in general terms, all evidence items affect all answer items. However, BVAL computes regions containing elements that may hold for all evidence (or answer) items, or may hold for just some evidence (or answer) items. Thus conclusions drawn from these regions may not take into account all test-case relationship information, wherein lies the problem.

This research applies to test-cases before BVAL is invoked, determining if the KB is sufficiently complete that BVAL will successfully function on a full test-case without encountering incompleteness problems. This research identifies specific test-case anomalies and proposes techniques to modify the KB to correct such problems. Alternatively, non-automatic methods are included wherein relevant information regarding test-case inconsistencies are presented to the user.

III. Incompleteness Methods

3.1 Definitions

3.1.1 Direct Dependence.

RV instances exhibiting direct dependence on each other are connected by a strict parent or child relationship. Thus, instance A is directly dependent on instance B if there is a either a sequence of parent nodes, or a sequence of child nodes, between A and B that connect the two nodes. Formally,

Definition 1 *An rv instance A is* **directly dependent** *on an rv instance B, if and only if there exists a sequence of n rv instances* $\{A, X_2, ..., X_{n-1}, B\}$*, where n is positive integer, and*

1. *Each element* S_i *in the sequence of rv instances is an element of a support condition of* S_{i-1}*, for all i,* $2 \leq i \leq n$*,*

   or

2. *Each element* S_i *in the sequence of rv instances is an element of a support condition of* S_{i+1}*, for all i,* $1 \leq i \leq n-1$.

3.1.2 Indirect Dependence.

RV instances exhibit indirect dependence on each other if there is a connection between them, consisting of parent and child nodes. Formally,

Definition 2 *An rv instance A is* **indirectly dependent** *on an rv instance B, if and only if there exists a sequence of n rv instances* $\{A, X_2, ..., X_{n-1}, B\}$*, where n is a positive integer, and*

1. *Each element* S_{i+1} *in the sequence of rv instances is an element of a support condition of* S_i*,*

   or

2. *Each element* S_i *in the sequence of rv instances is an element of a support condition of* S_{i+1}*,*

for all i, $1 \leq i \leq n-1$.

3.1.3 Test-Case. A test-case is a collection of rv instances separated into evidence and answer sets. The test-case is limited by the fact that neither set contains more than one instance of a particular rv; and no rv is present in both the answer and evidence sets. Formally,

Definition 3 *Let A be the set of all rv instances specific to the test-case as answers.*

Let E be the set of all rv instances specific to the test-case as evidence.

Let $rv(A_i)$ be the random variable of the i^{th} element of A.

Let $rv(E_j)$ be the random variable of the j^{th} element of E.

- *For all elements A_i in A, there does not exist an element A_j in A, $i \neq j$, such that the $rv(A_i) = rv(A_j)$.*
- *For all elements E_i in E, there does not exist an element E_j in E, $i \neq j$, such that the $rv(E_i) = rv(E_j)$.*
- *For all elements A_i in A, there does not exist an element E_j in E such that the $rv(A_i) = rv(E_j)$.*
- *For all elements E_i in E, there does not exist an element A_j in A such that the $rv(E_i) = rv(A_j)$.*

3.2 Test-Cases

As previously mentioned, test-cases are an acceptable and widely used method to validate KBSs[18, 20]. For the purpose of this research, certain assumptions about test-case design were made. By definition, a test-case consists of a set of evidence items and a set of answer items (see above). A further constraint is the nature of the evidence and answer items. For the purpose of the automatic portion of this research, it is assumed that the entered case is correct in its entirety. A correct test-case, while in essence a query of the system, is also a statement of fact: Given the specified evidence, the specified answers **are** true. In addition, each evidence item contributes to each answer item, and vice versa. If this were not the case, the test-case could be as easily specified without the non-contributor. Consider the following example: Suppose a test-case consists of three pieces of evidence – "A = 1", "B = 1", and "C = 1"; and two answers – "X = 1" and "Y = 1". Given the evidence, the answers should be true. If, however, "B = 1" has no direct influence on the answers, the test-case could be better expressed without "B = 1" in the evidence set (the answers are independent of "B = 1"). Thus, assuming the correctness of the test-case as specified, each evidence item should directly affect each answer item. In essence the evidence items should have an overlap in the instances they affect in the KB, and each answer should reside in this area. More

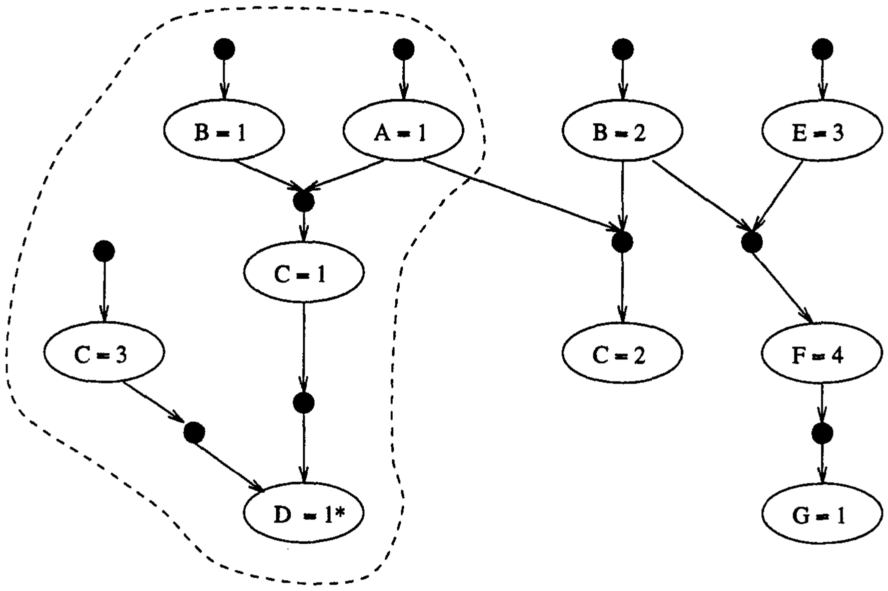

Figure 3.1 **Direct Dependency Region.** Region within dotted line is direct dependency region of evidence item "D = 1*".

formally, this additional test-case constraint is as follows:

- For each item E_i in the evidence set, every item A_j in the answer set is directly dependent on E_i. Thus all answer items are in the direct dependency region[1] of every evidence item(see

[1]The direct dependency region of any rv instance, X, is the set of all rv instances in the BKB directly dependent on X.

Figure 3.1). 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.

NOTE: This final constraint is key to the determination of incompleteness paths discussed in the next section.

3.3 Incompleteness Paths

As mentioned, correct test-cases hinge on the notion of direct dependence. If such a dependence cannot be established with the test-case (i.e. an answer instance is not in the direct dependency region of some evidence element), it indicates incompleteness. Incompleteness results from any combination of the following:

1. Missing rvs.
2. Missing instantiations.
3. Missing links.
4. Incorrect links.

Incompletenesses resulting from items 1 and 2 is not addressed in this research; therefore it is assumed that all the needed rvs and states are present in the BKB. The current focus is on the incompleteness caused by linking problems, since the test-case format specifies a direct link between evidence and answer items. If there is no connection between the particular evidence and answer instances, there is no information to guide potential connections between the two to resolve the incompleteness other than the fact that the two are somehow dependent on each other since they belong to the same test-case. To allow the test-case to be submitted to BVAL, direct dependencies must exist, and therefore links must be corrected or created to facilitate this. There are many possible "fixes" that will put an answer instance, not previously in an evidence's dependency region, into a dependency region of an evidence item. The goal is to determine a "fix" that is better than an arbitrary placement. Thus, information about connections between the evidence and answer items is gathered and used to direct the fixing mechanisms.

Connections between i-nodes can be complex, but the simpler the connection, the more direct the information relating two i-nodes[2]. Connections between i-nodes are formed from a combination of straight links between interior i-nodes. In the simplest case, there is a straight link between i-nodes, either a strict parental or child link; however, in the complex case, there is a combination of child and parent links between interim nodes, with the source and destination nodes forming the end-points of a complicated "zig-zag" type path. As mentioned, the simpler the connection between nodes, the more the direct the relationship between them. For this reason, only four "simple" connections were used, composing two distinct path types: Straight , and "V". Further variations on the path types (i.e. increasing the "zig-zag"), make i-nodes more removed from each other, and thus their relationship to each other weaker and less clear. Thus paths more complex than the simple "V" are not examined since it is difficult to determine what useful information (if any) they could offer to the system.

3.3.1 Straight Paths - Case 1. Case 1 involves paths that consist strictly of a child link[3] between an element in the upper evidence dependency region and the answer item (Case 1 High), or strictly of a parental link[4] between an element in the lower evidence dependency region and the answer item (Case 1 Low). Case 1 paths are shown in Figure 3.2.

Because of the provisions of a test-case, the answer is directly dependent on the evidence. However, in the high case, the answer is directly dependent on an item in the upper dependency region of the evidence (see Figure 3.2 (a)). While simply moving the answer to an arbitrary position within the dependency region would correct the incompleteness problem, it does not use

[2]Connections can be thought of as familial relationships. A tie, such as a grandparent or a great-grandchild, is a stronger, more direct causal link than, say, a great aunt or a step-sister.
[3]Successive elements in the path are some child i-node of the preceding path item.
[4]Successive elements in the path are some parent i-node of the preceding path item.

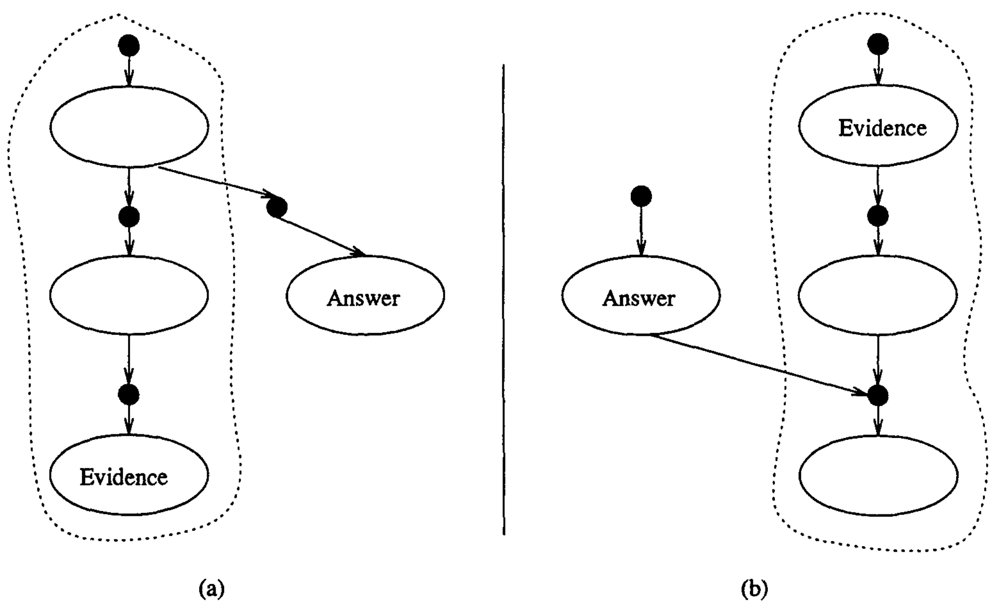

Figure 3.2 **Case 1 Straight Paths.** Dotted lines show direct dependency region of evidence items. (a) High case - path originates at instance in parent portion of evidence dependency region. (b) Low case - path originates at instance in child portion of evidence dependency region.

the information stored in the BKB representation to make an intelligent placement. When moving the answer, most links are kept intact to maintain as much of the relationship information as possible between the answer and other i-nodes. The correction removes all instances of the next to last item in the path from the supports of the answer. The evidence instance is then added to each support, and duplicate supports removed (see Figures 3.3, 3.4). Formally, this procedure is as follows:

- Let $dr(E)$ be the dependency region of evidence E, such that an answer A is not in this region.
- Let P be a path of length n, from an element X in the upper $dr(E)$ to A, such that $P = \{p_1, p_2, p_3, ..., p_{n-1}, p_n\}$, where $p_1 = X$ and $p_n =$ A, and p_{i+1} is a child of p_i for $1 \leq i \leq n-1$.
- Let $rv(p_{n-1})$ be the rv of path element p_{n-1}.

- Remove all arcs between states of $rv(p_{n-1})$ and the support s-nodes of A.
- Add E as a support tail for each support of A, by adding arcs from E to each s-node support of A.
- Remove duplicate supports of A.

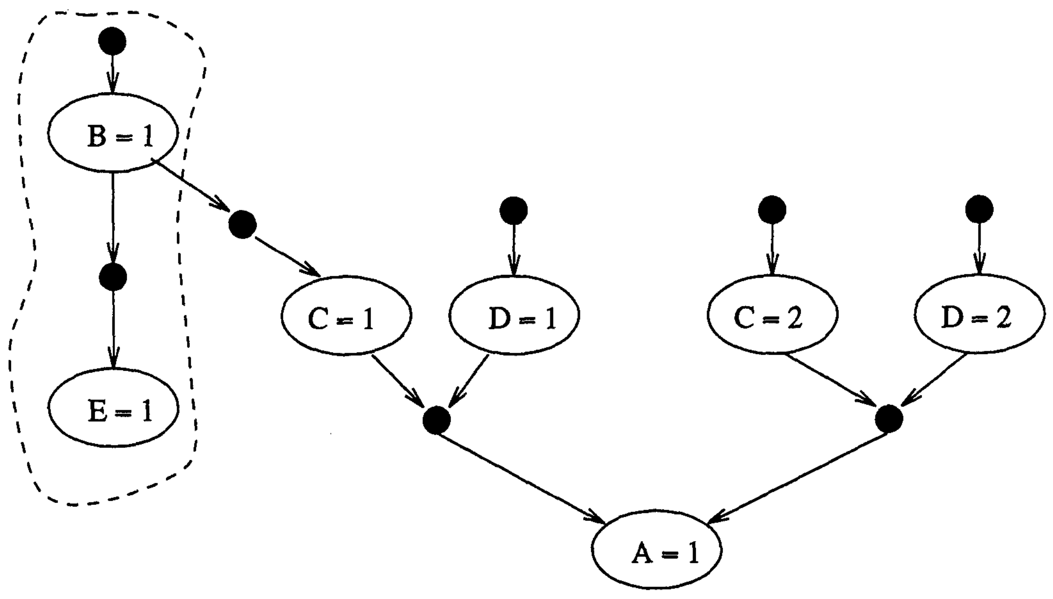

Figure 3.3 **Case 1 High - Before Fix.** Dotted line shows direct dependency region of evidence item before fix is made. Evidence is "E = 1", answer is "A = 1".

If the BKB was consistent before the change, it remains so afterwards as well. Of concern, however, is whether the movement will cause other evidence/answer problems – will moving the answer so it is in the dependency region of one evidence item cause it to move out of the dependency region of another? Suppose the answer is moved to be in the dependency region of evidence "E = 1", and all instances of rv "C" were removed as part of the move (see Figures 3.3, 3.4). If the answer was in the upper part of the dependency region of another evidence item, say "E = 2", it would still be there after the change, since no changes are made to the child connections of the answer. If the lower dependency region of "E = 2" contained an i-node in the support of the answer (not

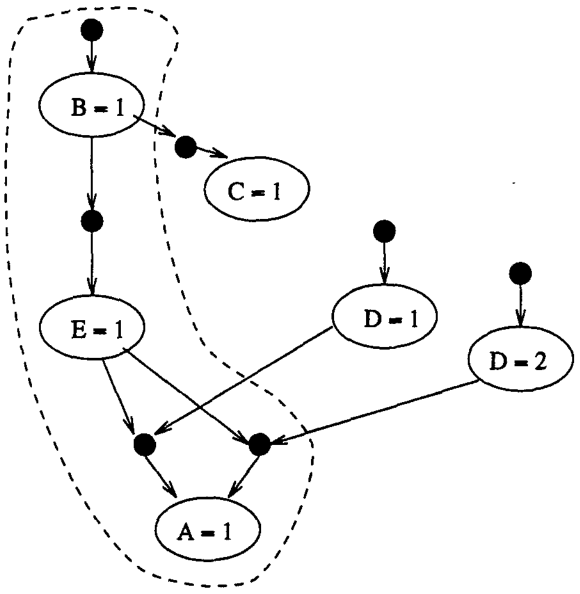

Figure 3.4 **Case 1 High - After Fix.** Dotted lines show direct dependency region of evidence item after fix is made. Evidence is "E = 1", answer is "A = 1".

an instance of rv "C"), it still contains that i-node after the change and, therefore, the answer is still in the dependency region of "E = 2". Thus, the only way the move will cause the answer to move into one dependency region and out of another, is if the only connection between "E = 2" and the answer was a child link through one of the instances of "C" removed from the supports of the answer. But some, if not all, of the removed instances are connected through parental links to an element in the upper dependency region of "E = 1", and to an element in the lower dependency region of "E = 2". In this instance, after the move, the answer still remains in the dependency region of "E = 2". Thus, while it is possible that moving the answer will correct one problem and cause another, it occurs in a relatively few number of cases, and was never actually encountered in the process of this research.

By moving the answer as previously described, little meaning was lost and the correct dependency established. The answer still depends on the first element in the path (via the evidence item); and the specific i-node tail within the path can still be active while the answer is on since it depends only on the first path item. For the test-cases examined in this research, this fix corrected support problems, reinforcing and creating more intuitive causal relationships (see Appendix C.1).

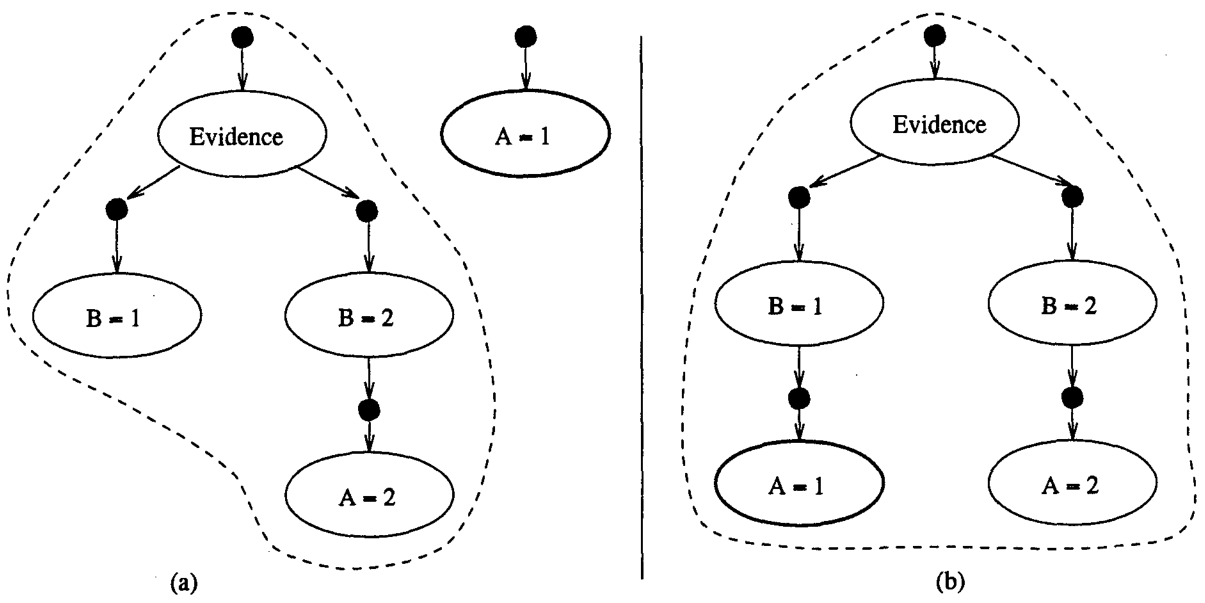

Figure 3.5 **Case 1 Low "Special Case".** Dotted lines show direct dependency region of evidence item. (a) Before change. Answer, "A = 1", is outside of dependency region. Note other instance of answer (namely "A = 2"), is in dependency region. (b) After change. Answer is now in dependency region, supported by "B = 1". Note "B = 2" supports alternate instance of answer rv, (namely "A = 2").

Like the high case, in the low case the answer is directly dependent on an element in the dependency region of the evidence. However, this element is in the lower dependency region as opposed to the upper region (see Figure 3.2 (b)). Key to the corrections made in this case is the relationship between i-nodes. In Bayesian Networks, all required connections between nodes are present; however in BKBs, part of the strength of the representation is that complete connections are not necessary. In a sense, it is understood that all nodes may influence all other nodes, but only certain relationships are actually enumerated. Within this case, this implicit relationship is exploited – the idea that whether or not the relationship is explicit in the BKB, rv states affect other rv states with certain probabilities (which could be zero). Before the general low case fix is discussed, a "special case" is examined that draws heavily on the previously mentioned implicit relationship between rv states.

The presence of an alternate state of the answer in the lower dependency region of the evidence item is considered the "special case". For example, given "A = 1" is the answer, and "A = 2" is in the dependency region of the evidence, the special case looks at the supports of the alternate instance "A = 2" and checks to see if support tails in the evidence dependency region have a sibling also in the region. If so, this sibling is added as a support for the answer (See Figure 3.5 (a)). This can be done for the following reasons:

- "B = 2" implies "A = 2" (given).
- "B = 1" implies "A = 2" (probability could be zero).
- "B = 1" implies "A = 1" (probability could be zero).

These relationships would be present if the knowledge base were fully connected like a Bayesian network. The connection, therefore, makes sense and the new/modified answer support probability, initially set to a default value, should be corrected during validation.

The general low case also uses the implicit relationship between i-nodes as mentioned. The fix works upward through the dependency region of the evidence, from the support of the item in the dependency region leading to the answer. As the procedure works its way upward, for each i-node encountered, it checks its supports to see if sibling instances[5] of the i-node's supporting tails are in the dependency region of the evidence. If so, the procedure checks if these other instances appear as tail conditions for some support of the i-node. If they do not, instances of the rv found

[5]Sibling instances of an i-node, say "I = 1", are those i-nodes who share the same rv as "I = 1", but are different states. Thus if rv "I" had three states, "1", "2", and "3", the sibling instances of "I = 1" would be "I = 2" and "I = 3".

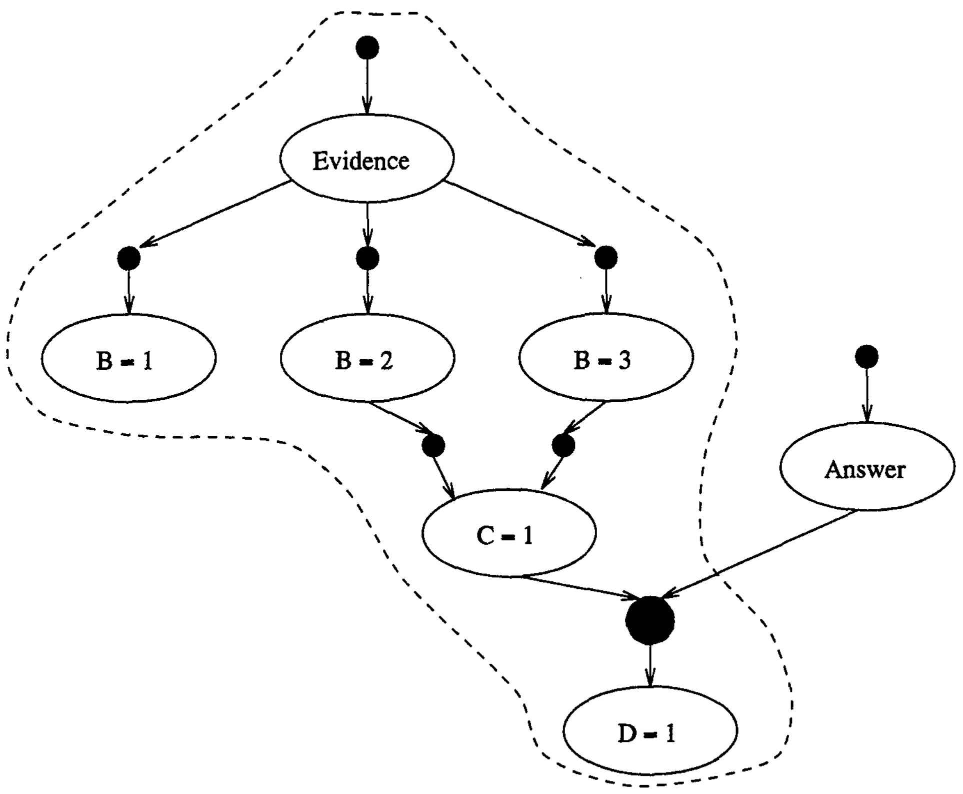

Figure 3.6 **Case 1 Low - Before Fix.** Dotted lines show direct dependency region of evidence item. The large filled circle is the s-node support from which movement toward the evidence begins.

within the dependency region are added as supports for the answer (see Figures 3.6 and 3.7). If a point is reached where further traversal upward within the dependency region is not possible, or the evidence item is reached, the default fix is applied[6]. Formally, the procedure for the general fix is as follows:

- Let $dr(E)$ be the dependency region of evidence E, such that an answer A is not in this region.

[6]The default fix simply adds the evidence as a direct support for the answer.

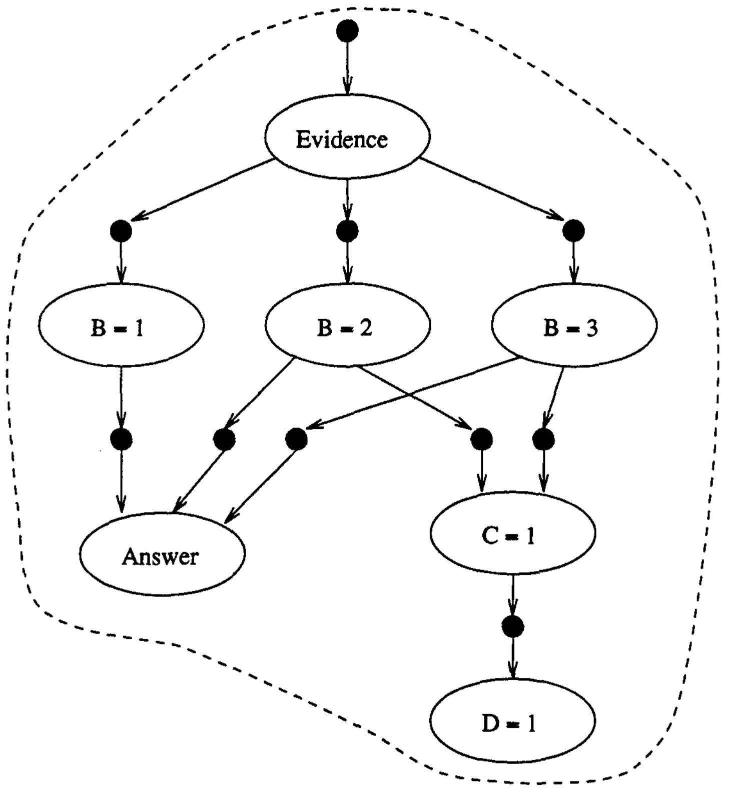

Figure 3.7 **Case 1 Low - After Fix.** Dotted lines show direct dependency region of evidence item.

- Let P be a path of length n, from an element X in the lower *dr*(E) to A, such that $P = \{p_1, p_2, p_3, ..., p_{n-1}, p_n\}$, where $p_1 = X$ and $p_n = A$, and p_{i+1} is a parent of p_i for $1 \leq i \leq n-1$.
- Let S be the s-node supporting p_1, one of whose tails is p_2.
- For each tail of S, T_s, inside the lower *dr*(E),

  1. for any support tail of T_s, $Tail_i$, in the lower *dr*(E), is there a sibling instance of $Tail_i$, $Sibling_i$, in the lower *dr*(E).
  2. if $Sibling_i$ is in the lower *dr*(E), and $Sibling_i$ is not a tail in any of the supports of T_s, add $Tail_i$ and all $Sibling_i$ in the lower *dr*(E) as supports for the answer p_n.

3. if no change is made, repeat steps 1 thru 3 with T_s equal to S.

- If no changes are made, add evidence E as support for answer A.

It is important to note the basic difference in the high and low fixes of this case. Case 1 Low places the answer in the dependency region of the evidence simply through the addition of supporting answer links. Thus existing dependencies between i-nodes are unaltered, only the direct dependency known to exist between the evidence and answer (see Section 3.2) is added to the KB. In short, Case 1 Low adds links previously missing in the KB.

Case 1 High, however, does not simply add missing links; rather it corrects links. Again the relationship between answer and evidence mandated by their inclusion in the test-case is the driving factor. Before the fix is made, the relationship between key i-nodes is as follows (see Figure 3.3):

1. The evidence, "E = 1", is directly dependent on "B = 1".
2. The answer, "A = 1", is directly dependent on "B = 1".
3. "A = 1" is directly dependent on "D = 1" and "D = 2".
4. "A = 1" is directly dependent on "C = 1" and "C = 2".
5. "A = 1" is indirectly dependent on "E = 1".

The correction results in the following relationships (see Figure 3.4):

1. The evidence, "E = 1", is directly dependent on "B = 1".
2. The answer, "A = 1", is directly dependent on "B = 1".
3. "A = 1" is directly dependent on "D = 1" and "D = 2".
4. "A = 1" is indirectly dependent on "C = 1".
5. "A = 1" is directly dependent on "E = 1".

Thus the decision is to switch the dependency relationship between the evidence-answer pair and the item-answer pair[7] (compare relationships #4 and #5). This is performed since the evidence-answer direct dependency is judged more important than the other direct dependency because of the nature of test-cases (see Section 4.1 for a specific application of the high path fix).

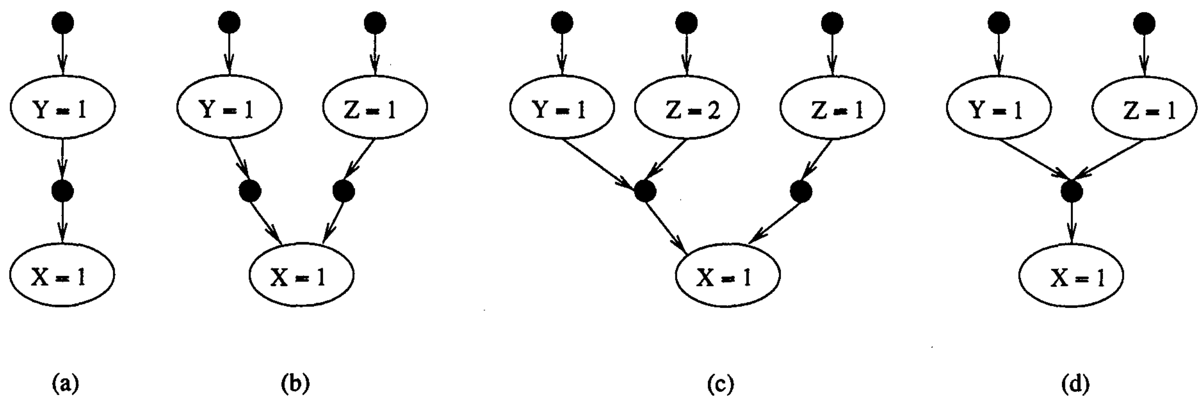

Figure 3.8 **Case 1 Low - Mutex Problems and Corrections..** (a) Before a change is made, "X = 1" has a single support, "Y = 1". (b) As part of the Case 1 Low fix, "Z = 1" is added as a new support for "X = 1". The two supports are not mutually exclusive, since "Y = 1" and "Z = 1" can both be active at the same time. (c) A correction to the mutex problem that adds an alternate instance of "Z", namely "Z = 2" as another tail of the "Y = 1" support. (d) Another mutex correction joins the two supports into one.

Another key difference between the high and low fix is the idea of mutual exclusion (mutex). As previously mentioned, the high fix maintains BKB consistency; however the low fix, as it adds supporting answer links, often causes mutex problems (see Section B.1 for a more complete discussion of mutual exclusion). These problems arise when the answer item already has a non-empty support[8]. If an i-node, say "X = 1", is supported by a single s-node with no tails, the addition of a support tail (from the low case) to this s-node clearly does not cause mutex problems. If,

[7]The item in this pair is the next to last element in the Case 1 High path, and its associated siblings. In this example, this item is "C = 1", along with "C = 2".
[8]A non-empty support is a support that has at least one tail condition.

however, "X = 1" has a non-empty support of "Y = 1" (see Figure 3.8(a)), and the low fix resulted in making "Z = 1" another support for "X = 1" (Figure 3.8(b)), mutex problems result between "Y = 1" and "Z = 1", since both can be true simultaneously. There are a number of corrections that can be made to resolve the mutex problems, from combining supports to the addition of new tail conditions (see Figure 3.8(c) and (d)). Since there are so many possible resolutions to the mutex problem (see Section B.1), optimal resolution depends on the availability of further information that can either eliminate poor mutex corrections, or indicate correct ones. Thus current methods, which do not utilize outside resources, do not attempt to automatically resolve mutex disputes; however, such methods would be invaluable (see Section 6.2 and Chapter V). Section 4.2 details a specific application of the low path fix and the resulting mutex problems.

3.3.2 "V" Paths – Case 2. Case 2 paths involve connections between an i-node in the dependency region of the evidence, and the answer, where the path consists of a combination of a strict parent edge and a strict child edge meeting at a pivot point (which can be either an i-node or an s-node). Like Case 1, there is a high and a low path. In the high path, the first element in the path is in the upper dependency region of the evidence, hence the "V" shape is right-side-up (see Figure 3.9). Unlike the high path, the first element in the low case path is in the lower dependency region, hence the path has an inverted "V" shape (see Figure 3.10). Note that in the high case there are two possible pivot points in the "V", either at an i-node[9] (Figure 3.9 (a)), or at an s-node (Figure 3.9 (b)). However, there is only a single low case, namely an i-node pivot[10].

Whether high or low, each "V" consists of two "straight" edges, one from the dependency region item to the pivot, and one from the pivot to the answer. By breaking each "V" path into two straight cases and performing Case 1 fixes, the answer will end up in the dependency region of the

[9]While only one s-node can be on at any time for a given i-node, it is possible for both supports tails to be active (if the s-node probabilities are equal). Thus, this is a valid path.

[10]If the pivot point in the low case were an s-node, it would mandate two outbound arcs from the s-node for each arm of the inverted "V" path. This would contradict the representation convention that each s-node has only a single originating arc.

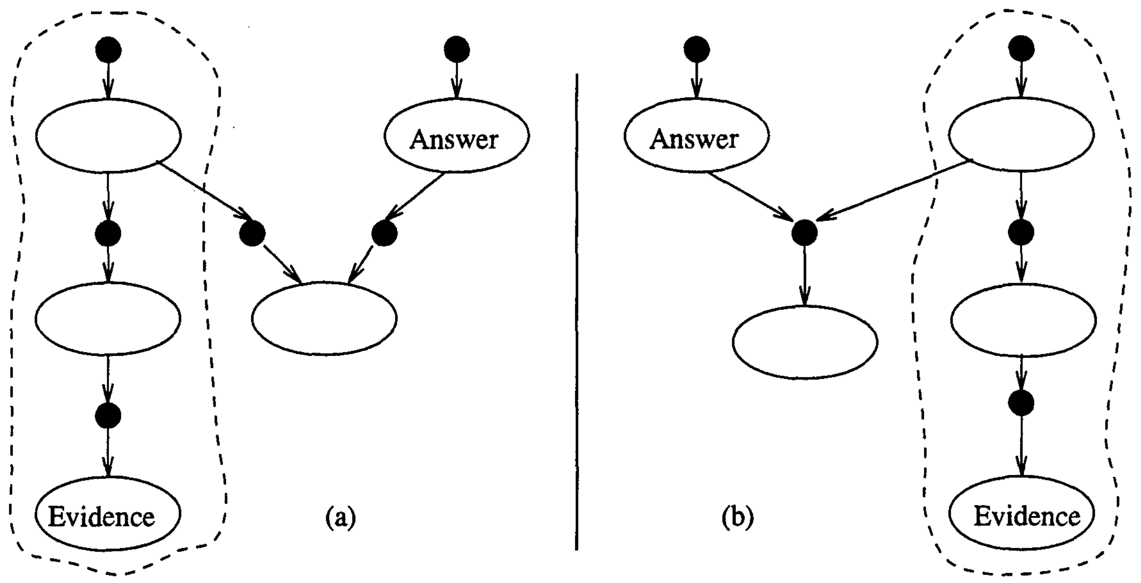

Figure 3.9 **Case 2 High "V" Paths.** Dotted lines show direct dependency region of evidence item. (a) Pivot point in "V" path is i-node. (b) Pivot point in "V" path is s-node.

evidence as desired by the test-case. For the high case, the portion of the "V" path to the pivot[11] is submitted to the Case 1 High method, resulting in the pivot's placement in the lower dependency region of the evidence. A subsequent Case 1 Low fix of the path from the pivot to the answer will ensure the answer's place in the evidence dependency region. A similar strategy applies to the low "V" case, except that the pivot is first included in the evidence dependency region using the Case 1 Low method. Once the pivot is placed in the lower dependency region of the evidence, per the Case 1 Low fix, no further changes need to be made to get the answer in the evidence dependency region[12].

[11]If the actual high case pivot is an s-node, the i-node supported by the s-node is considered the pivot.

[12]Since the answer is in the lower dependency region of the pivot through a strict child connection, by definition of direct dependency, it becomes part of the dependency region of the evidence once the pivot is moved.

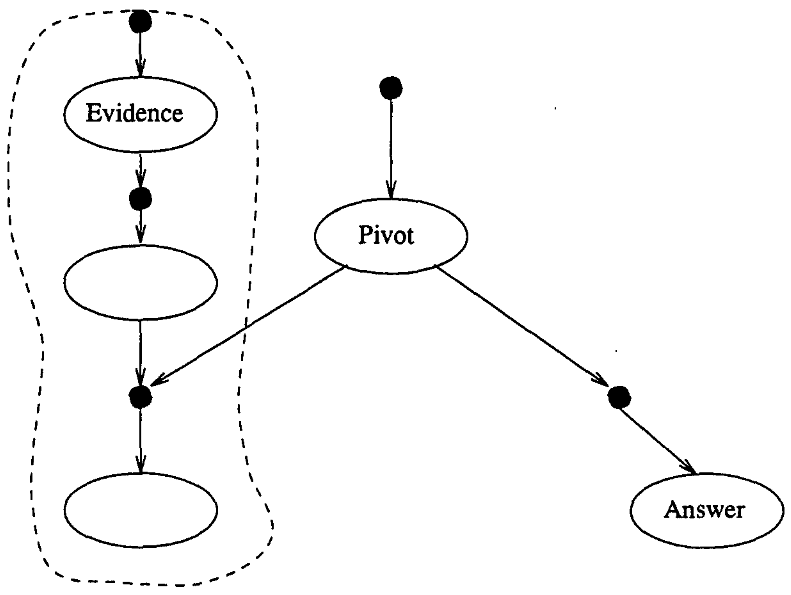

Figure 3.10 **Case 2 Low "V" Path.** Dotted lines show direct dependency region of evidence item. Pivot point in "V" path is i-node.

IV. Results

RV Name	Healthy Fish/Tank State	Unhealthy Fish/Tank State
Fin Position	spread	clamped to body
Breathing	normal	rapid; labored
Nitrite level	0	> 0
Appetite	normal	limited/none
Stress level	none	mild;moderate;major
Organ failure	false	true
Scales	normal	protruding; missing
Fish color	normal	yellowish; greyish; poor
Fish appearance	healthy	swollen/bloated; thin/weak
Fish behavior	normal	listless; darting; scratching; poor
Diseases	false	true
Clarity	clear	murky; powdery
Color	clear	yellow; green; yellow and green
Odor	none	decaying
Ammonia level	0	> 0
Cleanliness	clean	brown/green film or coating
Algae	none	present
Chloramine	not present	present
Nitrate	not Present	present
Fish waste	not present	present
Decaying food	not present	present
Water quality	good	poor

Table 4.1 **Data source for healthy and unhealthy goldfish and conditions.** Random variable "Diseases" can represent any of a number of disease rvs represented in the KB.

The previous chapter presented the methods used for resolving incompleteness problems within test-cases. The following are selected evidence/answer mismatches which illustrate each path and the effect of the corrections made. Table 4.1 includes some of the data used to create the test-cases, and Appendix C contains more complete transcripts of the following examples, including each example's full test-case description.

4.1 Case 1 - High Straight Path

This test-case addresses unhealthy tank conditions:

```
Evidence is:
[Objects] Cleanliness = Brown/Green film or coating

Answers are
[W] Fish waste = Present

case1 hi - item in upper portion of DR
[W] pH Level = Less than 6.0
[W] pH Level >= 7.2 = False
[W] Fish waste = Present
```

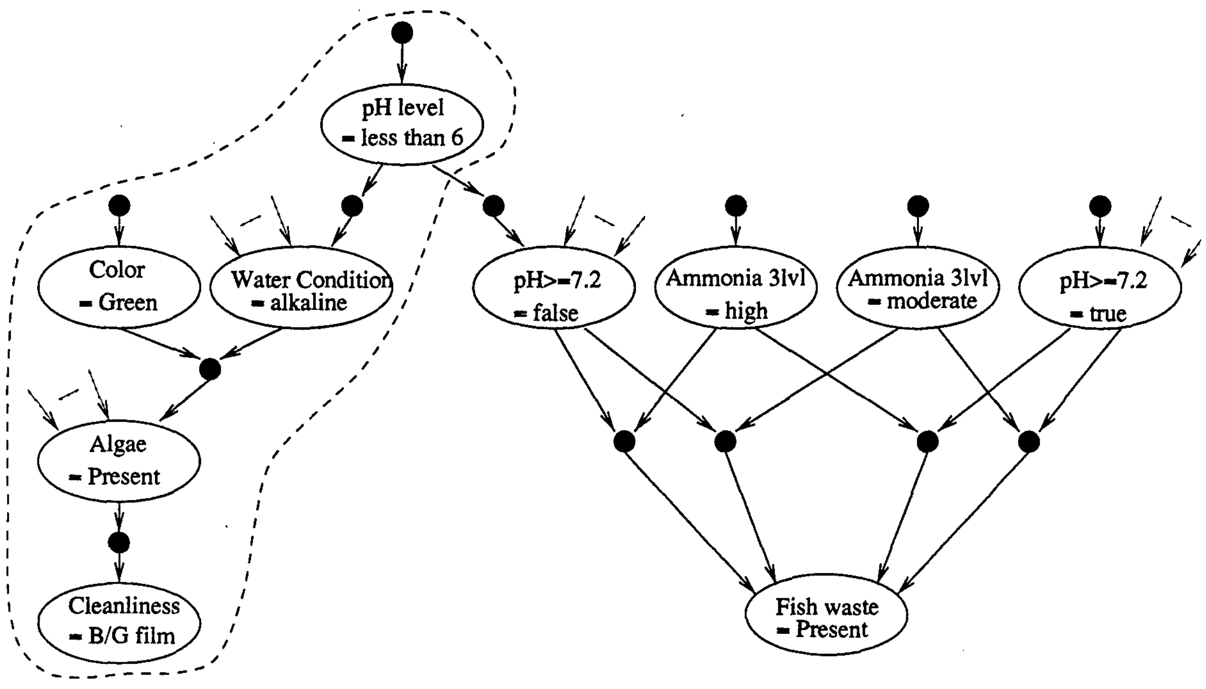

Figure 4.1 **Case 1 High - BKB Structure Before Change.** Graphical depiction of selected instances in dependency region of "Cleanliness = Brown/Green Film or coating", path to answer "Fish waste = Present", and answer supports. Dotted line arrows represent multiple i-node supports not pictured.

As illustrated in Figure 4.1, the answer depends on the random variables "pH >= 7.2" and "Ammonia 3-lvl". However, because of test-case restrictions, the answer must be directly influenced by the evidence "Cleanliness = Brown/Green film or coating". After the correction (see Figure 4.2),

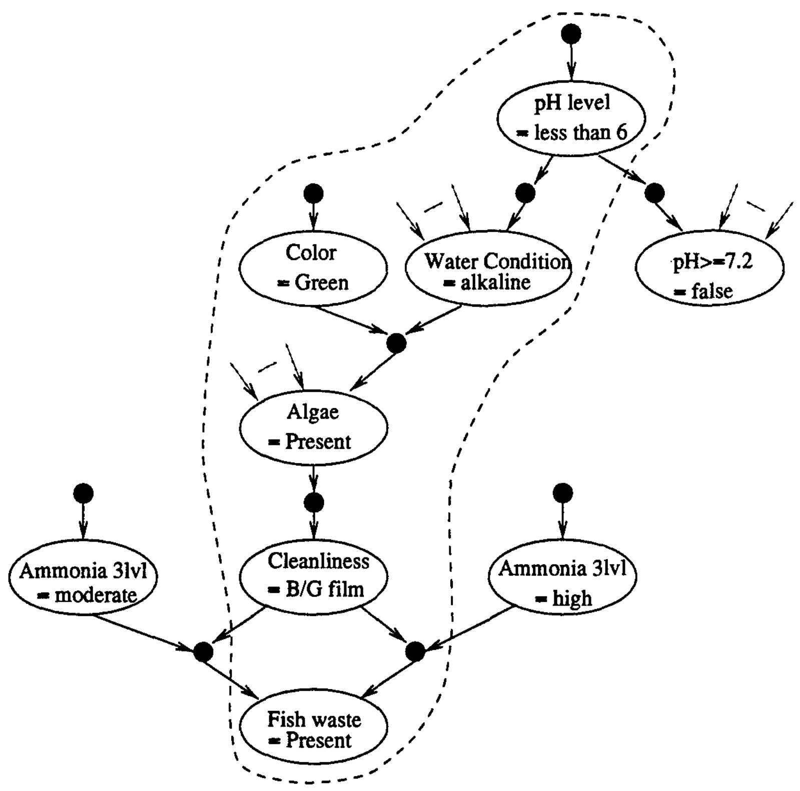

Figure 4.2 **Case 1 High - BKB Structure After Change.** Graphical depiction of selected instances in dependency region of "Cleanliness = Brown/Green Film or coating". Note answer "Fish waste = Present" is now in this region. Dotted line arrows represent multiple i-node supports not pictured.

this is indeed the case but the the answer's immediate supports no longer contain instances of "pH >= 7.2", which are replaced by the evidence instance. This fix corrected a weakness in the BKB since pH level really has no direct bearing on the presence of fish waste in the tank. The important tail of the answer supports is related to the ammonia present in the tank, which is a direct consequence of waste material (whether fish waste or uneaten food) remaining in the water. This condition remains as an immediate support, along with the evidence, which also intuitively makes sense as being a probable result of the answer condition. In this case, both instances of the rv "pH >= 7.2" can still be true, despite the severing of their links to the immediate supports of the answer. This is because all instances of the rv "Water Condition" depend on all instances of the "pH level" rv, which in turn affects all instances of "pH >= 7.2". Thus, no real meaning is lost in the BKB as a result of this correction; rather the direct link between "pH >= 7.2" and the answer is made an indirect link, and the indirect link between the evidence and the answer is made direct. Thus, the correct dependency, based on the test-case, is established and the consistency of the BKB is maintained.

4.2 Case 1 - Low Straight Path

This test-case addresses healthy tank conditions:

```
Evidence is:
[W] Nitrite Level = 0 ppm

Answers are
[W] Water Quality = Good

case1 low - item in lower portion of DR
[F] Parasites = Present
[W] Water Quality = Good
```

As mentioned in Section 3.3, the low case does not move links like the high fix in an attempt to correct incorrect dependencies; rather it create links to establish missing dependencies. Figure 4.3 shows the state of the BKB before the change is made. It is intuitive that for the quality of the water to be considered good, no harmful chemicals or materials should be in the tank. However, there is no direct connection between the evidence and answer; rather there is an indirect connection through "Parasites = present". While it is possible for the water condition to be good and parasites to be present, it seems more likely that if the water condition is good, fish will remain healthy and so will not be infected with parasites. Figure 4.4 shows the state of the BKB after a change is made. The addition of the support to the answer now makes it possible for the answer to be true:

1. based directly on the evidence.

2. with or without parasites being present.

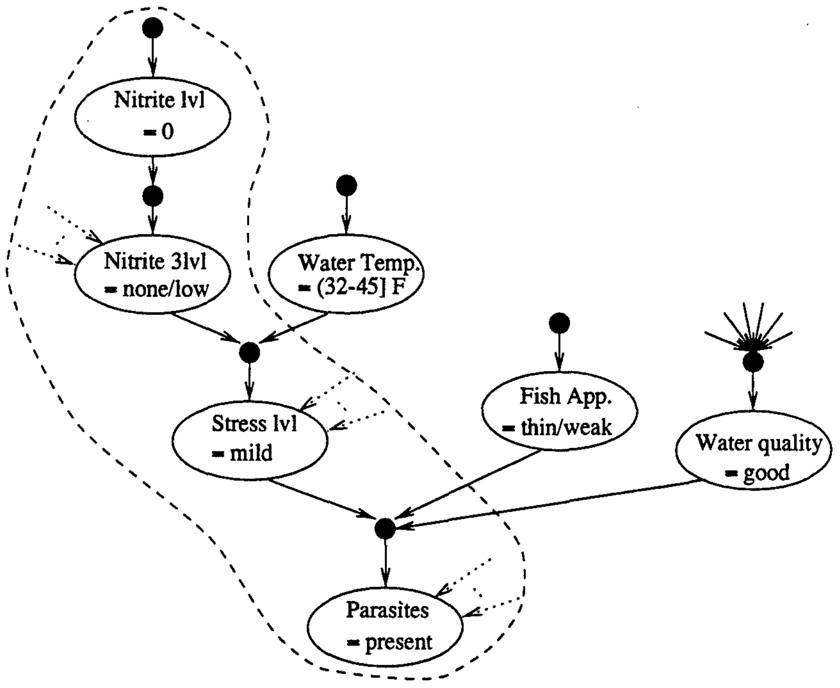

Figure 4.3 **Case 1 Low - BKB Structure Before Change.** Graphical depiction of selected instances in dependency region of "Nitrite lvl = 0", path to answer "Water Quality = good", and answer supports. Dotted line arrows represent multiple i-node supports not pictured.Note answer has one support with six tail conditions.

4.2.1 Mutual Exclusion Problems. Before the change was made, the answer had one support with six tails. These tails did not contain an instance of the evidence rv, and thus mutual exclusion problems resulted from the addition of the new support (see Appendix C.2). With knowledge of what good water quality is, it is simple to say that the evidence should not be added as a separate support, but should become part of the existing support of the answer. However, the system cannot be expected to "know" what "good" means, or to be able to apply "common sense" to the resolution of mutual exclusion problems. While in this case the simple addition of the evidence to the existing supports makes sense, in the next case it might not make sense; there may be multiple existing supports from which to choose. In addition, as mentioned in Section 3.3, and shown in Figure 3.8, there can be numerous fixes to any mutex problem, and the selection of

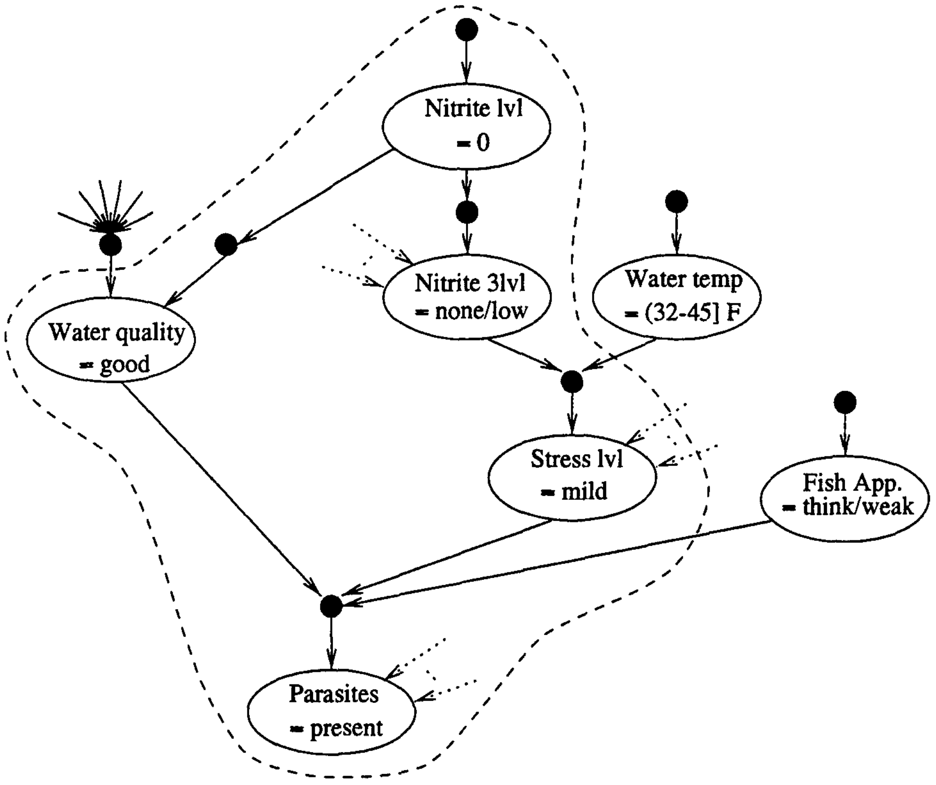

Figure 4.4 **Case 1 Low - BKB Structure After Change.** Graphical depiction of selected instances in dependency region of "Nitrite lvl = 0". Note answer "Fish waste = Present", with a new support, is now in this region. Dotted line arrows represent multiple i-node supports not pictured.

which one is best is not possible without knowledge currently not contained in the representation. Thus, the tool developed in this research does not attempt to correct mutex problems. Instead, it alerts the user (who should have the required expertise to resolve the problem) of the mutex violations (see Appendix C.2) in the hopes that they will resolve them. In the future, tools such as data-mining should provide the system with the expertise to automatically resolve such problems without user involvement (see Chapter V and Section 6.2).

4.3 *Case 2 - High "V" Path*

This test-case addresses unhealthy tank conditions:

```
Evidence is:
[W] Clarity = Murky

Answers are
[W] Water Quality = Poor

case2 ('V') hi - item in upper portion of DR
[W] Ammonia 3-Level = High
[F] Stress Level = Major
[F] Parasites = Present
[W] Water Quality = Poor
```

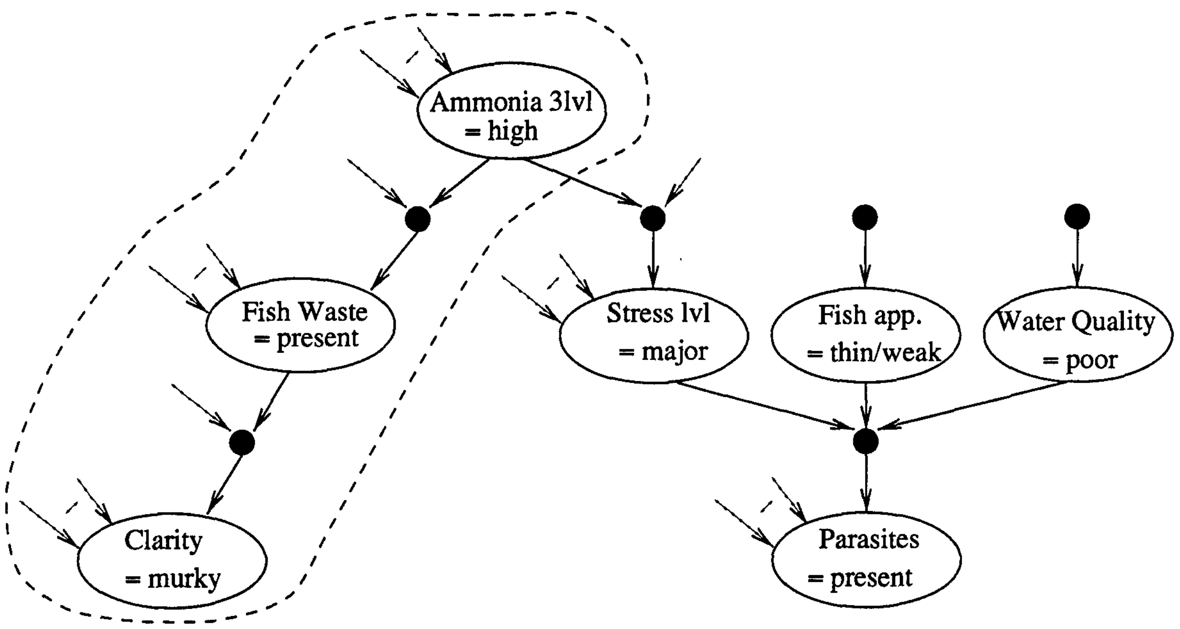

Figure 4.5 **Case 2 High - BKB Structure Before Change.** Graphical depiction of selected instances in dependency region of "Clarity = murky", path to answer "Water Quality = poor", and answer supports. Dotted line arrows represent i-node supports not pictured. Note pivot point is "Parasites = present".

As discussed in Section 3.3, this fix uses an incremental approach to getting the answer in the dependency region of the evidence, breaking the "V" of the path into two distinct pieces: the

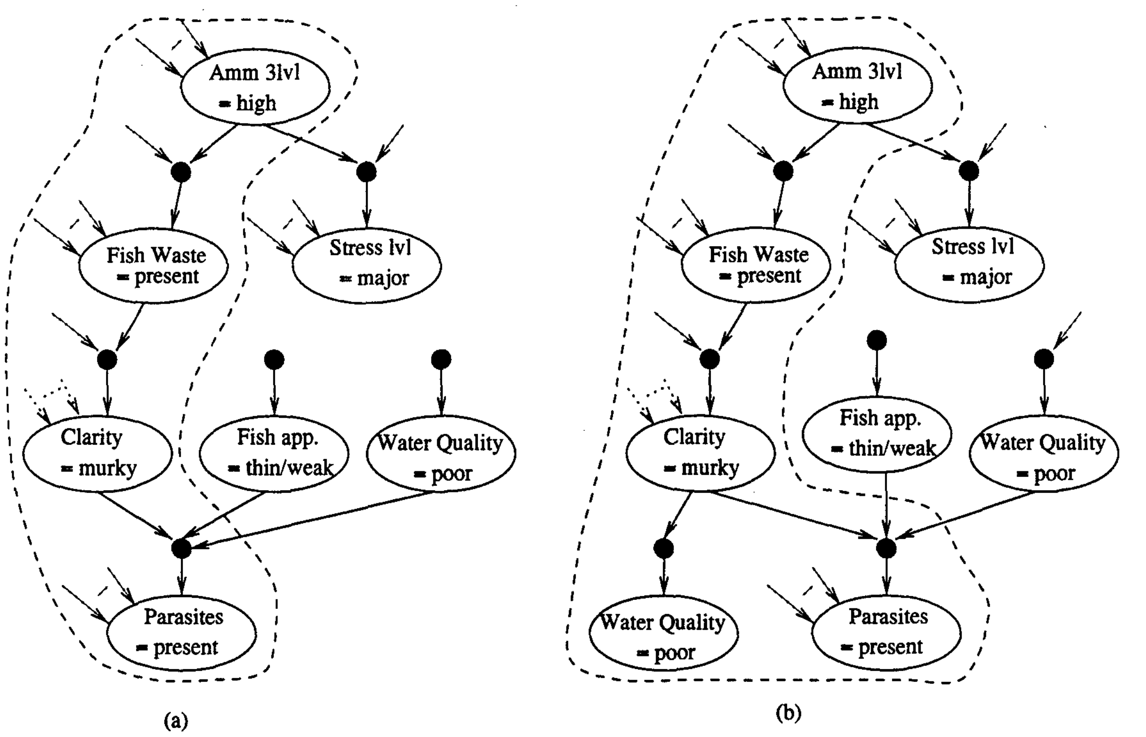

Figure 4.6 **Case 2 High - BKB Structure After Change.** Graphical depiction of selected instances in dependency region of "Clarity = murky". (a) Pivot "Parasites = present" is first placed in dependency region with a Case 1 High fix. (b) The answer, "Water Quality = poor", is placed in the region via a Case 1 Low fix. Dotted line arrows represent i-node supports not pictured.

first from the dependency region path item to the pivot, the other from the pivot to the answer (see Figures 4.5). Since the Case 1 High fix, applied to the path ending at the pivot of "Parasites = present", will result in the pivot element residing in the lower dependency region of the evidence, it is clear that the answer will not be in this region after the first path correction (see Figure 4.6 (a)). Furthermore, it is clear that the connection between the pivot and the answer now meets the requirements for a Case 1 Low fix, which will result in a direct dependency between evidence and answer (Figures 4.6 (a) and (b)).

4.3.1 Mutual Exclusion Problems. Since part of this fix involves utilization of the Case 1 Low fix, mutual exclusion can be a problem. In this particular example, however, no mutex problems arise from the correction since the answer had no non-empty supports (see Section 3.3). Again, if necessary, automatic correction would not be attempted; rather information indicating which supports were problematic would be presented to the user for their correction. (See Appendix C.3, Chapter V, and Sections 4.2 and 6.2).

4.4 Case 2 - Low "V" Path

This test-case addresses unhealthy tank conditions:

```
Evidence is:
[W] Color = Yellow

Answers are
[W] Fish waste = Present

case2 ('V')low - item in lower portion of DR
[F] Appetite = Limited/None
[F] Stress Level = Major
[W] Ammonia 3-Level = High
[W] Fish waste = Present
```

Section 3.3 detailed this path's correction as a single use of the Case 1 Low fix, using the part of the "V" path from the item in the evidence dependency region to the pivot point. Figure 4.7(a) shows the state of the BKB before a change is made; the "V" path pivot is "Ammonia 3-lvl = high". After the change is made it is clear to see why no further corrections are needed to the BKB, since the answer is now in the dependency region of the evidence item (Figure 4.7 (b)).

4.4.1 Mutual Exclusion Problems. Since part of this fix involves utilization of the Case 1 Low fix, mutual exclusion is a problem. Again, automatic correction is not attempted; rather information indicating which supports are problematic is presented to the user for correction. (See Appendix C.4, Chapter V, and Sections 4.2 and 6.2).

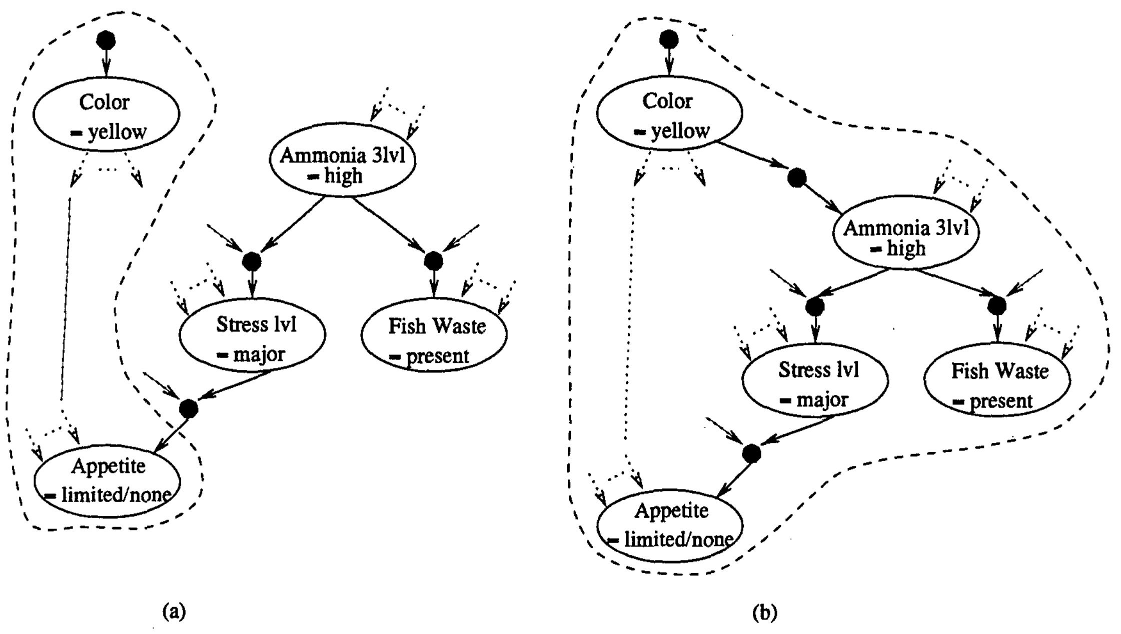

Figure 4.7 **Case 2 Low - BKB Structure Before and After Change.** Graphical depiction of selected instances in dependency region of "Color = yellow", path to answer "Fish Waste = present". Dotted line arrows represent i-node supports not pictured. (a) Before changes is made. (b) After Case 1 Low fix placing pivot node, "Ammonia 3lvl = High", in evidence dependency region. Note after pivot so place, answer is now in evidence dependency region.

V. Conclusions

The tool developed for this research incorporates manual and automatic processes to handle incompletenesses within a test-case. In the manual case, manipulations on the evidence and answer instances are performed and results returned to the user. These manipulations involve examination of the dependency regions of sibling instances to determine if incompleteness problems are corrected using an alternate instance. Answers not residing in an evidence dependency region but present within a sibling dependency region may indicate an incorrectly specified evidence item. Regardless, this information, along with specifics on which evidence item dependency regions do not contain answer instances, is returned to the user for manual correction.

While manual results are available, of primary importance to this effort is the automatic case, as outlined in Chapter III. Selected test-cases were drawn from the following areas:

1. Healthy and unhealthy fish.

2. Healthy and unhealthy tank/conditions.

Methods rely heavily on the direct dependency constraint levied on the evidence and answer items within a test-case. When the test-case violates this constraint, the closest indirect connections, comprising one of four paths, are examined and fixes made. While modifications to the KB are made automatically, user involvement is still necessary when mutual exclusion (mutex) problems arise. Since the tool attaches no meaning to the KB beyond structural data, and no external tools (see Section 6.2) are currently available to provide meaning, resolution of mutex problems without complete enumeration, or without arbitrary selection of supports satisfying mutex constraints is not possible.

For the test-cases examined, the total number of evidence-answer dependency mismatches after modifications made was always less than or equal to pre-change mismatches minus the number of actual fixes made. Formally,

$$N_{after} \leq (N_{before} - \Delta made)$$

- where N_{after} is the number of mismatches between evidence and answer items after changes were made;
- where N_{before} is the number of mismatches between evidence and answer items before changes were made;
- where $\Delta made$ is the number of actual corrections made.

This adds credence to the belief that corrections in one case do not cause mismatches between evidence and answer pairs previously exhibiting correct dependence on each other. While incompleteness remains an issue within BKBs, used in conjunction with knowledge acquisition methods and BVAL, the methods outlined in this research facilitate "completing" the KB to the minimal levels required by a test-case.

While the BKB modifications made corrected incompleteness problems within a BKB, limitations remain to the approach. Of primary concern is the idea of "choice". With any choice, there is the possibility that a different choice would have yielded better results. Thus, it is important that choices made are as informed as possible, so that an arbitrary choice is replaced by an intelligent one. While choices made by this research were not arbitrary, information that could be used to guide corrections was limited by what was available in the BKB representation. In essence, for the system to "know" one fix was better than another, the knowledge of the "better" fix would have to be in the system, in which case the problem would not exist. In other words, to intelligently fix a problem, the system has to know how to fix it, and if it knew how to fix it, since this information can only be garnered from the representation itself, the information would already be present in the KB so a fix would be unneccessary[1]. If the system could automatically access an outside source which had domain knowledge, this knowledge could be used to guide more intelligent system choices. Simply, the system could recognize its own knowledge deficiency, query an outside source,

[1]This is the Catch-22 of intelligent choices.

and learn what it was missing. Incorporation with PESKI (see Appendix A) would provide such an outside source via data-mining (see Section 6.2).

Another limitation of this research that PESKI incorporation would alleviate is the mutual exclusion problem. Optimally, mutual exclusion problems should be corrected as they are encountered so that operations are always performed on a consistent BKB, and since during correction other incompleteness problems may disappear. Currently, there is no mechanism to suspend the incompleteness methods to rectify mutual exclusion problems. In PESKI, however, methods for inter-tool communications are in place that would allow movement between the V & V and KA tools. Thus, complete incorporation would provide the user with the mechanism to begin test-case analysis, to suspend this analysis while accessing KA tools to correct mutual exclusion problems caused by the test-case incompleteness methods, and to resume the analysis before invoking BVAL with a complete test-case.

VI. Recommendations

6.1 Test-Case Suite

The current work operates on a single test-case at a time. The inherent problem with such a scheme is that the corrections made to fix one test-case may contradict another. A method whereby a collection of test-cases, or perhaps the entire validation suite of test-cases, could be used to guide the incompleteness detection and correction would be a significant improvement. Using all the dependency information from a collection of cases, it may be possible to automatically determine a more optimal fix than that currently proposed based on limited information base of a solitary test-case.

6.2 Data-Mining Tie-In

The more information available about the links between BKB items, the less arbitrary the nature of the incompleteness fix may be. As mentioned in Section 3.3, incompleteness results from missing rvs or instances, and missing or incorrect links between rv instances. Utilization of a data-mining tool would greatly improve automated techniques to correct incompleteness. Theorized relationships between nodes could be checked before corrections made, and relationships not initially identified by the incompleteness method could be discovered through data-mining. Rather than the path technique described in this research, the incorporation of the additional information provided by a data-mining tool into the incompleteness correction methods could drive corrections. More importantly, the data-mining tool could potentially remove the need for human approval of BKB modifications, since the database itself would perform the role of expert.

6.3 Knowledge Acquisition

Since corrections sometimes cause mutual exclusion problems, a bridge between the incompleteness tool and KA tool would be beneficial. In addition, methods within KA to automatically generate mutually exclusive tail conditions would assist the user in returning a BKB to consistency. Automatic support generation would also benefit from a data-mining assistant, so complete enumeration of all possible supports would not be generated. The resulting comprehensive subset of all possible supports would therefore include only viable support conditions derived from database knowledge.

6.4 PESKI Incorporation

Corrections or modifications to a BKB resulting from the incompleteness determined from a test-case can have widespread ramifications. In addition, the computer currently has little semantic understanding of the domain and, therefore, may not be able to discriminate between a sensical and a non-sensical fix to an incompleteness problem. Thus, some level of human involvement in the validation process would seem preferable over a completely automated system. Currently, the PESKI interface is virtually entirely a point-and-click mouse-based environment. Interaction between the user and a tool of the system (say BVAL) via a "command-line" type mechanism is not implemented[1]. For advanced features such as validation, such a capability is crucial since interaction between multiple areas (particularly between validation and KA), where automated tools may make corrections or suggestions that require user responses, is not uncommon.

[1]BVAL operates as a forked process. There is currently no facility in place to allow directly interaction between a user and the running BVAL.

Bibliography

1. Allen, Arnold O. ***Probability, Statistics, and Queueing Theory with Computer Science Applications, Second Edition.*** Academic Press, Inc., 1990.

2. Banks, Darwyn O. *Acquiring Consistent Knowledge for Bayesian Forests.* MS thesis, AFIT/GSO/ENG/95M-01. Graduate school of engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, March 1995.

3. Buchanan, Bruce G. and Edward H. Shortliffe, editors. *Rule-Based Expert Systems.* Addison-Wesley Publishing Company, 1984.

4. Charniak, Eugene. "Bayesian Networks without Tears," *AI Magazine*, 50–63 (Winter 1991).

5. Cooke, Nancy J. "Varieties of knowledge elicitation techniques," *International Journal of Human-Computer Studies*, *41*(6):801–849 (1994).

6. de Velde, Walter Van and Guus Schreiber. "The Future of Knowledge Acquisition: a European Perspective," *IEEE Expert*, 64–66 (April 1996).

7. Duda, R., et al. "Model Design in the PROSPECTOR Consultant System for Mineral Exploration," 153–167 (1979).

8. Feigenbaum, E.A. "Themes and Case Studies of Knowledge Engineering," *Expert Systems in the Micro–Electronic Age*, 3–25 (1995).

9. Gleason, Howard T. *Probabilistic Knowledge Base Validation.* MS thesis, AFIT/GSO/ENG/95D-04. Graduate school of engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1995.

10. Gonzalez, Avelino J. and Douglas D. Dankel. *The Engineering of Knowledge-based Systems.* Englewood Cliffs, NJ: Prentice Hall, 1993.

11. Harrington, Robert A., et al. "Development of an Intelligent User Interface for a Generic Expert System." *M. Gasser (Ed.), Online Proceedings of the 1996 Midwest Artificial Intelligence and Cognitive Science Conference.* 1996.

12. Harrington, Robert A., et al. "GESIA: Uncertainty-Based Reasoning for a Generic Expert System Intelligent User Interface." *To appear in the Proceedings of the 1996 International Conference on AI Tools.* 1996.

13. Kahn, Gary, et al. "MORE: An Intelligent Knowledge Acquisition Tool." *Proceedings of the Ninth International Joint Conference on Artificial Intelligence1.* 581–584. August 1985.

14. Kazman, Sue. "The FDS's Deadly Approval Process," *Consumers' Research Magazine*, *74*(4):31–34 (April 1991).

15. McDermott, John and Judith Bachant. "R1 Revisited: Four Years in the Trenches," *AI Magazine*, *5*(3):21–32 (Fall 1984).

16. Nazareth, Derek L. "Issues in the verification of knowledge in rule-based systems," *International Journal of Man-Machine Studies*, *30*:255–271 (1989).

17. Ng, Keung-Chi and Bruce Abramson. "Uncertainty Management in Expert Systems," *IEEE Expert*, 29–48 (April 1990).

18. O'Keefe, Robert M., et al. "Validating Expert System Performance," *IEEE Expert*, 81–89 (Winter 1987).

19. Pearl, Judea. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, CA, 1991. (Revised Second Printing).

20. Ram, Sudha and Sundaresan Ram. "Design and Validation of a Knowledge-Based System for Screening Product Innovations," *IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans*, *26*(2):213–221 (March 1996).

21. Santos, Jr., Eugene. "A Fully Integrated Probabilistic Framework for Expert Systems Development." Research proposal from Air Force Institute of Technology to Air Force Office of Scientific Research. March 1993.

22. Santos, Jr., Eugene. "Verifying and Validating Uncertain Knowledge-Bases." Research proposal from Air Force Institute of Technology. September 1995.

23. Santos, Jr., Eugene and Darwyn O. Banks. "Acquiring Consistent Knowledge in the Face of Uncertainty," *IEEE Transactions on Knowledge and Data Engineering* (1995). (Submitted to).

The Bayesian Knowledge Base is a key element of the expert system architecture called PESKI (Probabilities, Expert System, Knowledge and Inference)[21].

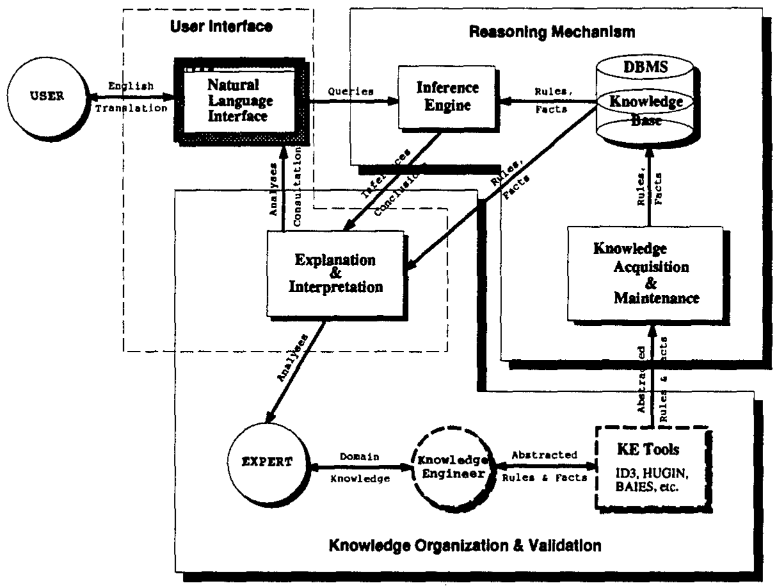

Figure A.1 **The PESKI architecture.** The broken boarder components **Knowledge Engineer** and **KE Tools** are considered optional.

As shown in Figure A.1, the PESKI architecture is composed of four main components:

- **Natural Language Interface**—provides for user–system communication by translating user queries, and system responses, into English.
- **Inference Engine**—the mechanism responsible for the reasoning actions of the system, controlling choice and application of information contained in the knowledge base.
- **Explanation & Interpretation**—monitors paths used by the inference engine in reaching its conclusion.

- **Knowledge Acquisition & Maintenance**—provides the tools for incorporating new, or modified, expert knowledge into the system.

There is some measure of overlap between these four components, hence PESKI facilitates the combination of these components into three overlapping subsystems:

- *User Interface*—combines the Natural Language Interface and the Explanation & Interpretation components.
- *Knowledge Organization & Validation*—combines the Explanation & Interpretation component with the human expert, the optional knowledge engineer, and the methods for knowledge engineering. Communication with the the Knowledge Acquisition & Maintenance component facilitates organization, and then assists in Validation when coupled with feedback from the Reasoning Mechanism through Explanation &Interpretation.
- *Reasoning Mechanism*—combines the Inference Engine and the Knowledge Acquisition & Maintenance components. The merging of these two components stems from the belief that in order for new knowledge to be placed in the knowledge base, some form of reasoning (and possibly learning) must be involved. Additionally, the inclusion of the Knowledge Acquisition & Maintenance component provides some degree of information hiding with respect to the knowledge representation used with the knowledge base.

The PESKI architecture is sufficient for the construction of knowledge-based systems in nearly any domain.

A.1 Current Status

Initial work on the physical realization of the PESKI idea has begun. Currently, PESKI works from an intelligent graphical user interface (IUI)[11, 12], which combines Knowledge Acquisition, Inference Engine, Verification & Validation, and Data-Mining tools. Each tool works from a main window, with possible sub-windows[1]. The Knowledge Acquisition tool replaces command line KA, performed previously with MACK[2] and MACKJr[2], and the Verification & Validation portion incorporates a reengineered BVAL[9], altered to operate in the new environment.

[1]Knowledge Acquisition currently is the only tool with a sub-window, namely Edit Supports. Use of sub-windows was necessary to reduce confusion in the main window.

[2]MACKJr, written by Santos[22, 23, 21], was a refinement of MACK, which operated on Bayesian Forests.

Appendix B. MUTUAL EXCLUSION AND CYCLIC KNOWLEDGE

B.1 Mutual Exclusion

Mutual exclusion (mutex) mandates the support conditions of a target rv state must be uniquely distinguishable from each other. So, if an rv instance is "on", only one of its supports can be active. The following excerpt from *GF*2[1] illustrates this property. It shows the six supports for a specific state, "Present", of the rv "Parasites". Each support has a probability of occurrence (the number before "TAIL"), and three tail conditions comprising the support itself.

```
13 [F] Parasites

      7 Present

            SUPPORTS: 6
              0.548703 TAIL: 3
                   16 [F] Stress Level 5 Major
                   19 [F] Fish Appearance 9 Thin/weak
                   17 [W] Water Quality 4 Poor

              0.315973 TAIL: 3
                   16 [F] Stress Level 4 Mild
                   19 [F] Fish Appearance 9 Thin/weak
                   17 [W] Water Quality 4 Good

              0.50959 TAIL: 3
                   16 [F] Stress Level 8 Moderate
                   19 [F] Fish Appearance 9 Thin/weak
                   17 [W] Water Quality 4 Good

              0.533448 TAIL: 3
                   16 [F] Stress Level 5 Major
                   19 [F] Fish Appearance 9 Thin/weak
                   17 [W] Water Quality 4 Good

              0.432017 TAIL: 3
                   16 [F] Stress Level 4 Mild
                   17 [W] Water Quality 4 Poor
                   19 [F] Fish Appearance 9 Thin/weak

              0.385675 TAIL: 3
                   16 [F] Stress Level 8 Moderate
```

[1] *GF*2 is an augmented, MACKJr version of the MACK[2] BKB, Goldfish. Created by this author, it consists of 57 rvs, with each rv having between 1 and 17 states. Like Goldfish, *GF*2 contains information related to goldfish and their tank surroundings.

```
19 [F] Fish Appearance 9 Thin/weak
17 [W] Water Quality 4 Poor
```

B.1.1 Effects on KA and Validation. The difficulty the mutual exclusion property makes during KA is there are no tools to automatically generate all combinations of specified supports[2] to account for the mutex provision. In addition, there is sometimes a tendency to change the dependency order to avoid complete enumeration of supports. Consider the following example: Examining the supports for "Stress Level = Mild", taken from GF2.

```
16 [F] Stress Level

      4 Mild

            SUPPORTS: 3
              0.62052 TAIL: 2
                    21 [W] Water Temperature 11 (32 - 45] F
                    19 [W] Nitrite 3-Level 8 Moderate

              0.633951 TAIL: 2
                    21 [W] Water Temperature 11 (32 - 45] F
                    19 [W] Nitrite 3-Level 8 None/Low

              0.675196 TAIL: 2
                    21 [W] Water Temperature 11 (32 - 45] F
                    19 [W] Nitrite 3-Level 4 High
```

It is intuitive that if a fish is suffering from a disease (such as Beta Finch Fin), it will experience some stress. Assume that if a fish is so afflicted, it causes mild stress. Thus, "Beta Finch Fin = True" causes "Stress Level = Mild", and so should be included among its supports. The simple addition of a fourth support consisting of a single tail condition ("Beta Finch Fin = True"), violates the mutex constraint, since it would be possible for the new support to be true in conjunction with any of the three supports shown above. In addition, it is possible the fish may be mildly stressed but not have Beta Finch Fin (it could have some other disease which is causing it to experience stress). Incorporatation of the Beta Finch Fin data into the supports for "Stress Level = Mild"

[2]For a support with three tails, X, Y, and Z, where X, Y, and Z each have four possible states, there are 4^3 combinations of states of the three random variables which would result in a unique support condition.

would consist of modifying the existing supports and adding three new supports, thus accounting for the case that the fish had Beta Finch Fin and the case when it did not.

So, support 1, would now be expressed as two supports with the following mutex tail conditions:

```
Support 1a:
    21 [W] Water Temperature 11 (32 - 45] F
    19 [W] Nitrite 3-Level 8 Moderate
    18 [D] Beta Finch Fin 4 True

Support 1b:
    21 [W] Water Temperature 11 (32 - 45] F
    19 [W] Nitrite 3-Level 8 Moderate
    18 [D] Beta Finch Fin 5 False
```

An alternative would be to place "Stress" as a support condition for Beta Finch Fin. If the latter instance had fewer supports and/or fewer tails than the former, the incorporation of "Stress" into the supports could entail significantly less time and effort on the part of the expert. Potentially, however, this placement could be less than optimal.

B.2 Cyclic Knowledge

The cyclic knowledge constraint precludes an rv instance, A, from supporting and being supported by the same rv instance, say X.

B.2.1 Effects on KA and Validation. This constraint affects KA particularly in cases where cause and effect between rv instances are not purely one directional. In *GF*2, for example, a support condition for "Parasites = Present" contains "Stress Level = Moderate", since stress makes goldfish more prone to parasitic attack. Yet, it is also true that if a goldfish has parasites, so "Parasites = Present", the goldfish is stressed. In other words, stress may cause parasites, and parasites may cause stress – a cyclic dependency. The problem faced during KA is where to place the support, which location is best. This decision is often arbitrary, especially when one directional implication is not significantly more likely than another.

Appendix C. Incompleteness Runs

C.1 Case 1 - High Path

The following partial transcript is for a test-case addressing unhealthy tank conditions:

```
Evidence is:
[W] Clarity = Murky
[W] Color = Yellow
[W] Odor = Decaying
[W] Ammonia Level = [.5 - 1] ppm
[Objects] Cleanliness = Brown/Green film or coating
[W] Algae = Present
[W] Nitrite Level = [.25 - .5) ppm
[F] Stress Level = Mild
[W] Chloromine = Present
[W] Nitrate = Present

Answers are
[W] Fish waste = Present
[W] Decaying Food = Present
[W] Water Quality = Poor


TESTING EVIDENCE #4:  [Objects] Cleanliness = Brown/Green film or coating
   Answer #0, [W] Fish waste = Present,  not dependent on evidence
   Answer #1, [W] Decaying Food = Present,  not dependent on evidence
   Answer #2, [W] Water Quality = Poor,  not dependent on evidence


Continue auto on evidence:
 [Objects] Cleanliness = Brown/Green film or coating and answer #0?  (Y/N)  y

case1 hi - item in upper portion of DR
num elements = 3
[W] pH Level = Less than 6.0
[W] pH Level >= 7.2 = False
[W] Fish waste = Present

BKB modified, BKB is consistent.
```

Supports of answer before Case 1 High Fix resulted in the inclusion of the answer in dependency region of evidence.

```
14 [W] Fish waste

        7 Present
```

```
                SUPPORTS: 4
                  0.478348 TAIL: 2
                       19 [W] Ammonia 3-Level 8 Moderate
                       19 [W] pH Level >= 7.2 5 False

                  0.553183 TAIL: 2
                       19 [W] Ammonia 3-Level 4 High
                       19 [W] pH Level >= 7.2 5 False

                  0.37888 TAIL: 2
                       19 [W] Ammonia 3-Level 8 Moderate
                       19 [W] pH Level >= 7.2 4 True

                  0.447621 TAIL: 2
                       19 [W] Ammonia 3-Level 4 High
                       19 [W] pH Level >= 7.2 4 True
```

Answer supports after changes made

```
    14 [W] Fish waste

            7 Present

                SUPPORTS: 2
                  0.3788800000 TAIL: 2
                       19 [W] Ammonia 3-Level 8 Moderate
                       21 [Objects] Cleanliness 27 Brown/Green film or coating

                  0.5531830000 TAIL: 2
                       19 [W] Ammonia 3-Level 4 High
                       21 [Objects] Cleanliness 27 Brown/Green film or coating
```

C.2 Case 1 - Low Path

The following transcript is for the regular low case.

```
Evidence is:
[W] Clarity = Clear
[W] Odor = None
[W] Ammonia Level = 0 ppm
[W] Algae = None
[W] Nitrite Level = 0 ppm
[W] Nitrate = Not Present

Answers are
[W] Fish waste = Not Present
[W] Water Quality = Good
```

```
TESTING EVIDENCE #4:  [W] Nitrite Level = 0 ppm
  Answer #0, [W] Fish waste = Not Present,  not dependent on evidence
  Answer #1, [W] Water Quality = Good,  not dependent on evidence

Continue auto on evidence:  [W] Nitrite Level = 0 ppm and answer #1? (Y/N)  y

case1 low - item in lower portion of DR

[F] Parasites = Present
[W] Water Quality = Good

not special low case
BKB modified, new SDU status is as follows
Two s-nodes are not mutually exclusive
  Component/state: [W] Water Quality = Good
    Supports are 1 and 0
```

Before changes were made, the BKB supports were as follows:

```
17 [W] Water Quality

        4 Good

              SUPPORTS: 1
                0.673736 TAIL: 6
                     17 [W] Ammonia Level 5 0 ppm
                     11 [W] Clarity 5 Clear
                      9 [W] Color 5 Clear
                      8 [W] Odor 4 None
                     14 [W] Chloromine 11 Not Present
                     11 [W] Nitrate 11 Not Present
```

After the change, the supports reflect the addition of a fifth support, the cause of the mutual exclusion problems, but the correction of the answer not belonging to the dependency region of the evidence.

```
17 [W] Water Quality

        4 Good

              SUPPORTS: 2
                0.6737360000 TAIL: 6
                    17 [W] Ammonia Level 5 0 ppm
                    11 [W] Clarity 5 Clear
                     9 [W] Color 5 Clear
                     8 [W] Odor 4 None
                    14 [W] Chloromine 11 Not Present
        11 [W] Nitrate 11 Not Present

    0.0500000000 TAIL: 1
        17 [W] Nitrite Level 5 0 ppm
```

C.3 Case 2 - High "V" Path

The following transcript is for the high "V" case.

```
Evidence is:
[W] Clarity = Murky
[W] Color = Yellow
[W] Odor = Decaying
[W] Ammonia Level = [.5 - 1] ppm
[Objects] Cleanliness = Brown/Green film or coating
[W] Algae = Present
[W] Nitrite Level = [.25 - .5) ppm
[F] Stress Level = Mild
[W] Chloromine = Present
[W] Nitrate = Present

Answers are
[W] Fish waste = Present
[W] Decaying Food = Present
[W] Water Quality = Poor


TESTING EVIDENCE #0:  [W] Clarity = Murky
  Answer #2, [W] Water Quality = Poor,  not dependent on evidence
Number of mismatches = 1


Continue auto on evidence:  [W] Clarity = Murky and answer #2?  (Y/N)  y

case2 ('V') hi - item in upper portion of DR
num elements = 4

[W] Ammonia 3-Level = High
[F] Stress Level = Major
[F] Parasites = Present
[W] Water Quality = Poor

BKB modified, BKB is consistent
```

Before changes were made, the BKB entries for the pivot element and for the answer instance were as follows:

```
13 [F] Parasites
```

```
        7 Present

               SUPPORTS: 6
                 0.548703 TAIL: 3
                     16 [F] Stress Level 5 Major
                     19 [F] Fish Appearance 9 Thin/weak
                     17 [W] Water Quality 4 Poor

                 0.315973 TAIL: 3
                     16 [F] Stress Level 4 Mild
                     19 [F] Fish Appearance 9 Thin/weak
                     17 [W] Water Quality 4 Good

                 0.50959 TAIL: 3
                     16 [F] Stress Level 8 Moderate
                     19 [F] Fish Appearance 9 Thin/weak
                     17 [W] Water Quality 4 Good

                 0.533448 TAIL: 3
                     16 [F] Stress Level 5 Major
                     19 [F] Fish Appearance 9 Thin/weak
                     17 [W] Water Quality 4 Good

                 0.432017 TAIL: 3
                     16 [F] Stress Level 4 Mild
                     17 [W] Water Quality 4 Poor
                     19 [F] Fish Appearance 9 Thin/weak

                 0.385675 TAIL: 3
                     16 [F] Stress Level 8 Moderate
                     19 [F] Fish Appearance 9 Thin/weak
                     17 [W] Water Quality 4 Poor


17 [W] Water Quality

        4 Poor

               SUPPORTS: 1
                 0.326264 TAIL: 0
```

After changes were made, modified BKB supports for the pivot and answer were as follows:

```
13 [F] Parasites

         7 Present

               SUPPORTS: 2
                 0.4320170000 TAIL: 3
```

```
                    19 [F] Fish Appearance 9 Thin/weak
                    17 [W] Water Quality 4 Poor
                    11 [W] Clarity 5 Murky

                0.5095900000 TAIL: 3
                    17 [W] Water Quality 4 Good
                    19 [F] Fish Appearance 9 Thin/weak
                    11 [W] Clarity 5 Murky


17 [W] Water Quality

         4 Poor

             SUPPORTS: 1
               0.3262640000 TAIL: 1
                   11 [W] Clarity 5 Murky
```

C.4 *Case 2 - Low "V" Path*

The following transcript is for the low "V" case.

```
Evidence is:
[W] Clarity = Murky
[W] Color = Yellow
[W] Odor = Decaying
[W] Ammonia Level = [.5 - 1] ppm
[Objects] Cleanliness = Brown/Green film or coating
[W] Algae = Present
[W] Nitrite Level = [.25 - .5) ppm
[F] Stress Level = Mild
[W] Chloromine = Present
[W] Nitrate = Present

Answers are
[W] Fish waste = Present
[W] Decaying Food = Present
[W] Water Quality = Poor


TESTING EVIDENCE #1:  [W] Color = Yellow
  Answer #0, [W] Fish waste = Present,  not dependent on evidence
  Answer #1, [W] Decaying Food = Present,  not dependent on evidence
  Answer #2, [W] Water Quality = Poor,  not dependent on evidence

Continue auto on evidence:  [W] Color = Yellow and answer #0?  (Y/N)  y

case2 ('V')low - item in lower portion of DR
```

```
num elements = 4

[F] Appetite = Limited/None
[F] Stress Level = Major
[W] Ammonia 3-Level = High
[W] Fish waste = Present

BKB modified, new SDU status is as follows
Component is [W] Ammonia 3-Level

Two s-nodes are not mutually exclusive
  Component/state: [W] Ammonia 3-Level = High
    Supports are 4 and 2

Two s-nodes are not mutually exclusive
  Component/state: [W] Ammonia 3-Level = High
    Supports are 4 and 1

Two s-nodes are not mutually exclusive
  Component/state: [W] Ammonia 3-Level = High
    Supports are 4 and 0
```

Before changes, supports were as follows:

```
19 [W] Ammonia 3-Level

        4 High

              SUPPORTS: 4
                0.954622 TAIL: 1
                    17 [W] Ammonia Level 11 [2 - 3] ppm

                0.975635 TAIL: 1
                    17 [W] Ammonia Level 11 [4 - 5] ppm

                0.932823 TAIL: 1
                    17 [W] Ammonia Level 11 [6 - 7] ppm

                0 TAIL: 1
                    17 [W] Ammonia Level 5 0 ppm
```

After changes, modified supports are as follows:

```
19 [W] Ammonia 3-Level

        4 High

              SUPPORTS: 5
                0.9546220000 TAIL: 1
                    17 [W] Ammonia Level 11 [2 - 3] ppm
```

```
0.9756350000 TAIL: 1
    17 [W] Ammonia Level 11 [4 - 5] ppm

0.9328230000 TAIL: 1
    17 [W] Ammonia Level 11 [6 - 7] ppm

0.0000000000 TAIL: 1
    17 [W] Ammonia Level 5 0 ppm

0.0500000000 TAIL: 1
    9 [W] Color 6 Yellow
```

Vita

Louise Lyle [redacted]. She was awarded a Bachelor of Science in Mathematics/Computer Science from Pepperdine University in 1987. She received her commission in 1992 from Officer Training School, and was assigned to Whiteman AFB as a missile launch officer. While at Whiteman, she received a Master of Arts degree in Management from Webster University. She remained in Missouri until May of 1995, when she began attending the Air Force Institute of Technology.

[redacted] [redacted]