1

An Investigation and Interpretation of Selected Topics in Uncertainty Reasoning

THESIS

Scott E. Deakin
First Lieutenant, USAF

AFIT/GSO/ENS/89D-3

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

1

AFIT/GSO/ENS/89D-3

An Investigation and Interpretation of Selected Topics in Uncertainty Reasoning

THESIS

Scott E. Deakin
First Lieutenant, USAF

AFIT/GSO/ENS/89D-3

AFIT/GSO/ENS/89D-3

An Investigation and Interpretation of Selected Topics in Uncertainty Reasoning

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 in Space Operations

Scott F. Deakin, B.S.
First Lieutenant, USAF

December, 1989

Approved for public release; distribution unlimited.

Preface

My thesis deals with topics in uncertainty reasoning, which is appropriate since for most of the duration I was uncertain about just what my topic was. I spent many days wondering just what I was supposedly doing. Although it could hardly be said that I was apoplectic with apprehension, I was alternately concerned about, and oblivious to my plight. Practicing avoidance and denial became a hobby, but unfortunately, as fate would have it, the task before me never unilaterally abated. Many times the future of this thesis seemed so very uncertain (at least to me). However, in the end the work was done, my mission completed.

My advisor, Maj Bruce W. Morlan, told me that he knew where I was going (existentially, that is). To him, I give thanks; the type of thanks one gives a dentist after an unwanted root canal. He gave me encouragement and, at times, fear, both of which proved to be motivational.

In retrospect, all that has passed has changed me, but all I really know is that I feel older now. The people around me, my classmates and friends, I am beholden too. I thank my advisor, Maj Bruce W. Morlan, for those interesting, frustrating, exasperating debates which always strayed from their original convoluted course. I thank my reader, Lt Col Skip Valusek, for questioning the things I took for granted, for catching the abundant small errors, and most of all, for not *keelhauling* me for my lack of communication. For all those who supported me in this endeavor, I bubble with gratitude. You helped me by just laughing with me and, at times, at me. I like most of you and will miss some of you.

Scott E. Deakin

Table of Contents

List of Figures

List of Tables

AFIT/GSO/ENS/89D-3

Abstract

Incorporating techniques for coping with uncertainty in the decision support systems has proven to be a fertile environment for creative ideas. Representations of uncertainty abound and no representation can be said to be inherently incorrect. From a theoretical standpoint, a viable solution must be coherent and logically consistent. Probability theory demonstrates these characteristics while, as of yet, other methods do not.

The purpose of this study was to investigate specific topics in uncertainty reasoning: 1) *Probability ratio graphs* as a representation of the probability model; 2) Dealing with missing information when system parameters are left unspecified; 3) Investigating the difference between probabilistic and causal independence; and, 4) Characterizing secondary uncertainty as spurious evidence and including it in the inference process.

It was shown that probability ratio graphs are a viable method for representing uncertainty, and a method for representing independence with probability ratio graphs is presented. Assuming probabilistic independence for missing information is shown to have intuitive and computational benefits; also shown is that where secondary uncertainty is included in the inference process has great impact on the computational complexity of an inference process.

An Investigation and Interpretation of Selected Topics in Uncertainty Reasoning

I. Organization

> ***I was to learn early in life that we tend to meet any new situation by reorganizing... and a wonderful method it can be for creating the illusion of progress while producing inefficiency and demoralization.***
>
> –Petronius (died A.D. 66)

The need for access to expert opinion and analysis when qualified experts are a scarce commodity is driving the development of artificial experts. These artificial experts are known as *expert systems*, and through their use, expert opinion and analysis is accumulating, and becoming transportable and standardized. If these systems are to be useful they must address and meet certain requirements. Some important requirements are the capability for handling large quantities of information, dealing with uncertainty, and heuristic control of the search space. Computer based expert systems provide the important capability of rapidly handling large quantities of data. However, combining the computer's capabilities with experts' experience and heuristic reasoning is proving to be a formidable task subject to much debate.

Expert systems are complex automated checklists. Though the term *expert system* sprang from the field of artificial intelligence (AI), they have existed in principle under other names and in other forms. Artificial intelligence provided an alternative to classical operations research (OR) in solving complex problems; OR applies optimization techniques to complex problems expressed in real numbers, whereas, AI applies heuristic search to complex problems that defy classical OR techniques. Most complex problems, especially those expert systems face, would benefit from a combination of both OR and AI techniques (17:11): OR can provide probability and decision theoretic techniques for dealing with uncertainty, and AI can provide heuristic control for managing the exponential explosion to which probabilistic systems are vulnerable.

Researchers have recognized the usefulness of a symbiosis between OR and AI. They are incorporating probability and decision theory into the development of expert systems (Kalagnanam and Henrion:1988, Breese and Fehling:1988, Heckerman:1988). Along these lines, Hollenga codified a method for developing expert systems that utilizes the synergistic effect of combining OR and AI. In his paper "A Decision-Theoretic Model for Constructing Expert Systems," he outlines a five step process that incorporates Bayesian reasoning in the development of the expert system rule base:

1. A group of decision makers decides on strategy implementation.
2. The output of the first step is then translated into a decision analytic framework where hypotheses, evidences, strategies, probabilities, and utilities are identified.
3. From the second step, probability thresholds are identified. These thresholds indicate the strategy with highest utility.
4. Executable rules are then generated such that only the evidence is needed to determine an action.
5. The last step is encoding the executable rules into a format that an expert system can manipulate (8:3-8).

Hollenga's method involves using decision analysis to generate the expert system rule base. In effect, the decision-analytic framework becomes the expert for rule development (8:8).

1.1 Thesis Research

Hollenga's process is a conceptual model which needs an architecture for research into its practical feasibility. A research tool that enables research into the capability of the aforementioned process goes far in supporting investigations into the implications of the proposed method.

Morlan proposes *probability ratio graphs* as a method for manipulating relational information (13). Represented in this way, decisions and inferences depend on the relative weight of the propositions. The odds-ratio structure presents a graphically appealing method for user comprehension where conditional relationships are visually evident.

This thesis used Hollenga's proposed five step method as a starting point for research into uncertainty reasoning. The development of a research tool for investigation into Hollenga's proposed five step method (specifically, step three) provided a means for further refining Morlan's concept of probability ratio graphs and generating questions about uncertainty reasoning. Several interesting topics of concern arose: representing independence in probability ratio graphs, generating missing evidence, the meaning of independence, and secondary uncertainty and spurious events.

1.2 Research Objective

The objective of this research was to investigate questions about uncertainty reasoning and further refine the probability ratio graph concept. The methodology centered on developing a PC-based research tool for investigating the nuances of Hollenga's five step method for isolating questions about uncertainty reasoning and applying probability ratio graphs. The resulting computer program is a secondary deliverable; it is conceptually concerned with the propagation and visual representation of probability and utility relational information using an odds-ratio approach.

1.3 Research topics

During this research several topics arose that were of some interest. These topics spawned questions of interpretation where the answers were ill-defined and not immediately obvious. The design heuristics of decision support systems depend in part on the interpretations that result from these considerations.

- When system parameters are unspecified, how should a diagnostic system deal with the missing information?
- Under the common assumption of disjoint hypotheses, what does hypothesis-evidence independence mean?
- What is the meaning of spurious evidence, and how does it affect the diagnostic system?

1.4 Scope

There are many proposed methods for representing and coping with uncertainty (certainty factors, fuzzy sets, Dempster-Sheafer, Bayesian inference); each method has good and bad facets. As in any situation, the problem at hand drives the choice of the tool. Misapplication of a tool can have potentially damaging effects which cannot always be predicted.

This research uses probabilities for representing uncertainty due to demonstrated advantages: it provides well-known ways of incorporating empirical data, has well developed methods for evaluating judged or computed probabilities by comparison with empirical frequencies, and has been shown that for any reasonable scoring rule, any scalar measure of uncertainty is either worse than probability or is equivalent to it (7:2).

1.5 Summary

Growing interest in expert systems, specifically incorporating probability and decision analytic techniques in the generation of expert systems, was the motivation for this research effort. Hollenga's five step method for expert system generation and Morlan's probability ratio graphs for probabilistic knowledge representation present two interesting ideas that address the problem of including uncertainty reasoning in such systems. Program development dealing with both of these areas uncovered the research topics described above. Chapter II presents a view of the current state of uncertainty representation and reasoning; there does not seem to be a "correct" interpretation, in that the validity of one method does not rule out other methods. Chapter III addresses Morlan's probability ratio graph concept and briefly describes the current state of the created software that deals with probability ratio graphs and Hollenga's five step method. Chapter IV presents the research topics on missing information, independence, and second order uncertainty. Chapter V contains conclusions reached about the topics in this thesis, as well as possible areas for further research.

II. State of Uncertainty Reasoning

2.1 Background

When problems become even moderately complex the human mind founders in a quagmire of information. Recognizing and reacting in situations when the available information is incomplete[1] or insufficient,[2] or filtering an abundance of data[3] for pertinent information, can quickly overwhelm even the brightest among us. At such times we invariably search for help in making sense of this seemingly chaotic information. Managerial sciences, operations research, artificial intelligence, statistics, decision theory...all of these fields' purpose is to condition and massage an abundance of information into an organized structure with a relatively small set of discriminating features so we can make sense of the data and make rational decisions based on the data. In general, experts employ similar conditioning and massaging schemes and either make decisions or advise those who do.

More complex and costly problems require more precise and accurate answers. Faced with limited experts and increasing demand, decision makers have to rely on basic rules of thumb and procedures provided by experts which may apply to their problem. For complex problems these rules of thumb and procedures become inadequate causing the decision maker to suffer from the information ailments (incomplete or insufficient information, or information overload); he is unable to process the available information coherently and thus, the decision suffers.

Expert systems are an attempt to create automated checklists that are capable of handling complex problems. They fill a need for access to expert knowledge and judgment when true experts are a scarce commodity. Through the use of expert systems, experts' knowledge and judgment becomes transportable and cumulative. Knowledge engineers distill experts' knowledge and judgment, and condense it into a predefined structure

[1] Incomplete information–In many cases of decision making the situation is under specified. The available information may be either ill-defined (vague) or imprecisely defined. (9:666)

[2] Insufficient information–In these cases the additional information is potentially there, but a separate and specific effort is required to bring it out (projections, correlations). (9:666)

[3] Information overflow–This is the case of too much information. The decision maker drowns in information which by far exceeds what he can process or comprehend at the time of decision. (9:667)

(knowledge base) that an "inference engine"[4] can logically manipulate to arrive at the same conclusion as an expert given similar circumstances. Expert systems that rely on these translated rules are heuristic in nature and are subject to translation errors resulting in rules that do not capture the experts' true diagnostic process.

As outlined above, there are two facets to the problem of automation: 1) Conditioning abundant information to produce relatively few measures that approximately define the state of the world, and 2) Interpreting the resulting measures and making decisions based on them. In developing systems with greater autonomy researchers must support these facets, meeting the implied requirements of well defined scope, large information handling capabilities, information filter, projection or forecasting, recognize incomplete information, handling uncertainty, coherence and rationality, while avoiding the problems of imprecise translation. These requirements call for both mathematical precision and heuristic control.

Operations Research (OR) and Artificial Intelligence (AI) are two fields that are concerned with the problem of supporting decision making by providing high level interpretation of the state of the world. Simon provides working definitions of these two fields with the understanding that both fields are not bounded by these definitions.

> Operations research may be defined as the application of optimization techniques to the solution of complex problems that can be expressed in real numbers. The criterion function, which determines what is to be optimized, must also be quantitative. This definition is clearly too narrow to encompass all the things that operations research professionals do...but it characterizes the predominant emphasis upon formal mathematical models and optimization...
>
> By contrast, artificial intelligence is the application of methods of heuristic search to the solution of complex problems that (a) defy the mathematics of optimization, (b) contain non-quantitative components, (c) involve large knowledge bases (including knowledge expressed in natural language), (d) incorporate the discovery and design of alternatives of choice, and (e) admit ill-specified goals and constraints.
>
> This characterization of AI does not set very definite boundaries. It might be regarded more as a hunting license than as a proper definition. It emphasizes

[4] Inference engine - also known as control structure or conflict resolution: similar to an algorithm but more general and less precise. The way facts, rules, and parts of rules are manipulated is controlled by the inference engine (16:99).

> the aspiration of AI to deal with all the aspects of managerial decision making that stretch beyond the limits of classical OR. (17:10-11)

Most problems have components that are best handled with OR methods and other components that are better addressed with AI's heuristics. It is, therefore, advantageous to combine and synthesize OR and AI, supporting, reinforcing, and extending each other (17:11). Others have also recognized the usefulness of a symbiosis between OR and AI. One area of interest is the incorporation of decision theory into the development of expert systems. Kalagnanam and Henrion, Breese and Fehling, and Heckerman, to name a few, have produced research in this area. Combining OR and AI methods has great potential for producing more powerful decision support and autonomous systems. Though united in the goal, researchers are divided on the path and what final form these symbiotic expert systems should take.

2.2 Automating the Expert

Automation is the "automatically controlled operation of an apparatus, process, or system by mechanical or electronic devices that take the place of human organs of observation, effort, and decision." When speaking of automating a process, care must be taken to specify to what degree the process is automated. There is some automation in every action that relies on a machine or tool. The goal of automation is not to remove the power of human self determination but to relieve the human from the burden of mundane or information intensive tasks so that more worthy undertakings can be pursued. It also allows for the speed many operations require and which humans cannot provide.

A major point of contention on achieving automation is the underlying philosophy of current and proposed methods. Expert systems are proving valuable in automating decision support and process control. However, the quality of these expert systems is questionable due to imprecision, uncertainty, and, in part, to the "discouraging observation...that today's systems seem to be successful because they are 'hand crafted' rather than because they apply a set of proven techniques and methods" (10:750). These 'hand crafted' methods and ingenious heuristics are the source of much philosophic debate on the form that these systems should take. The methods for manipulating uncertainty come under scrutiny as does expert emulation and rationality in the artificial decision maker.

2.2.1 Uncertainty Uncertainty is generally characterized as resulting from stochastic processes, linguistic vagueness, and subjective belief. Stochastic uncertainty is usually measured with statistics and arises when referring to random events such as rolling a die, or spurious events like accidents or faulty readings. Linguistic vagueness results from imprecise definition where terms have variable meanings; compare numbers which are discrete and well defined, and therefore, not vague (two, three,...), with "a few." Zadeh developed the concept of fuzzy sets to deal with such vagueness. Subjective beliefs are the predominant source of uncertainty; subjective uncertainty prevails in one-of-a-kind situations where someone makes an assessment. Bayesian statisticians argue that it is the only type since subjective interpretation is involved in communication and data assimilation (18:8).

Whatever its source, a successful expert system must be able to deal with uncertainty. Probability is perhaps the method best known for representing uncertainty, as is classical Bayesian inference, for reasoning under uncertainty. It is valid and has a sound theoretical foundation. However, classical Bayesian evidential reasoning becomes so computationally intensive as to be intractable when applied to a non-trivial decision problem. "It requires a detailed listing of all possible scenarios which is impossible. The apparent need for [a] huge distribution of cases is the major objection to using Bayesian probability theory in a real expert system." (18:8) Various methods (certainty factors, fuzzy set theory, Dempster-Sheafer theory) arouse controversy when they avoid Bayesian conditioning in an attempt to skirt computational intractability (12:271). Kyberg states, "non-Bayesian updating yields more determinate belief states as outcomes, but the benefits afforded by non-Bayesian updating are limited and questionable."(12:285)

All methods for dealing with uncertainty must have some measure which characterizes the amount of uncertainty there is in a proposition and some interpretation of that measure. The following four methods are some of the more well known theories for characterizing uncertainty.

2.2.1.1 Bayesian Probability theory Probability is an assessment about the frequency of events. Its roots lie in games of chance (dice, cards, roulette, etc...), where the probability of an event is the number of possible occurrences of $\mathcal{E}$ divided by the total number of possible outcomes $\mathcal{S}$:

$$P(\mathcal{E}) = \frac{\#\ \ of\ \ \mathcal{E}s}{\#\ \ of\ \ \mathcal{S}s}$$

An example is the probability of drawing an ace out of a deck of cards:

$$P(\mathcal{A}) = \frac{\#\ \ of\ \ Aces}{\#\ \ of\ \ Cards} = \frac{4}{52} = 0.07692$$

Probability reasoning is based on Bayes' theorem[5], which plays a central role in elementary probability. It is a general rule for the computation of a posterior probability $P(\mathcal{A}_k \mid \mathcal{B})$ from prior probabilities $P(\mathcal{A}_i)$ and conditional probabilities $P(\mathcal{B} \mid \mathcal{A}_i)$ (Devore:60). This theorem presents the mathematical equation for combining probabilistic assessments coherently and consistently:

$$P(\mathcal{A} \mid \mathcal{B}) = \frac{P(\mathcal{B} \mid \mathcal{A})P(\mathcal{A})}{P(\mathcal{B})}$$

Bayes' theory is based on conditional probabilities and allows for probabilistic inferences when evidence is observed. For instance, in the card example above, the $P(\mathcal{A}) = 4/52$ given that there are four aces in the deck. This is a conditional probability, as are all measures of uncertainty, being conditioned at least upon the set of possible cards. All events are conditional in that there is a set of conditions whether explicit or implicit that exits or defines the situation for which a proposition is valid.

There arr various arguments against using probability for reasoning with uncertainty. Two important arguments are that it doesn't reflect the way people reason and that it becomes computationally intractable. Arguments against probability spurred the

[5] Bayes' theorem is named after its eighteenth century originator, Reverend Thomas Bayes.

development of other methods for representing and reasoning with uncertainty.

2.2.1.2 Certainty Factors Certainty factors originated from Bayes' theorem. They were developed to handle uncertainty in MYCIN[6] when the Bayesian inferential method became intractable both computationally and with respect to data requirements. In their book, "Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project," Buchanan and Shortliffe explain the development of certainty factors (1).

Certainty factors represent uncertainty as reasons of belief using measures of belief, $MB(\mathcal{H},\mathcal{E})$; and disbelief, $MD(\mathcal{H},\mathcal{E})$ where $\mathcal{H}$ is a hypothesis and $\mathcal{E}$ is an evidence state. These measures are computed separately, and are then combined to represent the total uncertainty as a certainty factor, $CF(\mathcal{H},\mathcal{E}) = MB(\mathcal{H},\mathcal{E}) - MD(\mathcal{H},\mathcal{E})$.

The MB and MD both range between 0 and 1, giving the CF a range of -1 to 1. A positive CF indicates more belief than disbelief and, conversely, a negative CF indicates more disbelief than belief. A CF of 0 indicates that there is equal belief in both propositions $\mathcal{H}$ and *not*$\mathcal{H}$ (4:561).

Some criticisms levied against certainty factors are 1) that the combining rules are arbitrary, 2) they assume evidence independence (5:9), and 3) they do not have a sound theoretic basis (4:562). Although certainty factors are derived from probability theory, they are not probabilistic. They abandon probabilistic rules in an attempt to reduce the computational burden and data requirements. They work quite well as long as the reasoning path is short and the inherent errors do not build up. However, "it is not difficult to come up with an example in which, of two hypotheses, the one with the lower probability would have a higher certainty factor... This failure to rank according to probability is an undesirable feature of certainty factors." (1:269)

2.2.1.3 Fuzzy Set theory Fuzzy sets are sets where the borders are not crisply defined (tall, fast, heavy, etc...). Linguistic uncertainty comes from just these types of terms where the meaning can vary from person to person. The uncertainty arises from

[6] MYCIN is a rule-based expert system used for medical consultation.

the interpretation of what someone else means when they use these terms. Fuzzy sets attempt to represent sets as analog or continuous where conventional set theory is digital or discrete.

With conventional set theory an item can be in one of two states with respect to a set: in or out. Probability would represent the uncertainty between these two states. The degree of membership is not in question; it is either all in one or all in the other. With fuzzy sets, the degree of membership is in question: how much in or how much out. A member can, at the same time, be in both a set and its complement.

Zadeh used fuzzy set theory to extend two-valued syllogistic reasoning to allow an indication of doubt in the premises or conclusion. He accomplished this by attaching arbitrary predicates to each term of the syllogism:

$$\begin{array}{cccc} Q1 & \mathcal{A}'s & are & \mathcal{B}'s \\ Q2 & \mathcal{C}'s & are & \mathcal{D}'s \\ \hline Q3 & \mathcal{E}'s & are & \mathcal{F}'s \end{array}$$

where $Q1$, $Q2$, and $Q3$ are numerical or, more generally fuzzy quantifiers (0.8, most, many, etc.), and $\mathcal{A}, \mathcal{B}, \ldots$ are crisp or fuzzy predicates.

Fuzzy set theory lends itself to applications using a rule base. However, the rules are directional and inflexible; they cannot take into account factors that are not explicitly stated in the rules, making them context insensitive (independent from conditions not included in the rule). Conversely, with Bayesian probability theory the connections are directionless and conditional independence is explicit, not imposed by the formal structure (18:8-10).

2.2.1.4 Dempster-Sheafer theory With Dempster-Sheafer theory the uncertainty in the probability assessment is implicitly represented as unattributed probability. For instance, given two mutually exclusive and collectively exhaustive events: $\mathcal{A}$ and $not\mathcal{A}$, $P(\mathcal{A}) = .6$ and $P(not\mathcal{A}) = .1$, the remaining .3 is left unspecified. The unspecified probability represents the subjective ignorance about the objective probability of $\mathcal{A}$, which could lie anywhere between .6 and .9. The inclusion of the unspecified probability constitutes a type of sensitivity analysis. It delineates the precision of the information and the improvement that may be possible in gathering further information about A. Probability theory can accomplish this same task by explicitly representing the subjective ignorance (18:9).

> Today it seems as if there were a conflict between the Bayesians and the Shaferians in the field of AI applications of probabilistic inference. True, this seems rather to be the Bayesians' stand...The view of the other position is that Shafer's theory is a generalization of Bayesian theory, thus seemingly implying broader possibility of applications...
>
> The point here is that Dempster's rule can be understood as a generalization of Bayes theorem, but it is not the unique possible generalization. It is this non-uniqueness that creates the justification problem...
>
> In general, we can say that recent work in the field called, among philosophers, 'probability kinematics'...has shown that there exists an entire class of kinematics or generalised conditionals in the belief space given by all admissible probabilities in a frame of discernment, and that conditionals (both Bayes and generalised) are nothing more than rules for combining posterior bodies of evidence with prior bodies of evidence. The belief space is endowed with certain structure and a particular operation: provided that different conditions hold, this operation reduces to different conditionals or different combinations rules as its own particular cases. (2:709-710)

The choice of which method to use is not clear. The debate seems to lean in the favor of the Bayesians. However, both methods claim a following, and the choice of which to support seems to depend on context. Garbolino goes on to suggest that,

> There is no mechanical method for deciding which probabilistic rules of inference to apply in a given decision problem. This decision is a meta-decision which depends upon the ingenuity of the decision-maker and his associates in analyzing the problem, upon the logical structure of the frame in which it is possible to embed the problem, upon the 'quality' of the available evidence and upon the constraints in time and resources (even computational resources) which could prevent a refinement of the frame of discernment. (2:715)

So the method is left up to the decision-maker and is context dependent. Still a possible guideline is that the chosen method should adhere to logical coherence. With this in mind, Garbolino provides a basis for choice:

> The updating procedure for a knowledge-base is a *two-step* procedure; the first step is calculating the degree of support provided by the new evidence; the second step is updating the process properly said, that is, *propagating* the effect of the new evidence through the knowledge-base, maintaining its coherence. (3:735)

Bayesian updating accomplishes these two steps simultaneously; Shafer's updating accomplishes the first one only and, if one is interested in coherence, it raises the need to supply to the inference engine a "coherence maintenance" mechanism (3:736).

It is not clear that there is a "correct" method for dealing with uncertainty. Application determines the tool that should be used. Of the four methods listed, fuzzy set theory attempts to quantify linguistic uncertainty and certainty factors are a heuristic derivation of probability theory attempting to control computational and data burdens; probability theory and Dempster-Sheafer theory attempt to describe general uncertainty and support it with a theoretical foundation.

Of the four methods above, Bayesian probability is the only method that demonstrates consistence and logical coherence. The other methods, though are in some sense appealing representations, are restricted due to their lack of demonstrated consistence and coherence.

> [Kyberg concludes]...ii) That the treatments of uncertain evidence in both Bayesian and non-Bayesian updating are reducible to the corresponding treatments of certain evidence, and iii) that non-Bayesian updating yields more determinate belief states as outcomes, but that the benefits afforded by non-Bayesian updating is limited and questionable. (12:285)

This doesn't end the debate, for the real roots of controversy lie in the criticism that people do not use probability for reasoning, and thus follows the disagreement between those who support the *normative* view and those who support the *descriptive* view. This disagreement over representation is more philosophical than methodological. Probabilists take the normative view, saying that uncertainty should be represented not as people see it, but as they should if they want to act consistently and logically. Proponents subscribing to the descriptive view believe that uncertainty should be represented in a manner consistent with the way people represent it (15:3).

2.2.2 Emulation When the expert system rule base is compiled, knowledge engineers attempt to capture the knowledge, judgment, and diagnostic process of the expert. Often the translation from the expert's framework into the rule base is imprecise, resulting in rules that do not capture the true experience of the expert. This is not surprising since humans are not machines and do not think as computers process information. "The foundation for human reasoning is...rather vague, since on the whole the mind must still be considered as a black box" (10:745). Because the human reasoning process evades explicit understanding, the task of emulating the expert is practically impossible. The best that can be hoped for is good pattern matching (9:676).

For uses where rational reasoning is desired, attempting to emulate humans seems to be misdirected. As Garbolino states, "a procedure which models natural reasoning yields a conclusion based upon natural reasoning, not a *reasonable* conclusion based upon reasonable reasoning" (3:730). However, natural reasoning should not be abandoned just yet, because, though we lack the quantitative skills to handle data efficiently, humans can adapt and function in unknown territory:

> Human reasoning is generally acknowledged to be inefficient in terms of accuracy and speed, but highly efficient in terms of versatility and the ability to comprehend novel events... Artificial systems, on the other hand, must be told everything beforehand. If the domain is sufficiently simple and well-described, an artificial system may do well and may even surpass humans in terms of sheer reasoning power (speed, endurance, precision). But if the domain is more complex, and in particular if novel situations can arise, the artificial system will probably encounter serious difficulties. (10:743-746)

The strength of human reasoning lies in its adaptability; its ability to reason about the unknown. If the fundamental mechanisms underlying human thought can be captured in a computer system, then artificial intelligence will no longer be an oxymoron. Capturing the underlying process is a worthy goal, but until the fundamental mechanism is captured, attempts to emulate the human mind will remain empirical in nature and, therefore, be restricted in applicability.

2.2.3 Rationality The importance of a rational decision maker in automated systems stems from the issue of responsibility. "Users often rely blindly on the programs they use, e.g. large statistical packages, and the faults in these programs therefore have potentially serious consequences" (9:676). To avoid unpredictability, a system must exhibit *logical coherence*[7]. A system should reach the same conclusion when given the same evidence regardless of the order in which it is presented[8].

> A perennial question concerns the level of rationality of the decision maker, hence the quality of the cognitive processes that must be described. It is generally acknowledged that decision makers are far from being rational in any normative sense. The question remains, however, whether decision makers should be considered as inherently irrational–and accordingly in conflict with established decision principles–or rather as quasi-rational, i.e. striving to perform according to rational principles, but failing to do so (because of cognitive limitations, etc.). (9:671)

The term *quasi-rational* is essentially bounded rationality. Bounded rationality is a model of how decisions are made. People make decisions based on maximizing utility; limitations on the amount of information and the ability to process it coherently cause the appearance of bounded behavior (14).

> If your task is to build a machine for simulating [the] human mind, then it is true that coherence is not relevant, and if you will succeed in building an incoherent machine, you will be a good scientist indeed. But if your task is, more modestly, to build a machine for "intelligent decision support," then, if you do not care about the logical coherence of your machine, you are a new Dr. Strangelove. (3:735)

[7] To be *logically coherent*, beliefs must not be self-contradictory; a knowledge-base must not contain at the same time a proposition and its negation (3:729).

[8] This does not mean that sequence of occurrence is irrelevant. However, if sequence is important then it evidence that should be included for consideration.

It seems the prudent course of action would be to apply artificial systems in ways that they are efficient and capable, helping the decision maker remain rational. This means exploiting their ability to keep track of large amounts of information and the inter-relationship of it, while keeping focused on the guiding principle of logical coherence throughout system development.

2.2.4 Decision Analysis Decision analysis is ideally suited for decision support if logical coherence is a criterion for selection. Decision analysis tackles rare problems that are large and cumbersome, providing organization and methodology for logical reasoning in decision making. Decision analysis has been employed for complex decision problems because it is rational and logical, consistent, and can incorporate utility into the decision process.

> Decision analysis (D/A) provides the techniques to allow for an explicit representation and organization of the decision factors, such that logic can be applied to identify the preferred decision strategy. The axioms on which it is based provide a set of criteria for consistency among beliefs, preferences, and choices that "should" be adhered to by a rational decision maker. The D/A methodology provides a systematic way to choose among alternatives by considering the problem structure, uncertainties, and relative utilities of pursuing different options. Finally, the process of developing probability and utility estimates yields a model that can be validated piecemeal, yet with a structure that ensures a level of validity in the completed model, and a methodology for validating the completed decision aid. (8:2)

The pursuit of a rational decision maker can succeed through the use of decision analysis. Furthermore, because decision analysis uses Bayesian inference for evidence manipulation, it is coherent and therefore conforms to the philosophical requirements outlined above. Recent research is lending suppo:t to the conclusions of this philosophical approach. Kalagnanam and Henrion compared decision analysis and expert rules for sequential diagnosis:

> The results of this study clearly indicate that the test sequences provide[d] by the experts (in the task domain) are suboptimal. Unfortunately, there is

> uncertainty regarding the objectives which motivate the expert test sequences. This restrains us from drawing firm conclusions about the efficacy of human intuition for this task domain. But it is important to remember that one of the experts accepted the validity of the C/p[9] sequence and felt that the results are likely to be of practical interest. This suggests that the normative theories of decision making are capable of obtaining results which go beyond current expert opinion. (11:211)

Heckerman relates a situation where an expert user noticed a marked improvement in the performance of a diagnostic system when they changed the inference scheme from one based on Dempster-Shafer theory of belief to one based on a special case of Bayes' theorem. In this study, Heckerman compared three inference approaches:

- a special case of Bayes' theorem
- an approach related to the parallel combination function in the certainty-factor (CF) model
- a method inspired by the Dempster-Safer theory of belief

Heckerman points out the observed superiority of the method based on the Bayesian approach (6:158, 166-168).

To date, automated systems are still just computer code and cannot replace humans for the chore of facing unique situations. The domain of an artificial system must be completely specified beforehand. "Put simply, unless the system knows about something, it is unable to reason about it" (10:746). Decision analysis has proven to be a rational, consistent decision maker. This makes it ideally suited for reasoning with uncertainty in automated systems. Until a sound foundation for understanding human thought can be quantified and codified, emulation of the human expert by computers is a field where smoke and mirrors prevail and only systems endowed with artfully constructed heuristics play.

[9] C/p refers to the algorithm used in the study. 'C' refers to the cost of testing a component and 'p' refers to the probability of that component failing.

2.3 Summary

This chapter presented a view of the present state of uncertainty reasoning and expert systems from both an operational and a philosophical perspective. As presented above, the representation of uncertain knowledge has many possible paths that are intuitively pleasing in presenting different concepts of uncertainty. Questions about applicability arise when the methods are used for reasoning with uncertain information. The fact that the methods are different isn't an immediate problem; the lack of consistency and logical coherence is a problem. Probability theory demonstrates these properties while the other methods do not. From a theoretical standpoint, the lack of consistency and logical coherence is a major litmus test; if a method does not demonstrate these properties they are basically empirical in nature and have a restricted range of validity. For operational use an empirical method is as good as any as long as it operates within its valid range.

Though probability has a firm theoretical foundation, there remains room for representation and interpretation of the probability model. Chapter III presents *probability ratio graphs* which is a representation of the probability model. Chapter IV presents several interpretations on how probability deals or can deal with some differing facets of uncertainty.

III. Probability Ratio Graphs: An interpretation

And what is a weed? A plant whose virtues have not been discovered.
–Ralph Waldo Emerson

The basic architecture addressed by the research tool is an implementation of the format described by Morlan in his paper *"A Decision Analytic Approach to Building Expert Systems."* In this paper Morlan develops a format based on odds-ratios for representing a decision analytic framework with probability ratio graphs. He points out that the chosen representation for a mathematical model only serves to clarify meaning for the model builder and has no effect on the underlying mathematics. (13:1) The odds-ratio format is ideal for representing the information required for the decision analytic approach:

1. $P(\mathcal{E}_j \mid \mathcal{H}_i)$ – the conditional probabilities or likelihoods for the evidence and hypotheses.
2. $P(\mathcal{H}_i)$ – the prior probabilities for the hypotheses.
3. $U(A_k \mid \mathcal{H}_i)$ – the utility information for the hypotheses and actions[1].

The information listed is needed for making optimal decisions under uncertainty. Morlan shows that the relative utilities between competing actions is important, not the absolute values. The information needed for determining those relative utilities is contained in the ratios between the hypotheses and actions. (13:3) The following discussion on probability ratio graphs is adapted from Morlan's work and is only an overview of the material contained therein[2].

[1] This notation, $U(A_k \mid \mathcal{H}_i)$, represents the utility of performing action A_k when hypothesis $\mathcal{H}_i$ is the true state of the world (13:23).

[2] The forthcoming discussion is a condensed, adapted version of Morlan's presentation. Some areas have been simplified and other aspects interpreted. For an in-depth mathematical presentation, the original document should be solicited.

3.1 Probability ratio graphs

The research tool employs probability ratio graphs for representing the likelihood, prior, and utility information necessary for using the decision analytic approach. There are many approaches for representing probability (Venn diagrams, contingency tables, causal nets, probability trees, to name a few). Probability ratio graphs present another face in the crowd representing the same mathematics of probability. They are similar to influence diagrams and causal nets, providing a method for decomposing ungainly probability models into manageable configurations. However, probability ratio graphs offer advantages when addressing the concerns of completeness and validation. They also clarify implications of independence assumptions by explicitly mapping the transfer of statements about causal dependence into the mathematics of probability distributions.

The fundamental concept underlying probability ratio graphs is that in decision making the discriminating feature between competing alternatives is their relative measure. The ratio of their measures captures the important information in the decision analytic sense.

The function of probability ratio graphs is the representation and manipulation of probability and utility relational information. The representation involves two basic constructs: vertices and arcs. The manipulation function uses two basic procedures: triangulation and aggregation.

3.1.1 Representation The first function of the probability model is representing marginal and conditional probabilities. Figure 1 shows how the two basic constructs are related in a compound probability ratio graph (G will hereafter represent a probability ratio graph). Various combinations of these constructs can represent quite complex Bayesian inference problems involving many levels of disjoint hypotheses and their corresponding evidence states. The following definitions will help facilitate discussion:

Event: An assignment to a random variable, or set of random variables.

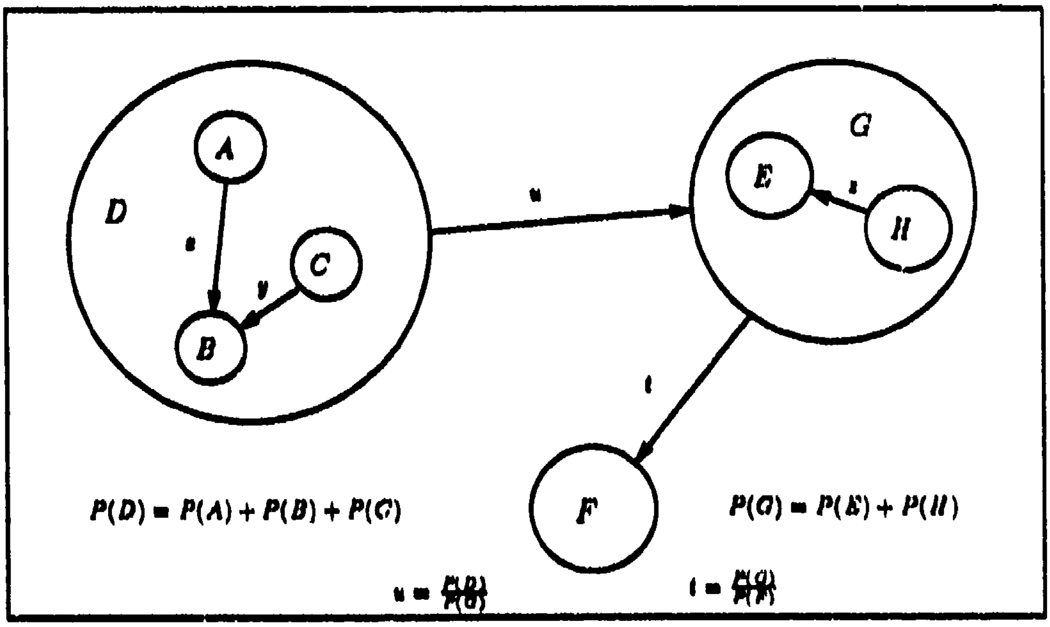


Figure 1. A compound probability ratio graph.

Vertex: An event and an embedded probability ratio graph. The event could be a hypotheses representing a possible state of the world or a possible evidence state which is used to reason about the possible hypotheses. The embedded G represents conditional probabilities, conditioned on the parent vertex.

Arc: A directed weighted arc connecting two vertices which are on the same level. It has three associated parameters: Head, Tail, and Ratio.

$$((Head, Tail), Ratio)$$

The *head* can be thought of as where the arc originates; likewise, the *tail* is where the arc ends. The *ratio* is the ratio of the probabilities, *head* to *tail*: $\frac{P(head)}{P(tail)}$.

Graph: A set of vertices, V, and a set of arcs, A, connecting those vertices. To be a legitimate *probability ratio graph* two conditions must be met:

1. The set of vertices, V, must form a set of mutually exclusive events.
2. The arcs, E, must form a minimal spanning tree (no cycles).

3.1.2 Manipulation The second function of the probability model is implementing probability calculus consistently. Probability ratio graphs use two basic types of functions to perform probability calculus: 1) triangulation, and 2) aggregation and disaggregation.

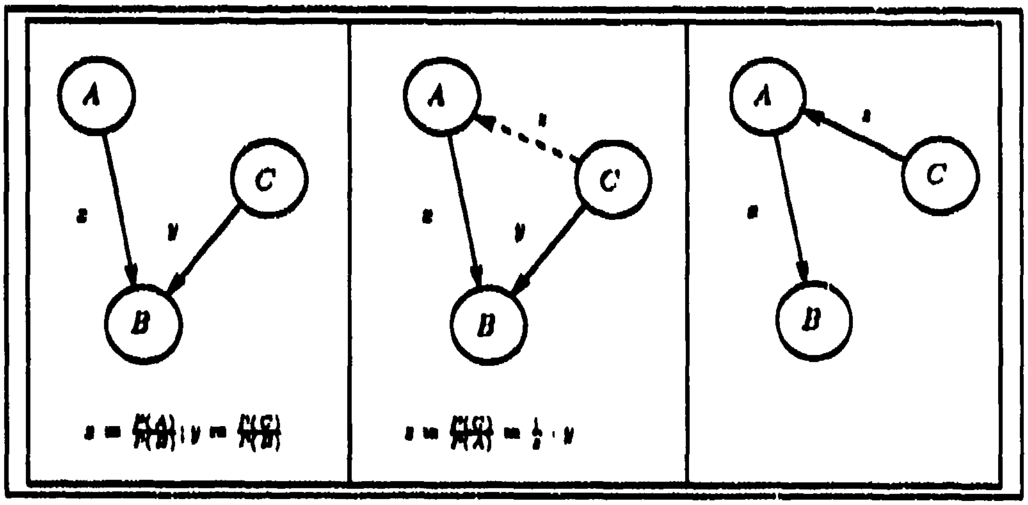

A
C
B
x
y
$x = \frac{P(A)}{P(B)}; y = \frac{P(C)}{P(B)}$
A
C
B
z
x
y
$z = \frac{P(C)}{P(A)} = \frac{1}{x} \cdot y$
A
C
B
z
x

Figure 2. A graphic depiction of triangulation.

Triangulation: Allows the computation of a missing third arc, given the two arcs which together with the third arc form a triangle. Figure 2 shows the function of triangulation. Its use will become evident in later examples.

Example 3.1 **Triangulation**

Given the situation as described in figure 2, triangulate between vertices from C to A:

$$x = \frac{P(A)}{P(B)}; y = \frac{P(C)}{P(B)}$$

$$z = \frac{P(C)}{P(A)} = \left(\frac{P(B)}{P(A)}\right) \cdot \left(\frac{P(C)}{P(B)}\right) = \frac{1}{x} \cdot y$$

Note: The arcs are directional. If the triangulation was from A to C, then,

$$z = \frac{P(A)}{P(C)} = x \cdot \frac{1}{y}$$

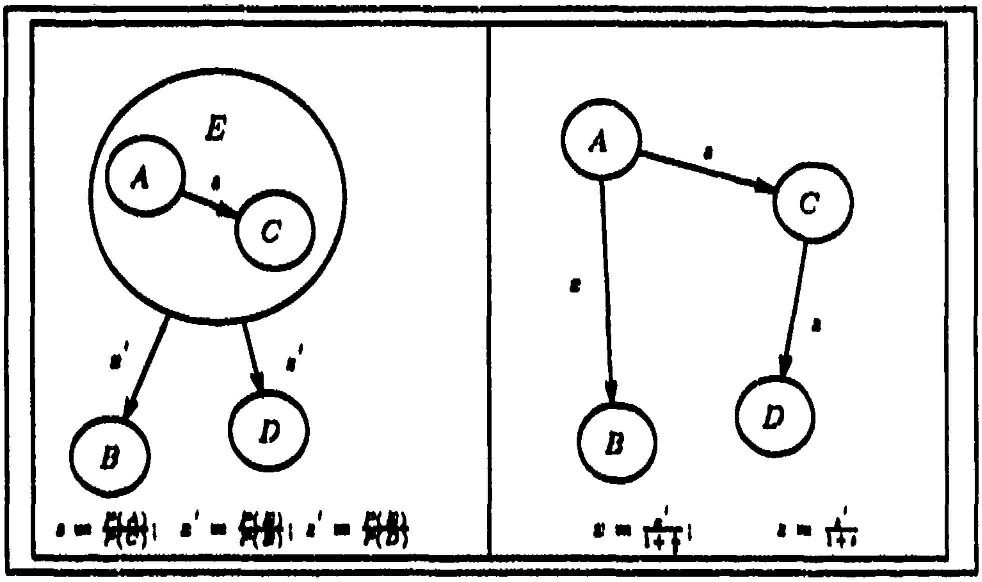

E
A
s
C
x'
s'
B
D
$s = \frac{P(A)}{P(C)}$; $x' = \frac{P(B)}{P(E)}$; $s' = \frac{P(E)}{P(D)}$
A
s
C
x
s
B
D
$x = \frac{x'}{1+\frac{1}{s}}$;
$s = \frac{s'}{1+s}$

Figure 3. A graphic depiction of aggregation.

Aggregation: Allows for the expansion of an embedded G up one level while maintaining the original probabilistic relationships. Aggregation changes the probabilistic relationships of the embedded graph from conditional probabilities into joint probabilities. Figure 3 is a graphic representation of aggregation.

Example 3.2	**Aggregation**

Given the situation as described in figure 3, aggregate the vertices A and C from E:

$$s = \frac{P(A)}{P(C)};\, x' = \frac{P(B)}{P(E)};\, s' = \frac{P(E)}{P(D)}$$

$$x = \frac{P(B)}{P(A)} = \frac{x'}{1+\frac{1}{s}}$$

$$s = \frac{P(C)}{P(D)} = \frac{s'}{1+s}$$

Note: The original connections and arc directions determine how the arc weights are combined.

Note: According to Morlan, the term "aggregation" refers to the transformation of two graphs into one graph.

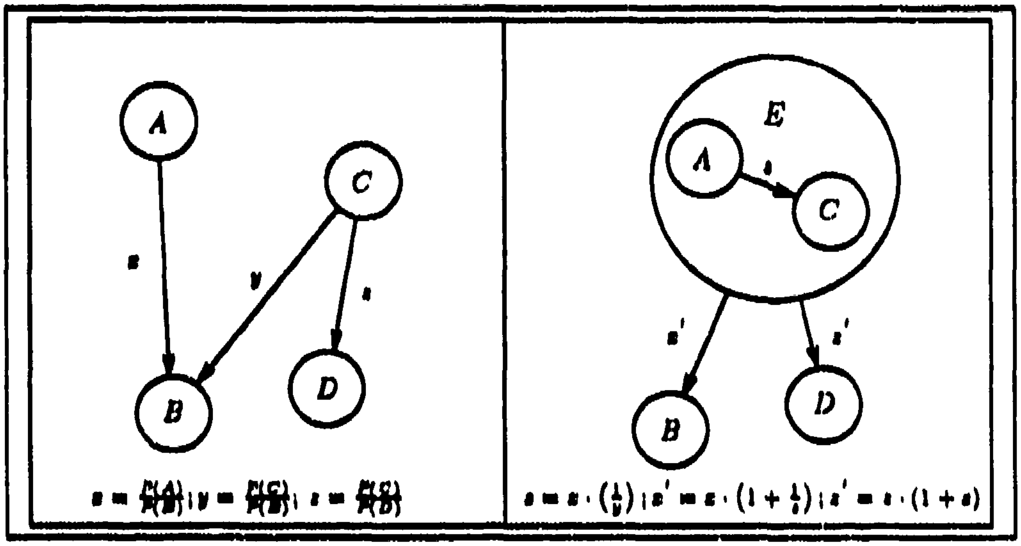

A
C
x
y
z
B
D
$x = \frac{P(A)}{P(B)}; y = \frac{P(C)}{P(B)}; z = \frac{P(C)}{P(D)}$
E
A
s
C
x'
z'
B
D
$s = x \cdot (\frac{1}{y}); x' = x \cdot (1 + \frac{1}{s}); z' = z \cdot (1 + s)$

Figure 4. A graphic depiction of disaggregation.

Disaggregation: Allows for the collection of several vertices into one vertex which then occupies the disaggregated vertices places in the original $\mathcal{G}$ and contains the disaggregated vertices as an embedded $\mathcal{G}$ in the new vertex. The reorganising leaves the overall probability relationships intact while representing the disaggregated vertices as conditional probabilities.

Example 3.3 Disaggregation

Given the situation as described in figure 4, disaggregate the vertices A and C: First, a triangulation must be performed between A and C, the direction of triangulation is of no consequence.

Triangulate from A to C giving $s = x \cdot \left(\frac{1}{y}\right)$.

$$s = \frac{P(A)}{P(C)}; x = \frac{P(A)}{P(B)}; z = \frac{P(C)}{P(D)}$$

$$x' = \frac{P(A) + P(C)}{P(B)} = x \cdot (1 + \frac{1}{s})$$

$$z' = \frac{P(A) + P(C)}{P(D)} = z \cdot (1 + s)$$

Note: The original connections and arc directions determine how the arc weights are combined. However, the differences are just a matter of inverting the weights as needed.

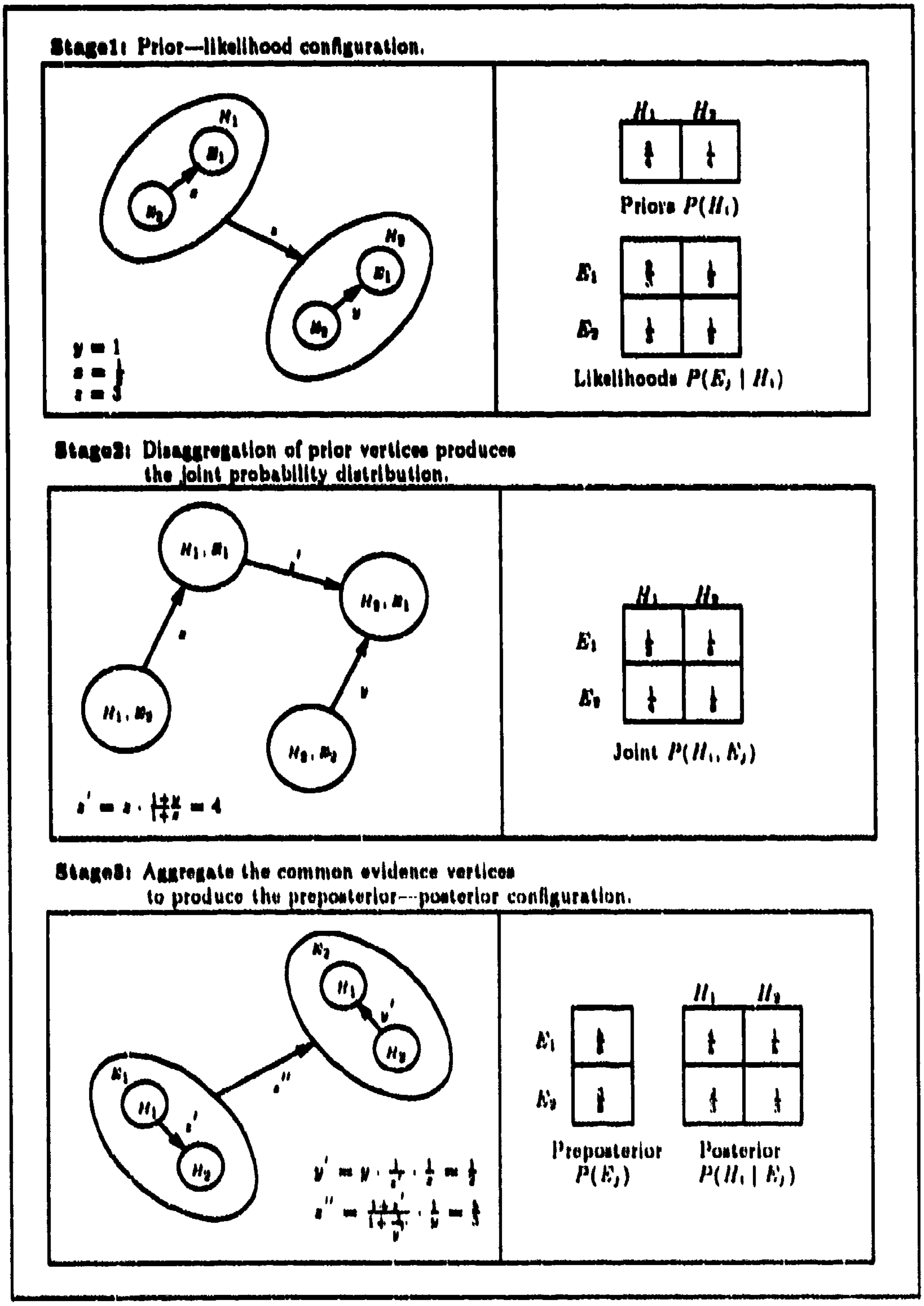
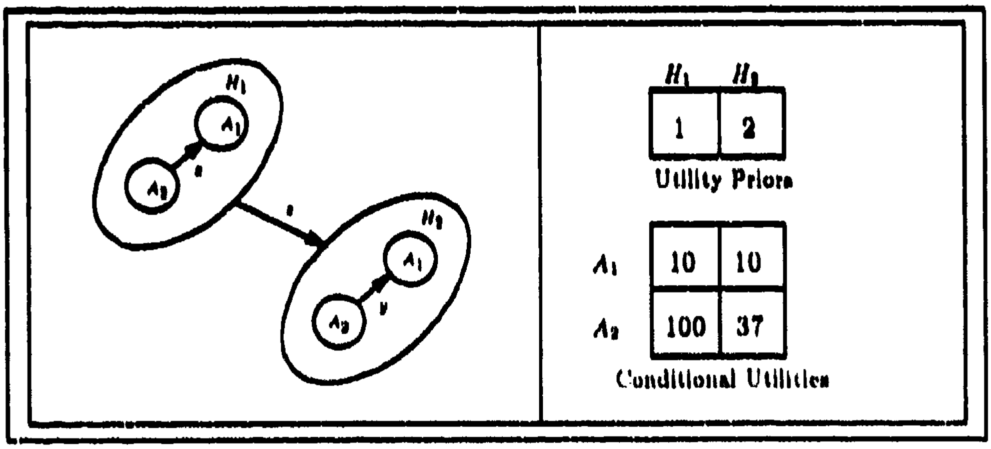

Stage1: Prior—likelihood configuration.
H_1
H_2
E_1
E_2
x
y
z
y = 1
z = 3
Priors $P(H_i)$
Likelihoods $P(E_j \mid H_i)$
Stage2: Disaggregation of prior vertices produces the joint probability distribution.
H_1, E_1
H_2, E_1
H_1, E_2
H_2, E_2
z'
Joint $P(H_i, E_j)$
Stage3: Aggregate the common evidence vertices to produce the preposterior—posterior configuration.
y'
x'
z''
Preposterior $P(E_j)$
Posterior $P(H_i \mid E_j)$

Figure 5. Simple application of Bayes' theorem using a probability ratio graph compared with tables.

3.1.3 Bayesian inference With the forgoing tools for representing and manipulating probabilities, an example of Bayesian inference is possible. Figure 5 is a simple example of a Bayesian cycle using probability ratio graphs with the corresponding tabular state. The example in figure 5 shows that the $\mathcal{G}$ transitions through three different configurations: 1) Prior—Likelihood configuration, 2) Joint configuration, and 3) Preposterior—Posterior configuration.

3.1.4 Utility representation The utility representation follows the same path as the hypotheses–evidence representation. Instead of conditional probabilities (likelihoods), $P(\mathcal{E}_j \mid \mathcal{H}_i)$, we have conditional utilities, $U(\mathcal{A}_k \mid \mathcal{H}_i)$, which represent the utility of performing action $\mathcal{A}_k$ given that Hypotheses $\mathcal{H}_i$ has occurred. The process is exactly the same, except that instead of the posterior probabilities, we are now after the preposterior utility information.



H_1	H_2
1	2

Utility Priors

A_1	10	10
A_2	100	37

Conditional Utilities

Figure 6. Utility ratio graph representation.

With the utility ratio graph, the same interpretations are applied to different information. The "prior" information is a scaling factor between the isolated conditional utility information. Figure 6 shows that the conditional utilities for $U(\mathcal{A}_1 \mid \mathcal{H}_1) = 10$ and $U(\mathcal{A}_1 \mid \mathcal{H}_2) = 10$, and that the scaling factor between $\mathcal{H}_1$ and $\mathcal{H}_2$ is $\frac{1}{2}$. This information states that performing $\mathcal{A}_1$ if $\mathcal{H}_2$ has occurred is worth twice as much as performing $\mathcal{A}_1$ if $\mathcal{H}_1$ has occurred, because $U(\mathcal{A}_1, \mathcal{H}_1) = 10$ while $U(\mathcal{A}_1, \mathcal{H}_2) = 20$.

3.1.5 Combining probability with utility Combining the probability and utility information takes place with the posterior probabilities, $P(\mathcal{H}_i \mid \mathcal{E}_j)$, and the utility scaling factors, $U(\mathcal{H}_i)$. The posterior probabilities are multiplied with their respective scaling factors and the utility ratio graph is manipulated to produce the utility information conditioned on the state of evidence, $U(\mathcal{A}_k \mid \mathcal{E}_j)$. Mathematically, this process is similar to the law of total probability:

$$U(\mathcal{A}_k \mid \mathcal{E}_j) = \sum_{i=1}^{n} U(\mathcal{A}_k \mid \mathcal{H}_i) \cdot P(\mathcal{H}_i \mid \mathcal{E}_j) \cdot U(\mathcal{H}_i) \tag{1}$$

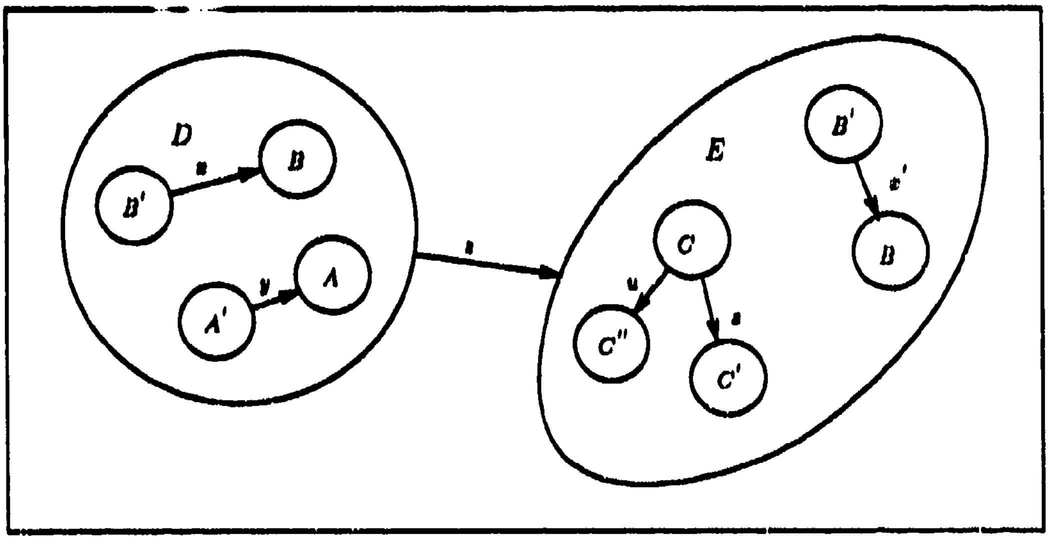


Figure 7. Independence representation with ratio graphs.

3.2 Independent evidence

According to the description of probability ratio graphs, the vertices of a graph represent disjoint events. This extends to the evidence states that are contained in a hypotheses vertex as well. Requiring enumeration of disjoint evidence states brings combinatorial explosion with independent evidences. The number of disjoint evidence states grows exponentially (2^n) with the number of independent evidences. This is clearly an undesirable situation and combating this exponential growth requires a "tweak" in the probability ratio graph concept.

Representing independence in a probability ratio graph requires separate graphs. In Morlan's description, he does not allow for the possibility of multiple (disconnected) embedded graphs, however, this addition to the concept is a natural outgrowth of independence. Figure 7 is a depiction of two independent probability ratio graphs transitioning into their joint representation. The mathematics of probability ratio graphs treat the vertices in a connected graph as disjoint, collectively exhaustive states. The semi-complex graph in figure 7 also treats each separate graph in this way.

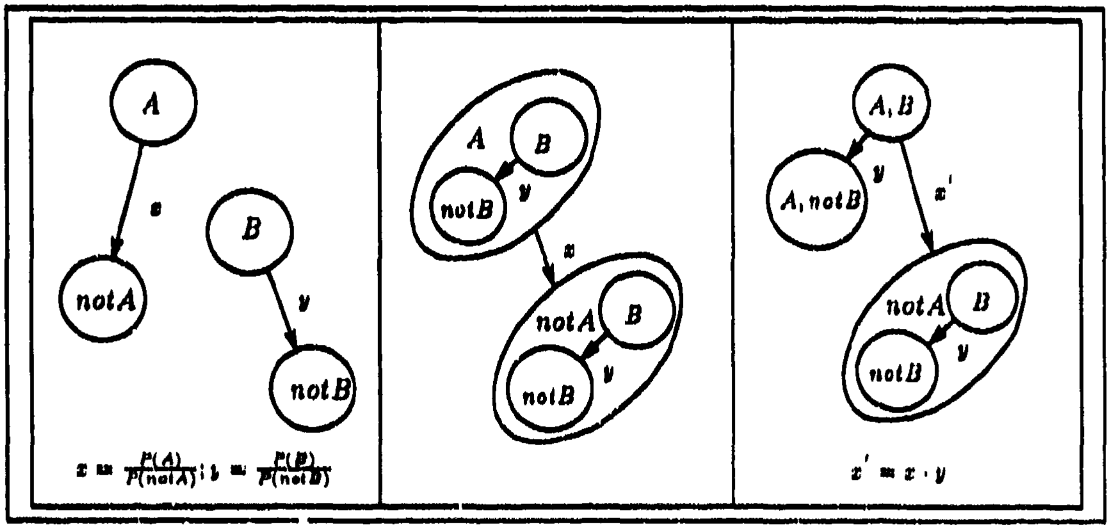


Figure 8. Joint occurrence of independent events.

There are two ways in which to use independent evidence for developing the posterior distribution using probability ratio graphs. One method uses the nesting approach shown in figure 8, the other involves sequential probability updating. The first method is the same as combining the independent likelihoods then applying Bayes' theorem. Sequential probability updating involves using the posterior probabilities to replace the priors and continuing the inference process. If the evidence states are independent then performing sequential updating will produce an equivalent result to using the disjoint evidence state and performing the inference step once. Both methods apply only when the evidences are independent.

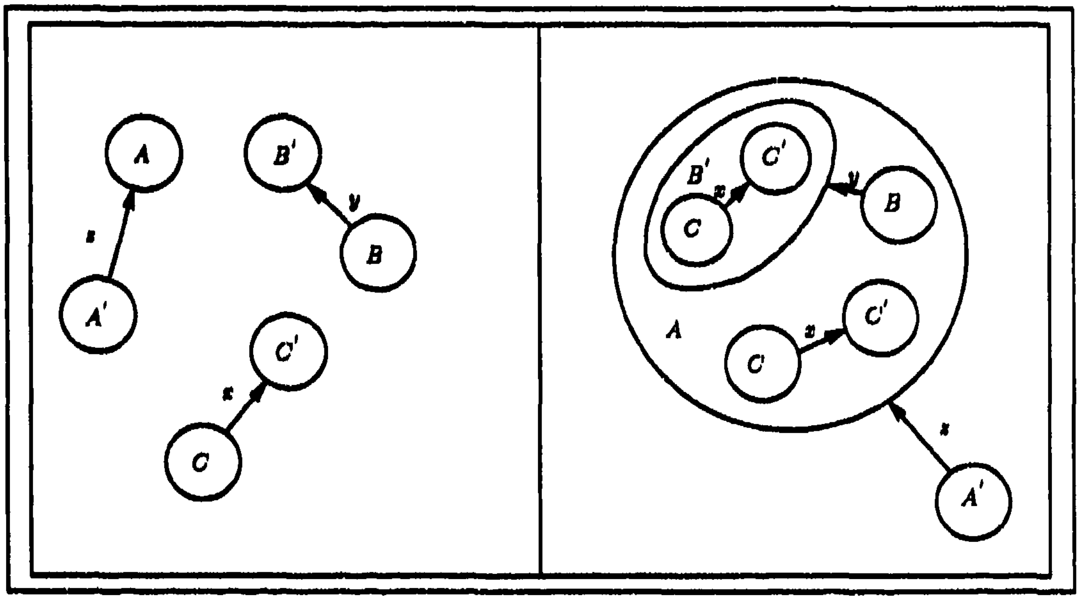


Figure 9. Nesting independent events.

The independence representation described requires support which is not supplied by Morlan's initial concept. To represent independence, vertices must be able to have multiple embedded graphs. This is an easy step to implement, it is merely a redefinition to include the possibility for multiple embedded graphs:

Vertex: An event and embedded probability ratio graphs. The event could be a hypotheses representing a possible state of the world or a possible evidence state which is used to reason about the hypotheses. An embedded $\mathcal{G}$ represents conditional probabilities, conditioned on the parent vertex. Multiple $\mathcal{G}$'s represent independent event states.

Supporting the mathematics associated with independence is more involved. A "nesting" procedure needs to be added to triangulation, aggregation, and disaggregation. And aggregation needs to be enhanced to account for the propagation, or replication of the remaining independent graphs in the aggregated graph. A possible nesting procedure is depicted in figure 9.

3.3 Current Software

The software developed for this work is generally an implementation of Morlan's probability ratio graph concept on a small scale (it does not include the independence capability). It can support the simple probability functions of representing conditional and joint probabilities, and can accomplish Bayes' theorem manipulations. The simple features can also represent quite complex systems if a nested[3] approach is used. Vertices on any level can contain embedded graphs. Using this approach, the graph is basically a tree representation as shown in figure 10. The software limits the total number of vertices that can be active[4] at any one time, but conceptually there is no limit to the possible number of levels.

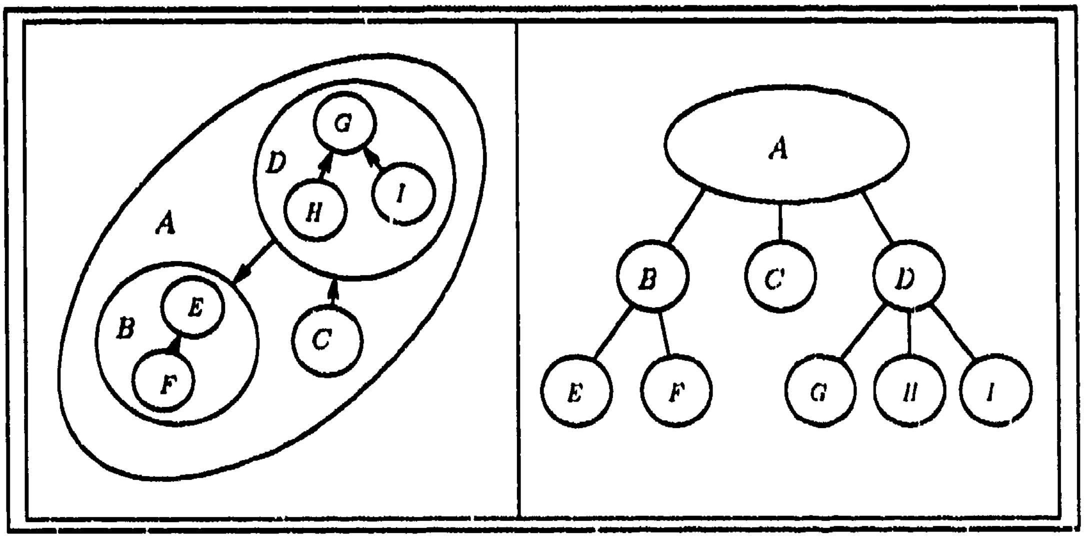


Figure 10. Comparison of a compound vertex to a tree representation.

The software divides the screen in four separate *viewports*, each viewport supporting different aspects of the program. Figure 11 shows a typical screen with the four viewports (clockwise from the top left corner): viewport 1, the prompting port for input; viewport 2, the menu port; viewport 4, results port for typed output; and viewport 3, the graphics

[3] "Nested" in this context refers to the multiple layers of graphs, not multiple graphs in a single layer.

[4] In this context "active" means all of the vertices which are specified at all levels in the system.

port. The program can present a vertex in two ways: 1) as a large pie graph which shows the relative probabilities of each embedded vertex as the angle subtended in the chart, or 2) as a small pie chart in the corner of the the graphics screen and its embedded graph shown as a connected graph with each embedded vertex shown as a pie chart of its likewise embedded graph. The net representation presents all the available information in a simple probability problem showing two complete levels and their relative proportions at once.

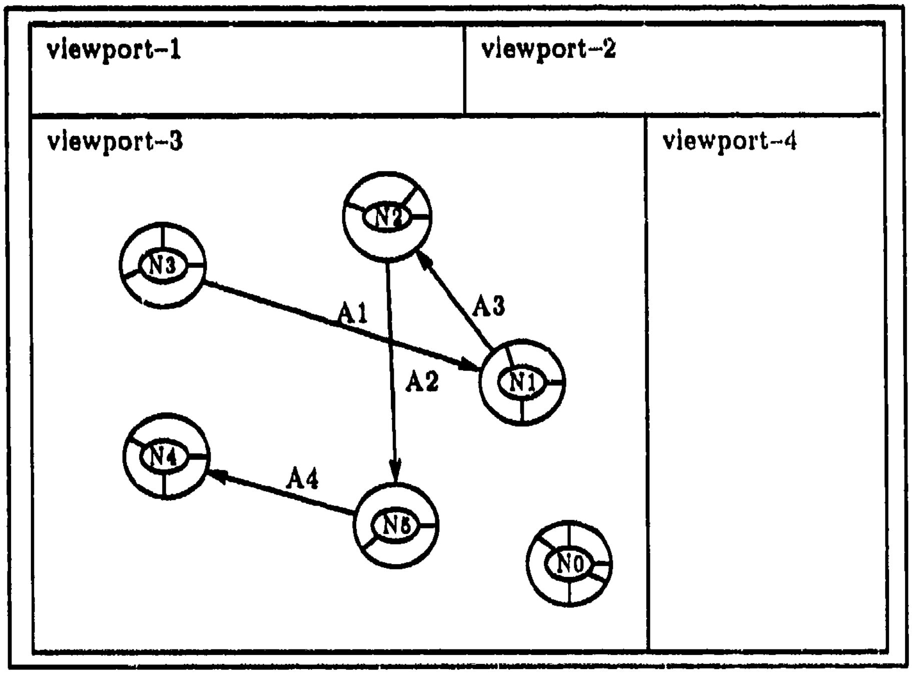


Figure 11. Net presentation of a compound vertex.

The software has strayed from Morlan's description in several areas. The triangulation function has been extended so that any two vertices in a common graph can be connected regardless of their current connection path[5]. If two previously linked nodes are triangulated it either has no effect or it reverses the arc direction, the outcome of which depends on the original arc orientation with respect to the new arc orientation. The aggregation function is "hot-wired" in that it figures the probabilities for the aggregated graph

[5] This extension equivalently triangulates between the vertices along the connection path.

and uses their probability measure to reconnect the vertices on the new level. Appendix A contains a discussion of the program's abilities, limitations, and use.

3.4 Summary

This chapter introduced an interpretation of Morlan's *probability ratio graphs* with possible enhancements. It also briefly introduced the PC-based tool that implements the probability ratio graph concept on a restricted level.

The addition of the independence enhancement will increase probability ratio graph's versatility and ability to represent complex decision problems. The current program is still very devlopmentally oriented. It would , no doubt, benefit from code simplification and technical modifications.

IV. Missing Information, Independence, and Secondary Uncertainty

During software generation several interesting topics arose that spawned questions of interpretation.

- When system parameters are unspecified, how should a diagnostic system deal with the missing information?
  A related question is, how does the difference between causal implication and conditional probability affect likelihood generation?
- Under the common assumption of disjoint hypotheses, what does hypothesis-evidence independence mean?
- What is the meaning of spurious evidence, and how does it effect the diagnostic system?

The design heuristics of decision support systems depend in part on the interpretations that result from these considerations. If the information described above is estimated it is important to understand the impact of the original parameters so its effect can be taken into account in the estimation.

4.1 Missing information

Morlan points out that during system development, "it cannot be guaranteed that all combinations of evidence will be generated for all hypotheses." (13:11) Under this situation the problem is how to continue the decision process when pertinent information is available but how it relates to some of the hypotheses under consideration is unknown. Clearly it is desirable to include all the available information as long as it is not a detriment to the decision process. The intent of this discussion is not to prove that a method is correct, but merely to present a proposed method and its intuitive interpretation.

Morlan presents a method wherein the missing evidence–hypotheses information is estimated under the assumption of probabilistic independence.

$$P(\mathcal{E} \mid \mathcal{H}) = P(\mathcal{E})$$

This does not mean there is no causal link between $\mathcal{H}$ and $\mathcal{E}$; in fact, it implies that there is a causal link between the two (see the section on independence for a discussion of this facet).

Another interpretation is that the lack of information implies that $\mathcal{H}$ and $\mathcal{E}$ are causally independent. This may be the case, but such an assumption has a potentially drastic effect on the posterior distribution. If the causality between $\mathcal{H}$ and $\mathcal{E}$ is truly unknown then an assumption of no link is a bold move.

Under the assumption of probabilistic independence, Morlan starts with the law of total probability to develop an equation for determining the missing information. For a case involving three hypotheses, with no information for $P(\mathcal{E} \mid \mathcal{H}_1)$, estimate:

$$P(\mathcal{E} \mid \mathcal{H}_1) = P(\mathcal{E}) = \frac{P(\mathcal{E} \mid \mathcal{H}_2) \cdot P(\mathcal{H}_2) + P(\mathcal{E} \mid \mathcal{H}_3) \cdot P(\mathcal{H}_3)}{1 - P(\mathcal{H}_1)}$$

By using this assumption the posterior probability $P(\mathcal{H}_1 \mid \mathcal{E})$ equals the prior probability $P(\mathcal{H}_1)$ and the posterior probabilities for the other hypotheses change (unless they too are probabilistically independent). In effect, the observance of $\mathcal{E}$ alters those hypotheses which have information relating to $\mathcal{E}$ and does not affect the others. Figure 12 shows a graphic interpretation of such a process.

Such an operation can be seen as partitioning the hypotheses into two sets: one set, $\mathcal{H}_{ind}$, containing the hypotheses which are probabilistically independent of $\mathcal{E}$, and one set, $\mathcal{H}_{dep}$, containing the hypotheses which are probabilistically dependent on $\mathcal{E}$. Figure 13 shows such a partitioning. The probability conditioning takes place on the dependent set, $\mathcal{H}_{dep}$ as if it were the entire hypotheses space. Then the two sets are recombined to provide the final posterior distribution, $P(\mathcal{H}_i \mid \mathcal{E})$.

The process can be performed in this manner without actually generating any missing information and the results are equivalent.

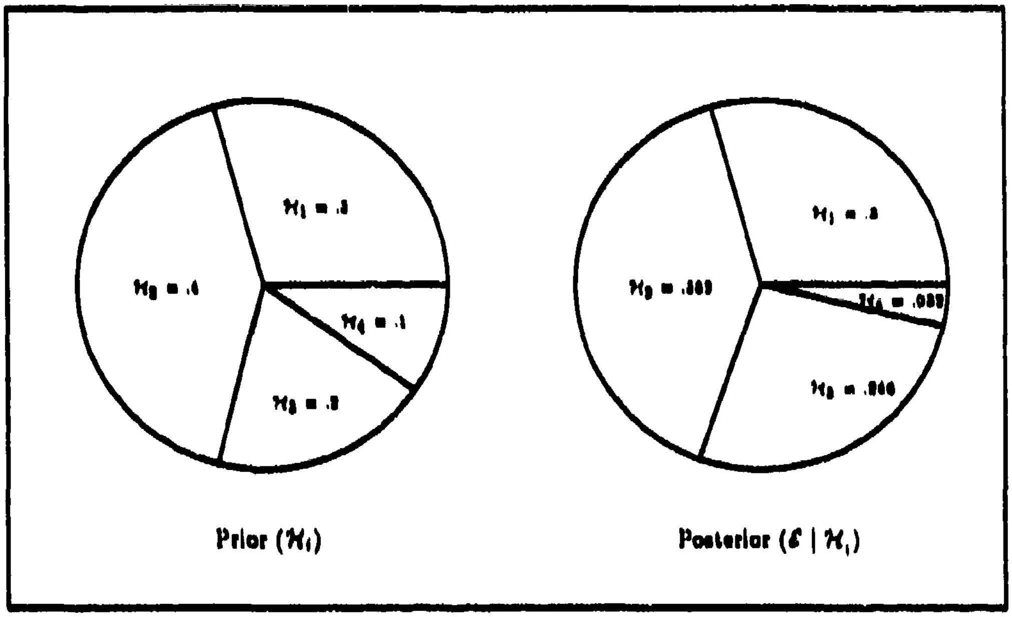


Figure 12. Graphic depiction of estimating missing information as probabilistically independent.

Example 4.1	**Missing Information**

Given the following information,

Priors			Likelihoods		
$P(\mathcal{H}_1)$	=	.3	$P(\mathcal{E} \mid \mathcal{H}_1)$	=	(missing)
$P(\mathcal{H}_2)$	=	.4	$P(\mathcal{E} \mid \mathcal{H}_2)$	=	.6
$P(\mathcal{H}_3)$	=	.2	$P(\mathcal{E} \mid \mathcal{H}_3)$	=	.9
$P(\mathcal{H}_4)$	=	.1	$P(\mathcal{E} \mid \mathcal{H}_4)$	=	.2

$$\begin{aligned} P(\mathcal{E}) &= \frac{P(\mathcal{E}|\mathcal{H}_2)\cdot P(\mathcal{H}_2)+P(\mathcal{E}|\mathcal{H}_3)\cdot P(\mathcal{H}_3)+P(\mathcal{E}|\mathcal{H}_4)\cdot P(\mathcal{H}_4)}{1-P(\mathcal{H}_1)} \\ &= \frac{(.6)(.4)+(.9)(.2)+(.2)(.1)}{1-.3} \\ &= .629 \end{aligned}$$

Assuming $\mathcal{E}$ and $\mathcal{H}_1$ are probabilistically independent, $P(\mathcal{E} \mid \mathcal{H}_1) = .629$. Applying Bayes' theorem to the data yields the following posterior probabilities,

$$\begin{aligned} P(\mathcal{H}_1 \mid \mathcal{E}) &= .3 \\ P(\mathcal{H}_2 \mid \mathcal{E}) &= .382 \\ P(\mathcal{H}_3 \mid \mathcal{E}) &= .286 \\ P(\mathcal{H}_4 \mid \mathcal{E}) &= .032 \end{aligned}$$

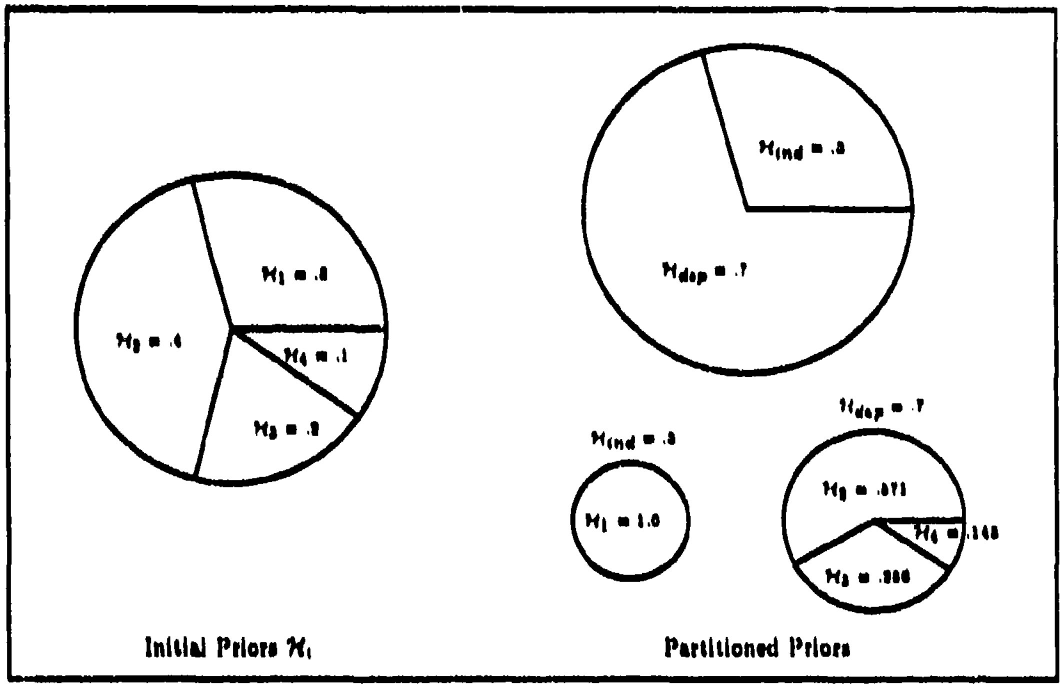


Figure 13. Graphic depiction of applying $\mathcal{E}$ by partitioning hypotheses with respect to probabilistic dependence.

Using the partitioning method, Bayes' theorem is applied only to the hypotheses that are connected to $\mathcal{E}$,

Priors	Likelihoods
$P(\mathcal{H}_2) = .4$	$P(\mathcal{E} \mid \mathcal{H}_2) = .6$
$P(\mathcal{H}_3) = .2$	$P(\mathcal{E} \mid \mathcal{H}_3) = .9$
$P(\mathcal{H}_4) = .1$	$P(\mathcal{E} \mid \mathcal{H}_4) = .2$

This delivers the intermediate posteriors,

$$P(\mathcal{H}_2 \mid \mathcal{E})' = .545$$
$$P(\mathcal{H}_3 \mid \mathcal{E})' = .409$$
$$P(\mathcal{H}_4 \mid \mathcal{E})' = .045$$

Recombination of the independent hypotheses, $\mathcal{H}_{ind}$, with the dependent hypotheses, $\mathcal{H}_{dep}$, involves scaling the intermediate posteriors with respect to their prior probabilities.

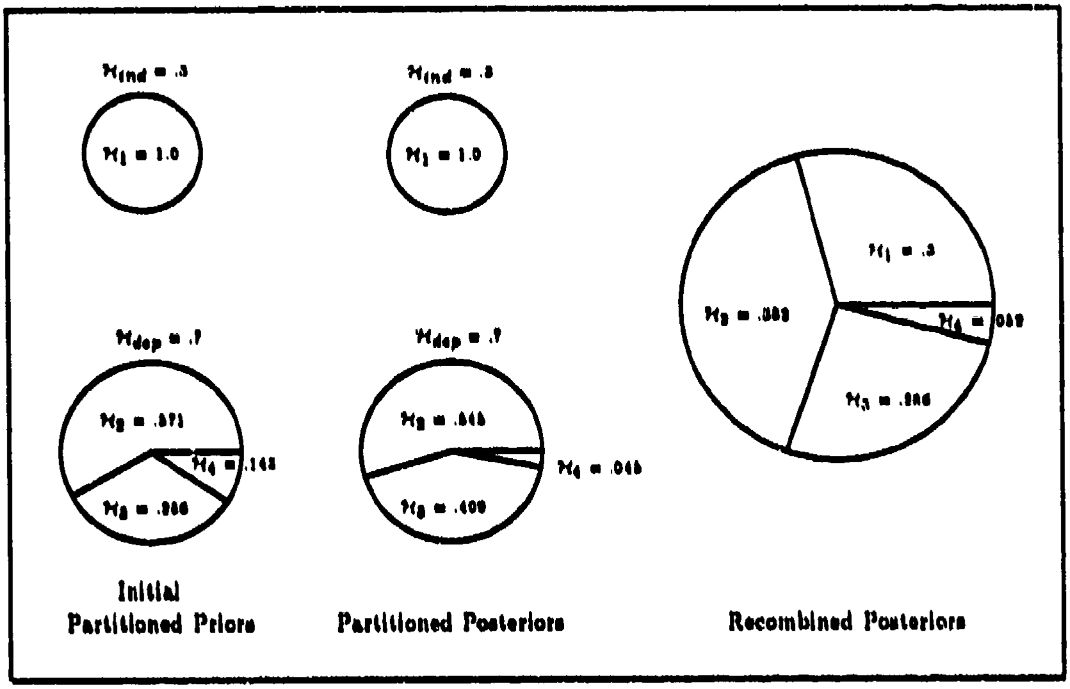


Figure 14. Graphic depiction of applying $\mathcal{E}$ by partitioning hypotheses with respect to probabilistic dependence (continued).

$$\begin{aligned} P(\mathcal{H}_{dep}) &= \textstyle\sum_{i=2}^{4} P(\mathcal{H}_i) \\ &= .4 + .2 + .1 \\ &= .7 \end{aligned}$$

$$P(\mathcal{H}_i \mid \mathcal{E}) = P(\mathcal{H}_2 \mid \mathcal{E})' \cdot P(\mathcal{H}_{dep})$$

Giving the same posterior distribution as the first method.

$$\begin{aligned} &P(\mathcal{H}_1 \mid \mathcal{E}) = .3 \\ &P(\mathcal{H}_2 \mid \mathcal{E}) = (.545) \cdot (.7) = .382 \\ &P(\mathcal{H}_3 \mid \mathcal{E}) = (.409) \cdot (.7) = .286 \\ &P(\mathcal{H}_4 \mid \mathcal{E}) = (.045) \cdot (.7) = .032 \end{aligned}$$

This example shows the intuitive appeal of assuming probabilistic independence when likelihood information is missing and there are no grounds for making a subjective assessment. It seems beneficial to include information when its availability can serve to differentiate between some of the competing hypotheses. At the same time, it would seem folly to exclude hypotheses just because the connections are unknown or not understood[1]. Using the assumption of probabilistic independence, as presented by Morlan, seems to balance the need to include the available information without treating the apparently underdefined hypotheses unfairly.

4.2 Likelihoods

The relationship between hypotheses and evidence is confused by the perceived equality of causality with conditional probability. Only under highly constrained conditions is $P(\mathcal{E} \mid \mathcal{H})$, the conditional probability, synonymous with $P(\mathcal{E} \mid_c \mathcal{H})$, the causal probability[2], ("$|_c$" indicates *causal* probability, not *conditional* probability). This difference causes problems in system development because in many situations it is easier to think of and estimate the probability that $\mathcal{H}$ caused $\mathcal{E}$, $P(\mathcal{E} \mid_c \mathcal{H})$, than it is to estimate the likelihood, $P(\mathcal{E} \mid \mathcal{H})$[3]. However, it is the likelihoods that are used in Bayesian inference, not the causal implications. Generating missing likelihood information brings out the important difference between conditional probability and causal implication.

The conditional probability, $P(\mathcal{E} \mid \mathcal{H})$, is information that is usually supplied through statistical analysis or expert opinion. However, in the event that either the existing data base is insufficient for valid statistical results, or the expert can, at best, supply causal implications, then likelihoods must be estimated. The likelihood $P(\mathcal{E} \mid \mathcal{H})$ is equal to the probability that $\mathcal{H}$ caused $\mathcal{E}$ plus the probability that $\mathcal{E}$ happened independently of $\mathcal{H}$ given that $\mathcal{H}$ did not cause $\mathcal{E}$.

Example 4.2 Likelihoods and causal information

Given the situation described in figure 15, how do the conditional probabilities, $P(\mathcal{E} \mid \mathcal{H}_i)$, relate to the causal probability, $P(\mathcal{E} \mid_c \mathcal{H}_i)$? The conditional probabilities of interest are readily evident from the Venn diagram in figure 15:

[1] Excluding hypotheses because of insufficient information is basically denying their existence due to ignorance of the process.

[2] The $P(\mathcal{E} \mid_c \mathcal{H})$ is the probability that $\mathcal{H}$ caused $\mathcal{E}$, where "$|_c$" indicates *causal* probability, not *conditional* probability.

[3] The likelihood $P(\mathcal{E} \mid \mathcal{H})$ is the probability that $\mathcal{E}$ has or will occur given that $\mathcal{H}$ has occurred. It is simply a conditional probability.

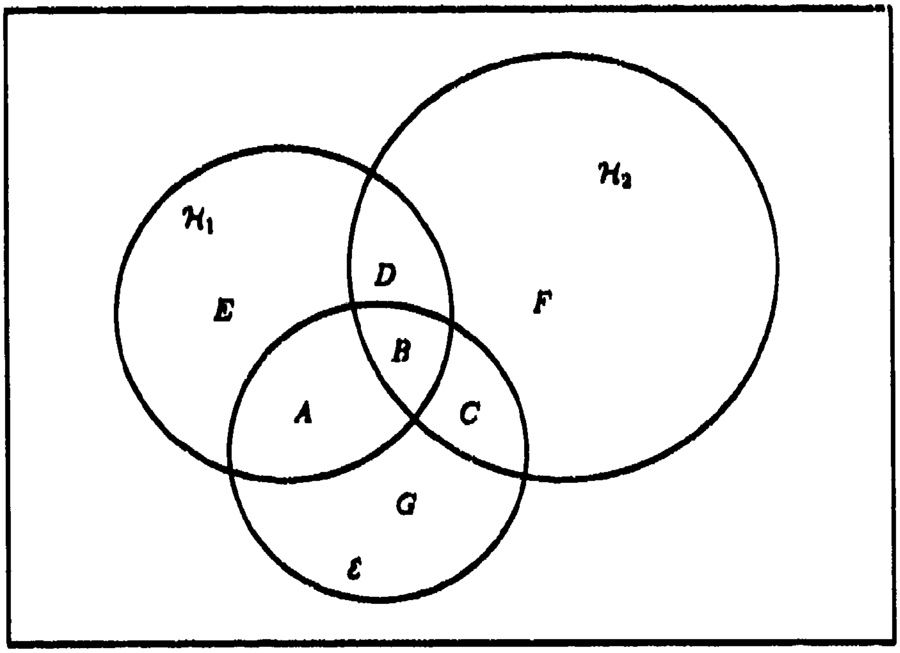


Figure 15. Venn diagram for example 4.2.

$$P(\mathcal{E} \mid \mathcal{H}_1) = \frac{A+B}{A+B+D+E}$$
$$P(\mathcal{E} \mid \mathcal{H}_2) = \frac{C+B}{C+B+D+F}$$

However, the causal probabilities, estimated by an expert, are something other than the conditionals:

$$\frac{A}{A+B+D+E} \leq P(\mathcal{E} \mid_c \mathcal{H}_1) \leq \frac{A+B}{A+B+D+E}$$
$$\frac{C}{C+B+D+F} \leq P(\mathcal{E} \mid_c \mathcal{H}_2) \leq \frac{C+B}{C+B+D+F}$$

The problem arises from the fact that $P(\mathcal{E} \mid_c \mathcal{H}_1)$ and $P(\mathcal{E} \mid_c \mathcal{H}_2)$ are known, while the conditional probabilities (the regions in the Venn diagram) are unknown. Without some further information on regions A and C there is no way of generating the conditional probabilities.

A possible simplifying assumption is that regions B and D are equal to the empty set. This is the basic assumption of disjoint hypotheses. As long as $\mathcal{E}$ can only be caused by the hypotheses under consideration (this implies that G is also equal to the empty set), then the conditional probabilities and the causal probabilities are equal:

$$P(\mathcal{E} \mid \mathcal{H}_1) = P(\mathcal{E} \mid_c \mathcal{H}_1) = \frac{A}{A+E}$$
$$P(\mathcal{E} \mid \mathcal{H}_2) = P(\mathcal{E} \mid_c \mathcal{H}_2) = \frac{C}{C+F}$$

This simplifying assumption has some appeal. However, adding spurious evidence to the model is actually saying that there is a hypothesis, that is independent of the set of hypotheses of interest, which could cause E. In this case the problem becomes difficult to visualize with a simple Venn diagram.

In most situations there is some probability that an evidence can occur without being caused by one of the hypotheses under consideration. This probability of independent occurrence increases the conditional probabilities and does not affect the causal probabilities. The problem of dealing with causal estimates when using a frequency inference mechanism is important and not trivial.

4.3 Independence

In a Bayesian diagnostic system when an evidence and hypothesis are probabilistically independent it is generally true that they are causally dependent. The only way $\mathcal{E}$ and a member of the disjoint hypotheses set, $\mathcal{H}_s$, can be both probabilistically and causally independent is if every member of $\mathcal{H}_s$ is causally independent of $\mathcal{E}$. In such a situation $\mathcal{E}$ is of no value as evidence since it does not differentiate between the possible hypotheses.

Another way that a member of $\mathcal{H}_s$ and $\mathcal{E}$ can be probabilistically independent is if the $P(\mathcal{E} \mid \mathcal{H})$ happens to equal the $P(\mathcal{E})$ as given from the law of total probability:

$$P(\mathcal{E}) = \sum_{i=1}^{n} P(\mathcal{E} \mid \mathcal{H}_i)P(\mathcal{H}_i)$$

Where $\mathcal{H}_i$ are members of $\mathcal{H}_s$.

In such a situation, probabilistic independence is an interesting property but is of little actual significance. The interesting property is that if $\mathcal{H}$ and $\mathcal{E}$ are probabilistically independent then observing $\mathcal{E}$ does not alter the probability that $\mathcal{H}$ has occurred. It is of no real significance because, as Morlan points out, "the important information in a situation includes the relative probabilities of the possible states of the world and the relative losses. Decisions between two actions can be made based on a computation of the relative expected losses." (13:3) So, in a decision situation the relative probabilities of the hypotheses are important not their actual probabilities.

4.4 Secondary Uncertainty

The term *spurious* came up quite often, mostly in conjunction with arguments about independence. Spurious is a vague term which can encompass a broad range of phenomena. According to the dictionary spurious means false; however, in some situations it is applied where it refers to various ideas: unknown origin, unimportant origin, and false indication. The first two ideas, unknown and unimportant origin, apply when the event actually occurred but why it occurred is either unknown or unimportant. The last idea, false indication, applies when the indicated event hasn't actually occurred. False indication can happen in two ways: first, an indication of an event that hasn't actually occurred, and second, lack of an indication when an event has occurred. This false indication is a reporting problem, and its probability assessment is the uncertainty in the evidence or *secondary uncertainty*.

False indication can be characterized as the observance of some event when the event hasn't occurred. The probability of such an observance is stated as the $P(\mathcal{E}_o \mid not\mathcal{E})$ and $P(not\mathcal{E}_o \mid \mathcal{E})$, or $P(\mathcal{E} \mid not\mathcal{E}_o)$ and $P(not\mathcal{E} \mid \mathcal{E}_o)$, where $\mathcal{E}_o$ is the observed state and $\mathcal{E}$ is the actual state. The confidence (uncertainty) in the obse ved data is the probability that the observed state is the same as the actual state: $P(\mathcal{E}_o \mid \mathcal{E})$ and $P(not\mathcal{E}_o \mid not\mathcal{E})$, or $P(\mathcal{E} \mid \mathcal{E}_o)$ and $P(not\mathcal{E} \mid not\mathcal{E}_o)$. What we see and can gather is the observed state, $\mathcal{E}_o$ and our decisions are based upon observed information. Whether the assessment of the uncertainty is described as $P(\mathcal{E} \mid \mathcal{E}_o)$ or $P(\mathcal{E}_o \mid \mathcal{E})$, the goal is an assessment of the posterior probability $P(\mathcal{H}_i \mid \mathcal{E}_o)$, where $\mathcal{H}_i$ are the hypotheses under consideration.

There are at least two possible methods for incorporating secondary uncertainty in a probabilistic inference problem: 1) Jeffrey's rule, which uses the assessment of $P(\mathcal{E} \mid \mathcal{E}_o)$, and incorporates this information at the posterior level; and 2) an alternate method, which involves the assessments of $P(\mathcal{E}_o \mid \mathcal{E})$ and $P(not\mathcal{E}_o \mid not\mathcal{E})$, and incorporates this information at the likelihood level. Both methods perform the same type of mathematical transformation; however, there are potentially great differences in the computational complexity of a given problem.

4.4.1 Jeffrey's rule Jeffrey's rule is a method for generating the probabilities $P(\mathcal{H}_i \mid \mathcal{E}_o)$ when the uncertainty assessment yields $P(\mathcal{E} \mid \mathcal{E}_o)$. It is an application of the law of total probability conditioned with the confidence in the report, $P(\mathcal{E} \mid \mathcal{E}_o)$.

$$P(\mathcal{H} \mid \mathcal{E}_o) = \sum_{i=1}^{n} P(\mathcal{H} \mid \mathcal{E}_i) P(\mathcal{E}_i \mid \mathcal{E}_o) \tag{2}$$

Equation 2 is a weighted average of the possible probability states $P(\mathcal{H} \mid \mathcal{E})$ and $P(\mathcal{H} \mid not\mathcal{E})$, weighted on the uncertainty in the evidence state $\mathcal{E}$. Graphically, Jeffrey's rule is simply a linear interpolation between the possible pure evidence states weighted with the uncertainty in the evidence states.

Example 4.3 Jeffrey's Rule with 1 Evidence

Given the following probabilities,

$$\begin{aligned} P(\mathcal{H} \mid \mathcal{E}) &= .2 \\ P(\mathcal{H} \mid not\mathcal{E}) &= .8 \end{aligned}$$

With the confidence in $\mathcal{E}$ at 80%,

$$P(\mathcal{E} \mid \mathcal{E}_o) = .8$$

then the posterior probability of $\mathcal{H}$ given the observed evidence $\mathcal{E}_o$ becomes,

$$\begin{aligned} P(\mathcal{H} \mid \mathcal{E}_o) &= P(\mathcal{H} \mid \mathcal{E}) P(\mathcal{E} \mid \mathcal{E}_o) + P(\mathcal{H} \mid not\mathcal{E}) P(not\mathcal{E} \mid \mathcal{E}_o) \\ &= (.2)(.8) + (.8)(.2) \\ &= .32 \end{aligned}$$

This example is graphically depicted in figure 16.

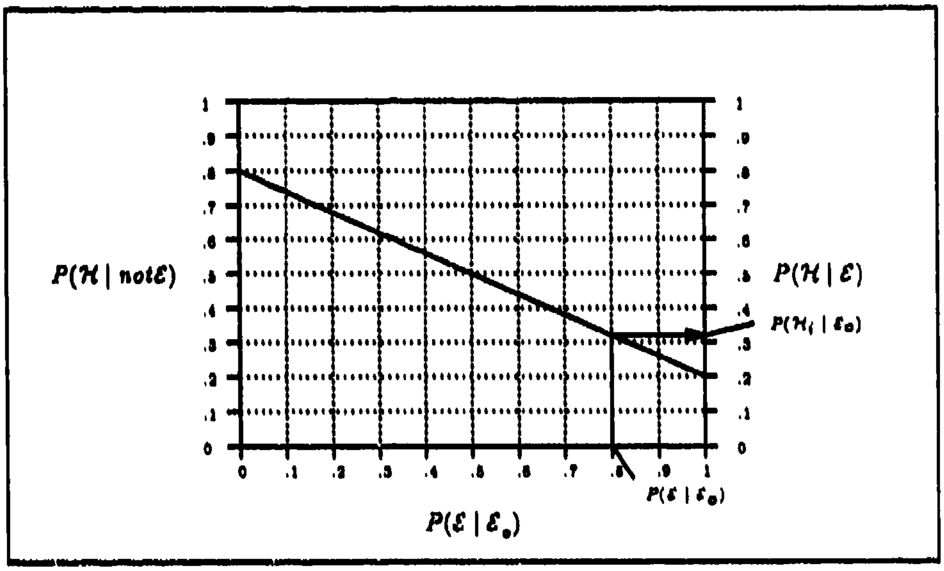


Figure 16. Graphic depiction of Jeffrey's rule.

If there are several evidence states, the result is the intersection of the interpolations conditioned on the evidences in question.

Example 4.4 Jeffrey's Rule with 2 Evidences

In this example there are two evidences, each with some degree of uncertainty. Given the evidence $\mathcal{E}_1$ is the evidence from example 4.3, and the following probabilities,

$$\begin{aligned} P(\mathcal{H} \mid \mathcal{E}_1, \mathcal{E}_2) &= .9 \\ P(\mathcal{H} \mid \mathcal{E}_1, not\mathcal{E}_2) &= .2 \\ P(\mathcal{H} \mid not\mathcal{E}_1, \mathcal{E}_2) &= .4 \\ P(\mathcal{H} \mid not\mathcal{E}_1, not\mathcal{E}_2) &= .8 \end{aligned}$$

With the confidence in $\mathcal{E}_1$ at 80%, and in $\mathcal{E}_2$ at 30%,

$$\begin{aligned} P(\mathcal{E}_1 \mid \mathcal{E}_o) &= .8 \\ P(\mathcal{E}_2 \mid \mathcal{E}_o) &= .3 \end{aligned}$$

With above information the assessment of the posterior probability of $\mathcal{H}$ given the observed state of the evidence, $\mathcal{E}_o$, becomes,

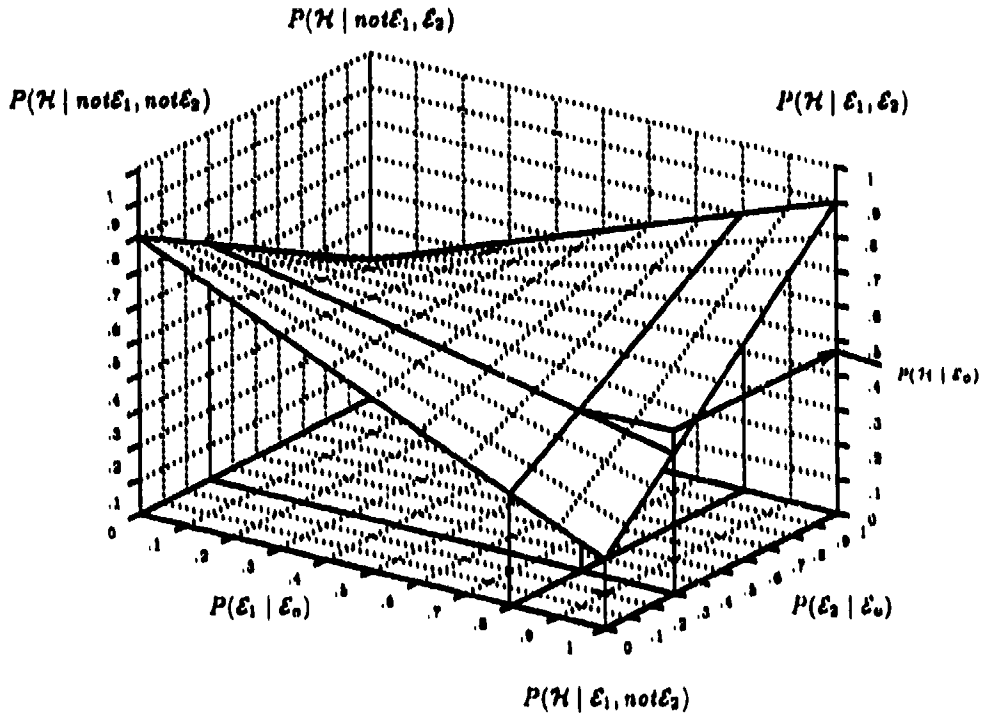


Figure 17. Graphic depiction of Jeffrey's rule with uncertainty in two evidences.

$$\begin{aligned}
P(\mathcal{H} \mid \mathcal{E}_o) &= P(\mathcal{H} \mid \mathcal{E}_1, \mathcal{E}_2)P(\mathcal{E}_1 \mid \mathcal{E}_o)P(\mathcal{E}_2 \mid \mathcal{E}_o) \\
&\quad + \ P(\mathcal{H} \mid \mathcal{E}_1, not\mathcal{E}_2)P(\mathcal{E}_1 \mid \mathcal{E}_o)P(not\mathcal{E}_2 \mid \mathcal{E}_o) \\
&\quad + \ P(\mathcal{H} \mid not\mathcal{E}_1, \mathcal{E}_2)P(not\mathcal{E}_1 \mid \mathcal{E}_o)P(\mathcal{E}_2 \mid \mathcal{E}_o) \\
&\quad + \ P(\mathcal{H} \mid not\mathcal{E}_1, not\mathcal{E}_2)P(not\mathcal{E}_1 \mid \mathcal{E}_o)P(not\mathcal{E}_2 \mid \mathcal{E}_o) \\
&= (.9)(.8)(.3) + (.2)(.8)(.7) + (.4)(.2)(.3) + (.8)(.2)(.7) \\
&= .464
\end{aligned}$$

This example is graphically depicted in figure 17. The value of at each corner of the graph is the "pure" posterior, conditioned on a combination of perfect information about the evidences.

4.4.2 Spurious Evidence Another method for incorporating uncertainty in the evidence is to use the assessment of the $P(\mathcal{E}_o \mid \mathcal{E})$ to generate new likelihood probabilities $P(\mathcal{E}_o \mid \mathcal{H})$. Like Jeffrey's rule, it is an application of the law of total probability. It involves conditioning on the confidence in the reporting system, $P(\mathcal{E}_o \mid \mathcal{E})$.

$$P(\mathcal{E}_o) = P(\mathcal{E}_o \mid \mathcal{E})P(\mathcal{E}) + P(\mathcal{E}_o \mid not\mathcal{E})P(not\mathcal{E}) \tag{3}$$

The effect of incorporating the uncertainty in the evidence in the likelihoods is to reduce the range of values that the likelihood $P(\mathcal{E}_o \mid \mathcal{H})$ can take.

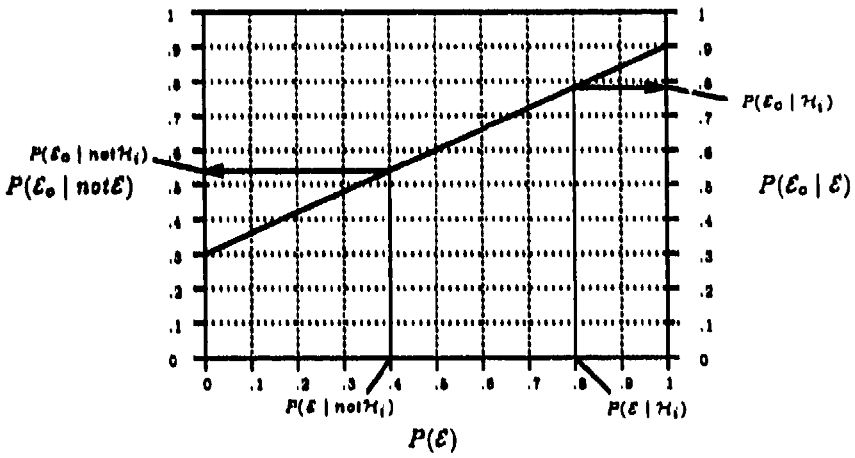


Figure 18. The effect of spurious evidence on likelihoods.

Example 4.5 Unsymeteric Spurious Evidence

Given that the probabilities for observing $\mathcal{E}$ are,

$$\begin{array}{lcl} P(\mathcal{E}_o \mid \mathcal{E}) & = & .9 \\ P(\mathcal{E}_o \mid not\mathcal{E}) & = & .3 \end{array}$$

and If the likelihoods are,

$$\begin{array}{lcl} P(\mathcal{E} \mid \mathcal{H}_i) & = & .8 \\ P(\mathcal{E} \mid not\mathcal{H}_i) & = & .4 \end{array}$$

then by applying equation 3 the likelihoods for the observed evidence become,

$$\begin{array}{lcl} P(\mathcal{E}_o \mid \mathcal{H}_i) & = & .78 \\ P(\mathcal{E}_o \mid not\mathcal{H}_i) & = & .54 \end{array}$$

In this example, the range of the observed likelihoods, $P(\mathcal{E}_o \mid \mathcal{H}_i)$, is between .9 and .3. Figure 18 depicts the range of values that $P(\mathcal{E}_o \mid \mathcal{H})$ can take. The $P(\mathcal{E}_o \mid \mathcal{H})$ will lie somewhere along the line depicted in figure 18.

Equation 3, when expressed in terms of the probability of seeing the correct state of the evidence, changes to,

$$P(\mathcal{E}_o) = P(\mathcal{E})\,[P(\mathcal{E}_o \mid \mathcal{E}) + P(not\mathcal{E}_o \mid not\mathcal{E}) - 1] + 1 - P(not\mathcal{E}_o \mid not\mathcal{E}) \qquad (4)$$

If the spurious nature of the sensor is independent of the sensor's actual state then $P(\mathcal{E}_o \mid \mathcal{E}) = P(not\mathcal{E}_o \mid not\mathcal{E})$, and is *symmetric*. When the probability of error is symmetric, equation 4 reduces to,

$$P(\mathcal{E}_o) = P(\mathcal{E})[2P(\mathcal{E}_o \mid \mathcal{E}) - 1] + 1 - P(\mathcal{E}_o \mid \mathcal{E}) \qquad (5)$$

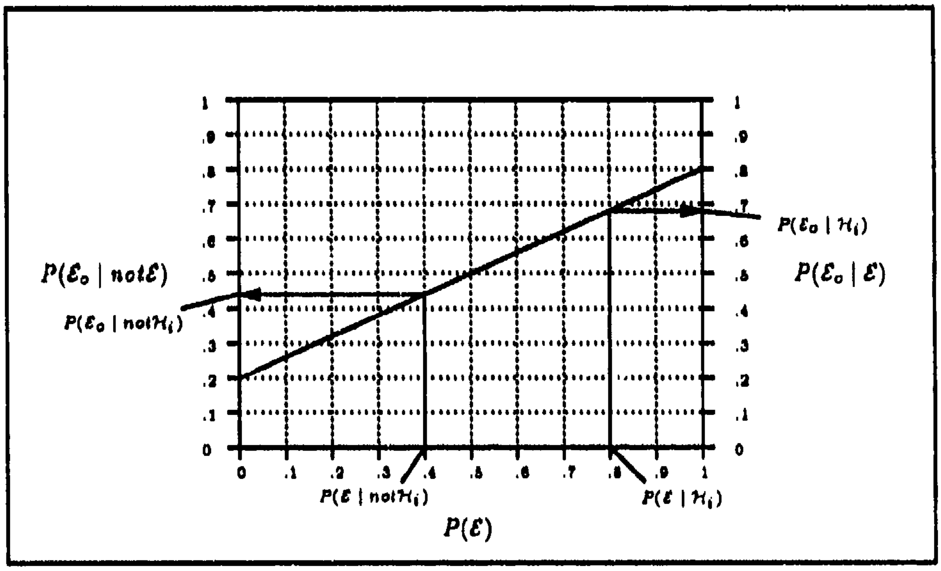


Figure 19. The effect of symmetric spurious evidence on likelihoods.

Example 4.6 Symmetric Spurious Evidence

Given the probabilities from example 4.5 except that the spurious probabilities are symmetric, and equal to,

$$P(\mathcal{E}_o \mid \mathcal{E}) = P(not\mathcal{E}_o \mid not\mathcal{E}) = .8$$

then the likelihoods for the observed evidence become,

$$\begin{aligned} P(\mathcal{E}_o \mid \mathcal{H}_i) &= .68 \\ P(\mathcal{E}_o \mid not\mathcal{H}_i) &= .44 \end{aligned}$$

This example is graphically depicted in figure 19.

The worst situation would be to have the $P(\mathcal{E}_o \mid \mathcal{E}) = P(not\mathcal{E}_o \mid not\mathcal{E}) = 0.5$. This, in effect, is a random sensor with a uniform distribution over the possible states. In such a case the $P(\mathcal{E}_o \mid \mathcal{H}) = 0.5$, regardless of the $P(\mathcal{E} \mid \mathcal{H})$; this case is depicted in figure 20.

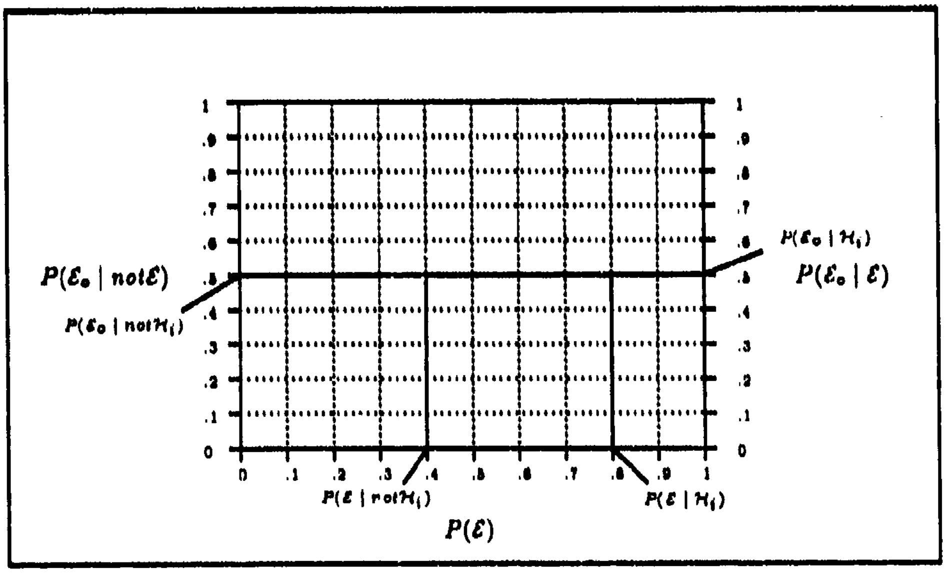


Figure 20. The effect of symmetric spurious evidence when $P(\mathcal{E}_o \mid \mathcal{E}) = 0.5$.

4.4.3 Comparisons Both methods estimate the $P(\mathcal{H} \mid \mathcal{E}_o)$. However, they go about it in different ways. With Jeffrey's rule the conditioning takes place after applying Bayes' theorem using the $P(\mathcal{E} \mid \mathcal{E}_o)$ and $P(not\mathcal{E} \mid \mathcal{E}_o)$, and the posterior probabilities $P(\mathcal{H} \mid \mathcal{E})$ and $P(\mathcal{H} \mid not\mathcal{E})$:

$$P(\mathcal{H} \mid \mathcal{E}_o) = \sum_{i=1}^{n} P(\mathcal{H} \mid \mathcal{E}_i)P(\mathcal{E}_i \mid \mathcal{E}_o) + P(\mathcal{H} \mid not\mathcal{E}_i)P(not\mathcal{E}_i \mid \mathcal{E}_o)$$

In the other method the conditioning takes place before applying Bayes' theorem using the $P(\mathcal{E}_o \mid \mathcal{E})$ and $P(\mathcal{E}_o \mid not\mathcal{E})$, and the likelihoods $P(\mathcal{E} \mid \mathcal{H}_i)$ and $P(not\mathcal{E} \mid \mathcal{H}_i)$:

$$P(\mathcal{E}_o \mid \mathcal{H}) = P(\mathcal{E}_o \mid \mathcal{E})P(\mathcal{E} \mid \mathcal{H}) + P(\mathcal{E}_o \mid not\mathcal{E})P(not\mathcal{E} \mid \mathcal{H})$$

Since both methods are concerned with estimating the same posterior probability $P(\mathcal{H} \mid \mathcal{E}_o)$, (by conjecture) they should arrive at the same conclusion given compatible initial conditions[4]. However, because the methods include the evidence uncertainty information at different levels in the Bayesian inference process the number of assessments and calculations required is different.

Examples 4.3 and 4.4 show the effect of applying Jeffrey's rule. The posterior probabilities for every combination of the set of evidences, $\mathcal{E}_s$, are needed to make an assessment of the $P(\mathcal{H} \mid \mathcal{E}_o)$. The problem grows exponentially with the number of evidences under consideration, and linearly with the number of hypotheses. Example 4.7 is a comparison of the assessments and calculations required by Jeffrey's rule and the alternate method.

[4] Determining compatible initial conditions is a formidable task in itself. Such a determination involves the use of the marginal distribution on either $\mathcal{E}_o$ or $\mathcal{E}$.

Example 4.7 **Method Comparisons**

Starting with an inference problem that has the prior probabilities $P(\mathcal{H}_i)$ and the likelihoods $P(\mathcal{E}_j \mid \mathcal{H}_i)$ already specified, how does Jeffrey's rule compare to the alternate method in assessing the posterior probabilities $P(\mathcal{H}_i \mid \mathcal{E}_o)$?

Initial conditions:		$\mathcal{H}_i$; i = 1 to a $\mathcal{E}_j$; j = 1 to n
Jeffrey's rule	alternate method	Conditional probability assessments
n		assessments of $P(\mathcal{E}_j \mid \mathcal{E}_o)$
	n or $2 \cdot n$	assessments of $P(\mathcal{E}_o \mid \mathcal{E}_j)$ if uncertainty is symmetric. assessments if not symmetric, n assessments of $P(\mathcal{E}_o \mid not\mathcal{E}_j)$, and n assessments of $P(\mathcal{E}_o \mid \mathcal{E}_j)$
		Posterior probability calculations
$a \cdot 2^n$		calculations of the posterior probabilities $P(\mathcal{H}_i \mid not\mathcal{E}_j)$, and $P(\mathcal{H}_i \mid \mathcal{E}_j)$
	a	calculations of the posteriors $P(\mathcal{H}_i \mid \mathcal{E}_o)$
		Conditional probability transformations
a		calculations of the posterior transformation $P(\mathcal{H}_i \mid \mathcal{E}_o) = \sum_{j=1}^{n} P(\mathcal{H}_i \mid \mathcal{E}_j)P(\mathcal{E}_j \mid \mathcal{E}_o) + P(\mathcal{H}_i \mid not\mathcal{E}_j)P(not\mathcal{E}_j \mid \mathcal{E}_o)$
	$a \cdot n$	calculations of the likelihood transformation $P(\mathcal{E}_{oj} \mid \mathcal{H}_i) = P(\mathcal{E}_{oj} \mid \mathcal{E}_j)P(\mathcal{E}_j \mid \mathcal{H}_i) + P(\mathcal{E}_{oj} \mid not\mathcal{E}_j)P(not\mathcal{E}_j \mid \mathcal{H}_i)$

Table 1. Comparison of Jeffrey's rule with the alternate method.

Table 1 shows that using Jeffrey's rule for incorporating secondary uncertainty grows exponentially in the number of posterior probabilities required. If secondary uncertainty can be incorporated in the likelihood information, then the inference problem grows linearly with the evidences in calculating the likelihoods, and decreases the number of posterior probabilities to one for each hypothesis under consideration.

4.5 Summary

This chapter covered 1) a possible method for dealing with missing information to continue with the inference process, 2) the difference between causal implication and conditional probabilities and how it affects likelihood generation, 3) the meaning of causal independence and probabilistic independence in the context of a Bayesian inference model, and 4) the effect of including secondary uncertainty in a Bayesian inference model.

The foregoing discussion only begins to show the ramifications of each area. With missing information, the assumption of probabilistic independence has intuitive appeal and is easily implemented. The difference between causal implication and conditional probability can have significant effects in the generation of the initial probabilities used in probabilistic inference systems. The difference in the meanings of causal and probabilistic independence with respect to Bayesian inference systems is an important distinction. Causal independence has an intuitive meaning where the independent parties have no connection. Probabilistic independence is an interesting phenomena but is of little use when interested in causality. Lastly, where secondary uncertainty is included has a great effect on the computational complexity of probabilistic systems.

V. Conclusions and Recommendations

> ***Isn't it sad how some people's grip on their lives is so precarious that they'll embrace any preposterous delusion rather than face an occasional bleak truth?***
> Calvin and Hobbes (comic strip) –Bill Waterson

This thesis was concerned with the representation and interpretation of uncertainty. Chapter II presented a view of the present state of uncertainty reasoning from an operational and a philosophical perspective. Chapter III examined *probability ratio graphs* as a representation of the probability model. Chapter IV discussed several topics of concern in uncertainty reasoning: 1) missing information and likelihood generation, 2) the meaning of independence from a Bayesian context, and 3) including secondary uncertainty and spurious indications. This research was motivated by the current interest in incorporating uncertainty reasoning into expert systems. The following are the conclusions resulting from this research.

5.1 Conclusions

Non-uniqueness of Uncertainty Representation. The representation of uncertain knowledge can take many forms, none of which can be said to be an exclusively "correct" method. While operationally any method is applicable if used within its valid range, for theoretical acceptance they must be consistent and logically coherent. Probability theory exhibits both characteristics and has a sound theoretical foundation while the other methods do not. This does not preclude research into the other methods; however, it does show that they have limited applicability as they now stand.

Probability Ratio Graphs as a Representation of Probability. Probability ratio graphs are an appealing method for representing the probability model. It is easily rendered in a graphical format where conditional and marginal probabilities are readily evident and are intuitively meaningful. As Morlan describes them, probability ratio graphs cannot represent probabilistic independence. The proposed method for adding an independence capability enhances probability ratio graph's ability to represent the probability model, and therefore increases their versatility for representing complex decision problems.

Missing Information and Likelihood Generation. The method of assuming probabilistic independence for generating missing likelihood information offers several benefits: 1) it has an intuitive appeal since it seems beneficial to include all the available relevant information in a given decision problem, 2) it appears detrimental to the decision problem to exclude hypotheses through ignorance, 3) Morlan's assumption of probabilistic independence seems to balance the need for including available information while not excluding hypotheses through ignorance, and 4) it obviates the need to generate any information since applying Bayes' theorem to the partitioned dependent hypotheses and recombining produces equivalent results.

The distinction between causality and conditional probability has ominous implications when experts make subjective estimates of conditional probabilities. Probability is a representation of relative frequencies, not causality. This is important because in many situations it is easier to think about and estimate causality than it is to estimate conditional probability. However, since Bayesian inference is probabilistic, it treats the information as relative frequencies, not causality. If experts' subjective causal assessments are to be used they must be transformed from a causal representation into a frequency representation.

Causal Versus Probabilistic Independence. In a Bayesian inference context, probabilistic independence is a coincidence of little significance to the decision problem. Causal independence, on the other hand, is of great significance since it is a strong implication when either an evidence state is present or absent.

Including Secondary Uncertainty. Secondary uncertainty is a product of the reporting system. It is termed secondary uncertainty because it is a characteristic of the evidence which is used to reason about the state of the world. It is thus secondary because it is not part of the primary uncertainty problem. Chapter IV presents Jeffrey's rule and an alternate method for including secondary uncertainty in the reasoning process. Jeffrey's rule includes the secondary uncertainty at the posterior level, whereas the alternative is to include the secondary uncertainty at the likelihood level. In a relatively simple example, it was shown that the alternate method, as compared to Jeffrey's rule, produced significant reductions in the computational complexity of including secondary uncertainty[1].

5.2 Areas for Future Research

Probability Ratio Graph Implementation. The probability ratio graph concept is maturing. With the conceptual addition of independence it has become capable of representing the full probability model. The developed program is capable of handling the restricted probability ratio graph model; however, it is still a fledgling research tool which is not computationally efficient. A next step would be to streamline and simplify the present coding, and implement the full probability ratio graph concept including the independence capability and the utility section.

Analysis of the Missing Information Assumption and Likelihood Generation. Applying the assumption of probabilistic independence to missing information is intuitively appealing and easily implemented from a calculation standpoint[2]. However, ease of implementation and being intuitively pleasing are not vigorous tests for validity. Other methods should be developed and then compared with this assumption to establish a basis for validity.

The generation of likelihood information is related to the missing information problem. As presented in this thesis, generating likelihood information from causal information necessitates an analysis of the underlying decision problem: specifically, assessing the possibility of the independent occurrence of evidence. Further research involving the likelihood generation scheme presented in chapter IV would provide a greater understanding of the

[1] By conjecture, given compatible initial conditions both methods should produce equivalent results since they are estimating the same posterior probability. However, determining those compatible initial conditions is itself a difficult undertaking.

[2] Under the assumption of probabilistic independence no missing information needs to be generated to continue the inference process.

subjective assessment processes and may explain the irrationality of some expert assessments when they occur.

Including Secondary Uncertainty. Including secondary uncertainty involves transforming a conditional probability involving the evidence, $\mathcal{E}$, into a conditional probability involving the observed evidence, $\mathcal{E}_o$. As shown in Chapter IV, where in the inference cycle this transformation occurs has a great influence in the computational complexity of the overall problem. Chapter IV only introduces the concept of including secondary uncertainty in the likelihood information; the preliminary indications are that this method greatly reduces the calculations required as compared with Jeffrey's rule. It was conjectured that Jeffrey's rule and the alternate method should produce equivalent estimates. This conjecture must now be substantiated or disproved whichever the case may be.

5.3 Summary

Uncertainty reasoning has proven to be fertile ground for creative minds. There is always room for further interpretation of uncertainty and how to objectively deal with it. This thesis dealt with several topics involving the representation and interpretation of various facets of uncertainty. Further research into the topics introduced in this thesis may prove fruitful for using a probabilistic model of uncertainty. However, probability is only one method for representing uncertainty; other methods may prove superior if they can solve the problems of consistency and logical coherence.

Appendix A. *Probability Ratio Graph Software*

This appendix contains a discussion of the probability ratio graph software created as part of this thesis effort. Turbo Pascal is the supporting language due to its 1) availability, 2) graphics capability, and 3) programming support. Another important point is that my advisor was familiar with Turbo Pascal, so help was available. The purpose of this software is to provide a beginning for encoding probability ratio graphs. The first section will cover the system structure, and the second section will cover program use.

A.1 Structure

Most of the program is support for information management functions and user interfacing. The software is separated into five units, each containing related functions, and one driver program.

Unit	Purpose
Lst	Contains procedures for list manipulations.
Ports	Contains procedures for screen formatting, interfacing, and control.
Parts	Contains program specific procedures for generating, loading, saving, and managing node and arc information.
Show	Contains program specific procedures for graphics display and probability ratio graph manipulations.
Menu	Contains procedures for program control and user interfacing.
Note: The driver program is "test.pas." Its only purpose is to initiate the program and hand over control to the menu unit.	

Table 2. Program units and their purpose.

The program uses record variables to represent the two data structures involved in probability ratio graphs, arcs and nodes (the term "node" is synonymous with "vertex" for purposes of discussion.). The operations are carried out by procedures (which, incidentally, do not necessarily go by the same names as described in Chapter III; the differences will be discussed when the need arises.). This discussion covers the general program concept and function. The individual procedures are listed in appendix B with specific information relating to there use and purpose.

A.1.1 Lst Unit The Lst unit contains procedures for manipulating list structures. Turbo Pascal does not readily support list structures[1]; however, there are some data structures in this program which are best represented with lists. The procedures in this unit provide a means for using Turbo Pascal's string variables as lists.

There are two types of list structures: 1) lists where the members are separated by commas, and 2) lists where the members are separated by blank spaces. The two types came about because the first lists were separated by commas and the majority of the program was written toward that end; the second list type was a late-comer, it grew from a need for displaying text in the different screen viewports (word wrapping in different port widths, and different screen modes–EGA, VGA).

This unit is independent of the specific program purpose; that is, it can be used in other applications where data structures are represented as lists.

A.1.2 Ports Unit The Ports unit contains procedures for screen formatting and interfacing. An important part of the program's function is the presentation of the information contained in probability ratio graphs, and to provide for easy user interfacing. Like the Lst unit, this unit is not program specific. The procedures can be used with other applications.

Some procedures in this unit provide the capability for separating the screen into four areas (viewports), each providing a means for presenting different types of information[2]. There are procedures that change the active viewport and perform the administrative overhead (maintaining last cursor position, changing color settings to the current viewport colors, etc...).

[1] LISP is an example of a language that relies on lists for the data structures.

[2] There are no procedures that specifically control the type of information displayed in the different viewports. Any information can be displayed in any port; where the information is displayed is up to the programmer.

Several procedures control the input and output of typed information. These specific procedures are necessary because Turbo Pascal does not have the capability for displaying text and graphics windows simultaneously. For presentation continuity the screen is in the graphics mode at all times. To provide for typed interaction, special procedures were needed.

A.1.3 Parts Unit The Parts unit contains the procedures that control the creation, deletion, identification, retrieving, and saving of the specific data types used by the program. It is program specific (meaning that the procedures relate to this specific application only). This unit specifies the record data types for the nodes and arcs, and declares global variables used for system data control.

A.1.4 Show Unit The Show unit contains the procedures that create the graphic representations and the procedures that implement the manipulation functions of probability ratio graphs. The majority of this unit is devoted to the graphics support function. The manipulation functions are encoded in three main procedures: 1) triangulate (triangulation), 2) explode (aggregation), and 3) cluster (disaggregation). This unit is program specific.

> **Graphics support.** The graphics support provides for two ways of viewing the system: 1) The nodes can be viewed individually as a pie chart where the pie sections represent embedded nodes. 2) The nodes can be viewed as a connected graph of miniaturized pie charts. The span of the arcs in each pie chart reflects the relative probability of the nodes conditioned on the parent node. The connected graph representation shows the parent node as a miniaturized pie chart in the lower right corner of the graphics viewport, and the connected graph shows the arc connections between the embedded nodes. The connected graph representation has the advantage of presenting more information at one time, while the pie chart has more resolution.
>
> **Triangulation.** The *triangulation* procedure is an extended form of the triangulation function described in Chapter III. It can triangulate between any two nodes in a connected graph regardless of the current connection path. It does this, in essence, by automatically performing repeated triangulations along the connection path until the two goal nodes are connected. If the two specified nodes are currently connected, then the triangulation function either reverses

the arc direction or does nothing. It reverses the arc direction if the original arc direction is opposite the specified direction. If the original and specified orientations are the same, then triangulation has no effect.

Aggregation. The *explode* procedure accomplishes the aggregation function. This procedure is "hot-wired" because instead of using the arc ratios directly to create the joint representation, it uses the arc ratios to calculate the conditional probabilities, and these probabilities to reconnect the arcs, forming the joint representat' ·1.

Disaggregation. The *cluster* procedure accomplishes the disaggregation function. This procedure uses the arc ratios for forming the conditional representation. However, it can only collect two nodes at a time and in doing so it creates a new parent node. That is, there is no direct way to add an existing node to an embedded graph. To include a node in an embedded graph one must, 1) cluster the object node, N_o, with the parent node containing the target embedded graph, N_{p1} (this creates a new parent node, N_{p2}, containing N_o and N_{p1} as a two element embedded graph.); then 2) explode N_{p1} which connects N_o to the original target embedded graph.

A.1.5 Menu Unit The Menu unit contains the procedures for program control and user interfacing. This unit is program specific in that the menu procedures are geared toward the procedures in this program. However, the overall menu format works with the screen structure defined by the Ports unit, and can be used for any application. If used for a different application, or even an extended version of the current program, the specific menu procedures would have to be rewritten or upgraded to reflect the change.

A.1.6 Nodes and Arcs In this program the nodes are the major objects (or data structures), and therefore have many attributes. Some of the attributes are used solely for graphics support. Table 3 lists the attributes associated with nodes.

Variable	Type	Purpose
Index	String	The array element pointer which delineates where the data relating to the node resides in heap memory.
Name	String	The label supplied by the user.
Children	String	A list of the node's embedded nodes.
Parent	String	The node in which this node is embedded.
ConArc	String	A list of the arcs which are connected to the node.
Prob	Real	The node's conditional probability, conditioned on the parent node.
Color	Integer	The color of the node (used for graphics).
Xpos	Integer	The "X" position of the center of the node (used for graphics).
Ypos	Integer	The "Y" position of the center of the node (used for graphics).
Rad	Integer	The radius of the node (used for graphics).

Table 3. Node record attributes.

Like the nodes, arcs also have various attributes. However, unlike the nodes, arcs do not need any parameters specifically for graphics support. Because they connect nodes, they use the nodes' parameters for the graphics information. Table 4 lists the attributes associated with arcs.

Variable	Type	Purpose
Index	String	The array element pointer which delineates where the data relating to the arc resides in heap memory.
Head	String	The node at which the arc originates.
Tail	String	The node at which the arc terminates.
Ratio	Real	The ratio of the nodes' probabilities, $Ratio = \frac{P(Head)}{P(Tail)}$.

Table 4. Arc record attributes.

A.1.7 Global Variables This program uses many global variables for defining the state of the system. They provide a means for access to system status, and for information transfer across procedures. The important system global variables are listed in Table 5 (these global variables define the state of the system).

Variable	Type	Purpose
N	Array	This is the array for the node record pointers. The "Index" parameter for the nodes is derived from this pointer.
A	Array	This is the array for the arc record pointers. The "Index" parameter for the arcs is derived from this pointer.
Ni	Integer	An index which is incremented when a new node is created. The term "new node" as used in this context refers to a new node record. Once a record is created it is never destroyed. When a node is deleted its index is added to the inactive node list so when a new system node is created the heap memory space can be recycled. "Ni" equals the number of nodes in the active and inactive node lists.
Ai	Integer	Performs the same function for the arcs as "Ni" does for the nodes.
ActiveN	String	A list of the active nodes' indices. An active node is any node that represented in the system.
InActiveN	String	A list of the nodes that have been deleted. Once deleted the node's record space becomes available for reuse. Therefore, the indices of and the memory for the deleted nodes are recycled.
ActiveA	String	Performs the same function for the arcs as "ActiveN" does for the nodes.
InActiveA	String	Performs the same function for the arcs as "InActiveN" does for the nodes.

Table 5. System defining global variables.

There are many more global variables which are used as temporary information storage for information transfer across procedure boundaries. The specific purpose of these additional variables is described in appendix B.

A.1.8 Data files The data files contain just enough information to reconstruct the system state when it was saved[3]. The information needed to reconstruct the system state involves 1) system global variables, 2) node parameters, and 3) arc parameters.

1. System global variables

   - ActiveN

[3] A data file could be created without saving a previously created system, however, this may cause syntax problems that would cause the program to crash while loading the data file.

- ActiveA
- InActiveN
- Ni

2. Node parameters
   - Index[4]
   - Name
   - Parent
3. Arc parameters
   - Head
   - Tail
   - Ratio

The structure of the data files follows this order[5],

1. ActiveN
2. Node parameters (Name, Parent)
3. ActiveA
4. Arc parameters (Head, Tail, Ratio)
5. InActiveN[6]
6. Ni

A.2 Program Use

To run this program you will need Turbo Pascal, version 4.0 or later, and at least the following files[7] : 1) Test.pas, 2) Lst.pas, 3) Ports.pas, 4) Parts.pas, 5) Show.pas, and 6) Menu.pas. There are also data files associated with this program that are ready to be loaded for an example of how the system works[8]: 1) SCOTT, and 2) ERIC.

To initiate the program, ensure that all of the necessary files above are present in the same directory, load "test.pas", and press <ctrl>F9 This will compile the main program and all of the necessary units[9].

[4] The index is contained in the "ActiveN" list.

[5] To see the order of a data file, simply view one—they are ASCII files.

[6] If the inactive node list is empty then this parameter is set equal to "NONE."

[7] The ".pas" files are the original code. The ".tpu" files for 2-5 are actually needed to run the program. These files will be created when the main program, "test.pas", is compiled.

[8] These data files are not necessary to run the program.

[9] If the ".tpu" files for the necessary units do not already exist, <ctrl>F9 will create them.

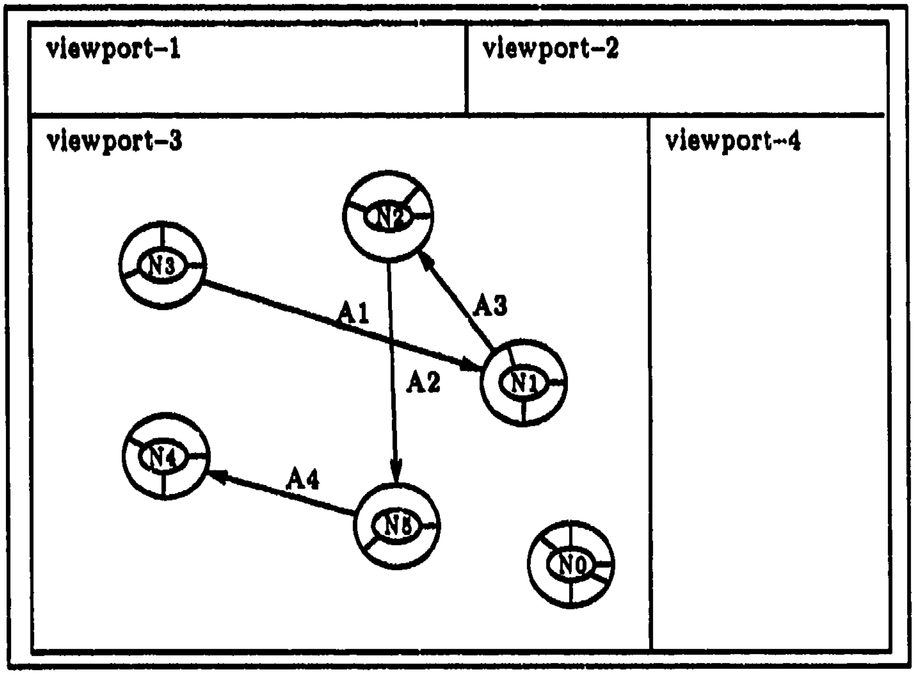


Figure 21. Screen format.

The screen will be split up into four separate areas as shown in figure 21. Viewport 2 is the menu viewport; all the main program options are displayed within viewport 2 using several different menus. The program currently has six available menus:

1. Mainmenu -- the top level menu procedure from which the other menus originate.
2. Setupmenu - submenu of mainmenu, offers options for basic system configuration.
3. Graphmenu - submenu of mainmenu, offers options for viewing the system and performing probability ratio graph manipulations.
4. Createmenu - submenu of mainmenu, offers options for creating the system objects (nodes and arcs).
5. Altermenu - submenu of setupmenu, offers options for altering object parameters.
6. Alterarcmenu - submenu of altermenu, offers options for altering arc parameters.

Because the program is in the development mode, the menus are somewhat chaotic and over-redundant. As the program develops the menus should mature into a well defined structure which separates the options in a meaningful way.

A.2.1 Menus Each menu can display up to six of the available options[10]. The listed options can be selected by either pressing the associated number or first letter of the option. There are three options that can be selected whether they are listed or not: 1) "Back", returns to the menu one level up; 2) "Main Menu", returns to the main menu; and 3) "Quit", terminates the program. If they are not listed, these three options can be selected by pressing the first letter (B, M, or Q). If they are listed, they can be selected by either the number or the first letter.

A.2.1.1 Main Menu The main menu is the top level control procedure. The available options and their purposes are defined in table 6.

Option	Purpose
1) Setup	Initiates the setup menu.
2) Create	Initiates the create menu.
3) Graph	Initiates the graph menu.
4) Initialize*	This option initializes the system parameters. All of the active nodes and arcs are added to their respective inactive lists and the active lists are set to nil.
5) Bit Image	This is a non-functioning option. It was to use Turbo Pascal's ability to save a bit image so the graphics could be printed. However, this may not be possible.
6) Quit	Terminates the program.
*Note: The initialise option does not prompt the user before resetting the system parameters. Currently, there is no way to recover lost data whether inadvertent or not.	

Table 6. Main-menu's options and their purpose.

As with all the menu procedures, the mainmenu procedure uses a while loop and a case statement to control the option selections. The the while loop provides for errors in selection (an accidental selection will not crash the program).

A.2.1.2 Setup Menu The setup menu provides the options for loading data files and saving current system information to a data file (there are other options but their

[10] An "available" option is an option that can be initiated from the current menu.

functions are somewhat limited.). The available options and their purposes are listed in table 7.

Option	Purpose
1) Load Data	Provides the capability to load a data file containing a previously constructed system configuration.
2) Save Data	Provides the capability for saving a system configuration for use at another time.
3) Initialize	Same option as listed in the main menu.
4) Alter	Brings up the alter menu.
5) Main Menu	Returns control to the main menu.
6) Quit	Terminates the program.

Table 7. Setup-menu's options and their purpose.

A.2.1.3 Graph Menu The graph menu provides the options for viewing the system and performing probability ratio graph manipulations. The available options and their purposes are listed in table 8.

Option	Purpose
1) Pie	Displays the specified node as a pie chart where the sections represent the conditional probabilities of the embedded nodes.
2) Net	Displays the specified node's miniaturized pie chart in the lower corner and the embedded nodes as a connected graph.
3) Explode	Performs the aggregation function on the specified node and displays the parent node's resulting configuration in the net representation.
4) Cluster	Performs disaggregation on the two specified nodes and displays the parent node's resulting configuration in the net representation.
5) Find path	Finds the connection path between the two specified nodes.
6) Triangulate	Performs triangulation with the two specified nodes.

Table 8. Graph–menu's options and their purpose.

A.2.1.4 Create Menu The create menu provides the options for creating system objects (nodes and arcs) and setting their parameters. The available options and their purposes are listed in table 9.

Option	Purpose
1) Node	Creates a new node and prompts the user for the node's parameters.
2)	(No option currently available)
3) Arc	Creates a new arc and prompts the user for the arc's parameters.
4)	(No option currently available)
5) Main Menu	Returns control to the main menu.
6) Quit	Terminates the program.

Table 9. Create–menu's options and their purpose.

A.2.1.5 Alter Menu The alter menu was intended to provide options for altering the objects' (nodes' or arcs') parameters. As it now exists, only arc parameters can be altered through this menu. The available options and their purposes are listed in table 10.

Option	Purpose
1) Hypothesis	(Currently, this option has no purpose.)
2) Evidence	(Currently, this option has no purpose.)
3) Arc	Initiates the alter-arc menu.
4) Cluster	Performs disaggregation on the two specified nodes and displays the parent node's resulting configuration in the net representation (same option as in the graph menu).
5) Main Menu	Returns control to the main menu.
6) Quit	Terminates the program.

Table 10. Alter–menu's options and their purpose.

A.2.1.6 AlterArc Menu The alterarc menu was intended to be one of three submenus which would provide options for altering the objects' parameters. Currently, it is the only one of these submenus which exists, and it really has no value in the current system. The available options and their purposes are listed in table 11.

Option	Purpose
1) Delete	(Currently, this option has no purpose.)
2) Reverse	Reverses the specified arc's orientation (this function can be accomplished with the triangulation option and, in any case, is of little practical use.).
3) Change Ratio	(Currently, this option is non-functioning)
4)	(No option currently available)
5) Main Menu	Returns control to the main menu.
6) Quit	Terminates the program.

Table 11. AlterArc–menu's options and their purpose.

A.2.2 Starting Out Once the program is loaded and running, the main menu should mysteriously appear in viewport 2. At this point you can either load a file or begin from scratch. To load a file choose the "Setup" option to initiate the setup menu; from the setup menu, choose "Load Data." You will be prompted for the name of the data file to load. If you enter nothing then the program will return and you can continue as if nothing happened. However, if you enter a file, then that file must exist and contain information in the correct syntax for the program to use or the program will crash (there is no error checking involved with loading data). To begin from scratch choose the "Create" option to initiate the create menu. To create system all that is needed is the "Node" option in the create menu. Using the node option will prompt for all the information needed during creation. Several system aspects are important: 1) A parent node must contain at least two nodes to be a parent (a node cannot contain a single node). 2) A node must be part of a connected graph (a node has to be connected to another node by an arc). The node option prompts for all the necessary information.

The best way to become familiar with the process is to create and run through an example. Figure 5 in Chapter III show a simple example; you may want to use this figure to check your results.

Appendix B. *Procedure and Function Reference Lookup*

This appendix contains the descriptions of the global variables, and the procedures and functions contained in table 12.[1] The global variables are arranged by units, and the procedures and functions are arranged alphabetically.

B.1 Global Variables

Ports — **Global variables**

GraphDriver Used to detect the system type.

GraphMode Used to detect the system graphics mode.

MaxX, MaxY Used for relative positioning within different viewports.

BC, FC These are the current port background and text colors respectively.

CP The current port number is stored here.

cpX1, cpY1 Current cursor position for port 1.

cpX4, cpY4 Current cursor position for port 4.

X2, X3, X4 Horizontal scaling variables for separating the screen into the four viewports.

Y2, Y3 Vertical scaling variables for separating the screen into the four viewports.

xAsp, yAsp Horizontal and vertical screen resolutions.

AspectRatio Screen aspectratio used for graphics.

Menu — **Global variables**

Quit Used for menu control between and within menu procedures.

Parts — **Global variables**

N, A Arrays for the "node" and "arc" pointers.

Nl, Al Counters to keep track of the total number of nodes and arcs that have been created.

[1] Most of the procedures from the Menu Unit have been omitted because they are of the same format and serve as control mechanisms. They are not generally applicable procedures.

Col Used to assign nodes different colors as they are created.

ActiveN List of active system nodes.

ActiveA List of active system arcs.

InActiveN List of deactivated nodes.

InActiveA List of deactivated arcs.

VisArc List of the visible arcs so they are only displayed once.

VisNode List of the visible nodes so arcs are only displayed between visible nodes.

Problst List of the nodes whose "Prob" parameters have been set so they are only set once during an iteration.

Lnode The last created node, used to automatically create new arcs.

Onode If there are two newly created nodes that have not yet been connected by an arc, then this is the other node.

Show Global variables

Goal Used in triangulation, this is the goal node.

Path List variable used to build the path between two nodes. Used for triangulation.

Carcs List of arcs connected to a node. Used in for triangulation.

B.2 Procedures and Functions

Probability Ratio Graph Software Procedures			
Append	Lst	Makenode	Parts
AppendT	Lst	MakePath	Show
Choice	Ports	Memlst	Lst
Cluster	Show	Minlst	Lst
Common	Lst	Net	Show
Connected	Show	NewArc	Parts
Delmem	Lst	Newnode	Parts
DrawArc	Show	Nummem	Lst
Drawpie	Show	OtherNode	Parts
Explode	Show	Out	Ports
FigNet	Show	OutT	Ports
FigProbs	Show	Pie	Show
FigRatio	Show	Rest	Lst
FillBk	Ports	RestT	Lst
FindPath	Show	ReverseArc	Show
First	Lst	Savedata	Parts
FirstT	Lst	SetPort	Ports
Growup	Show	SetProbs	Show
HideArc	Show	ShowArc	Show
InitParts	Parts	Showmenu	Menu
InitPorts	Ports	Sum	Show
Loaddata	Parts	Triangulate	Show
Makearc	Parts	Which	Parts

Table 12. Procedures used in probability ratio graph software.

The procedure and function look-up follows the order in table 12. They are listed in the following format (only the relevant items are listed with each entry).

Sample procedure	Unit contained in
Function	What it does
Declaration	How it's declared
Result type	What it returns if it's a function
Remarks	General information about the procedure
Restrictions	Things to be aware of

See also Related procedures/functions
Example Sample program or code fragment

This guide only lists procedures and functions contained in the units Lst, Ports, Parts, Show, and Menu. A Turbo Pascal guide should be referenced for procedures and functions contained in the original Turbo Pascal software.

Append function Lst

Function Returns the concatenation of **X** and **Y** in list format.

Declaration append(X, Y : String)

Result type String

Remarks This function joins the two specified strings in list format. That is, with a comma separating the two strings.

Restrictions The final string length is limited to 255 characters. If the two specified strings lengths are greater than 254 when added together (254 because a comma is inserted between the two strings), then the returned string will be truncated at the 255th character.

See also AppendT

Example
```
X := a,c,e,f;
Y := g,th,sdf,v;
S := append(X, Y);
in this example, S = a,c,o,f,g,th,sdf,v
```

AppendT function Lst

Function Concatenates two strings separated with a blank space.

Declaration appendt(X, Y : String)

Result type String

Remarks This function joins the two specified strings in in sentence format. That is, with a space separating the two strings. It is used for supporting the word wrapping capability while in the graphics mode.

Restrictions String lengths cannot exceed 255. This is a limitation of Turbo Pascal string data types. If the two string lengths exceed 254 when added (254 because a space is inserted between them), then the resulting string will be truncated at the 255th position.

See also Append, FirstT, RestT

Example

```
X := 'This unit is';
Y := 'made up of...':
S := appendt(X, Y);
```

In this example, S = 'This unit is made up of...'

Choice function — Ports

Function Accepts user input without disrupting the graphics mode.

Declaration choice

Result type String

Remarks This function enables the system to read, echo, and delete characters during input in the graphics mode without disturbing the graphics environment.

Cluster procedure — Show

Function Performs the "disaggregation" function of probability ratio graphs.

Declaration cluster

Remarks This procedure gathers two nodes (vertices) in to one node. After performing all of the necessary administrative functions on the system variables it redisplays the new system configuration in the net representation.

See also Explode, Triangulate

Common function — Lst

Function Checks if there any common members in the two specified lists.

Declaration common(lst1, lst2 : String)

Result type Boolean

Remarks This function operates on lists where the members are delineated with commas.

See also Memlst

Example

```
lst1 := 'a,s,d,f';
lst2 := 'w,r,t,d,q';
common(lst1, lst2);
```

In this example, common = true

Connected function Show

Function Checks if two specified nodes are connected by an arc.

Declaration connected(node1, node2 : String)

Result type Boolean

Remarks This function checks the two specified nodes' connected arc (ConArc) lists for common members.

Delmem function Lst

Function Deletes all instances of X from the list Y.

Declaration delmem(X, Y : String)

Result type String

Remarks This function operates on lists where the members are delineated with commas.

Example X := a;
Y := a,s,f,r,t,a,d;
S := delmem(X, Y);
In this example, S = s,f,r,t,d

DrawArc procedure Show

Function Draws the specified arc in the specified color.

Declaration drawarc(S : String; K : Integer)

Remarks This procedure performs the calculations and initiates the graphics for displaying arcs.

See also ShowArc, HideArc

DrawPie procedure Show

Function Draws the specified node as a pie chart.

Declaration drawpie(S : String)

Remarks This procedure displays the specified node as a pie chart with the pie slices representing the nodes in the embedded graph.

See also Pie, Net, FigProbs

Explode procedure Show

Function Performs the "aggregation" function of probability ratio graphs.

Declaration explode(S : String)

Remarks This procedure replaces the specified node with its embedded graph. The embedded graph's conditional representation is transformed into a joint representation.

See also Cluster, Triangulate

FigNet procedure Show

Function Sets the "Prob" relative to the arc ratios.

Declaration fignet(C, S : String)

Remarks This procedure sets the "Prob" values of the members of a connected graph relative to the ratios of the arcs connecting the graph.

See also FigProbs

FigProbs procedure Show

Function Initiates the update of the node "Probs" to reflect the connecting arc ratios.

Declaration figprobs(S : String)

Remarks This procedure is used in conjunction with FigNet, Sum, and SetProbs to update the probabilities of a connected graph.

See also FigNet, Sum, SetProbs

FigRatio function Show

Function Figures the ratio between two specified nodes. connecting arc ratios.

Declaration figratio(H : String)

Result type Real

Remarks This function uses the path generated by FindPath/MakePath to figure the ratio between two nodes. It is used in the triangulation procedure.

See also FindPath, MakePath, Triangulation

FillBk procedure Ports

Function Displays a filled polygon with a border.

Declaration fillbk(X1, Y1, X2, Y2, BC, TC : Integer)

Remarks This procedure is used for setting up the different viewports, and scrolling functions and clearing viewports. X1 and Y1 are the top right coordinates of the polygon, and X2 and Y2 are the lower left coordinates of the polygon. BC is the background color and TC is the border and text color.

See also Out, InitPorts

FindPath procedure Show

Function Finds the path between two nodes.

Declaration findpath

Remarks This procedure is used in conjunction with MakePath in the Triangulate and Cluster procedures. The two nodes must be members of the same connected graph.

See also MakePath, Triangulate, Cluster

First function Lst

Function Returns the first member of a list.

Declaration first(X : String)

Result type String

Remarks This function operates on lists where the members are delineated with commas.

See also FirstT, Rest, RestT

Example X := a,s,d,f;

S := first(X);

In this example, S = a

FirstT function Lst

Function Returns the first member of the specified list.

Declaration firstt(X : String)

Result type String

Remarks This function operates on lists where the members are delineated with blank spaces.

See also RestT

Example X := 'tho node is not active...';
S := firstt(X);
In this example, S = 'the'

Growup procedure Show

Function Changes the "Parent" of embedded nodes to their Grandparent.

Declaration growup(C, P : String)

Remarks This procedure is used in conjunction with the explode procedure.

See also Explode

HideArc procedure Show

Function Removes an arc from the screen.

Declaration hidearc(S : String)

Remarks This procedure draws over existing arcs in the background color rendering tho invisible.

See also DrawArc, ShowArc

InitParts procedure Parts

Function Initializes systom variables.

Declaration initparts

Remarks This procedure initializes the system variables. If used while the system has data, the will be lost. It gives no warnings.

InitPorts procedure Ports

Function Initializes the system screen and sets the output variables.

Declaration initports

Remarks This procedure initializes the screen and the output variables that deal with the screen parameters.

See also FillBk

Loaddata procedure — Parts

Function Reads a specified data file and recreates the system as read from the file.

Declaration loaddata

Remarks This procedure retrieves data from the specified file and creates the nodes and arcs specified in the file. The specified file must be in the proper format or it will cause a runtime error.

See also Savedata

MakeArc procedure — Parts

Function Controls the input and output functions for creating a new arc. siblings.

Declaration makearc

Remarks This procedure handles the input and output information gathering function for arc creation. It checks the nodes to ensure that they share the same parent. It will accept the ratio as a single number or as a fraction (.5 or 1/2).

See also MakeNode, NewArc

MakeNode procedure — Parts

Function Controls input and output for creating a new node.

Declaration makenode

Remarks This procedure handles the input and output information gathering function for node creation. It queries for the needed initial parameters, checks for valid parenthood, and ensures the node will be connected to a sibling node.

See also MakeArc, NewNode, OtherNode

MakePath procedure — Show

Function Creates a list which represents the connection path between the specified nodes.

Declaration makepath(H, T, Sa : String)

Remarks This procedure build a list of arcs which connects the specified nodes (H and T). This is a recursive procedure and "Sa" is a parameter used for controlling the recursion.

See also FindPath

Memlst function Lst

Function Determines whether a string, X is a singular member of the list, Y.

Declaration memlst(X, Y : String)

Result type Boolean

Remarks This function operates on lists where the members are delineated by commas. If X or Y are empty, then the value of memlst is false.

See also Common

Example

```
X := a;
Y := c,d,a,s,r;
if memlst(X, Y) then...
                    else...;
```

In this example, memlst = true

Minlst function Lst

Function Returns the specified list with all repeated members deleted.

Declaration minlst(X : String)

Result type String

Remarks This function operates on lists where the members are delineated with commas.

Example

```
X := a,d,a,c,s,c;
S := minlst(X);
```

In this example, S = a,d,c,s

Net procedure Show

Function Displays the specified node's embedded graph as a connected graph of pie charts.

Declaration net(S : String)

Remarks This procedure calculates the positions of the children nodes of "S" and displays the children as pie charts connected with arcs.

See also Pie

NewArc procedure Parts

Function Initializes a new arc pointer and sets the new arc's parameters.

Declaration newarc(H, T : String; R : Real)

Remarks This procedure creates a new arc and initializes the arc's parameters and echos the arc's creation to viewport 4. "H" is the head, "T" is the tail, and "R" is the ratio between them.

See also MakeArc, NewNode

NewNode procedure Parts

Function Initializes a new node pointer and sets the new node's parameters.

Declaration newnode(Na, P : String)

Remarks This procedure creates a new node, initializes the node's parameters, and echos the node's creation to viewport 4. "Na" is the node's name and "P" is its Parent.

See also MakeArc, NewArc

Nummem function Lst

Function Returns the number of members in a list.

Declaration nummem(S : String)

Result type Integer

Remarks This function operates on lists where the members are delineated with commas.

Example S := a,s,d,f;

I := nummem(S);

In this example, I = 4

OtherNode procedure Parts

Function This procedure is called by MakeNode when the specified parent contains no children.

Declaration othernode(P : String)

Remarks This procedure is called when a node is created and the parent is otherwise childless. Parent nodes must contain at least two children.

See also MakeNode

Out procedure Ports

Function Outputs a list string with word wrapping to the current port.

Declaration out(S : String)

Remarks This procedure outputs the string "S", which is a list where the members are separated with commas, to the current port and enables text wrapping while in the graphics mode.

See also OutT

OutT procedure — Ports

Function Outputs a text string with word wrapping to the current port.

Declaration outt(S : String)

Remarks This procedure outputs the string "S", which is a list where the members are separated with spaces, to the current port and enables text wrapping while in the graphics mode.

See also OutT

Pie procedure — Show

Function Sets the position and size parameters for displaying a large pie chart.

Declaration pie(S : String)

Remarks This procedure sets the position and radius for a large pie chart and calls drawpie to draw the actual chart.

See also Drawpie, Net

Rest function — Lst

Function Returns the specified list with the first member deleted.

Declaration rest(X)

Result type String

Remarks This function operates on lists where the members are delineated with commas.

See also RestT, First, FirstT

Example

```
X := a,s,d,f;
S := rest(X);
```

In this example, S = s,d,f

RestT function — Lst

Function Returns the specified list with the first member deleted.

Declaration restt(X : String)

Result type String

Remarks This function operates on lists where the members are delineated with blank spaces.

See also FirstT

Example X := 'the node is not active...';

S := restt(X);

In this example, S = 'node is not active...'

ReverseArc procedure — Show

Function Reverses an arc's orientation.

Declaration reversearc(S : String)

Remarks This procedure hides the specified arc, reverses its parameters, and shows the new arc.

See also HideArc, ShowArc

Savedata procedure — Parts

Function Writes current system data to the specified file.

Declaration savedata

Remarks This procedure writes the active and inactive node lists, the active arc list, the node index, the nodes' names and parents, and the arcs' parameters to an indicated file.

See also Loaddata

SetPort procedure — Ports

Function Changes the active graphics port.

Declaration setport(I : Integer)

Remarks This procedure changes the active viewport to the specified port. It handles all the administrative overhead involved with switching the active output port.

SetProbs procedure — Show

Function Normalises the "Prob" values over a connected graph.

Declaration setprobs(S : String)

Remarks This procedure is used in conjunction with FigNet, Sum, and FigProbs to update the probabilities of a connected graph.

See also FigNet, Sum, FigProbs

ShowArc procedure — Show

Function Draws the specified arc in the color of the head node.

Declaration showarc(S : String)

Remarks This procedure ensures that the specified arc is connecting two nodes that are indeed visible. If the two nodes are visible, ShowArc draws (by calling DrawArc) the arc in the head node's color and outputs the ratio to port 4.

See also DrawArc, HideArc

Showmenu procedure — Menu

Function Outputs the specified menu options to the menu port (2).

Declaration showmenu(S : String)

Remarks This procedure outputs the specified menu options to port 2. "S" is a list with six elements. The elements represent the displayed options for a particular menu.

Sum function — Show

Function Sums the "Prob" values for a connected graph.

Declaration sum(S : String)

Result type Real

Remarks This procedure is used in normalizing the "Prob" values of a connected graph. It calculates the sum of the "Prob" values.

See also FigProbs, FigNet, SetProbs

Triangulate procedure — Show

Function Performs the "triangulation" function of probability ratio graphs.

Declaration triangulate

Remarks This procedure prompts for the node between which to triangulate. It uses FindPath and MakePath and presents the path so the user can choose which arc to replace.

See also Cluster, Triangulate

Which function Parts

Function Returns the numerical value of the specified index.

Declaration which(S : String)

Result type Integer

Remarks This function is used in the identification of the nodes and arcs. The nodes and arcs are indexed in an array format and by retrieving the numerical indox the record values can be retrieved and manipulated.

Example

```
S := 'N12'
T := which(S)
```

In this example, T = 12.

Bibliography

1. Buchanan, Bruce G. and Edward H. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Reading, Massachusetts: Addison-Wesley Publishing Company, 1984. (November/December 1987).

2. Garbolino, Paolo. "A comparison of some rules for probabilistic reasoning," *International Journal of Man-Machine Studies, Volume 27 Numbers 5 & 6*: 709-716 (November/December 1987).

3. Garbolino, Paolo. "Bayesian theory and artificial intelligence: The quarrelsome marriage," *International Journal of Man-Machine Studies, Volume 27 Numbers 5 & 6*: 729-742 (November/December 1987).

4. Goicoechea, Ambrose *et al.* "Expert Systems Models for Inference With Imperfect Knowledge: A Comparative Study," *Proceedings of the 1987 IEEE International Conference on Systems, Man, and Cybernetics,2*. 559-563. New York: IEEE Press, 1987.

5. Groothuizen, R. J. P. "Inexact Reasoning in Expert Systems: An Integrating Overview," National Aerospace Laboratory NLR, Amsterdam, The Netherlands, January 1986.

6. Heckerman, David. "An Empirical Comparison of Three Inference Methods," *The Fourth Workshop on Uncertainty in Artificial Intelligence*: 158-169 (August 19-21, 1988).

7. Henrion, Max. "Uncertainty in Artificial Intelligence: Is Probability Epistemologically and Heuristically Adequate?" Department of Social and Decision Science, and Department of Engineering and Public Policy, Carnegie Mellon University, Pittsburgh, Pa. October 1986.

8. Hollenga, Dane and Bruce W. Morlan. "A Decision-Theoretic Model for Constructing Expert Systems," Working paper distributed in OPER 655, AI and the Operational Sciences. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1989.

9. Hollnagel, Erik. "Information and reasoning in intelligent decision support systems," *International Journal of Man-Machine Studies, Volume 27 Numbers 5 & 6*: 665-667 (November/December 1987).

10. Hollnagel, Erik. "Commentary: Issues in knowledge-based decision support," *International Journal of Man-Machine Studies, Volume 27 Numbers 5 & 6*: 743-751 (November/December 1987).

11. Kalagnanam, Jyant and Max Henrion. "A Comparison of Decision Analysis and Expert Rules for Sequential Diagnosis," *The Fourth Workshop on Uncertainty in Artificial Intelligence*: 205-212 (August 19-21, 1988).

12. Kyburg, Jr. Henry E. "Bayesian and Non-Bayesian Evidential Updating," *Artificial Intelligence: An International Journal, Volume 31 Number 3*: 271-293 (March 1987).

13. Morlan, Bruce W. Draft Dissertation. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September, 1989.

14. Pursch, William C. Class notes distributed in CMGT 523, Contracting and Acquisition Management. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, October 1989.

15. Reid, Thomas F. *Maintenance in Probabilistic Knowledge-Based Systems,"* MS Thesis, AFIT/GOR/ENS/86D-16. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1987.

16. Rowe, Niel C. *Artificial Intelligence Through Prolog.* Englewood Cliffs NJ: Prentice-Hall, Inc., 1988.

17. Simon, Herbert A. "Two Heads Are Better than One: The Collaboration between AI and OR," *Interfaces, Volume 17 Number 4*: 8-15 (July-August 1987).

18. Wise, Ben and Richard B. Modjeski. "Thinking About AI and OR: Uncertainty Management," *Phalanx,* 8-11 (December 1987).

Vita

First Lieutenant Scott E. Deakin was born in Mildenhal, England, 25 October 1963. His parents were so thrilled with his arrival that they decided four was enough. He graduated from Moapa Valley High School in 1981, and attended Brigham Young University, receiving a degree of Bachelor of Science in Mechanical Engineering in December 1985. He was commissioned upon graduation through the USAF ROTC program, and reported for active duty at Lowry AFB, Colorado, on 15 February 1986. He served as a Satellite Operations Officer in the Second Satellite Control Squadron, 2nd Space Wing, Falcon AFB, Colorado, until entering the School of Engineering, Air Force Institute of Technology, in May 1988.

Permanent address: Post Office Box 45
Moapa, Nevada 89025

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GSO/ENS/89D-3		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION School of Engineering	6b. OFFICE SYMBOL *(If applicable)* AFIT/ENS	7a. NAME OF MONITORING ORGANIZATION	
6c. ADDRESS *(City, State, and ZIP Code)* Air Force Institute of Tecnology Wright-Patterson AFB OH 45433-6583		7b. ADDRESS *(City, State, and ZIP Code)*	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS *(City, State, and ZIP Code)*		10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.	

11. TITLE *(Include Security Classification)*
An Investigation and Interpretation of Selected Topics in Uncertainty Reasoning

12. PERSONAL AUTHOR(S)
Scott E. Dankin, B.S., 1st Lt, USAF

13a. TYPE OF REPORT MS Thesis	13b. TIME COVERED FROM ____ TO ____	14. DATE OF REPORT *(Year, Month, Day)* 1989 December	15. PAGE COUNT

16. SUPPLEMENTARY NOTATION
Fr. back Keywords:)

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	Bayes Theorem, Probability, Reasoning, Theses. [illegible]
12	3		
0	5		

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Thesis Advisor: Maj Bruce W. Morlan
Instructor Operations Research
Department of Operational Sciences

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Maj Bruce W. Morlan	22b. TELEPHONE *(Include Area Code)* (513) 255-3362	22c. OFFICE SYMBOL AFIT/ENS

Previous editions are obsolete.

Incorporating techniques for coping with uncertainty in the decision support systems has proven to be a fertile environment for creative ideas. Representations of uncertainty abound and no representation can be said to be inherently incorrect. From a theoretical standpoint, a viable solution must be coherent and logically consistent. Probability theory demonstrates these characteristics while, as of yet, other methods do not.

The purpose of this study was to investigate specific topics in uncertainty reasoning: 1) *Probability ratio graphs* as a representation of the probability model; 2) Dealing with missing information when system parameters are left unspecified; 3) Investigating the difference between probabilistic and causal independence; and, 4) Characterizing secondary uncertainty as spurious evidence and including it in the inference process.

It was shown that probability ratio graphs are a viable method for representing uncertainty, and a method for representing independence with probability ratio graphs is presented. Assuming probabilistic independence for missing information is shown to have intuitive and computational benefits; also shown is that where secondary uncertainty is included in the inference process has great impact on the computational complexity of an inference process.