ABSTRACT

Title: HYBRID CAUSAL LOGIC METHODOLOGY FOR RISK ASSESSMENT

CHENGDONG WANG
Doctor of Philosophy, 2007

Directed by: Professor Ali Mosleh
Department of Mechanical Engineering

Probabilistic Risk Assessment is being increasingly used in a number of industries such as nuclear, aerospace, chemical process, to name a few. Probabilistic Risk Assessment (PRA) characterizes risk in terms of three questions: (1) What can go wrong? (2) How likely is it? (3) What are the consequences? Probabilistic Risk Assessment studies answer these questions by systematically postulating and quantifying undesired scenarios in a highly integrated, top down fashion. The PRA process for technological systems typically includes the following steps: objective and scope definition, system familiarization, identification of initiating events, scenario modeling, quantification, uncertainty analysis, sensitivity analysis, importance ranking, and data analysis.

Fault trees and event trees are widely used tools for risk scenario analysis in PRAs of technological systems. This methodology is most suitable for systems made of hardware components. A more comprehensive treatment of risks of technical systems needs to consider the entire environment within which such systems are designed and operated. This environment includes the physical environment, the socio-economic environment,

and in some cases the regulatory and oversight environment. The technical system, supported by an organization of people in charge of its operation, is at the cross-section of these environments.

In order to develop a more comprehensive risk model for these systems, an important step is to extend the modeling capabilities of the conventional Probabilistic Risk Assessment methodology to also include risks associated with human activities and organizational factors in addition to hardware and software failures and adverse conditions of the physical environment. The causal modeling should also extend to the influence of regulatory and oversight functions. This research offers such a methodology. It proposes a multi-layered modeling approach so that most the appropriate techniques are applied to different individual domains of the system. The approach is called the Hybrid Causal Logic (HCL) methodology. The main layers include: (a) A model to define safety/risk context. This is done using a technique known as event sequence diagram (ESD) method that helps define the kinds of accidents and incidents that can occur in relation to the system being considered; (b) A model that captures the behaviors of the physical system (hardware, software, and environmental factors) as possible causes or contributing factors to accidents and incidents delineated by the event sequence diagrams. This is done by common system modeling techniques such as fault tress (FT); and (c) A model to extend the causal chain of events to their potential human and organizational roots. This is done using Bayesian belief networks (BBN). Bayesian belief networks are particularly useful as they do not require complete knowledge of the relation between causes and effects. The integrated model is therefore a hybrid causal

model with the corresponding sets of taxonomies and analytical and computational procedures.

In this research, a methodology to combine fault trees, event trees or event sequence diagrams, and Bayesian belief networks has been introduced. Since such hybrid models involve significant interdependencies, the nature of such dependencies are first determined to pave the way for developing proper algorithmic solutions of the logic model. Major achievements of this work are: (1) development of the Hybrid Causal Logic model concept and quantification algorithms; (2) development and testing of computer implementation of algorithms (collaborative work); (3) development and implementation of algorithms for HCL–based importance measures, an uncertainty propagation method the BBN models, and algorithms for qualitative-quantitative Bayesian belief networks; and (4) development and testing of the Integrated Risk Information System (IRIS) software based on HCL methodology.

Hybrid Causal Logic Methodology for Risk Assessment

By
Chengdong Wang

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

Advisory Committee:

Professor Ali Mosleh, Chair/Advisor
Professor Mohammad Modarres
Professor Marvin Roush
Professor Joseph Bernstein
Professor Gregory Baecher, Dean Representative

Acknowledgements

I wish to express my sincerest gratitude to my advisor, Professor Dr. Ali Mosleh, for his instruction, generosity, and patience. Without his dedicated support, this research and this dissertation would not have become reality. His support made the completion of my graduate studies and this achievement possible.

In fact, there are many great professors I wish to thank for their time and for many things that I learned for them during the course of my graduate studies. The experience in working as a Teaching Assistant for Professor Marvin Roush was a joy. Professor Mohammad Modarres taught me a great deal about reliability principle and accelerating testing. Professor Joseph Bernstein's electronic reliability class was a significant help. I was able to integrate into this research a great deal from Professor Gregory Baecher's wonderful decision making class, and I appreciate his time and participation as a member of my committee. Professor Carol Smidts shared her software reliability knowledge with me. I also thank Dr. Frank Groen whose insightful discussions were a great help in developing and testing of the HCL algorithms. I am grateful to Dr. Dongfeng Zhu and Ms. Katrina Groth for their dedicated help in the broader collaborative activities where my work was a part. I also deeply appreciate the warm and collegial atmosphere created by my friends Ms. Zahra Mohaghegh, Mr. Reza Kazemi-Tabriz, Mr. Seyed H. Nejad Hosseinian, and Mr. Thiago Pirest in the Center for Risk and Reliability. Many thanks go to Janna Allgood for her excellent editing assistance. Last but not least, I wish to recognize the late Lothar Wolf, of whom I will always have fond memories.

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1 Statement of Problem

Probabilistic Risk Assessment is being increasingly used in a number of industries such as nuclear, aerospace, chemical process, to name a few. Probabilistic Risk Assessment (PRA) characterizes risk in terms of three questions: (1) What can go wrong? (2) How likely is it? (3) What are the consequences? Probabilistic Risk Assessment studies answer these questions by systematically postulating and quantifying undesired scenarios in a highly integrated, top down fashion. The process typically includes the following steps: objective and scope definition, system familiarization, identification of initiating events, scenario modeling, failure modeling, quantification, uncertainty analysis, sensitivity analysis, importance ranking, and data analysis.

Fault trees (FT) and event trees (ET) are widely used tools in probabilistic risk assessment of technological systems. Risk scenarios are developed initially with a technique known as event sequence diagram (ESD) and then their binary logic version in form of event trees. The probabilities of these risk scenarios are quantified in terms of the probabilities of various constituent events. An event tree starts with the "initiating event" and progresses through a series of successes or failures of "intermediate events" until the various "end states" are reached. Details regarding the occurrences of the various "top events" of the event trees are further developed, when necessary, using fault tress. Figure 1 is a pictorial representation of the basic logic of PRA methodology as used today.

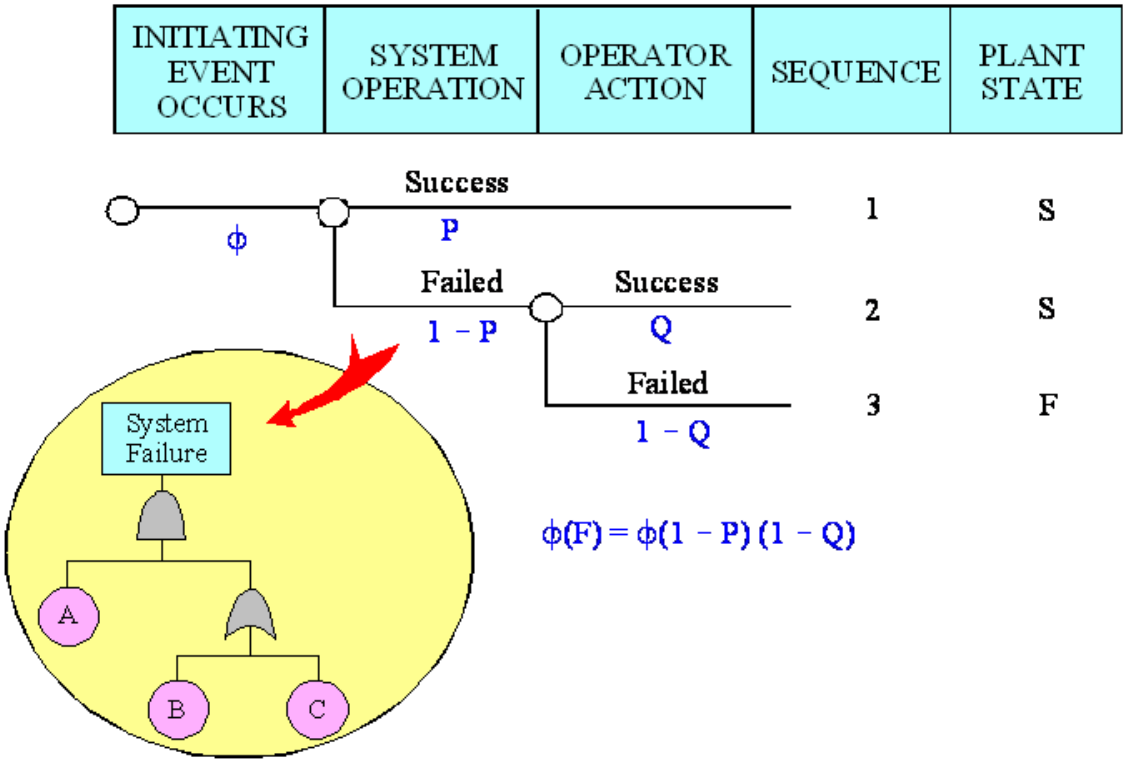


Figure 1: Conventional Probabilistic Risk Assessment structure

Figure 1, shows an event tree including its initiating event and two pivotal events – system operation, operator action; and one fault tree including top event – system failure and three basic events – A, B, C. The analysis includes the scenario generation, scenario quantification, cut set identification and importance measure.

The fault tree is a deductive analysis tool that evaluates the failure, including the type and probability, in the event tree. The fault tree consists of three parts. The top part is the "top event", which corresponds to the failure of a pivotal event in the risk scenario. The middle part consists of intermediate events causing failure of the top event. These events

are linked through logic gates to the bottom part of the fault tree - the basic events, whose failure ultimately causes the top event to occur.

Probabilistic Risk Assessment quantifies scenarios including estimating the frequency of and the consequence of the undesired end states. The frequency of occurrence of each end state is quantified using a fault tree linking approach. This results in a logical product of the initiating event frequency and the probabilities of each pivotal event along the scenario path from the initiating event to the end state.

The methodology is most suitable for hardware systems of components. A more comprehensive treatment of risks of technical systems needs to consider the entire environment within which such systems are designed and operated. This environment includes the physical environment, the socio-economic environment, and in some cases the regulatory and oversight environment. The technical system, supported by an organization of people in charge of its operation, is at the cross-section of these environments. This is depicted by Figure 2. A good example of such systems is the civil aviation system as a whole, which is an extremely complex web of private and governmental organizations, operating or regulating flights involving diverse types of aircraft, ground support, and other physical and organizational infrastructures. In contrast with many other complex systems, the aviation system may be characterized as an “open” system, as the there are large numbers of dynamic interfaces with outside organizations, commercial entities, individuals, physical systems, and environments.

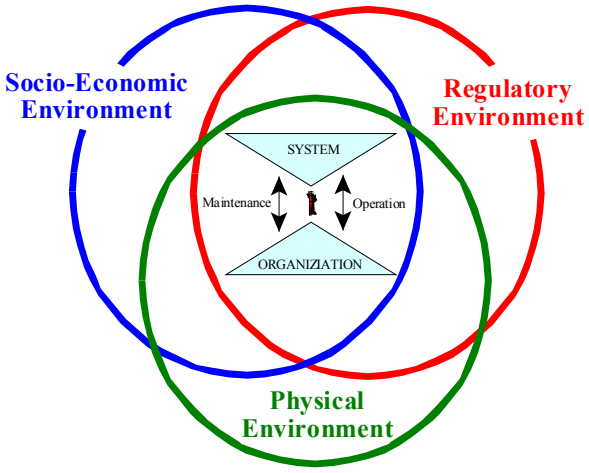


Figure 2: Systems and their complex environment

The external environments include the physical environment, the regulatory/ oversight environment, and the socio-economic environment. At the intersection of these environments, the physical system (e.g., aircraft, ground support infrastructure), is operated and maintained by one or more organizations, through individuals interfacing directly with the physical system. All interfaces are dynamic and interactions and interdependencies are subject to change in manners that may or may not be planned or anticipated. A truly "systems approach" to analyzing and assessing the safety performance of such systems would have to explicitly account for role and effects of these various elements in an integrated fashion.

In order to develop a more comprehensive risk model for these systems, and important steps is to extend the modeling capabilities of the conventional Probabilistic Risk Assessment methodology to also include risks associated with human activities and organizational factors in addition to hardware and software failures, and adverse conditions of the physical environment. The causal modeling should also extend to the influence of regulatory and oversight functions.

This research offers such a methodology. The method developed has its roots in conventional risk analysis techniques used for complex engineered systems, but it has been extended to include the organizational and regulatory environment of the physical system. It recognizes that in order to fully and realistically capture the factors that directly or indirectly impact system risk one has to look at all of the interactive elements and dimensions of this heterogeneous system.

This is achieved through a multi-layered modeling approach so that most appropriate modeling techniques are applied in the different individual domains of the system. The approach is called the Hybrid Causal Logic model. The main layers include:

a) A model to define safety context. This is done using the event sequence diagram (ESD) method that helps define the kinds of accidents and incidents that system being analyzed could experience.

b) A model to capture the behaviors of the physical system (hardware, software, and environmental factors) as possible causes or contributing factors to accidents and incidents delineated by the event sequence diagrams. This is done by common system modeling techniques such as fault trees.

c) A model to extend the causal chain of events to potential human and organizational roots. This is done using Bayesian belief networks. Bayesian belief networks are particularly useful, as they do not require complete knowledge of relation between causes and their potential effects. [Cowell, 1999][Jensen, 2001][Russell and Norvig, 2003][Pearl, 2001][Lauritzen, 1996]

The integrated model is therefore a hybrid causal logic (HCL) model with the corresponding sets of taxonomies and analytical and computational procedures. The modeling framework is pictorially shown in Figure 3.

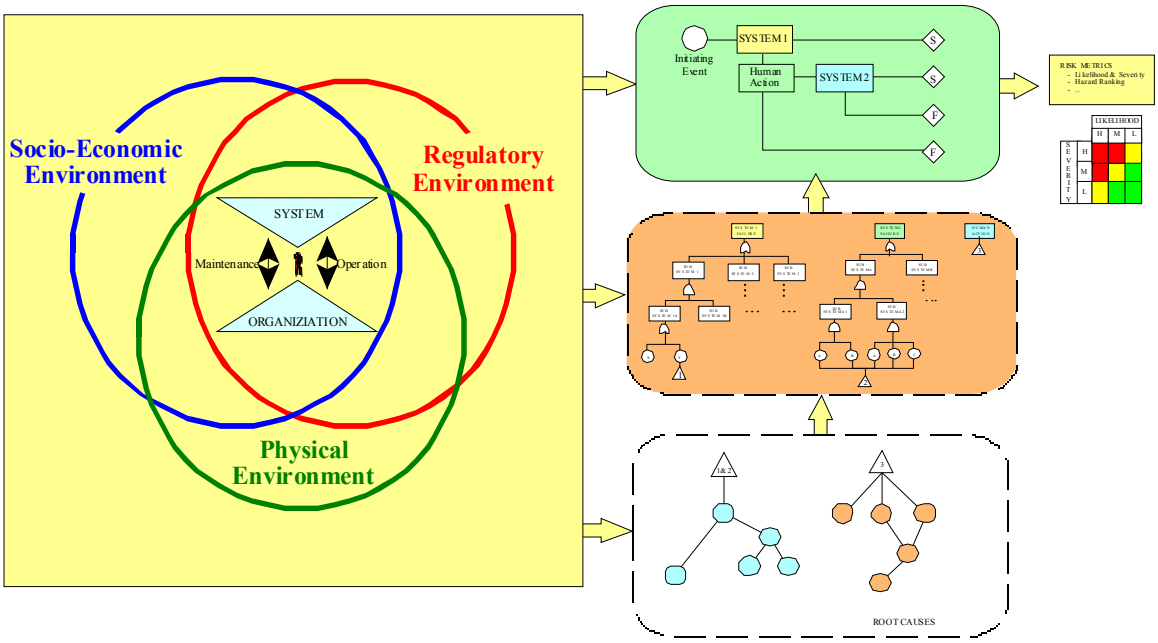


Figure 3: Hybrid risk assessment framework

Here, as in all modeling and analysis methodologies, the scope of analysis, domain of application, and the needs they intend to address, play a central role in defining the focus, general nature, and specific details. The proposed methodology is intended to address multiple requirements and practical needs. Some of the specific requirements include the ability to

1) Identify safety hazards and risks, including those rooted in the system and its external and internal physical, human, organizational, and regulatory environments, and
2) Support risk-informed decision making on safety matters

1.2 Major Achievements

In this research, a methodology to combine fault trees, event trees or event sequence diagrams, and Bayesian belief networks has been introduced. Since such hybrid models involve significant interdependencies, the nature of such dependencies are first determined to pave the way for developing proper algorithmic solutions of the logic model. Major achievements of this work are:

- development of the Hybrid Causal Logic model concept and quantification algorithms; (2) development and testing of computer implementation of algorithms (collaborative work);
- development and implementation of algorithms for HCL–based importance measures, an uncertainty propagation method the BBN models, and algorithms for qualitative-quantitative Bayesian belief networks; and
- development and testing of the Integrated Risk Information System (IRIS) software based on HCL methodology.

HCL successfully incorporates the capability of handling uncertain and incomplete information in risk assessment.

1.3 Dissertation Outline

The remainder of this dissertation describes the details of the various methods and algorithms of the methodology. In Sections 2 and 3, the three main layers of the framework are described. Section 4 shows how these three are integrated into the unified

methodology, Hybrid Causal Logic, including mathematical algorithms for solving the model. Sections 5, 6, and 7 generalize the various Probabilistic Risk Assessment techniques for the Hybrid Causal Logic-based modeling method. These include: the risk importance measures (Section 5), the safety performance indicators (Section 6,) the uncertainty analysis (Section7), and Qualitative-Quantitative Bayesian belief networks (Section 8). Section 9 is devoted to a short description of a software implementation of the methodology, the Integrated Risk Information System (IRIS) software package. Concluding remarks are provided in Section 10.

2. Basic Building Blocks of Conventional Probabilistic Risk Assessment

2.1 Event Sequence Diagram Methodology

2.1.1 Basics

To describe the essence and role of event sequence diagram, we refer to Figure 4. This graph is in the aviation safety context.

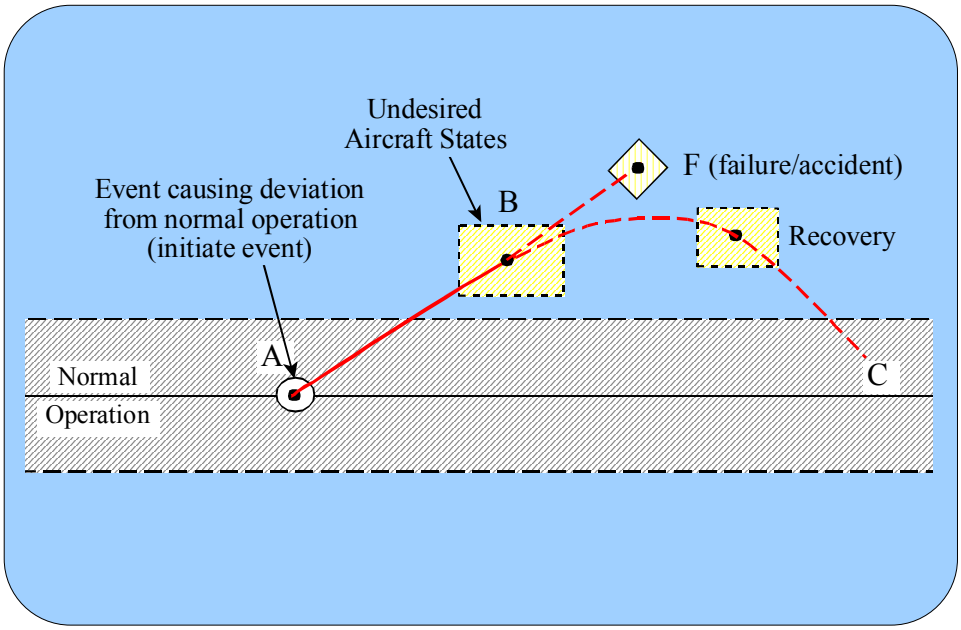


Figure 4: Accident scenario context for safety analysis

The figure depicts the change of state of an aircraft initially operating within the "safe functional/physical zone" (shaded area). At point "A" an event (e.g., equipment failure) occurs causing deviation from the safe zone, putting the aircraft in an undesired state (point "B"). Another event (e.g., crew recovery action) is initiated at that point, and

depending on the whether it succeeds or fails, the aircraft is put back into the safe zone (point "C") or an accident occurs (point "F"). The sequence of events from A (the initiating event) to the end states (C or F) forms two simple scenarios. These scenarios provide the context within which the events and their causes are evaluated as potential hazards or sources of risk.

Event sequence diagram method is a powerful modeling tool for developing possible scenarios. It enables one to visualize the logical and temporal sequence of causal factors, leading to various states of the system. A set of graphical symbols is used to describe the various elements of a scenario. Figure 5 shows a simple event sequence diagram. Event sequence diagrams start with a circle symbolizing the initiating event or condition.

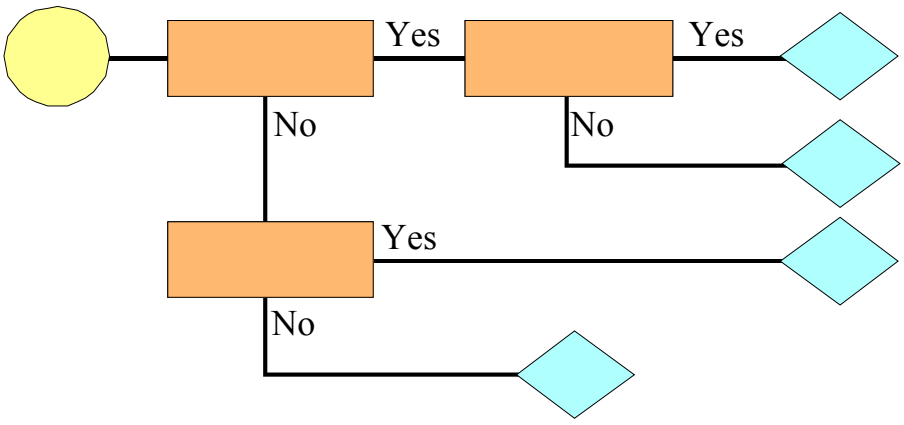


Figure 5: Event Sequence Diagram Concept

The possible events or conditions (rectangles in the diagram) that can follow the initiating event are then listed in the proper temporal or logical order with lines connecting them, forming various possible strings or sequences of events that ultimately end with a diamond symbol representing the end states of the sequences. Pivotal events have a

single input line, and a YES/ NO pair of output lines, depending on whether the pivotal event occurs (YES output) or otherwise (NO output). The same applies in the case of conditions where YES means the condition is satisfied, and NO means the opposite.

An event sequence diagram, therefore, is a visual representation of a set of possible risk scenarios originating from an initiating event. Each scenario consists of a unique sequence of occurrences and non-occurrences of pivotal events (point B or C in Figure 5). Each scenario eventually leads to an end state, which designates the severity of the outcome of the particular scenario. Figure 6 is an example of a very simple event sequence diagram where given the occurrence of the initiating event, the state of System 1 (a pivotal event) determines whether the sequence leads to success (end state S), when it works, or a human action is required, when it fails. Given the success of human action, another pivotal event (state of System 2) will determine the final outcome: success state (S) if System 2 works, and failed state (F) if it fails. The failure of human action also leads to failed state F. Therefore, this simple event sequence diagram depicts four possible risk scenarios, two leading to success, and two leading to a failed state (accident).

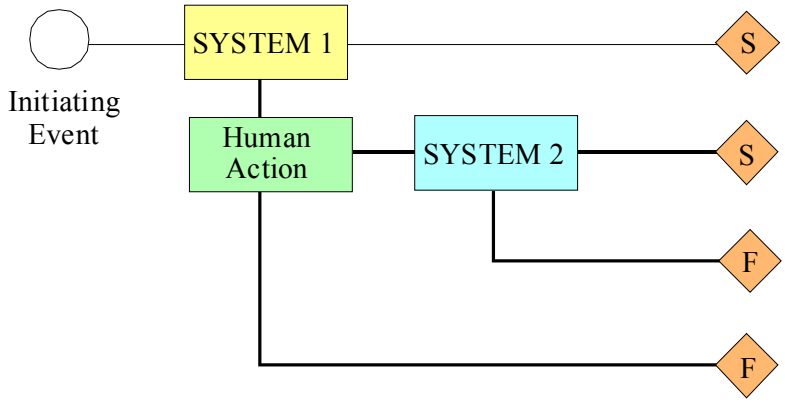


Figure 6: A simple event sequence diagram

Event sequence diagrams are extremely versatile and can be used to model many situations ranging from behavior of purely static systems to many types of dynamic systems. Historically, Event sequence diagram has been a loosely defined term and it has been used in a variety of industries for different purposes. They have been used in probabilistic risk analyses by the nuclear power industry to develop and document the basis for risk scenarios, and also to communicate risk assessment results and models to designers, operators, analysts, and regulators. Event sequence diagrams have also been used in the aviation industry as part of safety and reliability analysis of aircraft systems. NASA has used event sequence diagrams to help identify accident scenarios. In all three applications mentioned above the event sequence diagrams have been used both qualitatively for identification of hazards and risk scenarios as well as quantitatively to find probabilities of risk scenarios [NASA PRA Procedures Guide, 2001].

2.1.2 Event Sequence Diagram Components

Event sequence diagrams are constructed using graphical symbols representing events, conditions, and decision gates. These are defined in the following:

Events: Events can be any observable physical event or condition (e.g., failure or degraded state hydraulic system, crew action, air turbulence, etc). The symbols used to represent events can be seen in Table 1 along with a brief summary of event types.

An initiating event is the first event in an event sequence diagram. It is the one event that starts the sequence of events mapped out in the event sequence diagram. A circle is used to indicate the initiating event in the event sequence diagram. The event sequence diagram ends in one or more end states or terminating events, which are indicted by a

diamond. A pivotal event is an event that has two mutually exclusive outcomes: "yes" (event occurrence) and "no" (event non-occurrence).

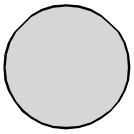
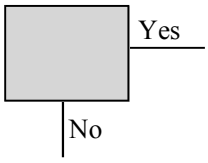
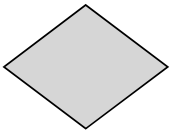
	**Initiating Event:** The first event in an event sequence diagram. Initiates a sequence of events. There is only one initiating event in an event sequence diagram.
	**Comments:** Comments offer information about the scenario without affecting the outcome of the scenario. Comments are often provided for user clarification and readability and have no influence on the scenario.
	**Pivotal Event:** An event with two mutually exclusive outcomes (paths) corresponding to the occurrence or non-occurrence of the event. Typical outcomes are "yes" and "no".
	**Deterministic Delay:** A situation in which no event can occur for the known duration of the delay. Following events may occur immediately after the delay, which effectively shifts the time to occurrence of events following it.
	Random Delay: Similar to a deterministic delay, except that the duration of the delay is random within a prescribed window and defined by a time to completion distribution.
	**End State:** The end point of an event sequence diagram scenario. Many end states are possible in an event sequence diagram.

Table 1: Events in an event sequence diagram

2.1.3 Event Sequence Diagram Construction and Quantification

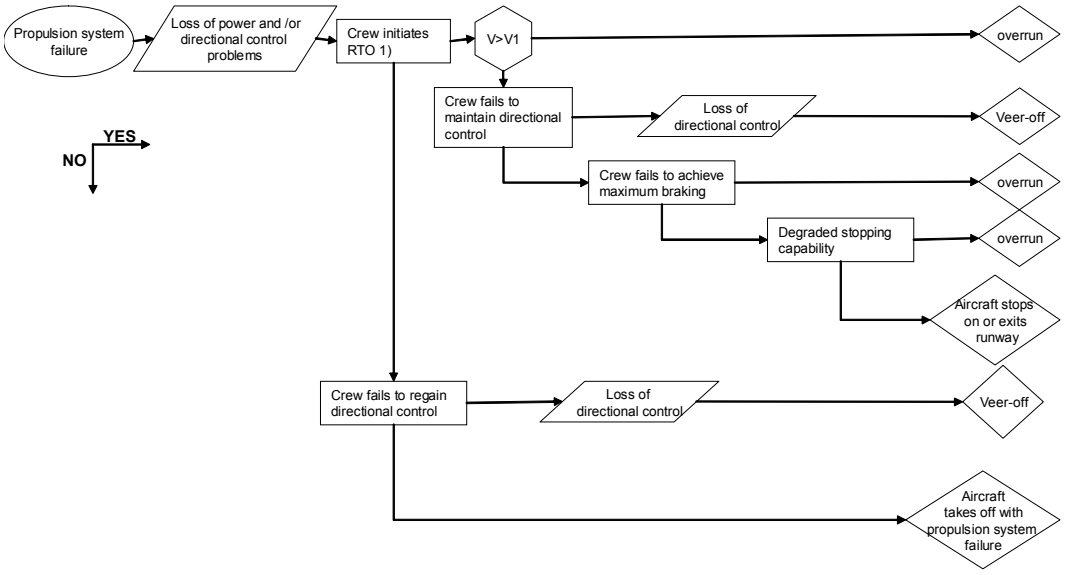
An event sequence diagram can be constructed from the set of events, conditions, and decision gates in combination with the physical parameters, constraints, and dependency rules. As in any modeling endeavor, event sequence diagram construction requires definition of the scope and identification of appropriate levels of abstraction. Both are dependent on the objectives of the analysis, as well as availability of data and resources. For instance while in principle it is possible to develop highly detailed scenario models with all possible causal chains of its events, it is often more useful to keep the event sequence diagrams at a relatively high level of abstraction, leaving the more detailed causal explanation to other models. A good example is the case where it is judged that the pivotal event "failure of a single engine" is of sufficient explanatory power in defining possible accident scenarios in an event sequence diagram. If necessary, the causes of the event (e.g., failures of various components of the engine) can be analyzed and represented by a corresponding fault tree (see discussion on Fault Tree methodology). Initiators are normally the causes of departures and deviation from the norm, or an event that puts the system in a path that makes it susceptible to events, which could cause the deviation from the norm, with a potential to lead to an accident. According to [NASA PRA Guidebook 2002], a useful starting point for identification of initiating events is a specification of 'normal' operations in terms of the nominal values of a suitably chosen set of physical parameters and the envelope in this variable space outside of which an initiating event would be deemed to have occurred. Initiating events are usually active rather than latent conditions or events.

Prior to constructing event sequence diagrams, it is helpful to identify important features or characteristics that would distinguish different classes of accidents scenarios, their initiation, progression, and dominant aspects of the system's physical or operational aspects. Examples are division by phase of flight, organization, location, system, crew, etc. Among other benefits is the fact that in doing so, interdependencies between different events are to some extent avoided, making further development of causal factors in other layers of the hybrid causal model easier and more transparent.

The pivotal events can be either a branch point (where several outcomes are possible) or "pinch points" where several upstream sequences are merged meaning that the future developments of the scenarios (downstream form the pivotal event) are not dependent on how we got to the pivotal event.

Care must be exercised that the path-independent character is not only checked for the basic nature of the event but also its probability, as all events in an event sequence diagram are in principle conditional on the past. Similar to initiating events, pivotal events are normally active events.

Figure 7 is an example of event sequence diagram [Work done at National Aerospace Laboratory of the Netherlands (NLR), under contract from FAA]



(1) appropriate response would be to initiate an RTO when V<V1

Figure 7: An example of generic event sequence diagram (Loss of Control during Takeoff)

2.2 Event Trees

Another method closely related to event sequence diagrams and used in risk and safety analysis of complex systems are event trees and decision trees. Both are inductive logic methods for identifying the various possible outcomes of a given initiating event.

As in event sequence diagrams, an event tree begins with a defined accident-initiating event. This event could arise from failure of a system component, or it could be initiated externally to the system. Different event trees must be constructed and evaluated to analyze a set of accidents. Once an initiating event is defined, all of the safety systems that can be utilized after the accident initiation must be defined and identified. These safety systems are then structured in the form of headings for the event tree. This is illustrated in Figure 8 for two safety systems that can be involved after the defined initiating event has occurred.

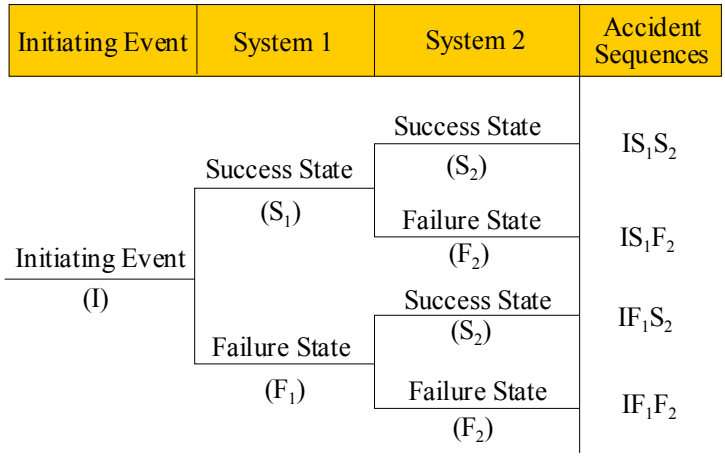


Figure 8: Illustration of an event tree

Once the system failure and success states have been properly defined, the states are then combined through the decision-tree branching logic to obtain the various accident sequences that are associated with the given initiating event. As illustrated in Figure 8, the initiating event is depicted by the initial horizontal line and the system states are then connected in a stepwise, branching fashion; system success and failure states have been denoted by S and F, respectively. The format illustrated follows the standard tree structure characteristic of event tree methodology, although sometimes the fault states are located above the success states.

The accident sequences that result from the tree structure are shown in the last column of Figure 8. Each branch of the tree yields one particular accident sequence; for example, I S_1 F_2 denotes the accident sequence in which the initiating event (I) occurs, system 1 is called upon and succeeds (S_1), and system 2 is called upon but is in a failed state so that it does not perform its defined function. For larger event trees, this stepwise branching would simply be continued.

2.3 Fault Tree Modeling

Fault tree analysis is a technique by which many events that interact to produce other events can be related using simple logical relationships (AND, OR, etc.); these relationships permit a methodical building of a structure that represents the system. Fault tree analysis is the most popular technique used for qualitative and quantitative risk and reliability studies [Henley et al. 1981, Malhotra et al. 1994]. Fault trees have been extended to include various types of logic relations - Priority AND gates, Sequence Dependency gates, Exclusive OR gates, and Inhibitor gates [Dugan et al. 1990] and have been extended to include multi-state systems [Veeraraghavan et al.1994, Wood 1985, Yu et al. 1994, and Zang et al.2003].

To conduct the construction of a fault tree for a complicated system, it is necessary to first understand how the system functions. A system function diagram (or flow diagram) is used to initially depict the pathways by which signals or materials are transmitted between components comprising the system. A second diagram, a functional logic diagram, is sometimes needed to depict the logical relationships of the components.

Only after the functioning of the system is fully understood should an analyst construct a fault tree. Of course, for simpler systems, the function and logic diagrams and a Failure Mode and Effects Analysis are unnecessary and fault tree construction can begin immediately.

In fault tree construction, the system failure event that is to be studied is called the top event. Successive subordinate (i.e., subsystem) failure events that may contribute to the occurrence of the top event are identified and linked to the top event by logical connective functions. The subordinate events themselves are then broken down to their

logical contributors and, in this manner, a fault tree structure is created. Progress in the synthesis of the tree is recorded graphically by arranging the events into a tree structure using connecting symbols called gates.

When a contributing failure event can be divided no further, or when it is decided to limit further analysis of a subsystem, the corresponding branch is terminated with a basic event. In practice, all basic events are taken to be statistically independent unless they are identified as "common cause failures". Such failures are those that arise from a common cause or initiating event, in which case, two or more primary events are no longer independent.

2.3.1 Fault Tree Building Blocks

There are two types of building blocks: gate symbols and event symbols. A sample collection of the most commonly employed is shown in Figure 9.

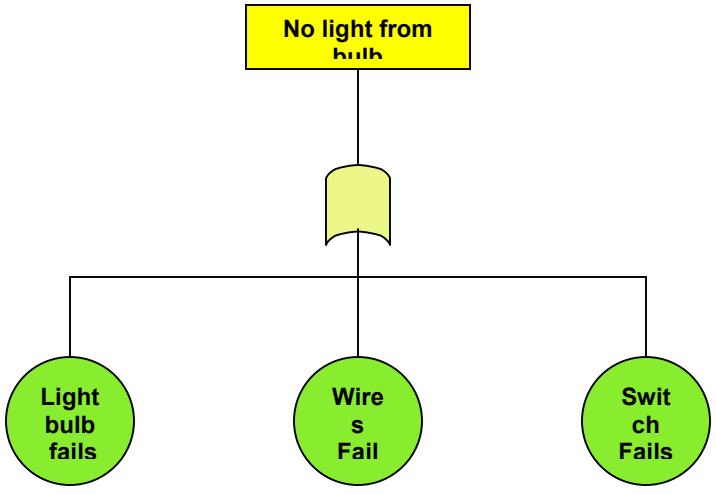


Figure 9: Example of a fault tree logic diagram

Gate symbols connect events according to their causal relations. A gate may have one or more input events but only one output event. The output events of AND gates occur if all input events occur simultaneously. On the other hand, the output events of OR gates happen if any one of the input events occurs. Example of these logic gates is shown in Figure 9 where we see that "Light Is Off" if "Light Bulb Is Failed" OR "Wire Is Failed" OR "Switch Fails Open". The causal relation expressed by an AND gate or OR gate is said to be deterministic because the occurrence of the output event is completely controlled by the input events. There are causal relations that are not deterministic. Consider the two events: "a person is struck by an automobile" and "a person dies". The causal relation of these two events is not deterministic but probabilistic because the accident does not always result in a death.

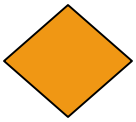
The basic event, a circle, represents a basic initiating fault event that requires no further development. In other words, the circle signifies that the appropriate limit of resolution has been reached.

AND – Out put occurs if all of the input events happens

OR – Out put occurs if at least one of the input events occurs

BASIC EVENT – A basic event requiring no further development

CONDITIONING EVENT – Specific conditions or restrictions that apply to any logic gate (used primarily with PRIORITY AND and INHIBIT gate)

 UNDEVELOPED EVENT – An event which is not further developed either because it is of insufficient consequence or because information is unavailable

 INTERMEDIATE EVENT – An event occurs because one or more antecedent causes acting through logic gates

Figure 10: Basic Fault Tree Building Blocks

2.3.2 Fault Tree Evaluation

Once the tree structure has been established, subsequent analysis is deductive and takes two forms, either qualitative or quantitative. The purpose of a qualitative analysis is to reduce the tree to a logically equivalent form in terms of the specific combinations of basic events sufficient to cause the undesired top event to occur. Each combination will be a "minimal cut set" of failure modes for the tree. One procedure for reducing the tree to a logically equivalent form is by the use of Boolean algebra. The second form of analysis is quantitative or numerical, in which case, the logical structure is used as a model for computation of the effects of selected combinations of primary event failures.

Quantitative analysis of the fault tree consists of transforming its established logical structure into an equivalent probability expression and numerically calculating the probability of occurrence of the top event from the probabilities of occurrence of the basic events. For example for a hardware failure basic event the probability could be that of the failure of component or subsystem during the mission time of interest, T. In such cases the failure probability may be independent of time, such as a demand failure probability or a steady-state unavailability, or the probability may change with time.

Once constructed, a fault tree can be described by a set of Boolean algebraic equations, one for each gate of the tree. For each gate, the input events (such as primary events) are

the independent variables, and the output event (such as an intermediate event) is the dependent variable. Utilizing the rules for Boolean algebra, it is then possible to solve these equations so that the top and intermediate events are individually expressed in terms of minimal cut sets that involve only basic events.

To determine the minimal cut sets of a fault tree, the tree is first translated to its equivalent Boolean equations and then either the "top-down" or "bottom-up" substitution method is used. The methods are straightforward and they involve substituting and expanding Boolean expressions. Two Boolean laws, the distributive law and the law of absorption, are used to remove the redundancies.

2.4 Linked Fault Tree and Event Sequence Diagram

In the probabilistic risk assessment, the event sequence diagram is often used to model a set of possible accident scenarios originating from an initiating event. Each scenario in an event sequence diagram consists of a unique sequence of occurrences and non-occurrences of pivotal events. Each scenario eventually leads to an end state, which designates the severity of the outcome of the particular scenario.

Initiating events as well as pivotal events in the event sequence diagrams are detailed using fault trees (This is the approach in the Quantitative Risk Assessment System (QRAS)) [Groen et al., 2002] similar to the event tree/fault tree combination in classical probabilistic risk assessment.

In this research, the fault tree and event sequence diagram quantification is carried out using Binary Decision Diagram technology, which are now widely recognized for their speed and accuracy.

2.5 Binary Decision Diagram Approach to Solve the Fault Trees or Linked Fault Trees and Event Sequence Diagrams

Boolean analysis allows the analyst to solve the fault tree through generation of cut sets. Boolean analysis quantifies the fault tree by relating the gate input events (the independent variables) to their corresponding output event (the dependent variables). In more recent years fault tree solution methods using binary decision diagram have gained popularity due to their unquestionable superiority over the conventional algorithms both in accuracy and efficiency. This method is therefore used in developing the algorithms for solving the hybrid causal diagrams in this research. The following describe the fundamentals.

2.5.1 Binary Decision Diagrams

A binary decision diagram is a directed, acyclic graph. It was introduced by Lee [Lee 1959] and Akers [Akers 1978], utilized by Bryant [Bryant 1987, Bryant & etc. 1990], improved by Rauzy [Rauzy 1993, Rauzy & etc. 1997], and enhanced in efficiency and accuracy by Sinnamon and Andrews [Sinnamon& Andrews 1997] in the fault tree analysis.

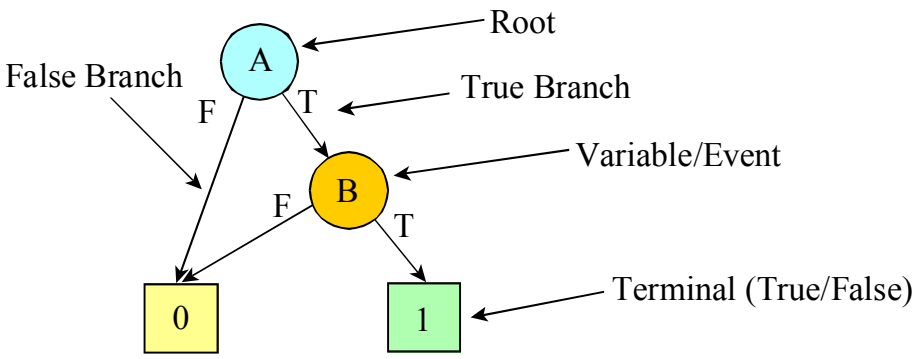


Figure 11: Binary decision diagram representation of a Boolean expression

In effect, a binary decision diagram is a binary decision tree over the Boolean variables with node unification occurring for identical Boolean expressions. A binary decision diagram is a rooted, directed, acyclic graph with an unconstrained number of in-edges and two out-edges, one for each two states of any given variable. As a result, the binary decision diagram has only two terminal nodes labeled 0 and 1, (one or zero representing the Boolean values true or false) representing the final value of the expression. (see for example Figure 11 in which A and B represent events).

Each non-terminal node or vertex has a 0 branch representing basic event non-occurrence and a 1 branch representing basic event occurrence. Each node X in the binary decision diagram can be written in an If-Then-Else (ITE) format [Rauzy & etc. 1997] derived from Shannon's formula:

$$\mathrm{ITE}[X, f_1, f_2] = Xf_1 + \overline{X}f_2$$

where f_1 and f_2 are Boolean functions with X = 1 and $\overline{X} = 0$, respectively which are of one order less than X.

2.5.2 Relation between Fault Tree and Binary Decision Diagram

A fault tree in the graph theory language is an acyclic graph with internal nodes that are logic gates (e.g., AND, OR, K-out-of-N) and external nodes (basic events) that represent system events. Since the events in fault trees are binary by definition and since the fault tree structure is essentially a Boolean expression, any fault tree can be represented by an equivalent binary decision diagram. Figure 12 shows a number of simple fault tree configurations and their binary decision diagram version.

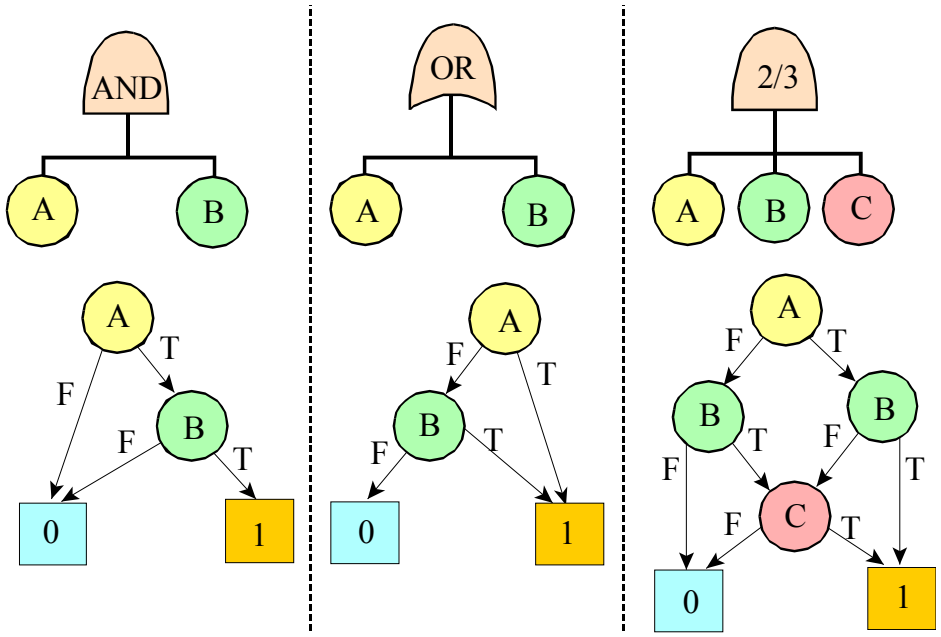


Figure 12: Binary decision diagram representations of simple fault tree

2.5.3 Recursive Formulation of If-Then-Else Engine

Binary decision diagram has the advantage that its quantification does not require the determination of the minimal cut sets or "prime implicants" as an intermediate stage, saving on computation time and improving the accuracy. Converting a fault tree structure to a binary decision diagram is the primary cost.

The probability of the top event of a fault tree is a function of the probabilities of its primary events, and can be calculated by means of a binary decision diagram traversal, with the Shannon decomposition being applied on each node of the binary decision gram. In its most basic form for a binary decision diagram structure f = ite (X_1, f_1, f_0),

$$\Pr(f) = \Pr(X) \cdot \Pr(f_1) + [1 - \Pr(X)] \cdot \Pr(f_0)$$

Where Pr(X) denotes the probability that X = 1, and Pr(f) the probability that f = 1.

Figure 13 shows the quantification of a case where the Top Event as a function of the basic events A, B, and C is f = {A · C or B · C}. The probability of this top event is calculated by summing over the probabilities of the various paths leading to 1. One path involves A = 1, then C = 1, and another is A = 0, B = 1, and C = 1.

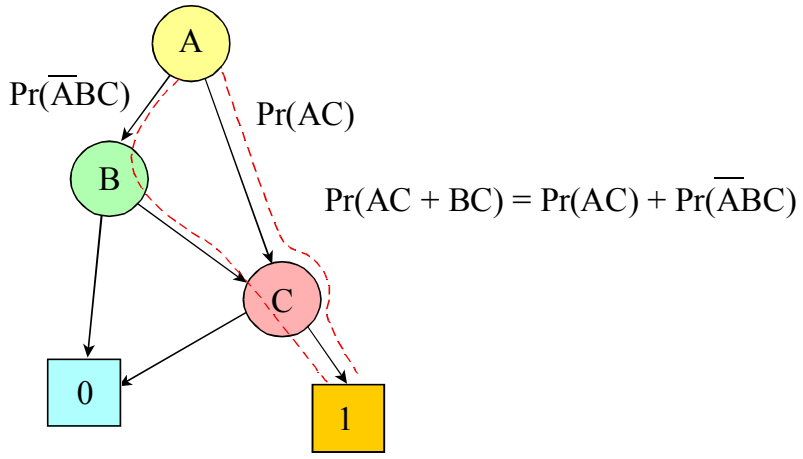


Figure 13: Example quantification of a Simple BDD structure of basic events A, B, and C

2.5.4 Solving Combined Fault Trees and Event Sequence Diagrams Using Binary Decision Diagrams

As discussed earlier one of the most effective ways of modeling risks associated with complex systems is by using event sequence diagrams as the first layer of describing system behavior in case of anomalies, and then providing the more detailed picture of the contributing causes (of the events in the event sequence diagram) by fault trees. Since event sequence diagrams in their most basic form are reducible to binary logic, the combination of event sequence diagrams and fault trees can be converted into a binary

decision diagram also. The process is depicted in Figure 14. This approach was used in the NASA's risk analysis computer code QRAS. [Groen et al., 2002]

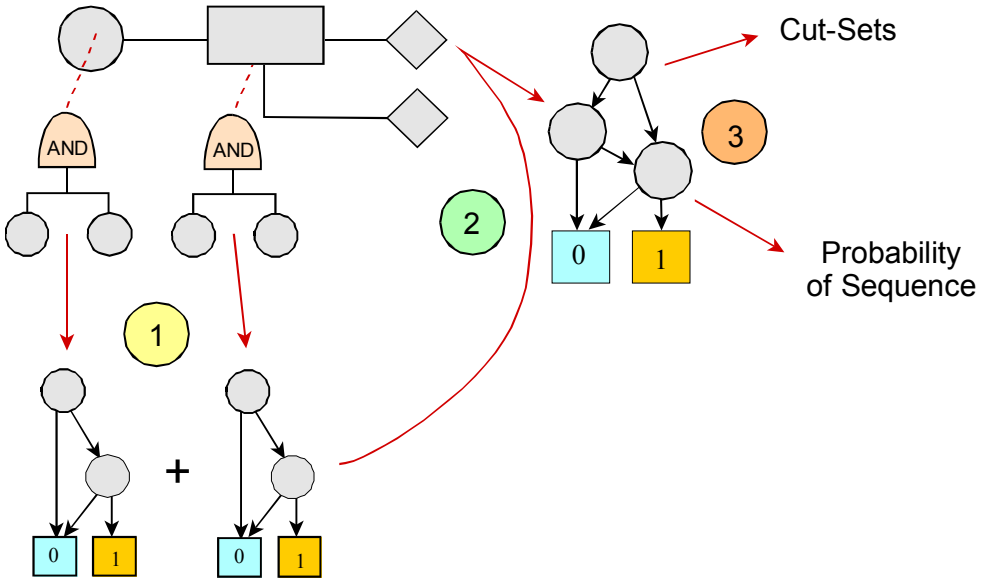


Figure 14: Use of binary decision diagrams to solve combined event sequence diagram and fault tree models

3. Bayesian Belief Networks Applied in Hybrid Causal Logic

3.1 Why Bayesian Belief Networks

Bayesian belief networks can offer a compact, intuitive, and efficient graphical representation of dependence relations and conditional independence between entities of a domain. The graphical structure shows properties of the problem domain in an intuitive way, which makes it easy for non-experts of Bayesian belief networks to understand and build this kind of knowledge representation. It is possible to utilize both background knowledge, such as expert knowledge, and knowledge stored in databases when constructing Bayesian belief networks.

The compact and efficient nature of Bayesian belief network models has been exploited to develop efficient algorithms for solving queries. Queries include diagnosis, explain away, and so on. [Jensen, 2001]

Answering a query is a basic function of Bayesian belief networks. Furthermore, the results of a query can be analyzed by using Bayesian belief networks. For instance, in the field of health care, where a patient has been prescribed a dangerous, high-risk treatment, the patient may want an explanation as to why he or she needs this treatment. Bayesian networks can be used to provide such an explanation. Similarly, in the practice of risk assessment, some observations about the state of the system may be conflicting. Considering the results of two different tests, one test result may indicate that the risk is not high enough to shut down the system, whereas the other test result may indicate the contrary. Data conflict analysis in Bayesian belief networks can be used to identify, trace, and resolve possible conflicts in the observations made. [Jensen, 2001]

In the expert judgment practice, the expert is so called in one domain instead of all areas. The question is how sensitive the query is to each expert when there are different experts in the model. This sensitivity analysis can be performed using Bayesian belief networks. [Jensen, 2001][Neapolitan, 2004] In this research, the measures and justification are shown.

In a decision making scenario, the decision maker may need to acquire additional information before a decision can be made. In this application, Bayesian belief networks can be extended to the influence diagram. A common example is a decision on whether or not to drill for oil at a specific site. The result of an additional test may change the decision, but is it worth the cost to perform the test? The influence diagram is a very useful tool for answering this kind of question. Analyzing the result obtained from queries against a Bayesian belief network can provide a lot of information.

Bayesian belief networks have been used as a practical technique for assessing uncertainty in large and complex systems. [Russell and Norvig 2003]

3.2 Why Do We Need Bayesian Belief Networks for Probability Computations

Bayesian belief networks make explicit the dependencies between different variables. In general, there may be relatively few direct dependencies (modeled by arcs between nodes of the network); this means that many of the variables are conditionally independent. The conditional independence in a network drastically reduces the computations necessary to work out all the probabilities. In general, all probabilities can be computed from the joint probability distribution. In addition, this joint probability distribution is far

simpler to compute when there are a large number of conditionally independent nodes. Comparing the polytree and the non-polytree graph, the computation cost differs significantly.

Suppose, for example in Figure 15, that we have a network consisting of five variables A, B, C, D, and E. If we do not specify the dependencies explicitly, then we are essentially assuming that all the variables are dependent on each other. The chain rule enables us to calculate the joint probability distribution $P(A,B,C,D,E)$ as:

$$P(A,B,C,D,E) = P(A \mid B,C,D,E)P(B \mid C,D,E)P(C \mid D,E)P(D \mid E)P(E)$$

However, suppose that the dependencies are explicitly modeled in a Bayesian belief network:

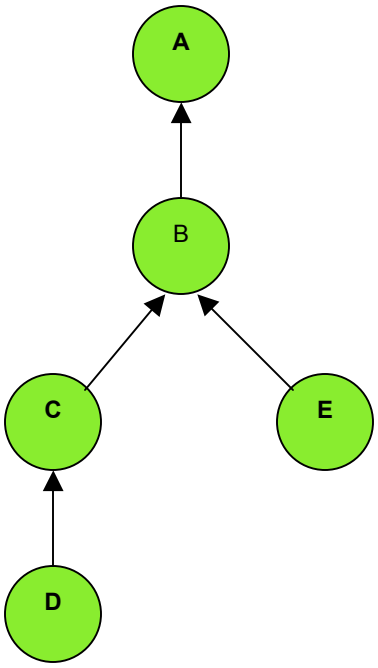


Figure 15: Bayesian belief networks reduce computation cost

Then the joint probability distribution $P(A,B,C,D,E)$ is greatly simplified:

$$P(A,B,C,D,E) = P(A \mid B)P(B \mid C,E)P(C \mid D)P(D)P(E)$$

Bayesian belief networks on their own enable us to model uncertain events and arguments about them. The intuitive graphical representation can be very useful in clarifying previously opaque assumptions or reasoning hidden in the head of an expert. With Bayesian belief networks, it is possible to articulate expert beliefs about the dependencies between different variables. Bayesian belief networks allow an injection of scientific rigor when the probability distributions associated with individual nodes are simply ‘expert opinions'. Bayesian belief networks can expose some of the common fallacies in reasoning due to misunderstanding of the probability.

The real power of Bayesian belief networks comes when we apply an inference algorithm to consistently propagate the impact of evidence on the probabilities of uncertain outcomes. A Bayesian belief network will derive all the implications of the beliefs that are input to it; some of these will be facts that can be checked against observations, or simply against the experience of the decision makers themselves.

3.3 Bayesian Belief Networks Basics

Bayesian belief networks are currently the predominant uncertainty knowledge representation and reasoning technique in artificial intelligence. [Cowell, 1999][Jensen, 2001][Russell and Norvig, 2003][Pearl, 2001][Lauritzen, 1996]. Figure 16 is a well-known example presented by Cowell [Cowell, 1999] in the area of artificial reasoning.

A Bayesian belief network represents the joint probability distribution (JPD) that may be written as

$$\Pr\left(X_1, X_2, \ldots, X_n\right) = \prod_{i=1}^{n} \Pr\left[X_1 \,\middle|\, \text{parent}\left(X_i\right)\right]$$

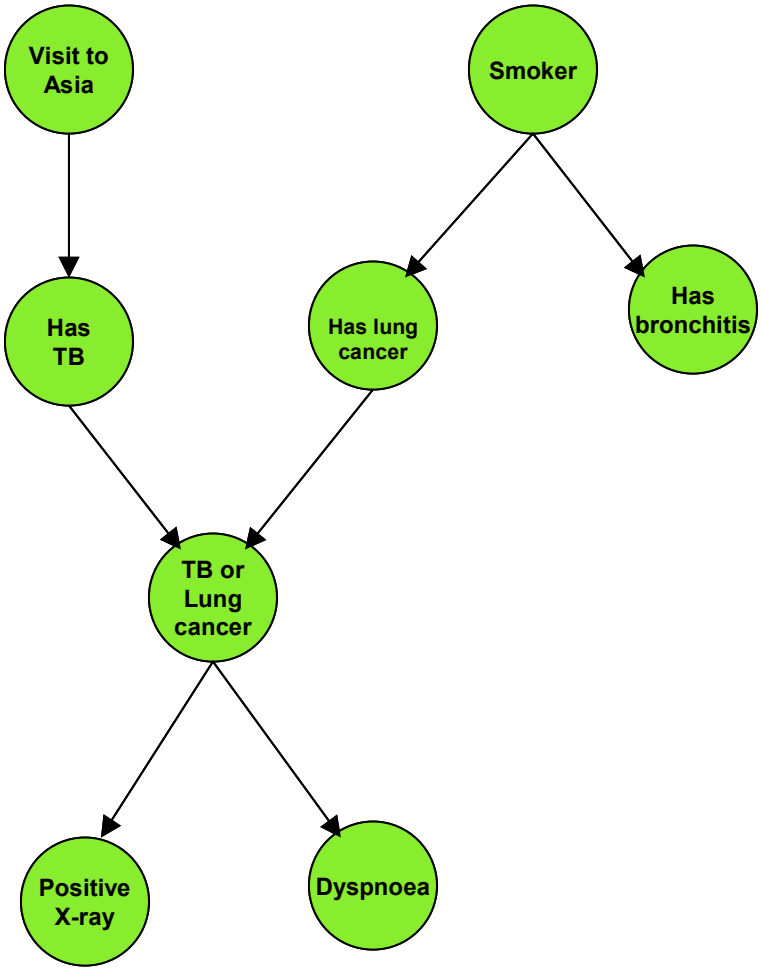


Figure 16: A medical diagnosis example [Cowell 1999]

During the early 1990s, Bayesian belief networks (also called Bayesian networks, belief networks, and causal probabilistic networks) were a hot topic not only among research institutions, but also in the private industry. In the field of artificial intelligence, much research work and application have been done based on Bayesian belief networks. The difference with other expert system techniques is that Bayesian belief networks require the user to have good insight, plus theoretical and practical experience in order to exploit the full function provided by probabilistic reasoning.

In all expert systems, one of the problems is how to treat uncertainty. Uncertainty has many sources, such as inaccurate observation or incomplete or vague information; additionally the relation in the domain may be of non-deterministic type.

The Bayesian belief network is the combination of graph theory and probability theory.

Bayesian Inference Rule

The probability $P(A)$ of an event A is a number between [0,1]. Probabilities obey the following basic axioms:

- $P(A) = 1$ if and only if A is certain.

- If A and B are mutually exclusive, then $P(A and B) = P(A) + P(B)$

The following is a basic rule of probability calculus for dependent events:

$P(A \mid B)P(B) = P(A, B)$, where $P(A, B)$ is the probability of the joint event A and B. Universally, it can be written as $P(A \mid B, H)P(B \mid H) = P(A, B \mid H)$, recognizing a body of knowledge, H.

Further, $P(A \mid B)P(B) = P(B \mid A)P(A)$

This results in the famous Bayes' theorem:

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

D-Separation: [Jensen et al., 1990]

Two variables, A and B, in a causal network are d-separated if, for all paths between A and B, there is an intermediate variable V, such that either

- the connection is serial or diverging and the state of V is known, or

- the connection is converging and neither V nor any of V's descendants has received evidence.

If A and B are not d-separated, we call them d-connected.

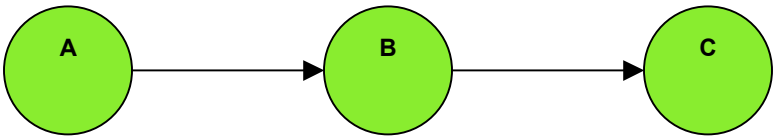


Figure 17: Serial connections example to explain d-separation

The d-separation is explained in the Figure 17 and Figure 18. In Figure 17, A has an influence on B which will influence C. Similarly, evidence on C will influence the certainty on A through B. Obviously, if the state of B is known, then the channel is blocked, and A and C become independent. This is called A and C d-separated given B.

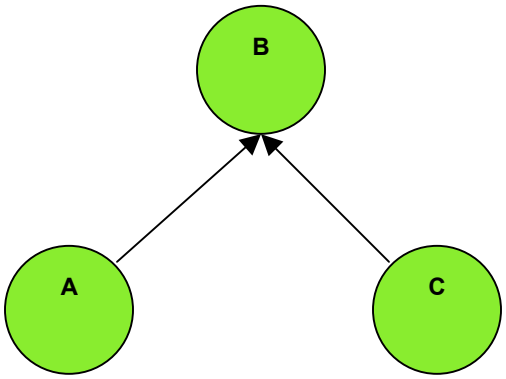


Figure 18: Converging connections example to explain d-separation

In the converging connection of Figure 18, the evidence may be transmitted through a converging connection if either the variable in the connection or one of its descendants has received evidence.

Conditional Independence [Jensen et al., 1990]:

Conditional independence occurs when two variables are d-separated. This means two variables, A and B, in a causal network if for all paths between A and B there is an intermediate variable V, such that either

-the connection is serial or diverging and the state of V is known, or

-the connection is converging and neither V nor any of V's descendants has received evidence

Inference in Bayesian Belief Networks

The purpose of Bayesian belief networks is to make inference under uncertainty. The task is to find the probability of an event given some or no instantiation/evidence of the system state. There are four types of inference [Russell and Norvig 2003]:

1) Diagnostic Inference - updating beliefs from effects to causes.
2) Causal Inference - updating beliefs from causes to effects. This is similar to inference in fault tree and event tree analyses.
3) Inter-Causal Inference - also called explaining away, involves updating beliefs between causes of a common effect.
4) Mixed Inference - this involves updating beliefs using a mixture of the other above inferences types.

The Hybrid Causal Logic approach in this research mainly uses causal inference.

3.4 Bayesian Belief Networks Inference Algorithms

Inference algorithms include exact and approximate inferences. Pearl published an efficient message propagation inference algorithm for polytrees [Pearl, 1982][Kim and Pear 1983] and developed the clustering algorithm for exact inferences in general

Bayesian belief networks, utilizing a conversion to a directed polytree of clusters [Pearl 1986-1][Pearl 1986-2]. For multiply connected networks, a conditioning algorithm is presented by [Pearl 1986-1][Darwiche, 1995][Peot and Shachter, 1991][Diez, 1996]. There are local conditioning algorithms [Diez, 1996], global conditioning [Shachter et al., 1994], dynamic conditioning [Darwiche, 1995], and recursive conditioning algorithms [Darwiche, 2001]. In our algorithm, the refined conditioning method is used when computing the information sent from a Bayesian belief network to a fault tree.

The variable elimination (VE) algorithm [Zhang and Poole, 1994][Shachter et al., 1990][Li and D'Ambrosio, 1994][Dechter, 1996][Zhang and Poole, 1996] is a simple and intuitive approach that eliminates other variables one by one by summing them out. The complexity of variable elimination algorithm can be measured by the number of numerical multiplications and numerical summations it performs. An optimal elimination ordering is one that results in the least complexity, but the problem of finding an optimal elimination ordering is NP-complete [Arnborg et al., 1987]. Later, the junction tree algorithm [Lauritzen and Spiegelhalter, 1988] was developed; this is a time-efficient and space-efficient version of a variable elimination algorithm. It makes it possible to get all node posterior probability in one run, as compared to the variable elimination algorithm that can only quantify one node at a time. In our algorithm, the iterative heuristic algorithm is based on the variable elimination algorithm.

Variable Elimination Algorithm [Zhang and Poole, 1994][Shachter et al., 1990][Li and D'Ambrosio, 1994][Dechter, 1996][Zhang and Poole, 1996]:

Define $f(X_1, X_2, ..., X_k)$ as a factor that is a table of numbers, one for each instantiation of the variables $X_1, X_2, ..., X_k$, $P(X_j \mid Parent(X_j))$ as the Conditional Probability Tables (CPT) for Bayesian belief networks, each CPT in a Bayesian belief networks is a factor.

Let F (a set of factors) be $\{P(X_j \mid Parent(X_j) : j = 1, ..., n)\}$

Choose an elimination ordering $Z_1, ..., Z_n$ of variables in Z.

For $j = 1, ..., n$, compute the new factor $g_j = \sum_{Z_j} f_1 \times f_2 \times ... \times f_k$, where the f_i are the factors in F that include Z_j

Remove the factors f_i (that mention Z_j) from F and add new factor g_j to F

Return $(\prod_{f \in F} F)$

Briefly, the method is moving all relevant terms inside of the innermost sum, performing the innermost sum to get a new term, inserting the new term into the product, then returning to execute it again until all irrelevant terms are summed out.

A common heuristic for eliminating variables is to construct the moral graph [Jensen 2001] and eliminate the variable that will add the fewest number of new edges. For polytrees, one needs to always eliminate singly connected nodes in the moral graph at each stage. In general Bayesian belief networks finding the optimal elimination ordering is NP-complete.

Conditioning Algorithm [[Pearl 1986-1][Darwiche, 1995][Peot and Shachter, 1991][Diez, 1996]

The conditioning method is mainly used to break a loop to a poly tree in Bayesian belief networks (See Figure 19). The conditioning method consists of instantiating a variable

and thereby blocking information path. For directed acyclic graphs with several loops, the technique is repeated until the resulting structure is a polytree. The number of polytree propagations will be the product of the number of states in the conditioning variables. For a DAG over $X = \{A,...,Z\}$, $P(X) = \sum_{a \in A} P(X,a)$ if A is instantiated to the state a [Pearl, 1981] [Diez, 1996].

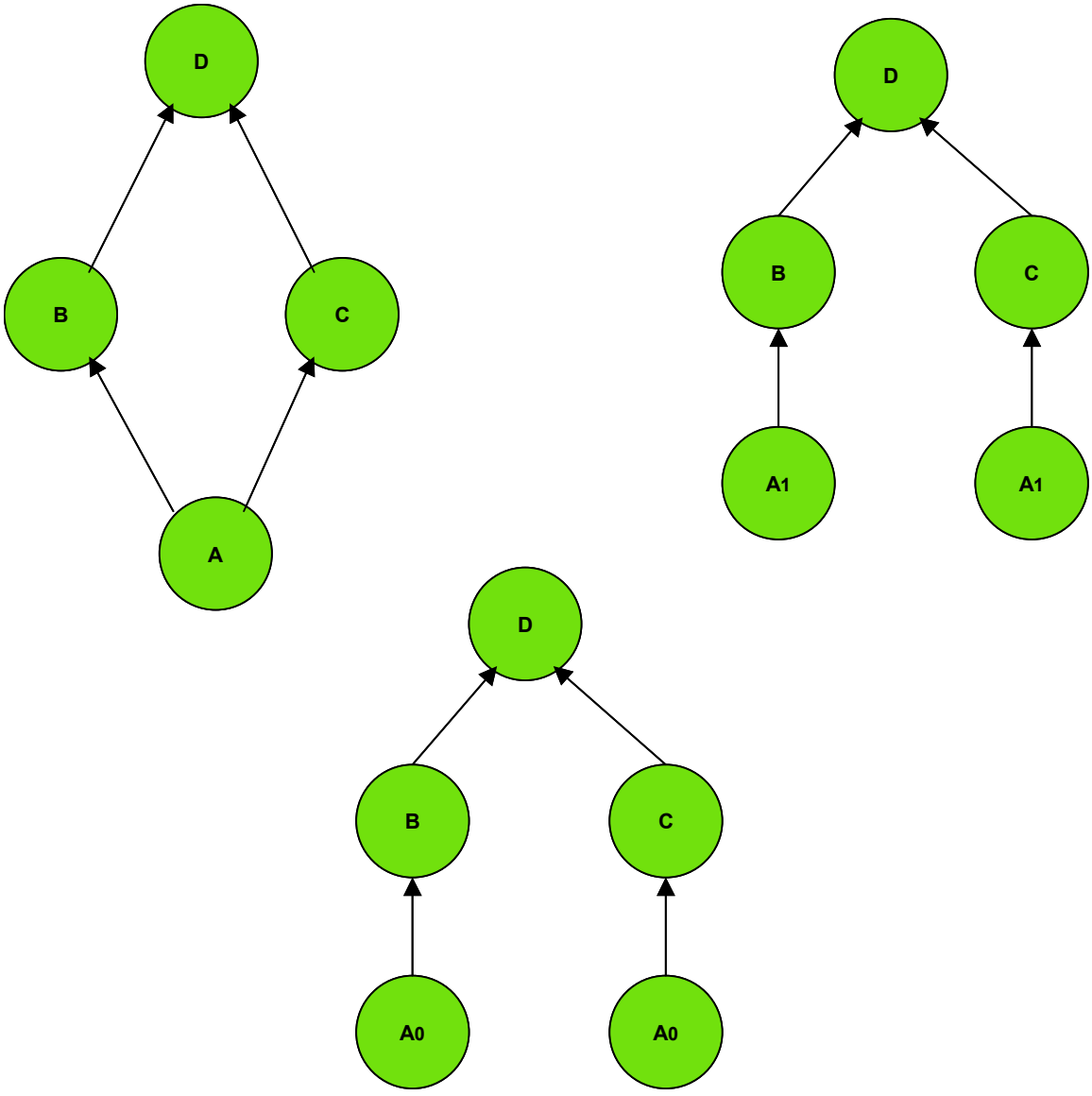


Figure 19: The conditioning breaks the Bayesian belief network into two new polytree Bayesian belief networks

The method of cut conditioning, which was developed as a constraint satisfaction problem, avoids the construction of exponentially large tables. In a Bayesian belief network, a cutset is a set of nodes that, when instantiated, reduces the remaining nodes to a polytree that can be solved in linear time and space. The query is answered by summing over all the instantiations of the cutset, so the overall space requirement is still linear [Pearl, 1988]. A recursive conditioning algorithm allows a complete range of space and time tradeoffs. [Darwiche, 2001]

Junction Tree Algorithm [Lauritzen and Spiegelhalter, 1988]

A junction tree algorithm is a more time efficient and space efficient version of variable elimination algorithm. In variable elimination algorithms, the query variable must be specified first. This means each time, for each different query, the entire algorithm must be run again. Junction tree algorithms can support the simultaneous execution of a large class of queries.

To apply this algorithm, one needs to first triangulate the Bayesian belief networks, and then build the junction tree. After that, initialization and global propagation are applied. If there are evidences entering the network, the set evidence and normalization step are then added. Figure 20 shows the flow of junction tree algorithm.

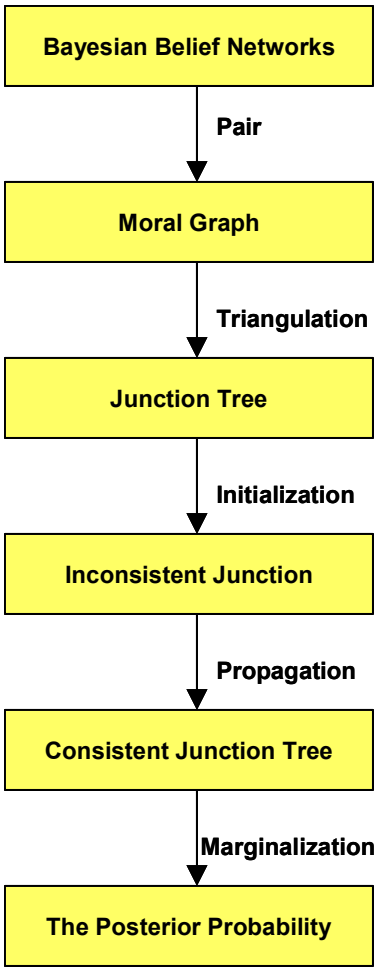


Figure 20: The junction tree algorithm flow chart

This algorithm has the following steps:

Step 1: Convert the Bayesian belief network into a corresponding moral graph, which is an undirected graph in which each variable X and its parents are linked pairwise. The moral graph G_M corresponding to a Bayesian belief network G is constructed by copying G, discarding the directions of the links, and then ensuring for all variables in G that links exist between each pair of parents of the variables, adding new links when necessary.

Step 2: Triangulate the moral graph G_M, by adding links to the graph until all cycles consisting of more than three links contain a link between two nonadjacent nodes in the cycle. Specifically, a triangulated graph can be constructed by creating a copy G_T of G_M. While nodes are left in G_T, select a node from G_T. Connect all nodes in the cluster formed by the selected node and its neighbors, by adding links to G_T and creating corresponding links in G_M. Remove the selected node from G_T.

Step 3: Identify the cliques in G_M, which are the subgraphs that are both complete, meaning that each pair of nodes is connected by a link, and maximal, meaning that the clique is not part of a larger complete subgraph. Cliques are identified during the triangulation of G_M (step 2), by saving each cluster that is not also included in a previously saved cluster.

Step 4: Construct the junction tree G_J by connecting the cliques obtained in step 3 such that the resulting tree satisfies the junction tree property. Start out with a set of trees, each consisting of a single clique. Create candidate separators for each pair of cliques in G_M. Then select a candidate separator, and remove it from the pool of candidate separators. If the cliques corresponding to the candidate separator belong to different trees, connect the trees by inserting the separator between the cliques. Repeat until a single tree remains.

The order in which candidate separators are selected must be such that the separators with the largest number of variables in the intersection are selected first. If two or more candidate separators have the same number of variables, select the separator that connects cliques for which the sum of the number of configurations is smallest.

Comparing conditioning methods with the junction tree method, the junction tree method transforms the network into a probabilistically equivalent, but topologically different polytree by merging the offending nodes; conditioning methods do the transformation by instantiating variables to definite values, and then evaluating a polytree for each possible instantiation.

3.5 Bayesian Belief Network Scale Capability

Bayesian belief networks consisting of more than one thousand nodes are not uncommon. For example in Quick Medical Reference (QMR-DT) network there are more than 4000 nodes [Shwe et al., 1991]. QMR-DT is a two-level or bi-partite graphical model. The top level of the graph contains nodes for the diseases, and the bottom level contains nodes for the symptom. This is shown in Figure 21 as diseases and findings.

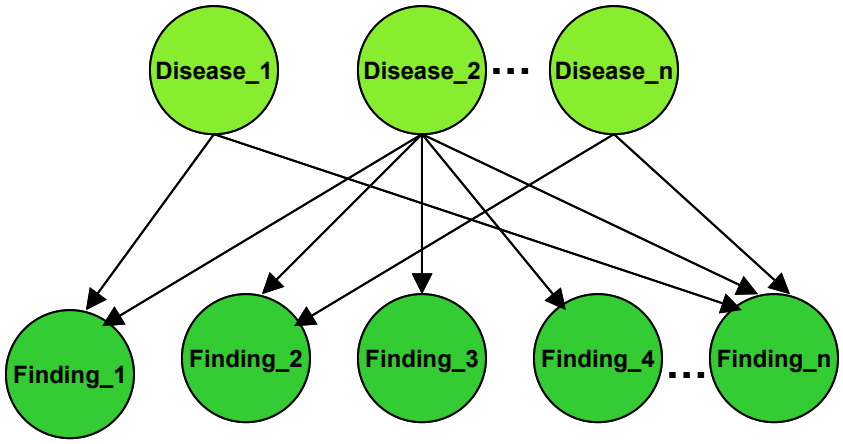


Figure 21: Part of the QMR-DT network

3.6 Bayesian Belief Network Application

Bayesian belief networks have been widely applied, especially in the field of artificial intelligence, including such applications as medical diagnosis [Heckerman 1990], a variety of Microsoft product's help systems [Horvitz, Breese, Heckerman, 1998], intelligent tutoring, and troubleshooting. Recently, it has been used for risk assessment and root cause analysis in the field of civil engineering and for application in software reliability estimation.

An appealing application of Bayesian belief networks in the area of probabilistic risk assessment is in their integration with the main modeling methods of PRA, namely the event sequence diagram (or alternatively even trees) and fault tress. For this purpose a full set of integrating algorithms need to be developed. This is the core of this dissertation. We will first review some possible options.

3.6.1 Fault Trees Converted to Bayesian Belief Networks

A the Bayesian belief network structure can also represent the logic of fault trees, and serve as method for solving (quantifying) fault trees. By extension a hybrid causal model of Bayesian belief networks and fault trees can be solved by converting the fault tree into an equivalent Bayesian belief network. Essentially Bayesian belief networks can be seen as the generalization of the fault trees, since Bayesian belief networks can handle multiple states, and deterministic analysis is a special case of probabilistic analysis. A FT-to-BBN mapping method has been described by [Bobbio et al., 1999] and [Bobbio et al., 2001], Figure 22, and

Figure 23 shows how fault trees can be modeled by using Bayesian belief networks in the case of the AND, and OR gates. The proof presented is slightly different from Bobbio's proof .

(a) Bayesian belief network equivalent of OR Gate.

Figure 22 shows a simple OR gate and its Bayesian belief network representation

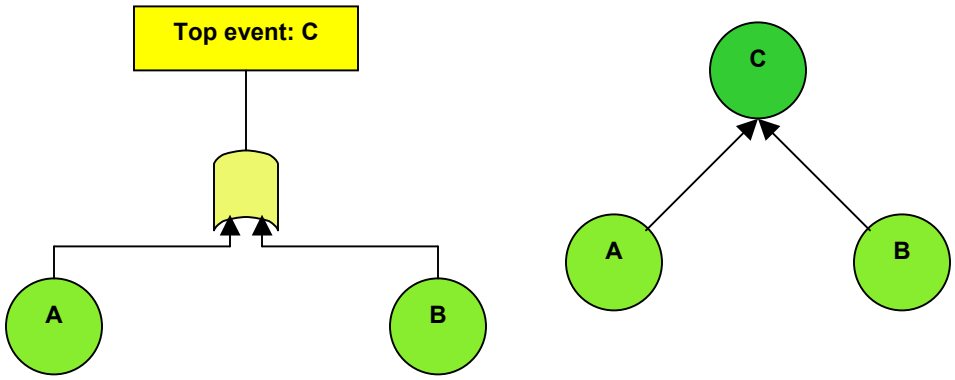


Figure 22: Fault tree OR gate mapping to Bayesian belief network

For a Bayesian belief network to be equivalent to the OR gate, conditional probability table should be as follows:

$$\Pr\left(C=1 \middle| A=0, B=0\right)=0$$

$$\Pr\left(C=1 \middle| A=0, B=1\right)=1$$

$$\Pr\left(C=1 \middle| A=1, B=0\right)=1$$

$$\Pr\left(C=1 \middle| A=1, B=1\right)=1$$

In the fault tree, the Boolean expression will be C = A + B. We may infer that $\Pr(C) = \Pr(A) + \Pr(B) - \Pr(A)\Pr(B)$ if A and B are assumed to be statistically independent.

In the equivalent Bayesian belief network with the above conditional probability table we have,

$$\Pr(C) = \Pr(C \mid A, B)\Pr(A, B) + \Pr(C \mid \overline{A}, B)\Pr(\overline{A}, B)$$
$$+ \Pr(C \mid A, \overline{B})\Pr(A, \overline{B}) + \Pr(C \mid \overline{A}, \overline{B})\Pr(\overline{A}, \overline{B})$$
$$= \Pr(A, B) + \Pr(\overline{A}, B) + \Pr(A, \overline{B})$$

Since

$$\Pr(A) + \Pr(B) = [\Pr(A, \overline{B}) + \Pr(A, B)] + [\Pr(\overline{A}, B) + \Pr(A, B)] = \Pr(A, \overline{B}) + \Pr(\overline{A}, B) + 2\Pr(A, B)$$

Then

$$\Pr(C) = \Pr(A) + \Pr(B) - \Pr(A)\Pr(B)$$

(b)Bayesian belief network equivalent of AND Gate

Figure 23 shows a simple AND gate and its Bayesian belief network representation

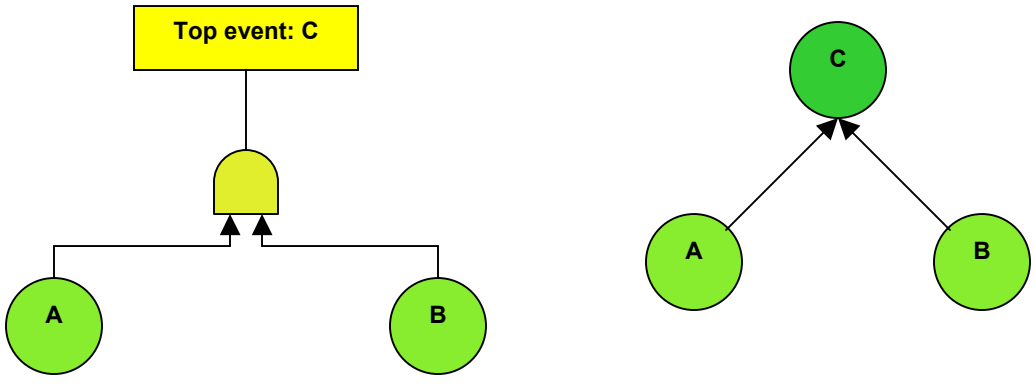


Figure 23: AND Gate and BBN Equivalent

The conditional probability table is as follows:

$$\Pr\left(C=1 \middle| A=0, B=0\right)=0$$
$$\Pr\left(C=1 \middle| A=0, B=1\right)=0$$
$$\Pr\left(C=1 \middle| A=1, B=0\right)=0$$
$$\Pr\left(C=1 \middle| A=1, B=1\right)=1$$

In fault trees, the Boolean expression will be C = AB; then the probability can be

$\Pr(C) = \Pr(A)\Pr(B)$ if A and B are assumed statistically independent.

In Bayesian belief network,

$$\begin{aligned}\Pr(C) &= \Pr\left(C \middle| A, B\right)\Pr(A, B) + \Pr\left(C \middle| \overline{A}, B\right)\Pr\left(\overline{A}, B\right) \\ &\quad + \Pr\left(C \middle| A, \overline{B}\right)\Pr\left(A, \overline{B}\right) + \Pr\left(C \middle| \overline{A}, \overline{B}\right)\Pr\left(\overline{A}, \overline{B}\right) \\ &= \Pr(A, B)\end{aligned}$$

The K-out-of-N gate modeling process is similar.

3.6.2 Attaching Bayesian Belief Networks to Fault Tree Basic Events

An approach to combining fault tree analysis and Bayesian belief networks has been shown in Pai and Dugan's work [Pai and Dugan, 2001] in the context of software reliability modeling (see Figure 24). The paper presents linking individual basic events to separate Bayesian belief networks; in other words, one node in the Bayesian belief networks serves as the basic event in the fault tree. The limitation is that the BBNs attached to various basic events cannot have common nodes, a condition that is difficult to meet in many practical applications.

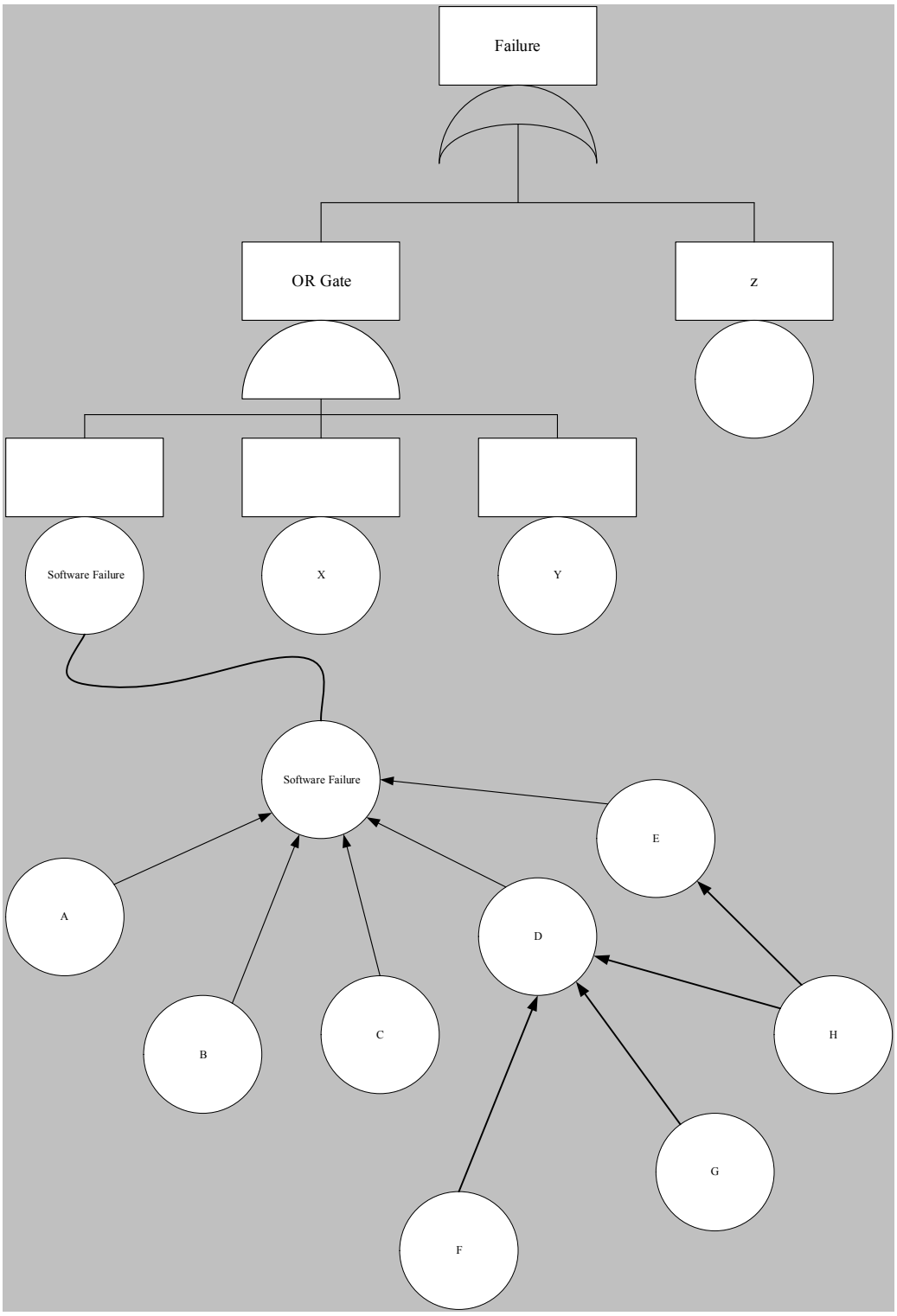


Figure 24: The Bayesian belief network serves one basic event in the fault tree

3.6.3 Modeling Event Trees using Bayesian Belief Networks

The following algorithm for modeling event trees using Bayesian belief networks was developed in this research. In general, an event tree can be converted into a Bayesian belief network without losing any information. Since the top events in event trees are usually dependent, Bayesian belief networks can show the dependency more transparently.

The procedure to model event trees using Bayesian belief networks is to define a node in a Bayesian belief network for each event in event trees with the respective conditional probability table obtained from the event tree:

1. For the initiating event in the event tree, create a root node in the Bayesian belief network, and assign the prior probability of the initiating event to this root node.
2. For each event in the event tree, create a corresponding leaf node in the Bayesian belief network.
3. Connect the nodes in corresponding order in the event tree.
4. Assign the equivalent conditional probability table in corresponding order to the conditional probability in the event tree.

The process is illustrated through the following example of Figure 25 and Figure 26.

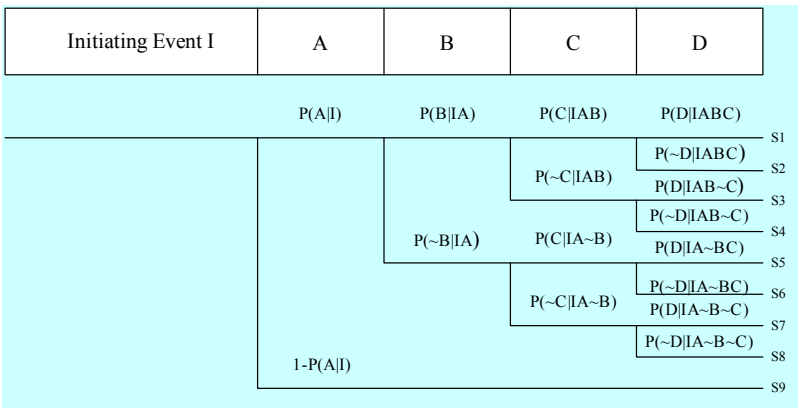


Figure 25: The event tree to be mapped into Bayesian belief network

Figure 25 shows a typical event tree with dependency between the events. As a numerical example, the quantification of Scenario 6 is shown below.

$$\phi(S6) = \phi(I)P(A \mid I)P(\overline{B} \mid IA)P(C \mid IA\overline{B})P(\overline{D} \mid IA\overline{B}C)$$

$$P(I) = 0.002$$
$$P(A \mid I) = 0.003$$
$$P(\overline{B} \mid IA) = 0.995$$
$$P(C \mid IA\overline{B}) = 0.055$$
$$P(\overline{D} \mid IA\overline{B}C) = 0.08$$

The probability of the scenario will be 2.6268E-8. The corresponding BBN is shown in Figure 26.

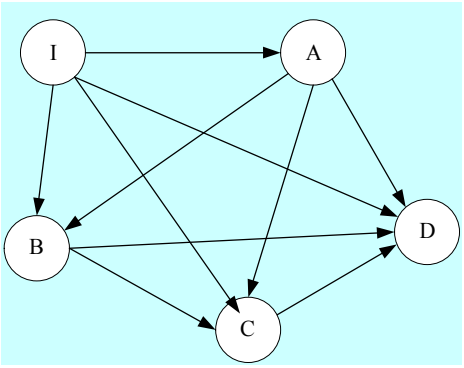


Figure 26: The Bayesian belief network mapped from Figure 25

The conditional probability table (Table 2) is assigned according to the conditional probability in the event tree. Here F denotes fail, S denotes success.

I	P(I)
0	0.002
1	0.998

I	A	P(A|I)
0	0	0.003
0	1	0.997
1	0	0
0	1	1

I	A	B	P(B|I, A)
0	0	0	0.005
0	0	1	0.995
0	1	0	0.008
0	1	1	0.992
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

I	A	B	C	P(C|I, A, B)
0	0	0	0	0.06
0	0	0	1	0.94
0	0	1	0	0.005
0	0	1	1	0.945
0	1	0	0	0.033
0	1	0	1	0.967
0	1	1	0	0.06
0	1	1	1	0.94
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0

1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

I	**A**	**B**	**C**	**D**	**P(D|I, A, B, C)**
0	0	0	0	0	0.06
0	0	0	0	1	0.94
0	0	0	1	0	0.05
0	0	0	1	1	0.95
0	0	1	0	0	0.08
0	0	1	0	1	0.92
0	0	1	1	0	0.011
0	0	1	1	1	0.989
0	1	0	0	0	0.05
0	1	0	0	1	0.95
0	1	0	1	0	0.08
0	1	0	1	1	0.92
0	1	1	0	0	0.1
0	1	1	0	1	0.9
0	1	1	1	0	0.06
0	1	1	1	1	0.94
1	0	0	0	0	0
1	0	0	0	1	1
1	0	0	1	0	0
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	0	1	1
1	0	1	1	0	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	0	1	1
1	1	0	1	0	0
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	0	1	1
1	1	1	1	0	0
1	1	1	1	1	1

Table 2: Conditional probability table for mapping the example event tree to equivalent Bayesian belief network

In Bayesian belief networks, Scenario 6 operates by querying the joint probability:

$$P(\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_\mathrm{n}) = \prod_{\mathrm{i}=1}^{n} P\left[X_i \,|\, Parent(X_i)\right]$$

for the unique joint states where $i = i,\ A = a,\ B = \bar{b},\ C = c,\ D = \bar{d}$, and the joint probability is

$$P(i, a, \bar{b}, c, \bar{d}) = P(i)P(a \,|\, i)P(\bar{b} \,|\, i, a)P(c \,|\, i, \bar{a}, \bar{b})P(\bar{d} \,|\, i, a, \bar{b}, c)$$
$$= 0.002 \times 0.003 \times 0.995 \times 0.055 \times 0.08 = 2.6268E - 8$$

After updating the networks for every query, the same numeric result 2.6268E-8 is obtained.

4. Hybrid Causal Logic Methodology and Algorithm

We have defined the combination of event trees/event sequence diagrams, fault trees and Bayesian belief networks as Hybrid Causal Logic diagram. In a Hybrid Causal Logic diagram, Bayesian belief networks can be used to model the basic events in the fault trees or the pivotal events in the event tree. The various BBNs can have common nodes. The combination can be made at any level to build a Hybrid Causal Logic diagram (see Figure 27).

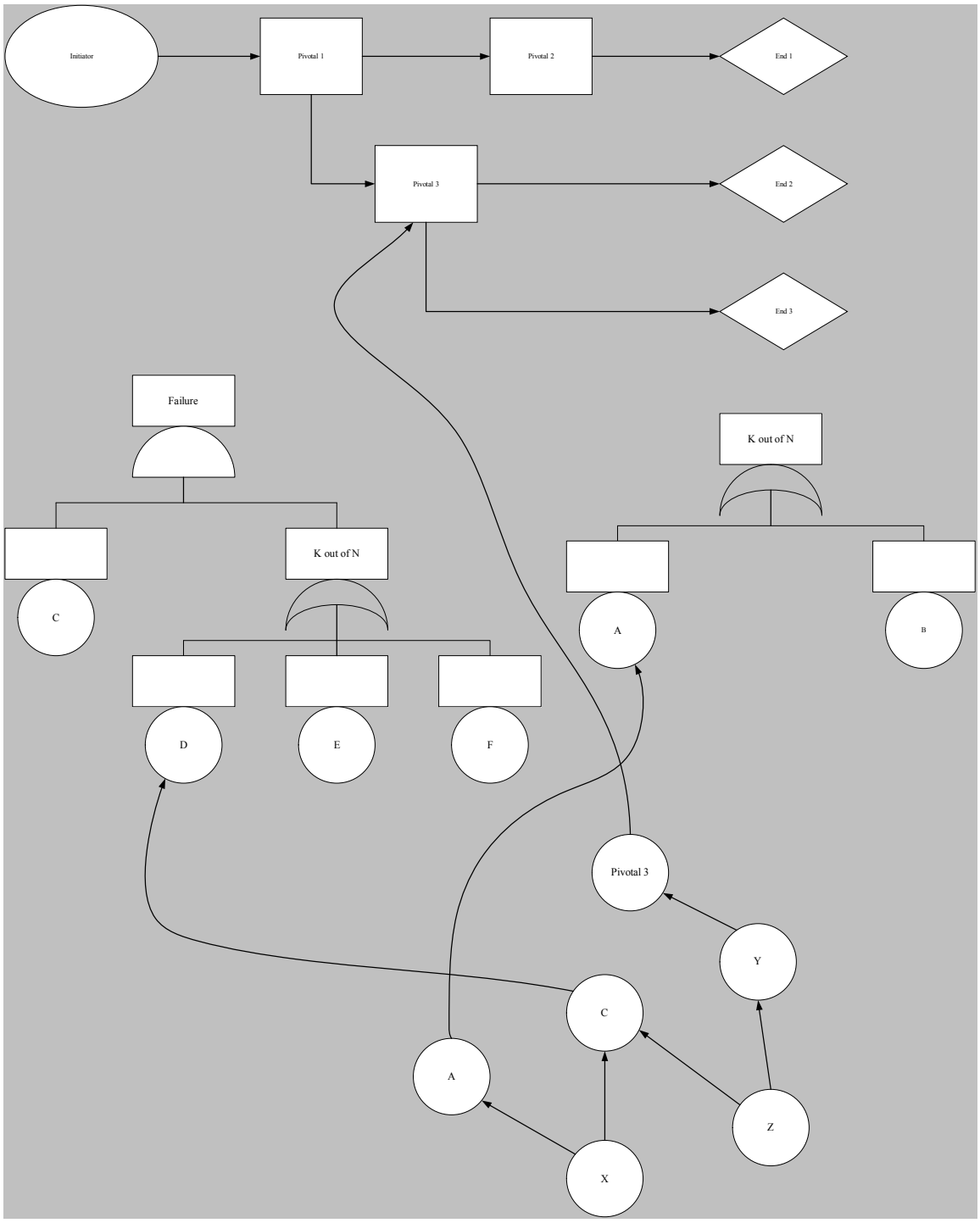


Figure 27: An example Hybrid Causal Logic diagram

The hybrid process can include: (1) Bayesian belief networks modeling the basic event in fault trees, for example the node A, C in Figure 27; (2) Bayesian belief networks modeling the pivotal event in the event sequence diagram, for example the node Pivotal 3

in Figure 27; (3) furthermore, the fault tree top event can model the root node in the Bayesian belief network.

4.1 The Nature of the Dependency in HCL Diagrams

What makes the combination (hybrid model) of complicated is the dependency introduced by the link in the fault tree context, or the loop introduced by the link in Bayesian belief networks (see Figure 28).

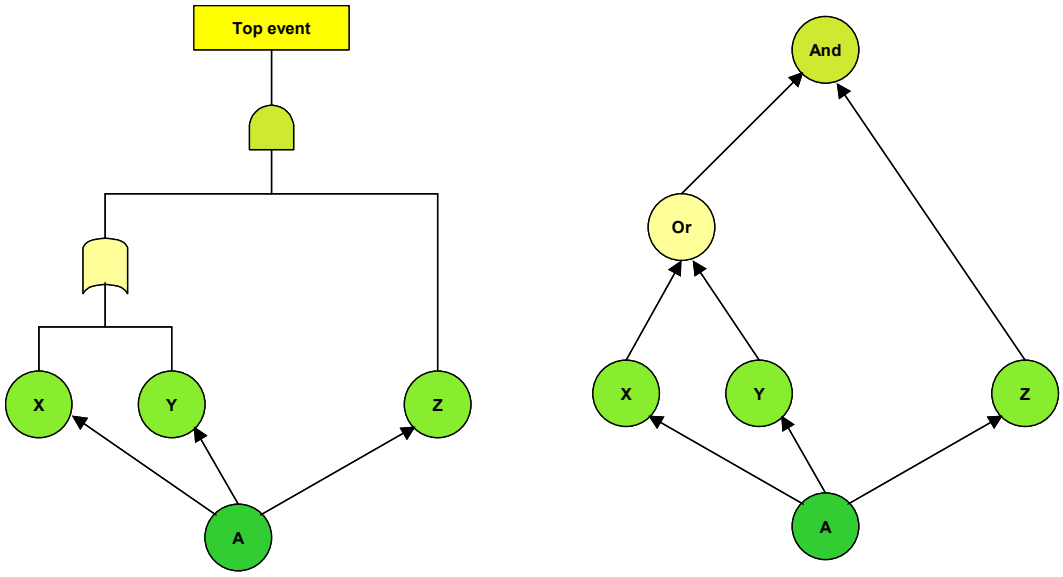


Figure 28: The Loop introduced in the combination of fault trees and Bayesian belief networks

By putting everything into a Bayesian belief network the dependency becomes obvious. The loop as seen in Figure 28 will introduce the double counting in the networks. There are three loops, one is A, X, Y, OR, the second is A, Y, OR, Z, AND, and the third one is A, X, OR, Z, AND. According to Weiss's research, Pearl's message-passing algorithm can still be directly applied in some cases [Weiss, 2000]. However, in general this will

become problematic, especially when the probability is very small. The double counting will cause errors in the output.

The following example (Figure 29) will show the difference between the case where one considers the dependency and the case where dependency is ignored. As seen in Figure 29, the fault tree basic events A, B are modeled by a BBN. The prior and conditional probabilities are shown in Table 3.

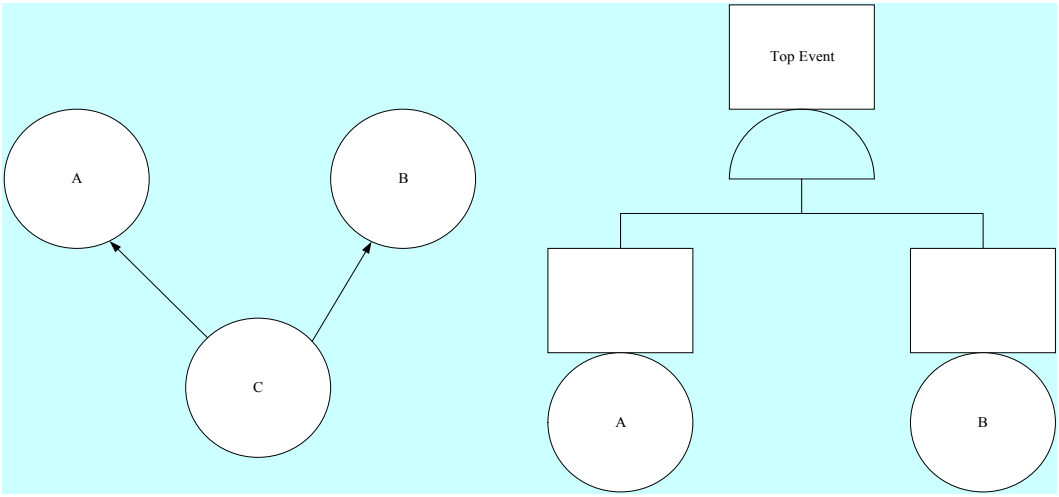


Figure 29: The Bayesian belief network and fault tree for dependency analysis example

P(C) (Prior probability of C)

State of Variable C	P(C)
0	0.99
1	0.01

P(A|C) (Conditional probability table of A)

C	A	P(A|C)
0	0	0.95
0	1	0.05

1	0	0.1
1	1	0.9

P(B|C) (Conditional probability table of B)

C	B	P(B|C)
0	0	0.95
0	1	0.05
1	0	0.60
1	1	0.10

Table 3: Prior and conditional probability tables for dependency analysis example

Only considering the Bayesian belief network itself, after running the Bayesian belief networks inference algorithm, we get $P(\overline{A}) = 0.0585$ $P(\overline{B}) = 0.0535$, and if we do not consider the dependency, the fault tree top event probability will be

P(Top Event) = $0.0585 \times 0.0535 = 0.00312975$

In contrast, if we apply the Hybrid Causal Logic algorithm to eliminate the dependency, the result will be 0.006075. (The same result is obtained when we convert the Bayesian belief network and fault tree in Figure 29 to a single Bayesian belief network.)

Therefore ignoring the dependency can lead to significant errors.

In this research, two HCL algorithms are developed based on different Bayesian belief network inference algorithms. (1) Based on the conditioning algorithm, the *refined conditioning algorithm* converts the combination to a Bayesian belief network and breaks

the loop in order to do the quantification. (2) Based on the variable elimination or junction tree algorithm, the *iterative algorithm* sets the evidence sequentially in the Bayesian belief network; this is done to quantify the posterior probability of the intersectional nodes between the fault tree and Bayesian belief network.

4.2 Role of Binary Decision Diagram in Hybrid Causal Logic Solution

In developing the Hybrid Causal Logic solution procedure, the binary decision diagram method is utilized. The reason for adopting the binary decision diagram is that, in Bayesian belief networks, the inference is probabilistic. That means the negated part also plays an important role in the qualitative analysis. The IF-THEN-ELSE [Rauzy and Dutit, 1997] engine in the binary decision diagram does consider the negated part. This is one reason why we use binary decision diagrams to quantify fault trees in Hybrid Causal Logic diagram. This means the common (intersectional) nodes, which are both in the Bayesian belief network and fault trees/ event sequence diagrams, can be well quantified. Additionally, binary decision diagram has been proven a very efficient way to represent and quantify Boolean logic.

4.3 The Refined Conditioning Method

This new method is based on conditioning algorithm for Bayesian belief networks. The algorithm refines the conditioning algorithm to ensure the message (information/evidence) passes exactly from the Bayesian belief network to fault trees or event trees. The conditioning propagation method in Bayesian belief networks was developed to break a

loop into a polytree [Pearl, 1986-1]. The conditioning method consists of instantiating a variable and thereby blocking the information path. For a directed, acyclic graph over $X = \{A,...Z\}$ we have $P(X) = \sum_{a \in A} P(X, a)$ if A is instantiated to the state a [Pearl, 1981] [Diez, 1996]. The basic approach is to treat the intersectional node in the Bayesian belief network as a cut set, instantiate it, and perform the summation in the intersectional part. In the intersectional part of fault trees (or event sequence diagram) and Bayesian belief networks, the conditioning method will prevent the information from double counting when a message passes from Bayesian belief networks to fault trees or event sequence diagram through a different route. When applying the conditioning method, the conditional independence property has been assumed. After breaking the loop, the procedure is the same as the variable elimination method. The refined conditioning algorithm does not need to convert the entire fault trees and event sequence diagrams to Bayesian belief networks. This improves efficiency and keeps the conventional probabilistic risk assessment infrastructure. The heuristic algorithm is used to set the boundary of summation where the variables in the BDD accept the message (information/evidence) from the Bayesian belief network and have the potential interaction.

For a Bayesian belief network singly connected to a fault tree (the BBN only links to one fault tree basic event), one can simply take the BBN node probability for the state corresponding to the meaning of the event in the fault tree (e.g., failure) as the probability of the basic event of the fault tree.

If the Bayesian belief network is multiply connected to fault trees, the quantification procedure is as follows:

1. *Convert the fault tree to a binary decision diagram.*
2. *Identify the intersectional node position.*
3. *If there is more than one intersectional node, apply Refined Conditioning Algorithm.*
4. *If not, continue Binary Decision Diagram recursive computation.*
5. *END*

The general approach for the Refined Conditioning Algorithm is:

$$P(FT) = \sum_{b \in B} P(FT \mid b_i) P(b_i)$$

Where B is the intersectional node in the Bayesian belief network, $P(FT \mid B)$ is the conditional probability table over the intersectional nodes in $FT\{X, Y, Z...\}$. The detail steps are described below using Figure 30.

The BDD sub-graph $\alpha = ITE(\alpha, \alpha_1, \alpha_0)$, depends on B if α directly depends on B, or α_1, or α_0 depends on B. The Figure 30 shows the refined conditioning algorithm quantification steps. Here, α, α_1, α_0 are in the BDD, and B represents the Bayesian belief network.

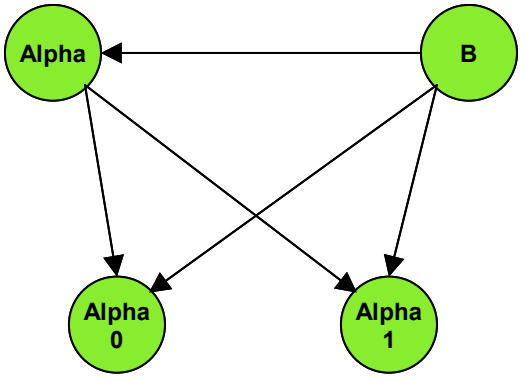


Figure 30: Graph for Illustrating the Refined Conditioning Algorithm

1) Given sub-graph $\alpha, \alpha_1, \alpha_0$ in BDD over B in BBN with $P(X_i \mid Parent(X_i))$ (conditional probability table for the intersectional nodes in binary decision diagram part),

$$F(\alpha, \alpha_1, \alpha_0) = \alpha\alpha_1 + \bar{\alpha}\alpha_0$$

2) If α, or α_1, or α_0 depends on B (B is parent of $\alpha\alpha_1 + \bar{\alpha}\alpha_0$)

   a. Cluster the variables in α, or α_1, or α_0 together to be a single expression

   b. Multiply the conditional probability table into the F expression

   c. Apply the conditioning method to compute respective joint probability

The three steps are expressed through the following equation:

$$P(F(\alpha, \alpha_1, \alpha_0) = \sum_{b \in B} [P((\alpha\alpha_1 + \bar{\alpha}\alpha_0) \mid Parents(\alpha\alpha_1 + \bar{\alpha}\alpha_0))]P(Parents(\alpha\alpha_1 + \bar{\alpha}\alpha_0))$$

$$= \sum_{b \in B} P((\alpha\alpha_1 + \bar{\alpha}\alpha_0), b_i) = \sum_{b \in B} P((\alpha\alpha_1 + \bar{\alpha}\alpha_0) \mid b_i)P(b_i)$$

$$\Pr(F(\alpha, \alpha_1, \alpha_0) = P_{failurestate}(F(\alpha, \alpha_1, \alpha_0)$$

3) Else

$$\Pr(F(\alpha, \alpha_1, \alpha_0) = \Pr(\alpha\alpha_1 + \bar{\alpha}\alpha_0)$$

4) Store the result in the table to α node

5) Recursive F

The above is the Refined Conditioning Algorithm for quantifying the Hybrid Causal Logic. The HCL diagram of Figure 31 is used to illustrate the method. The corresponding probability values are listed in Table 4.

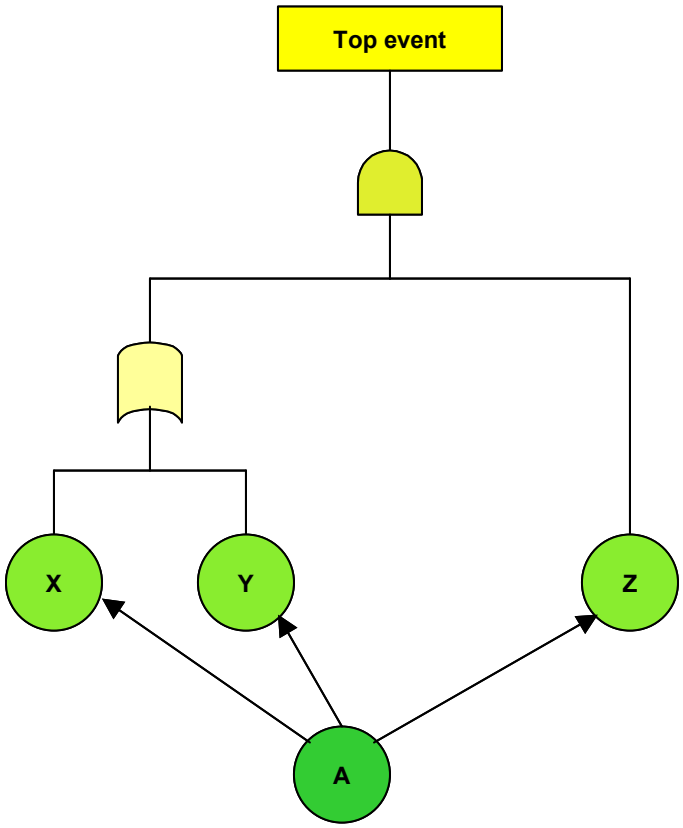


Figure 31: Example HCL to Demonstrate the Refined Conditioning Method

State of A	
0	0.0002
1	0.9998

A	**X**	**P(X|A)**
0	0	0.5
0	1	0.5
1	0	0.1

1	1	0.9

A	Y	P(Y|A)
0	0	0.4
0	1	0.6
1	0	0.2
1	1	0.8

A	Z	P(Z|A)
0	0	0.75
0	1	0.25
1	0	0.25
1	1	0.75

Table 4: Prior and conditional probability table for showing the Figure 31 Refined Conditioning Method Example

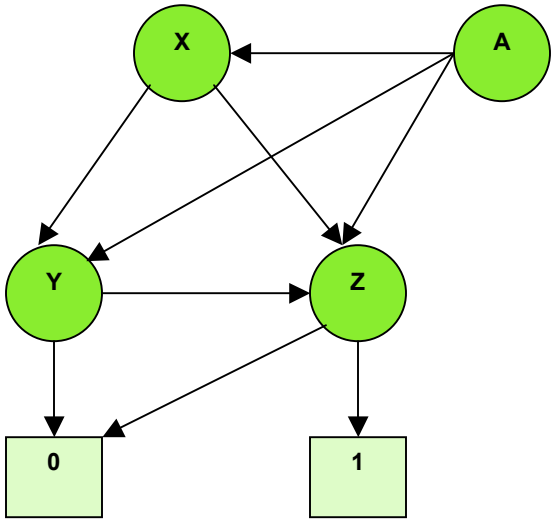


Figure 32: Binary Decision Diagram for the Fault Tree part of Figure 31

The quantification for Figure 31 is as follows:

The Boolean expression for the fault tree is

$$XZ + \overline{X}YZ$$

By applying refined conditioning algorithms, the top event failure probability is:

$$\sum_a (XZ + \overline{X}YZ \mid A)$$
$$= P(x_0 \mid a_0)P(z_0 \mid a_0)P(a_0) + P(x_1 \mid a_0)P(y_0 \mid a_0)P(z_0 \mid a_0)P(a_0)$$
$$+ P(x_0 \mid a_1)P(z_0 \mid a_1)P(a_1) + P(x_1 \mid a_1)P(y_0 \mid a_1)P(z_0 \mid a_1)P(a_1)$$

(Here x_0 indicates False and x_1 indicates True)

Using probability values from Table 4,

$$P(TopEvent) = 0.5 * 0.75 * 0.0002 + 0.5 * 0.4 * 0.75 * 0.0002$$
$$+ 0.1 * 0.25 * 0.9998 + 0.9 * 0.2 * 0.25 * 0.9998$$
$$= 0.070091$$

This result is the same as that obtained by converting the hybrid diagram in Figure 31 to a single Bayesian belief network. Because of the potential interaction between the nodes that receive message (information/evidence) from the Bayesian belief network, it is crucial to group them together to the extent possible near the top in the reduced ordered binary decision diagram (ROBDD). The order is critical in the reduced ordered binary decision diagram [Bryant, 1992][Bartlett and Andrews, 1999]. Improving the variable ordering of ordered binary decision diagrams (OBDDs) is NP-Complete [Bolling and Wegener, 1996]. Some heuristic algorithms, and a comparison of them, have been provided in [Bartlett and Andrews, 2001][Fujita et al., 1988][Minato et al., 1991][Boussiou, 1996] as well as for Bayesian belief networks when applying the variable elimination algorithm.

In Bayesian belief networks, finding an optimal elimination ordering is NP-Complete too [Arnborg et al., 1987], and some heuristic algorithms have been developed, such as minimum deficiency search [Bertel and Brioschi, 1972] and maximum cardinality search [Tarjan and Yannakakis, 1984].

In the variable order process, grouping variables that accept the impact from Bayesian networks in α, or α_1, or α_0 is crucial in order for the summation to work efficiently. In addition, grouping all of these variables in the top is a practical way to improve the summation efficiency and accuracy. This means that if these variables are far away, the recursive searching will take a long time and the summation scale will be very large.

4.4 The Iterative Method

In this algorithm, we first convert fault trees to binary decision diagrams, and apply the IF-THEN-ELSE engine [Rauzy and Dutit, 1997] to get the Boolean logic expression. In the numerical computation, in any Boolean logic expression, if there is more than one event is in the Bayesian belief network, then the second one should take the posterior probability after the first one is set to evidence. The reason is to make them conditionally independent. This must be done iteratively. The reason is that when we perform the quantification in the Hybrid Causal Logic diagram, except for the first intersectional node the following nodes are updated based on the previous nodes being instantiated, as the nodes are not independent. The probability of the visiting node depends on its ancestors' states. When any node is visited, it is instantiated to the fail or success state depending on the binary decision diagram branch to which it belongs. When converting fault trees to

binary decision diagrams, it is also critical to put the intersectional nodes close together in the graph. This improves computational efficiency.

By applying this algorithm to Figure 32, the quantification will be as follows:

$$F = XZ + \overline{X}YZ$$

$$F = XZ_{X=x} + \overline{X}Y_{X=\bar{x}}Z_{X=\bar{x},Y=y}$$

$$\Pr(f) = P(X = x)P(Z = z \mid X = x) + P(X = \bar{x})P(Y = y \mid X = \bar{x})P(Z = z \mid Y = y, X = \bar{x})$$

Using the probabilities listed in Table 4 in the above equation, the same numeric result (0.070091) is obtained.

The above two algorithms provide a heuristic way to quantify the Hybrid Causal Logic diagram. The following section will provide the detail of the quantification procedure that has been implemented in the IRIS software [Groth 2007] for quantification of the Hybrid Causal Logic models.

Compared with the Refined Conditioning Algorithm, from Graph Theory, the iterative algorithm is based on the variable elimination method (Junction tree is its time and space efficient version) and the graph is not divided. In Refined Conditioning algorithm, the graph is divided into multiple graphs at the cut set of the graph, which will break the loop to the polygraph. The strength of the Refined Conditioning algorithm is inherited from the conditioning algorithm in the Bayesian belief networks. The polytree algorithm can be directly applied. The limitation is due to the complexity when cutting the graph to multiple graphs, the optimized path to cut the graph is a NP-problem. The strength of the Iterative algorithm is inherited from the variable elimination algorithm, the graph doesn't have to be cut to multiple graphs, that is space efficient. The limitation is the optimizing

order to eliminate the variable is a NP-problem and requiring more times to do the calculation compared with Refined Conditioning algorithm.

4.5 Hybrid Causal Logic Quantification Algorithm

4.5.1 Computational Rule

The following algorithm is a joint contribution with another researcher, and subject of a US patent application. [Mosleh, Wang, Groen, 2006]

This algorithm uses the binary decision diagram IF-THEN-ELSE engine [Rauzy and Dutit, 1997]. In implementation, variable L is used to provide the stopping rule, where the hybrid recursive will stop to apply the standard binary decision diagram qualification rule.

Define a function *f*, which computes the probability of α conditional on L

$f(\alpha, L) = \Pr(\alpha \mid L)$.

Here *f* can be written in the form of a recursive equation

$$f(\alpha, L) = \Pr(x \mid L) \cdot f(\alpha_1, L \cdot x) + (1 - \Pr(x \mid L)) \cdot f(\alpha_0, L \cdot \overline{x})$$

corresponding to the probabilistic decomposition,

$$\Pr(\alpha \mid L) = \Pr(x \mid L) \cdot \Pr(\alpha_1 \mid L \cdot x) + \Pr(\overline{x} \mid L) \cdot \Pr(\alpha_0 \mid L \cdot \overline{x}).$$

Here, $L \cdot x$ denotes the logical union of L and x. In case α equals 1 (true), *f* returns 1. In case α equals 0 (false), *f* returns 0. When the binary decision diagram is a standard, non-hybrid binary decision diagram, $\Pr(\alpha \mid L) = \Pr(\alpha)$.

	Input Condition (α,L)	**Output $f(\alpha,L)$**
1	$\alpha = 0$	0
2	$\alpha = 1$	1

3	α is non-hybrid, $L \neq \varnothing$	$f(\alpha, \varnothing)$
4	(α, L) in computed table	Pr(α,L) (pre-computed)
5	X is independent	$\Pr(x) \cdot f(\alpha_1, L) + (1 - \Pr(x)) \cdot f(\alpha_0, L)$
6	X is dependent	$\Pr(x \mid L) \cdot f(\alpha_1, L \cdot x) + (1 - \Pr(x \mid L)) \cdot f(\alpha_0, L \cdot \overline{x})$

Table 5: The Hybrid Causal Logic Implementation Rule

Here, if the variable X associated with a binary decision diagram node α is independent; f can be computed more efficiently as

$$f(\alpha, L) = \Pr(x) \cdot f(\alpha_1, L) + [1 - \Pr(x)] \cdot f(\alpha_0, L)$$

taking advantage of the fact that X does not depend on any other variable

$$\Pr(x \mid L) = \Pr(x)$$

nor does any other variable depend on X

$$\Pr(y \mid L \cdot x) = \Pr(y \mid L \cdot \overline{x}) = \Pr(y \mid L)$$

This makes L to include only the necessary conditions. This speeds up the computation by increasing the chance that f(α, L) can return a previously computed probability value. Since in the BBNs nodes often have multiple states, expressions such as $y \mid x$ are not adequate. Instead expressions such as $y = 1 \mid (x = 0, A, B, C, D, E...)$ are more representative. Therefore, in the above equation, x means $X = x$, $\overline{x}$ means $X = \overline{x}$, the variable in L is set to a specific state according to the path where the binary decision diagram goes to this variable X. Those BBN variables not in the path are set to "un-instantiated".

Figure 33 presents an overview of the major procedural steps required for the quantification of Hybrid Causal Logic models.

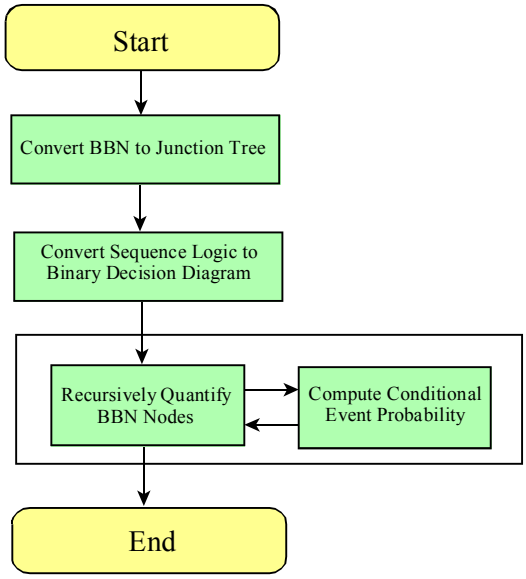


Figure 33: Major procedural steps in the quantification of Hybrid Causal Models

The procedure above does not pose any specific constraints on the variable order in the binary decision diagram. However, it is believed that from a computational standpoint, it is generally advantageous to place dependent variables together in the variable order, as it minimizes the number of binary decision diagram nodes that need to be quantified for multiple conditions L.

An additional ordering applies in case where the binary decision diagram variables are spread over multiple smaller Bayesian belief networks, rather than being connected by a single, large Bayesian belief network. In this case, only the variables within each graph depend on each other. The grouping of such variables in the variable ordering is likely to result in a faster quantification procedure.

For instance, consider a binary decision diagram with variables X, Y, V, W. Let variables X and Y be connected by one Bayesian belief network, and variables V and W be connected by another. If we apply the variable ordering X, Y, V, W, we see that any

binary decision diagram node with variable V or W is not dependent on variables X or Y. When quantifying these nodes, X and Y can thus be removed from the condition L

$$\Pr\left(\alpha_{V,W} \mid L\right) = \Pr\left(\alpha_{V,W} \mid L/\{X, Y\}\right)$$

As explained before, limiting L to include only the necessary conditions increases the chance that a previous probability value can be used, resulting in faster computations. The reduction would however not be possible in case of a variable order such as X, V, W, Y.

Once the binary decision diagram variable order is established, a determination must be made for each position in the order, of the set of variables higher in the order on which the variable in the selected position or any variables lower in the order depend. When computing $f(\alpha|L)$, L is then filtered by removing any variable not in the set. The filtering operation replaces condition 3 in the procedure of

Table **5**.

Stronger filtering, and thus additional speed optimizations, would be possible by determining the filter sets for individual binary decision diagram nodes rather than binary decision diagram variables, but this would significantly increase the memory required to store each binary decision diagram node.

Ordering rules for hybrid structures is a topic of additional investigation, which could focus on ordering the dependent variables such that the number of evaluations of Pr(x | L) is minimized.

4.5.2 Constructing Binary Decision Diagrams in Hybrid Causal Logic Diagram

This section includes a procedure for the construction of binary decision diagrams. The procedure follows well-known procedures [Bryant (1992)].

Let us define a binary decision diagram node $\alpha = \Delta(X_\alpha, \alpha_0, \alpha_1)$ as denoting the statement

$$\text{If } (X_\alpha = x_\alpha) \text{ then } \alpha_0 \text{ else } \alpha_1$$

Nodes α_1 and α_0 are either BDD nodes as defined above, or 0 (false) or 1 (true) terminals. We define the recursive procedure APPLY(α,β,op) for BDD nodes α and β and logical operation op. op can be such operations as AND, OR, and XOR, NOR and NAND. For cases where no trivial solutions exist, i.e., where neither α nor β is a terminal node (true or false), the following rules are used

Input Condition α, β	**APPLY(α, β, op)**
$X_\alpha > X_\beta$	$\Delta\left[X_\alpha, \text{APPLY}\left(\alpha \mid \overline{x}_\alpha, \beta, \text{op}\right) \text{APPLY}\left(\alpha \mid x_\alpha, \beta, \text{op}\right)\right]$
$X_\alpha < X_\beta$	$\Delta\left[X_\beta, \text{APPLY}\left(\alpha, \beta \mid \overline{x}_\beta, \text{op}\right) \text{APPLY}\left(\alpha, \beta \mid x_\beta, \text{op}\right)\right]$
$X_\alpha = X_\beta = X$	$\Delta\left[X, \text{APPLY}\left(\alpha \mid \overline{x}, \beta \mid \overline{x}, \text{op}\right), \text{APPLY}\left(\alpha \mid x, \beta \mid x, \text{op}\right)\right]$

Table 6: Constructing the binary decision diagram including dependent variables

In this table, $X_\alpha > X_\beta$ and $X_\alpha < X_\beta$ refer to an ordering of variables in the model which must be established in advance using an appropriate ordering heuristic. A simple heuristic adds variables to the order as they are encountered when traversing the fault tree.

Furthermore

$$\alpha \mid x_\alpha = \alpha_1 \text{ and } \alpha \mid \overline{x}_\alpha = \alpha_0.$$

A binary decision diagram corresponding to a fault tree is then constructed by successively applying the APPLY operation in order to construct the binary decision diagrams corresponding to all fault tree gates, starting at the lowest-level gates. Each basic event in the fault tree is represented by a corresponding binary decision diagram $\Delta(X, 0, 1)$. Binary decision diagrams corresponding to *k-out-of-n* gates are constructed using the following procedure in Table 7:

For $i = (n - k + 1)$ to 1 step –1
for $j = k$ to 1 step –1
$\beta_{i,j} = \text{Apply}[\text{OR}, \text{Apply}(\text{AND}, \alpha_{i+j-1}, \beta_{i,j+1}), \beta_{i+j,1}]$
where $\beta_{n-k+2,j} = 0$ and $\beta_{i,k+1} = 1$

Table 7: The rule constructing K-out-of-N gates in binary decision diagram

A binary decision diagram need to be created for each risk scenario represented by the occurrence of an initiating event, a sequence of occurrences and negations of pivotal events, and an end state, where fault trees are used to specify the conditions under which the pivotal event do or do not take place. The binary decision diagram representing the scenario logic is then found by combining the binary decision diagram obtained for the fault trees associated with the initiating event and pivotal events according to the logic of the sequence.

4.5.3 Quantification of Dependent Variables

In this discussion, no consideration has so far been given to the computation of conditional probabilities Pr(x|L) for dependent variables. Within the context of the quantification of Hybrid Causal Logic diagram, the quantification of these variables corresponds to the computation of marginal distributions, which is a standard operation for Bayesian belief networks, where L acts as the set of observations for which the network must be conditioned.

The selection of a Bayesian belief network solution method must consider that the Bayesian belief network will need to be solved a high number of times. Furthermore, only minor modifications of L should be expected between subsequent quantifications. For instance, the computation of Pr(x|L) is likely to be followed by a computation of $\Pr(y \mid L \cdot x)$ or $\Pr(y \mid L \cdot \overline{x})$.

The number of quantifications can be reduced by storing computed probabilities in the table of computed values. For this purpose, each dependent variable X can be encoded using a corresponding binary decision diagram node $\alpha_X = \Delta(X, 0, 1)$, such that the conditional probabilities for variables can also be indexed by α and L.

We note however that the procedure of

Table **5** is not limited to the quantification of Hybrid Causal Logic diagrams, and can generally be applied to problems in which the events in the Boolean logic models (variables in the binary decision diagram) are dependent, as long as the conditional marginal probabilities of the dependent variables can be found.

As stated above the quantification procedures described requires the repeated quantification of dependent variables, i.e., those variables in the binary decision diagram

that also appear as variables in the Bayesian belief networks. The quantification of these variables corresponds to the propagation of evidence in a Bayesian belief network in order to compute marginal probability distributions, which is a standard operation in Bayesian belief network theory.

4.5.4 An Example

This example is the same as in the Figure 31. The computational rule is applied step by step here. The graph is re-drawn as in Figure 34 complying with all the computational definition and rules.

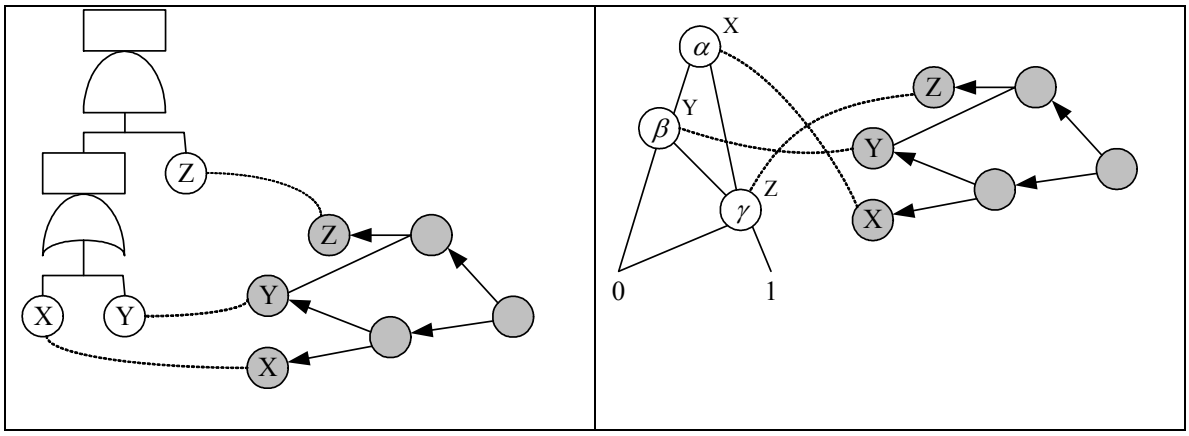


Figure 34: The Hybrid Causal Logic diagram (left) and corresponding hybrid binary decision diagram and Bayesian belief network structure

Quantification of this structure requires the following steps:

$$\Pr(\alpha) = \Pr(\bar{x}) \cdot \Pr(\beta \mid \bar{x}) + \Pr(x) \cdot \Pr(\gamma \mid x)$$
$$\Pr(\beta \mid \bar{x}) = \Pr(y \mid \bar{x}) \cdot 0 + \Pr(\bar{y} \mid \bar{x}) \cdot \Pr(\gamma \mid y, \bar{x})$$
$$\Pr(\gamma \mid y\,\bar{x}) = \Pr(\bar{z} \mid y, \bar{x}) \cdot 0 + \Pr(z \mid y, \bar{x}) \cdot 1$$
$$\Pr(\gamma \mid x) = \Pr(\bar{z} \mid x) \cdot 0 + \Pr(z \mid x) \cdot 1$$

By making the appropriate substitutions, we find that the procedure decomposes Pr(α) into

$$\Pr(\alpha) = \Pr(\bar{x}) \cdot \Pr(y \mid \bar{x}) \cdot \Pr(z \mid y, \bar{x}) + \Pr(x) \cdot \Pr(z \mid x)$$

This equation can be reduced to the expected result

$$\begin{aligned} \Pr(\alpha) &= \Pr(\bar{x}\, y\, z) + \Pr(x\, z) \\ &= \Pr(\bar{x} y\, z + x\, z) \end{aligned}$$

even though variables *X*, *Y*, and *Z* are dependent. The conditional marginal probabilities $\Pr(\bar{x}), \Pr(y \mid \bar{x}), \Pr(z \mid y\,\bar{x}), \Pr(x), \text{and } \Pr(z \mid x)$ follow from the Bayesian belief network (X, Y, Z, A).

Based on these probabilities, the top event probability is computed as

$$\begin{aligned} \Pr(\alpha) &= \Pr(\bar{x} y z + x z) \\ &= \Pr(\bar{x}) \cdot \Pr(y \mid \bar{x}) \cdot \Pr(z \mid y, \bar{x}) + \Pr(x) \cdot \Pr(z \mid x) \\ &= 0.89992 \times 0.200022 \times 0.250111 + 0.10008 \times 0.2505 \\ &= 0.070091 \end{aligned}$$

$\Pr(\bar{x})$	0.89992
$\Pr(y \mid \bar{x})$	0.200022
$\Pr(z \mid y, \bar{x})$	0.250111
$\Pr(x)$	0.10008
$\Pr(z \mid x)$	0.2505

Table 8: Conditional probability obtained from the corresponding Bayesian belief network (X, Y, Z, A)

This result is the same as that obtained by converting the Hybrid Causal Logic diagram to a single Bayesian belief network (Figure 35) with the prior and conditional probability in Table 9.

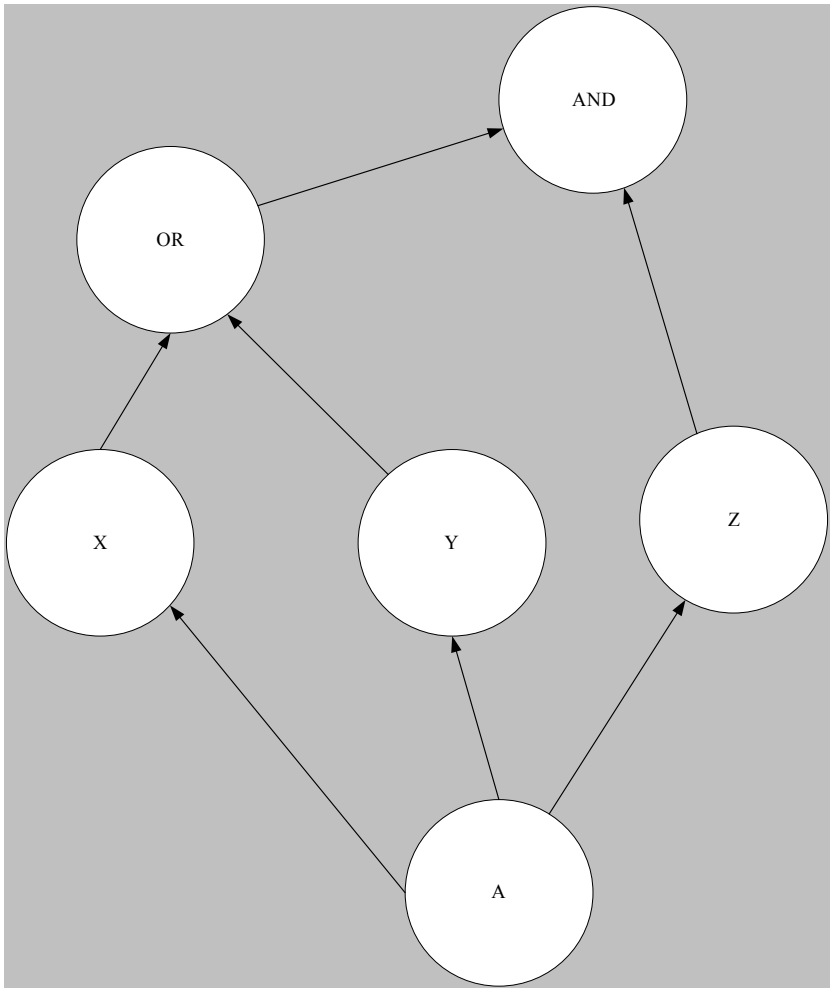


Figure 35: The single Bayesian belief network converted from Figure 31

A	P(A)
0	2.0E-4
1	0.9998

A	X	P(X|A)
0	0	0.5
0	1	0.5
1	0	0.1
1	1	0.9

A	Y	P(Y|A)
0	0	0.4
0	1	0.6
1	0	0.2
1	1	0.8

A	Z	P(Z|A)
0	0	0.75
0	1	0.25
1	0	0.25
1	1	0.75

X	Y	OR	P(OR|X,Y)
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

X	Y	AND	P(AND|X,Y)
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Table 9: The prior and conditional probability table for the Bayesian belief network in Figure 35

4.6 Hybrid Causal Logic Quantification Algorithm with Evidence Observed

In the previous section, the quantification algorithm for the Hybrid Causal Logic was described. In this section, we extend the procedure for propagation of the impact of observing evidence in a Bayesian belief network node of the Hybrid Causal Logic diagram.

Algorithm:

1. Apply the junction tree algorithm to the Bayesian belief network.
    a) Moralize: Pair all the parents and remove all the arrows.
    b) Triangulate: Add arcs to the nodes to give every cycle with more than four nodes a chord/arc.
    c) Identify cliques and separators.
    d) Build the junction tree.
    e) Initialization: Each clique or separator in the junction tree has an associated table over the configurations of its constituent variables. For each node X, choose one clique Y in the junction tree that contains X and all of X's parents, Multiply $P(X \mid Parents(X))$ on Y's table.
    f) Evidence entry: Evidence is added and propagated into the junction tree.
    g) Marginalize and normalize the variable.
2. After updating the Bayesian belief network, save the parameters of each variable for use in the next step.
3. Keep the evidence in the Bayesian belief network and run the Hybrid Causal Logic diagram quantification algorithm.

Example: Consider the Hybrid Causal Logic diagram of Figure 36. The binary decision diagram of the fault tree portion is shown in Figure 37. The probability values of the BBN part of the Hybrid Causal Logic are listed in Table 10.

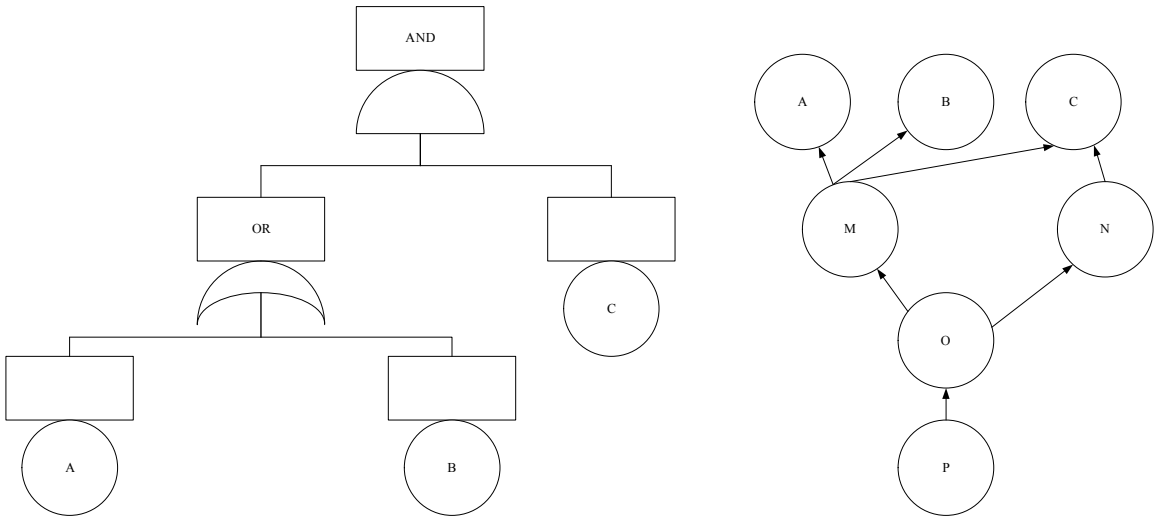


Figure 36: Hybrid Causal Logic example used for illustrating evidence propagation procedure

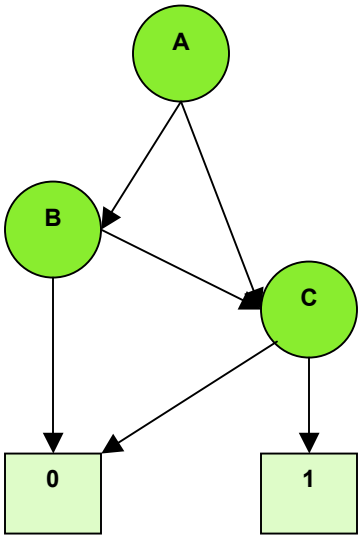


Figure 37: Binary decision diagram for the fault tree portion of the example Hybrid Causal Logic diagram

State of P	
0	0.1
1	0.9

State of P	State of O	P(O|P)
0	0	0.8
0	1	0.2
1	0	0.6
1	1	0.4

State of O	State of M	P(M|O)
0	0	0.8
0	1	0.2
1	0	0.3
1	1	0.7

State of O	State of N	P(N|O)
0	0	0.9
0	1	0.1
1	0	0.2
1	1	0.8

State of M	State of N	State of C	P(C|M,N)
0	0	0	0.9
0	0	1	0.1
0	1	0	0.6
0	1	1	0.4
1	0	0	0.7
1	0	1	0.3
1	1	0	0.1
1	1	1	0.9

State of M	State of B	P(B|M)
0	0	0.7
0	1	0.3
1	0	0.2
1	1	0.8

State of M	State of A	P(A|M)
0	0	0.6
0	1	0.4
1	0	0.3
1	1	0.7

Table 10: Prior and conditional probability tables for HCL of figure 36

Evidence Scenario 1: The evidence is observed on the root nodes or the intermediate nodes between the fault tree nodes and root nodes. Set evidence as N= 1, P= 0 which means that the probability that N is in state 1 is 1, and probability that P in state 0 is 1.

By applying the binary decision diagram IF-THEN-ELSE engine, the equation for the BDD in Figure 37 will be:

$$AC + \overline{A}BC$$

In the multiple states and probabilistic context the above expression translates into

$$\begin{aligned}&P(A=1\,|\,B,C,M,N=1,O,P=0)\times P(C=1\,|\,A=1,B,M,N=1,O,P=0)\\&+P(A=0\,|\,B,C,M,N=1,O,P=0)\times P(B=1\,|\,A=0,C,M,N=1,O,P=0)\\&\times P(C=1\,|\,A=0,B=1,M,N=1,O,P=0)\end{aligned}$$

After adding the evidence in the Bayesian belief networks, and based on the values in Table 10 we find

$$\begin{aligned}&P(A=1\,|\,B,C,M,N=1,O,P=0)=0.56\\&P(C=1\,|\,A=1,B,M,N=1,O,P=0)=0.733\\&P(A=0\,|\,B,C,M,N=1,O,P=0)=0.44\\&P(B=1\,|\,A=0,C,M,N=1,O,P=0)=0.482\\&P(C=1\,|\,A=0,B=1,M,N=1,O,P=0)=0.702\end{aligned}$$

$$\begin{aligned}&AC+\overline{A}BC\\&=0.56\times 0.733+0.44\times 0.482\times 0.702\\&=0.559\end{aligned}$$

This result is the same as when the whole HCL diagram is converted to the equivalent BBN in Figure 38.

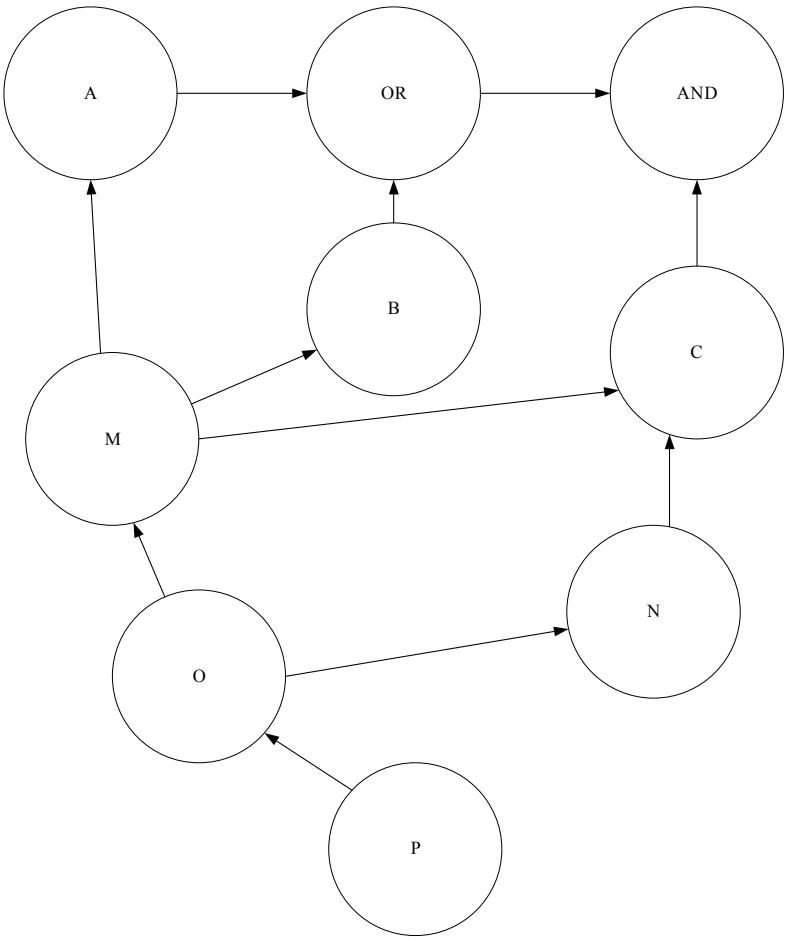


Figure 38: The corresponding Bayesian belief network

Evidence Scenario 2: The evidence is observed on the intersectional nodes, which are the basic events in the fault tree and a node in the BBN at the same time. We set the evidence as C=1, and M=0, which means the probability of C being in state 1 is 1, and the probability of M being in state 0 is 1.

By applying the binary decision diagram IF-THEN-ELSE engine, the equation for the BDD in Figure 37 will be

$$AC + \overline{A}BC$$

In the multiple states and probabilistic context, it is really

$$P(A=1\mid B,C=1,M=0,N,O,P)\times P(C=1\mid A=1,B,M=0,N,O,P)$$
$$+P(A=0\mid B,C=1,M=0,N,O,P)\times P(B=1\mid A=0,C=1,M=0,N,O,P)$$
$$\times P(C=1\mid A=0,B=1,M=0,N,O,P)$$

After adding the evidence to the BBN we find

$$P(A=1\mid B,C=1,M=0,N,O,P)=0.4$$
$$P(C=1\mid A=1,B,M=0,N,O,P)=1$$
$$P(A=0\mid B,C=1,M=0,N,O,P)=0.6$$
$$P(B=1\mid A=0,C=1,M=0,N,O,P)=0.3$$
$$P(C=1\mid A=0,B=1,M=0,N,O,P)=1$$

$$AC+\bar{A}BC$$
$$=0.4\times 1+0.6\times 0.3\times 1$$
$$=0.58$$

Again, the result is the same as for the case where the HCL diagram is converted to the BBN of Figure 38 and inference is ran with the corresponding evidence.

Some specific cases:

The process of adding evidence can lead to some conflicts. For example, in some cases, when evidence is set on a node, its child will be specified. The child will not be allowed to be given any evidence. In this situation, when the Hybrid Causal Logic diagram quantifies, the value taken from the child variable will be the same value as specified by its parents.

5. Importance Measure in Hybrid Causal Logic

5.1 Importance Measurement

By using importance measures PRA analyst can rank scenarios and identify the most important contributors in order to improve reliability or to reduce risk. Because of the dependency brought about by the Bayesian belief networks (common BBN nodes shared by FT basic events of ESD pivotal events, the conventional importance definition needs to be redefined in the Hybrid Causal Logic scenario and the measurement method needs to be redeveloped. Two conventional importance measures are adopted here. One is the risk achievement worth (RAW) [Modarres et. Al, 1999]; the other is Vesely-Fussel importance factor (VF) [Fussel 1975].

Risk achievement worth is also called risk increase factor. RAW measures the increase in system failure probability assuming failure of a component. It is an indicator of the importance of maintaining the current level of reliability for the component. RAW is defined as:

$$RAW(S, e) = \frac{\Pr(S \mid e)}{\Pr(S)}$$

Vesely-Fussel importance factor [Fussel 1975] indicates the part of the total system failure probability that involves the component failure. VF is defined as:

$$VF(S, e) = \Pr(e \mid S)$$

By applying fundamental probability theory

$$VF(S, e) = \frac{\Pr(S \cdot e)}{\Pr(S)} = \frac{\Pr(S \mid e)\Pr(e)}{\Pr(S)}$$

5.1.1 Procedure for Risk Achievement Worth Quantification in Hybrid Causal Logic

In the following procedure, we need to traverse the Hybrid Causal Logic diagram twice to obtain the importance measures of a given component *e*. In this process, the dependency caused by the BBN nodes is eliminated by applying the following procedure. The first step is to compute the top event probability from the basic events probabilities, which is done in a binary decision diagram by applying the IF-THEN-ELSE engine. If the basic events are in a BBN, the Bayesian belief network inference algorithm is called and the dependency is eliminated by applying the Hybrid Causal Logic quantification algorithms described in previous sections. . The quantification is done from bottom to top.

In this importance measurement procedure, the dependency of the dependent variables (referring to the variables in both the BBN and fault trees or event sequence diagrams) must be counted between themselves and their children.

In Bayesian belief networks, the assumption that the probability of "failed state" of one node is equal to 1 automatically impacts children nodes due to the conditional probability relationship between the node and its children. The probabilities of children nodes will be changed. This needs to be taken into account in Hybrid Causal Logic because this change will influence HCL output results. It is counted in the computation when quantifying the $\Pr(S \mid e)$.

The methodology is, first of all, to always set the component failing state equal to 1; then run the BBN inference algorithm one time and put the probabilities of each node in the array for the next use of the Hybrid Causal Logic diagram traverse.

The procedure is summarized in the following:

1. Find Pr(S) by traversing the Hybrid Causal Logic diagram once.
2. Assuming component *e* fails, if *e* is in the Bayesian belief network part of the HCL, assume *e* failing state probability is 1; propagate the BBN one time and store the intermediate probability in an array for use in the second Hybrid Causal Logic diagram traverse; marking this variable *e*/component *e* to keep its failing state in the second Hybrid Causal Logic diagram quantification traverse; if not, go directly to the next step.
3. Find $\Pr(S \mid e)$ by traversing the Hybrid Causal Logic diagram again.
4. The risk achievement worth measurement of component e is then calculated as $\frac{\Pr(S \mid e)}{\Pr(S)}$.

5.1.2 Procedure for Vesely-Fussel Importance Factor Quantification in HCL

For this measure, we need to traverse the Hybrid Causal Logic diagram twice to obtain the importance measures of component *e*. As with risk achievement worth quantification, the dependent variables referring to the variables in the Bayesian belief network must be considered first. The solution is to first run the Bayesian belief network inference algorithm one time and then put the probabilities of each node in the array for the next Hybrid Causal Logic diagram traverse.

The Vesely-Fussel importance measurement procedure is summarized in following:

1. Find Pr(S) by traversing Hybrid Causal Logic diagram one time.

2. Assuming component *e* fails, if e is in the Bayesian belief network, and assuming component *e* failing state probability is 1, propagate the Bayesian belief network one time and store the intermediate probability in an array for use in the second Hybrid Causal Logic diagram traverse, marking this variable *e*/component *e* to keep its failing state in the second Hybrid Causal Logic diagram quantification traverse; if not, go directly to next step.
3. Find $\Pr(S \mid e)$ by traversing Hybrid Causal Logic diagram again.
4. The Vesely-Fussel importance factor of component *e* is given by $\frac{\Pr(S \cdot e)}{\Pr(S)}$.

5.2 Example

Consider a fault tree including five basic events A, B, C, D, E; two of the events D, E are modeled by a Bayesian belief network (Figure 39). Table 11 is the conditional probability table for the Bayesian belief network. Table 12 is the analysis result.

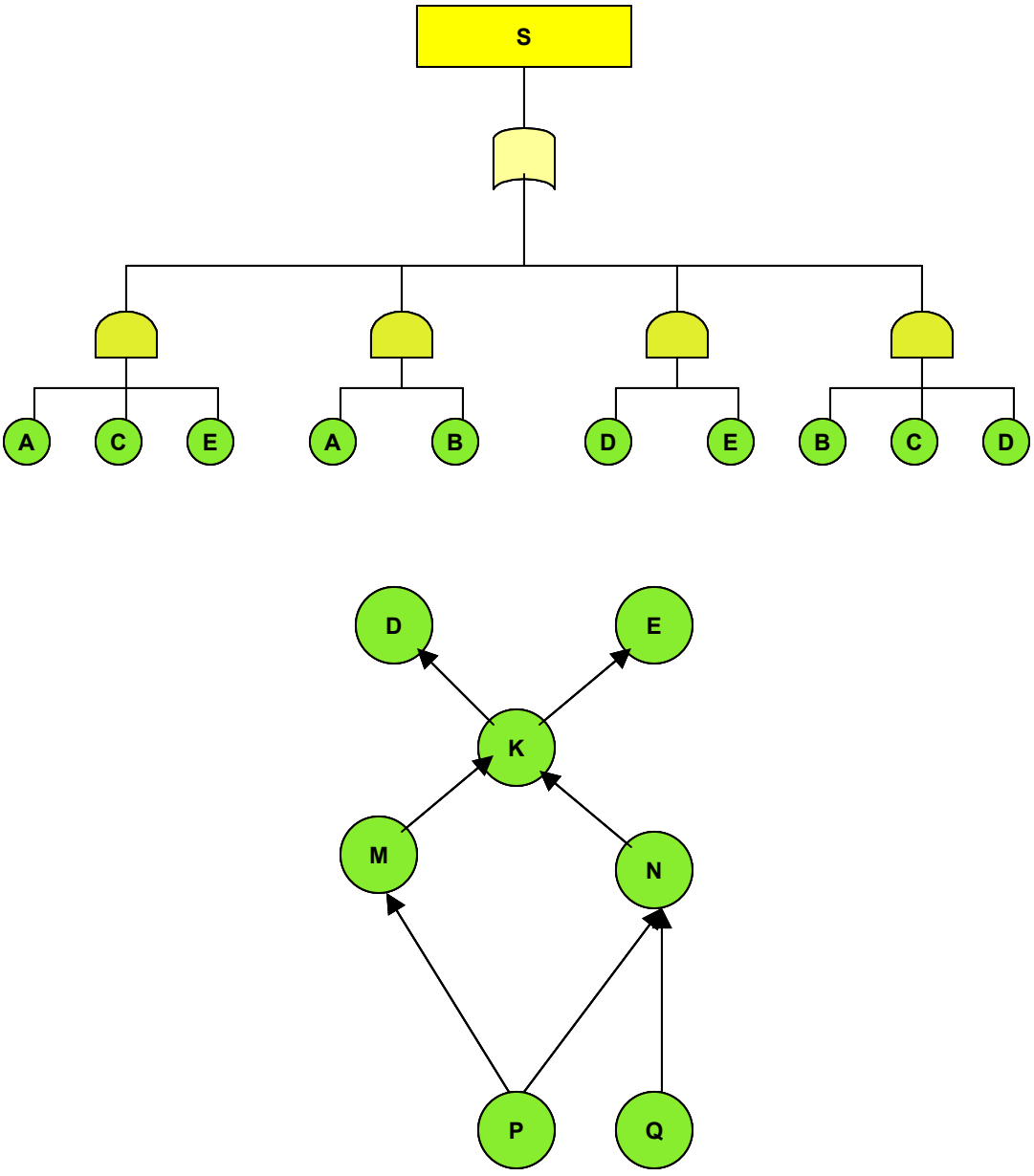


Figure 39: The Hybrid Causal Logic for Importance Measure Example

e	Pr
A	0.1
B	0.2
C	0.3

Prior probability of node P

T	0.2
F	0.8

Conditional probability of node M

P	T	F
T	0.8	0.75
F	0.2	0.25

Prior probability of node Q

T	0.1
F	0.9

Conditional probability of node N

P	T		F	
Q	**T**	**F**	**T**	**F**
T	0.9	0.65	0.55	0.1
F	0.1	0.35	0.45	0.9

Conditional probability of node K

M	**T**		**F**	
N	**T**	**F**	**T**	**F**
T	0.99	0.6	0.3	0.02
F	0.01	0.4	0.7	0.98

Conditional probability of node D

K	T	F
T	0.95	0.2
F	0.05	0.8

Conditional probability of node E

K	T	F
T	0.8	0.1
F	0.2	0.9

Table 11: The Conditional Probability Table for the BBN of Figure 39

e	**Pr(S|e)**	**V-F**	**RAW**
A	0.506689	0.145296	1.45296

B	0.410566	0.235464	1.17732
C	0.366413	0.315214	1.05071
D	0.86096	0.951995	2.46885
E	0.654419	0.963118	1.87658
K	0.73424	0.942137	2.10548
M	0.674701	0.464339	1.93474
N	0.415021	0.891384	1.1901
P	0.380103	0.871975	1.08997
Q	0.359118	0.926812	1.02979

Table 12: The Importance Measurement Analysis Result

Table 12 gives the importance measurement results for all the components in the Hybrid Causal Logic diagram of Figure 39.

6. The Safety Performance Indicator in Hybrid Causal Logic

6.1 Definition

Various elements of an HCL risk model can be viewed as Safety Performance Indicator (SI). For example failure of a major system (pivotal event of an ESD), or failure of a component of that system (a basic event of the corresponding FT), or poor maintenance quality (as basic node of a BBN that covers human and organizational causal factors) could used as SPIs . Normally the frequency of observing these events can be viewed as an indicator of level of safety or risk. The level of risk can be obtained as the product of the frequency of the SI and the conditional probability an undesired end state of the system given the occurrence of that SI.

Therefore safety performance indicators can be a Bayesian belief network root node, a fault tree basic event, a fault tree intermediate event, a fault tree top event, or an event sequence diagram pivotal event. In the Hybrid Causal Logic diagram, the safety performance indicator is selected by the user or decision maker according to some considerations such as ease of monitoring the frequency of the SI.

As stated earlier, each safety performance indicator has a frequency and risk weight value. The frequency is input by the user when building the diagram. The risk (in terms of a given end state S) for an individual safety performance indicator is defined as

$$Risk = \Phi(SI_N)\Pr(S \mid SI_N)$$

where

$\Phi(SI_N)$ denotes the frequency of the risk indicator SI_N observed, and $\Pr(S \mid SI_N)$ is the risk weight defined as the conditional probability of a selected end state category given the occurrence of the safety performance indicator--- $\Pr(S \mid SI_N)$. Here S denotes a selected end state or end states category, and SI_N is a specified indicator.

If S has more than one end state, then risk level is calculated as the sum of the risk weights given each single end state:

$$\Pr(S \mid SI_N) = \sum_{i=1}^{k} \Pr(S_i \mid SI_N)$$

where k indicates the number of end states.

The total risk is defined as the sum of the multiple single indictors' risks:

$$Total\,Risk = \sum_{1}^{m} \Phi(SI_m) P(S \mid SI_m)$$

In a single event sequence diagram, because the end states are exclusive, the total risk weight is the exact approach. If the end states category includes multiple event sequence diagrams, this summation works based on the rare events assumption for each single end state in this category.

6.2 Risk Weight Quantification for Different HCL Elements

The safety performance indicator can be the basic event in the fault tree, a node in the Bayesian belief networks, a pivotal event in the event sequence diagram, or an intermediate event in the fault tree.

Fault Tree Basic Event as Safety Performance Indicator: When the safety performance indicator is a basic event, the risk weight calculation is the same as

the conditional probability $P(S \mid e)$ calculation in the importance measure. By assuming $\Pr(SI_N) = 1$ and traversing the Hybrid Causal Logic diagram one time, the probability will be the conditional probability or risk weight of $\Pr(S \mid SI_N)$. (In the IRIS software code, this conditional probability is pre-calculated and stored for the use in importance measure and risk indicator quantification.)

Bayesian Belief Networks Node as Safety Performance Indicator:

When the safety performance indicator is a Bayesian belief network node, the risk weight is defined as: $\Pr(S \mid SI_{state\,x} = 1)$, where $SI_{state\,x} = 1$ means SI is instantiated to a specific state ($state\,x$).The reason is that the nodes in Bayesian belief networks can have multiple states.

The quantification procedure for this risk weight is:

1. Instantiate the node to a specific state in the Hybrid Causal Logic diagram
2. Run the standard Bayesian belief network and store the result for the use in Step 4
3. Select and keep that risk indicator in the specific state
4. Run the standard Hybrid Causal Logic diagram quantification procedure and using the result from Step 2

When the basic event in the fault tree is the intersectional node between the fault tree and the Bayesian belief networks, the quantification of the risk weight is the same as for the nodes in the Bayesian belief network.

Fault Tree Intermediate Event as Safety Performance Indicator:

When the safety performance indicator is a fault tree intermediate event or a gate event, the safety performance indicator is still defined as:

$Risk = \Phi(SI_N)\Pr(S \mid SI_N)$ where

1. SI_N is the intermediate event or a top event in the fault tree
2. $\Phi(SI_N)$ is the frequency observed for that event

The risk weight is calculated through the fundamental probability theorem

$$\Pr(S \mid SI_N) = \frac{\Pr(S \cdot SI_N)}{\Pr(SI_N)}$$

In the hybrid causal diagram application, $S \cdot SI_N$ is graphically done through the standard binary decision diagram AND operation.

Note that in this kind of safety performance indicator, the quantity $\Pr(SI_N)$ calculated from the model, should match the frequency $\Phi(SI_N)$.

Pivotal Event as Safety Performance Indicator: When the safety performance indicator is a pivotal event of an event sequence diagram, if it is linked to fault tree top event, the computation is the same as in the case of fault tree intermediate or top event. If the pivotal event is linked to the Bayesian belief networks, the computation is the same as in the case of Bayesian belief networks node.

7. Uncertainty Propagation in Hybrid Causal Logic Diagram

7.1 Introduction

The BBN probability values conditions could be subject to uncertainty themselves. Uncertainty propagation for Hybrid Causal Logic diagram is discussed in this Chapter. An example of uncertainty propagation with sampling from distributions of fault tree basic event probabilities, event sequence diagram pivotal event probabilities, and BBN prior and conditional probabilities is presented. The Dirichlet distribution is proposed as an appropriate distribution to model the prior and conditional probability distribution for may BBN applications.

The uncertainty in Hybrid Causal Logic diagram can also come from structures or variables and the BBN model (Model Uncertainty). However structural uncertainty assessment is not in the scope of this research.

7.2 Uncertainty Distributions

The Dirichlet distribution is the continuous multivariate probability distributions parameterized by the vector α of nonnegative reals. It is the multivariate generalization of the Beta distribution

$f(x;\,\alpha) \sim \prod_{i=1}^{K} x_i^{\alpha_i - 1} \delta(1 - \sum_{i=1}^{K} x_i)$ Where $\alpha = (\alpha_1, ..., \alpha_K)$ is a parameter vector with $\alpha_i \geq 0$

This distribution can be used for all discretized BBN nodes as the joint probability distribution over the domain of all variables and all conditional distributions.

7.3 Uncertainty Sampling

The uncertainty propagation will be the problem of how to propagate Dirichlet distributions in the BBN part of the Hybrid Causal Logic diagram. Uncertainty propagation is done using Monte Carlo sampling. It is especially useful for complex, nonlinear cases or cases involving many uncertain parameters, such as Bayesian belief networks. The procedures follow:

1) Create a Hybrid Causal Logic diagram.
2) Generate random values for input avriables:
   - prior probabilities of Bayesian belief network root nodes.
   - conditional probabilities of Bayesian belief networks.
   - probabilities of fault tree and event sequence diagram basic events that are not dependent on Bayesian belief networks nodes.
3) Evaluate the Hybrid Causal Logic diagram (e.g., calculate the probabilities of the end states).
4) Repeat step 2 and 3 enough times for desired accuracy.

The methodology for generating Dirichlet random numbers is to sample a random vector $x = (x_1, ..., x_K)$ here $\sum_{i=1}^{K} x_i = 1$ from K-dimensional Dirichlet distribution with

parameters $(\alpha_1, ..., \alpha_K)$. The first step of the procedure is to draw K independent random samples $y_1, ... y_K$ from Gamma distributions each with density $\frac{y_i^{\alpha_i - 1} e^{-y_i}}{\Gamma(\alpha_i)}$ and then set $x_i = \frac{y_i}{\sum_{j=1}^{K} y_i}$.

If we set $\alpha_0 = \sum_{i=1}^{K} \alpha_i$, then the random variables $x_1, ... x_K$ are $\frac{\alpha_1}{\alpha_0}$, …, $\frac{\alpha_K}{\alpha_0}$ respectively.

The approach here is to start from the uniform distribution and generate a random gamma distribution number; from the gamma distribution number we get the random Dirichlet distribution number. Ahrens and Dieter have shown an efficient method to generate the Gamma random number [Ahrens and Dieter, 1974] [Ahrens and Dieter, 1982]. We use this (from Randlib C language library.)

The following is an example of the procedure for uncertainty propagation in a Hybrid Causal Logic diagram. All variables are assumed to be binary. This will simplify the Dirichlet distribution to Beta distribution in the Bayesian belief networks. It includes distributions of prior probability of the Bayesian belief network nodes, distributions of the conditional probability table, and the distributions of the fault tree basic event probabilities and the event tree initiator probability.

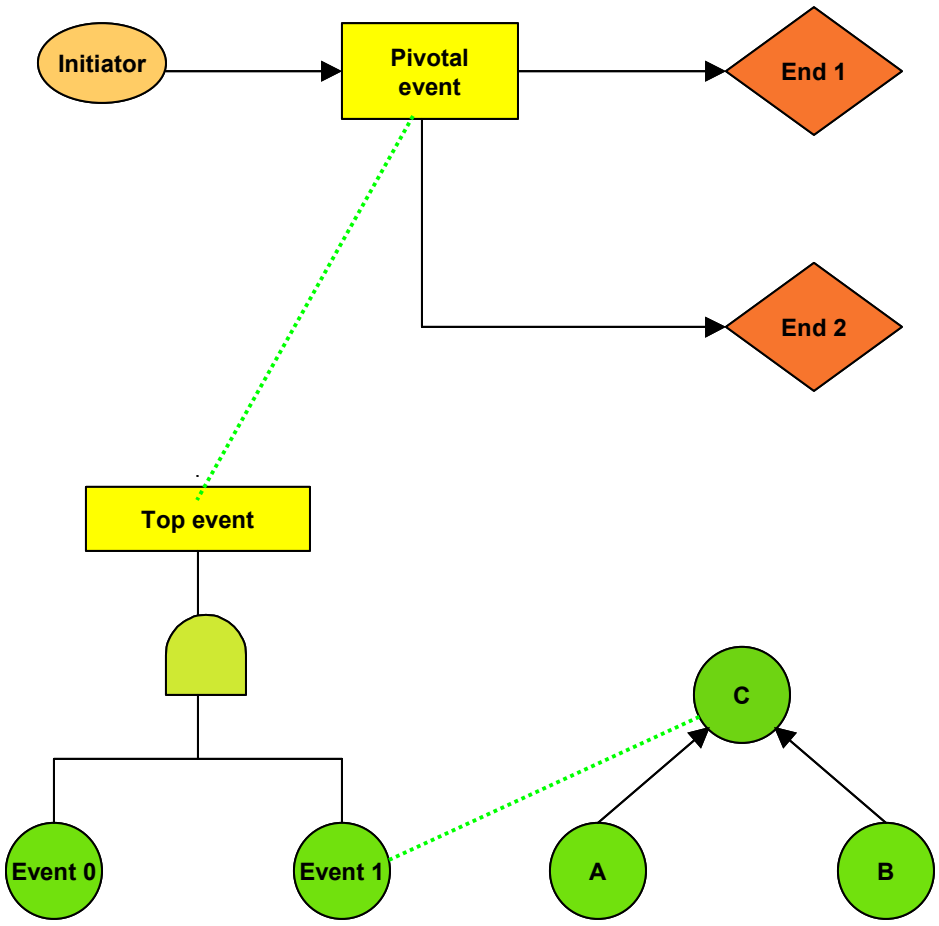


Figure 40: Hybrid Causal Logic uncertainty propagation for numerical example

The **mean values** of the prior probabilities and conditional probabilities are provided in Table 13:

Node A:

0	0.3
1	0.7

Node B:

0	0.2
1	0.8

Node C:

A	**B**	**C**	**P(C|A,B)**
0	0	0	0.9
0	0	1	0.1
0	1	0	0.6
0	1	1	0.4
1	0	0	0.45
1	0	1	0.55
1	1	0	0.2
1	1	1	0.8

Table 13: The mean values of the distributions of the prior and conditional probabilities of the uncertainty propagation example

The uncertainty distribution of the probability of the fault tree basic event 0 is a Beta distribution $\beta(1,7)$. Basic event 1 is linked to node C in the Bayesian belief network. The uncertainty distribution of the probability of the initiator is a Beta distribution $\beta(1,9)$. The pivotal event PE is linked to the fault tree top event.

Several of the input distributions and intermediate distribution resulting from uncertainty propagation are plotted in Figure 41, Figure 42, Figure 43, Figure 44.

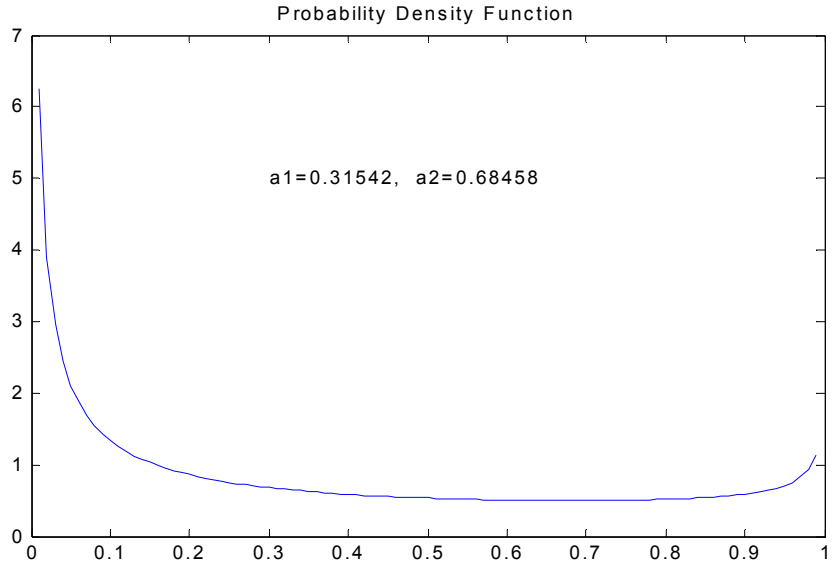


Figure 41: Probability density function plotting for state A_0 of node A

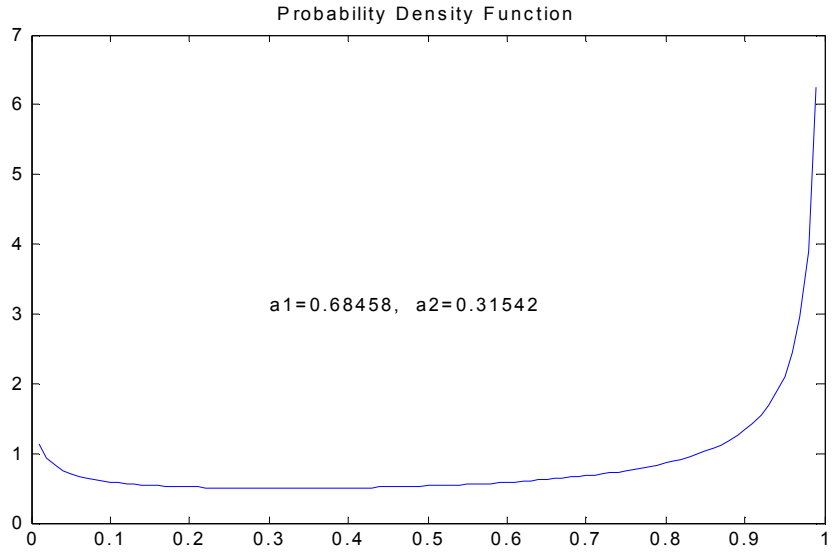


Figure 42: Probability density function plotting for state A_1 of node A

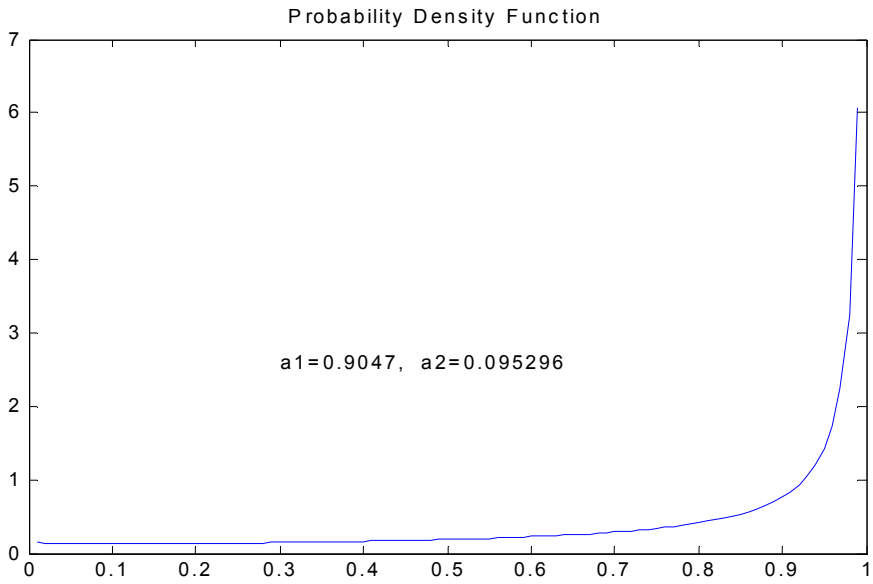


Figure 43: Probability density function plotting for conditional probability $C_0|A_0B_0$

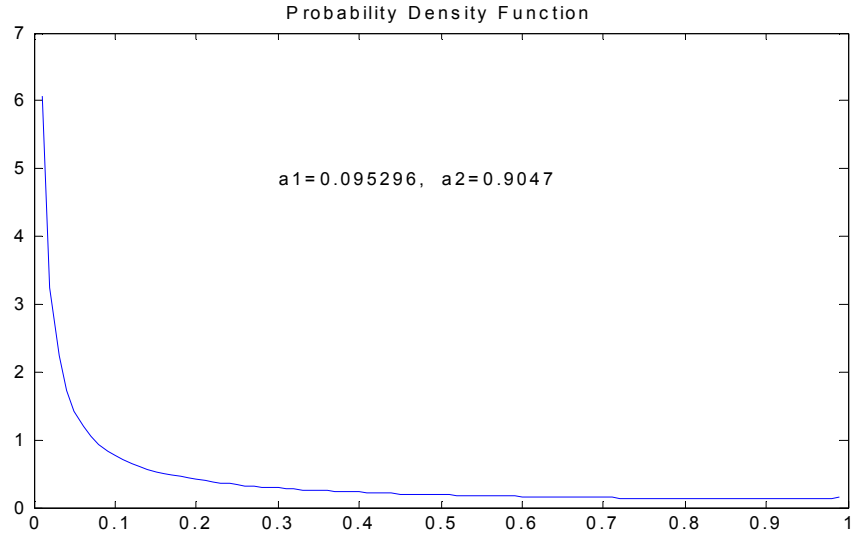


Figure 44: Probability density function plotting for conditional probability $C_1|A_0B_0$

Figure 45 is a plot of the distribution of State C1 of event C. The mean values obtained through uncertainty propagation are compared against the point estimate results in Table 14 and Table 15.

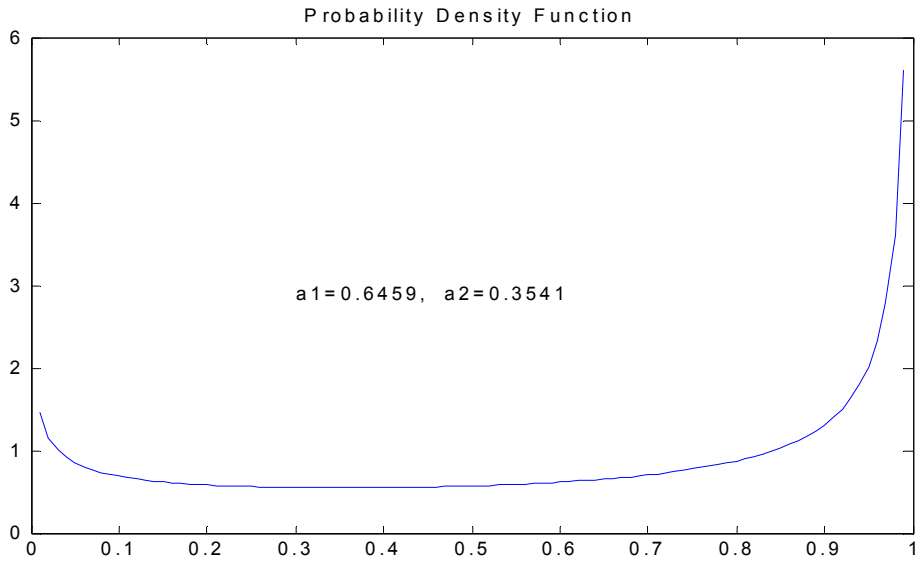


Figure 45: Probability density function plotting for state C_1

	Point Estimation	Mean of Distribution
A_0	0.3	0.315418
A_1	0.7	0.684582
B_0	0.2	0.218211
B_1	0.8	0.781789
$C_0\|A_0B_0$	0.9	0.904704
$C_1\|A_0B_0$	0.1	0.095296
$C_0\|A_0B_1$	0.6	0.634658
$C_1\|A_0B_1$	0.4	0.365342
$C_0\|A_1B_0$	0.45	0.488883
$C_1\|A_1B_0$	0.55	0.511117

$C_0\|A_1B_1$	0.2	0.198867
$C_1\|A_1B_1$	0.8	0.801133
Basic Event 0	0.125	0.124236
Initiator	0.1	0.101249

Table 14: The sampling data and the point estimation data

	Result from Point Estimation	Mean Values from Uncertainty Propagation
C_0	0.373	0.3541
C_1	0.627	0.6459
Probability of End State 1	0.0046	0.00454

Table 15: Comparison of results based on point estimates and mean values based on propagation of the uncertainty distributions

8. Qualitative-Quantitative Bayesian Belief Networks in Hybrid Causal Logic

8.1 Introduction

The Bayesian belief network is a powerful tool not only for graphically representing the relationships among a set of variables, but also for dealing with uncertainties in the domain. However even with this flexibility there are many situations where it is not possible to get numerical values for the probabilities of basis variables and conditional probabilities of the BBN model. In such cases subject matter experts may be more willing to express their opinions using qualitative scales. Also most survey instruments and assessment methods often used in social sciences and human behavior disciplines are based on qualitative scales. Therefore in most applications some BBN nodes and conditional relations can be expressed quantitatively, while others require a qualitative measure. The Qualitative-Quantitative Bayesian belief network (QQ_BBN) provides a solution. A QQ-BBN is a belief network where some of the model likelihoods are expressed with qualitative measures (e.g, High, Medium, Low) and others are based on probability scales. A set of likelihood propagation rules and inference procedures enables the mixed likelihood types to work together in a single BBN.

8.2 QQ-BBN Analysis Methodology

The qualitative-quantitative Bayesian belief network approach is accomplished in two ways. One way is through a "qualitative algebra" ---the prior and the conditional

probabilities are treated as symbols, or some specific words used to scale the subject of the assessment (e.g., impossible, unlikely, fair, probable, certain). The BBN inference is done through operations on symbols. The inference will stay in this symbolic set. Examples of the "qualitative algebra" needed are listed in Table 16 and Table 17.

	Low	**Medium**	**High**
Low	Low	Medium	High
Medium	Medium	Medium	High
High	High	High	High

Table 16: A Qualitative Addition Rule

	Low	**Medium**	**High**
Low	Low	Low	Medium
Medium	Low	Medium	Medium
High	Medium	Medium	High

Table 17: A Qualitative Multiplication Rule

An alternative is to assign a discrete value to the qualitative scales. For example the qualitative terms {impossible, little, fair, probable, certain} would be converted into numerical values such as {0, 0.3, 0.5, 0.7, 1}. Once this is done the usual BBN inference algorithm can be applied.

Example 1: A qualitative inference example is shown using Figure 46:

Figure 46: Qualitative Inference Example

$$P(Flue = present) = low$$
$$P(Fever = high \mid Flu = present) = high$$
$$P(Fever = normal \mid Flu = absent) = high$$

After the propagation, the posterior probability for "fever" will be

$$\begin{aligned} & P(Fever = high) \\ & = P(Fever = high \mid Flu = present) \times P(Flu = present) \\ & + P(Fever = high \mid Flu = absent) \times P(Flue = absent) \\ & = high \times low + high \times low \\ & = low \end{aligned}$$

Example 2: Using the second method in which the qualitative scales are converted into a specific finite quantitative scale (for instance: Low=0.2, Medium=0.5, High=0.8}, the BBN can be quantified using the typical probabilistic inference methods. This is illustrated using the BBN of Figure 47. The corresponding prior and conditional probabilities are listed in Table 18:

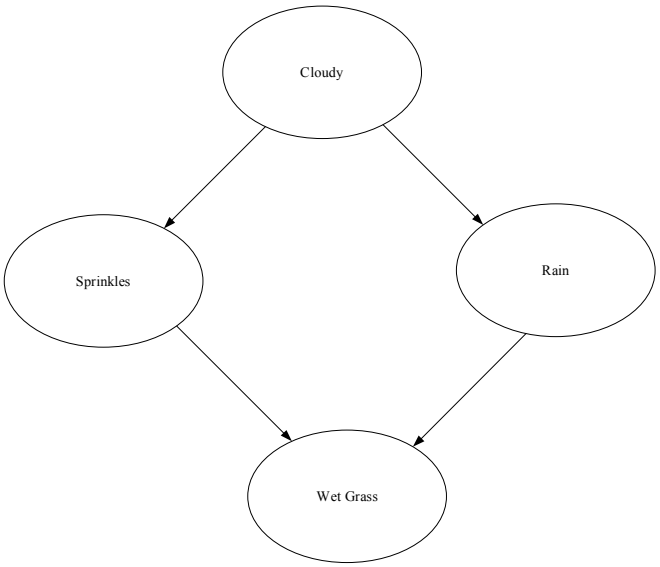


Figure 47: Bayesian belief network for example 2 (network based on [Russell and Norvig 2003])

P(C)=0.5

C	P(S)
True	0.2
False	0.5

C	P(R)
True	0.8
False	0.2

S	R	P(W)
True	True	0.8
True	False	0.8
False	True	0.8
False	False	0.2

Table 18: Prior and conditional probability table for the Bayesian belief network in Figure 47

The inference process is as follows: We first perform inference by using the numerical values in typical Bayesian belief network inference algorithms, and then convert the resulting value(s) to corresponding adjective(s). In this example, upon applying the junction tree algorithm, the probability for "wet grass" is 0.632. In the qualitative display, it will be shown as "medium".

To run the inference in a purely qualitative way, the prior and conditional probability tables (Table 19) can be used along with the qualitative algebra of Table 17 and Table 18 to arrive at the results.

P(C) = Medium

C	**P(S)**
True	Low
False	Medium

C	**P(R)**
True	High
False	Low

S	**R**	**P(W)**
True	True	High
True	False	High
False	True	High
False	False	Low

Table 19: The qualitative prior and conditional probability tables for Figure 47

We can simply apply the Pearl's message passing algorithm [Pearl 1986-1], and each time look up the computational Table 16 and Table 17 to find the corresponding result.

P(S)= Low * Medium + High * Medium =Low + Medium = Medium

P(R)= Medium * High + Medium *Low = Medium + Low = Medium

P(W)= High * Medium * Medium + High * Medium * Medium + High * Medium * Medium + Low * Medium * Medium = Medium + Medium + Medium + low = Medium

This result matches the result based on numerical conversion, but there are limitations in this qualitative inference. This is discussed in later. In this type of qualitative-quantitative Bayesian belief networks, all inferences are approximate, and as such Peal's message-passing algorithm is good enough [Weiss, 2000]. There is no need to apply more exact methods such as the junction tree algorithm or conditioning algorithm.

8.3 The Layer between Qualitative and Quantitative Domains

In QQ-BBNs, qualitative likelihood propagation "calculus" caries the inference from deeper layers of the network to the points where assessments of probabilities is possible based on observation or expert judgment. Such mixed BBNs require a linkage between the two scales at the boundary between the qualitative and quantitative parts. The assessment of the probabilities at the interface will almost always have to be done by subject matter experts, with much care to ensure the consistency with the qualitative and quantitative scales applied in the same BBN.

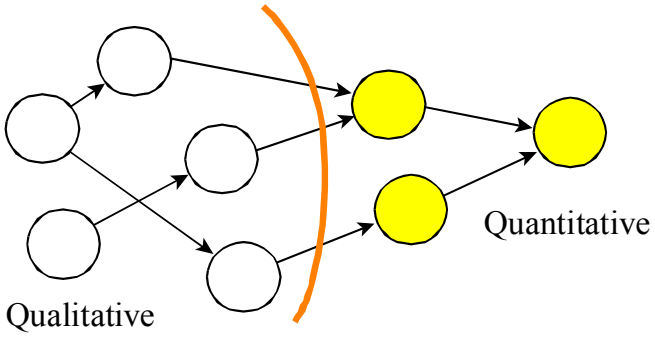


Figure 48: Boundary between qualitative and quantitative Bayesian belief networks

In the Integrated Risk Information System software, this is done by setting the property of the nodes. If the children nodes are quantitative, the parents will check the scale table to covert its scale to number, then sending to the children. If not, the qualitative scale symbol will be transformed to the children to continue the quantification. The qualitative and quantitative inference are transferable, it can go from qualitative to quantitative as shown in

Figure **48** and if needed, from quantitative to qualitative too.

9. Software Implementation of the Hybrid Causal Logic Methodology

The methodology described in previous chapters has been implemented in a computer software called the Integrated Risk Information System (IRIS) [Mosleh et al 2006]. The development involved a team of 10 students and software programmers. IRIS features a conventional probabilistic risk assessment tool for event sequence diagrams and fault tree analyses. Furthermore, it incorporates the power of Bayesian belief networks, and the combined logic in form of HCL models. This section briefly describes the main features of IRIS.

9.1 Main Features of the Integrated Risk Information System

9.1.1 Event Sequence Diagram Analysis

The event sequence diagram analysis as shown in Figure 49 is the top layer of Hybrid Causal Logic software modeling capability. It describes the possible risk scenarios following potential perturbations of normal system operation. The pivotal events in the event sequence diagram can be modeled by fault trees or Bayesian belief networks. Each scenario leads to an end-state and consequence that designates the severity of that scenario. IRIS allows the user to use "pinch points" when building the diagram. The use of pinch point makes the diagram more compact and more apparent to the user. The user can analyze the single scenario or a category of scenarios. In each project, the user can build as many event sequence diagrams as needed.

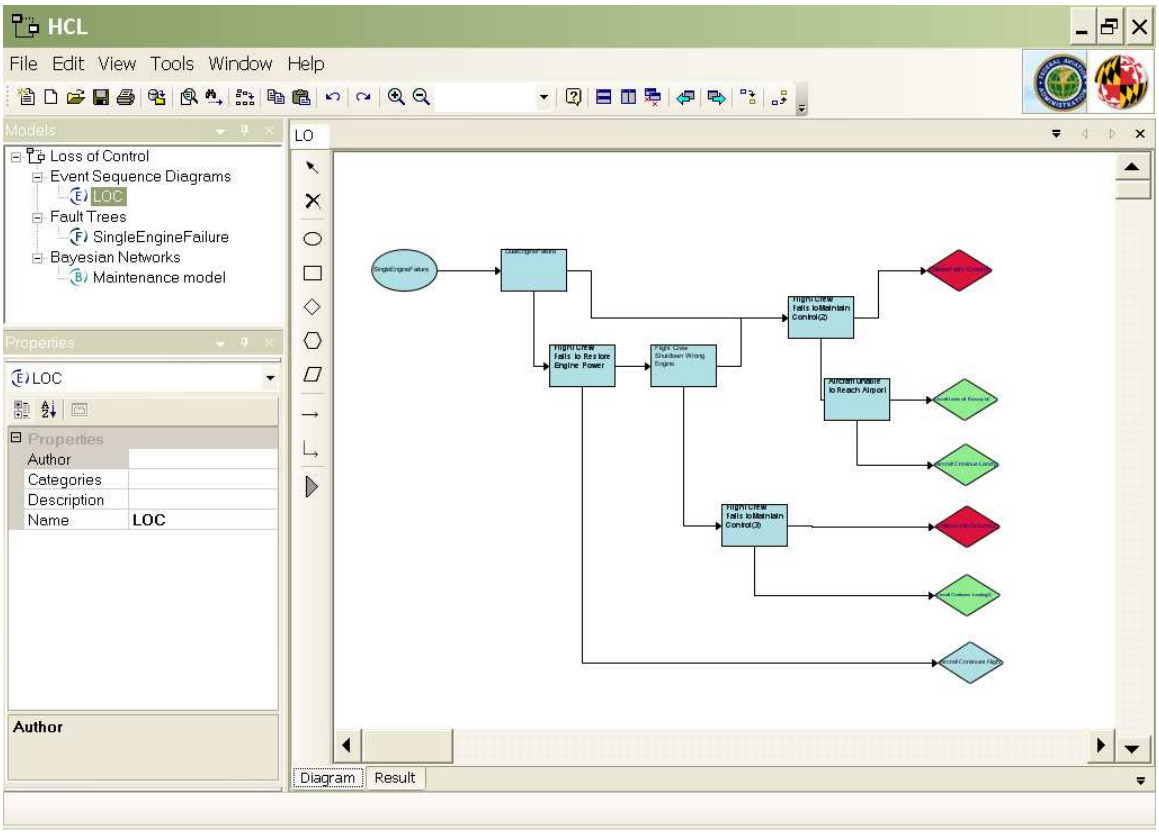


Figure 49: Event sequence diagram analysis in Integrated Risk Information System (IRIS)

Some key features in the event sequence diagram module are:

- Copy and paste function to easily copy and paste between different diagrams.
- A "pinch point" capability so that scenarios with identical futures given a particular point in their evolution could be grouped at that point, leading to more compact ESD diagram.
- Customizable result view (Figure 50). The user can select the end state type to filter the scenario ranking view, for example, when the user only wants to see a specific scenario. The user also can choose the end state type to filter the view.
- Comprehensive scenario analysis information, including scenario probability, scenario cut set identification, cut set quantification, and cut set ranking

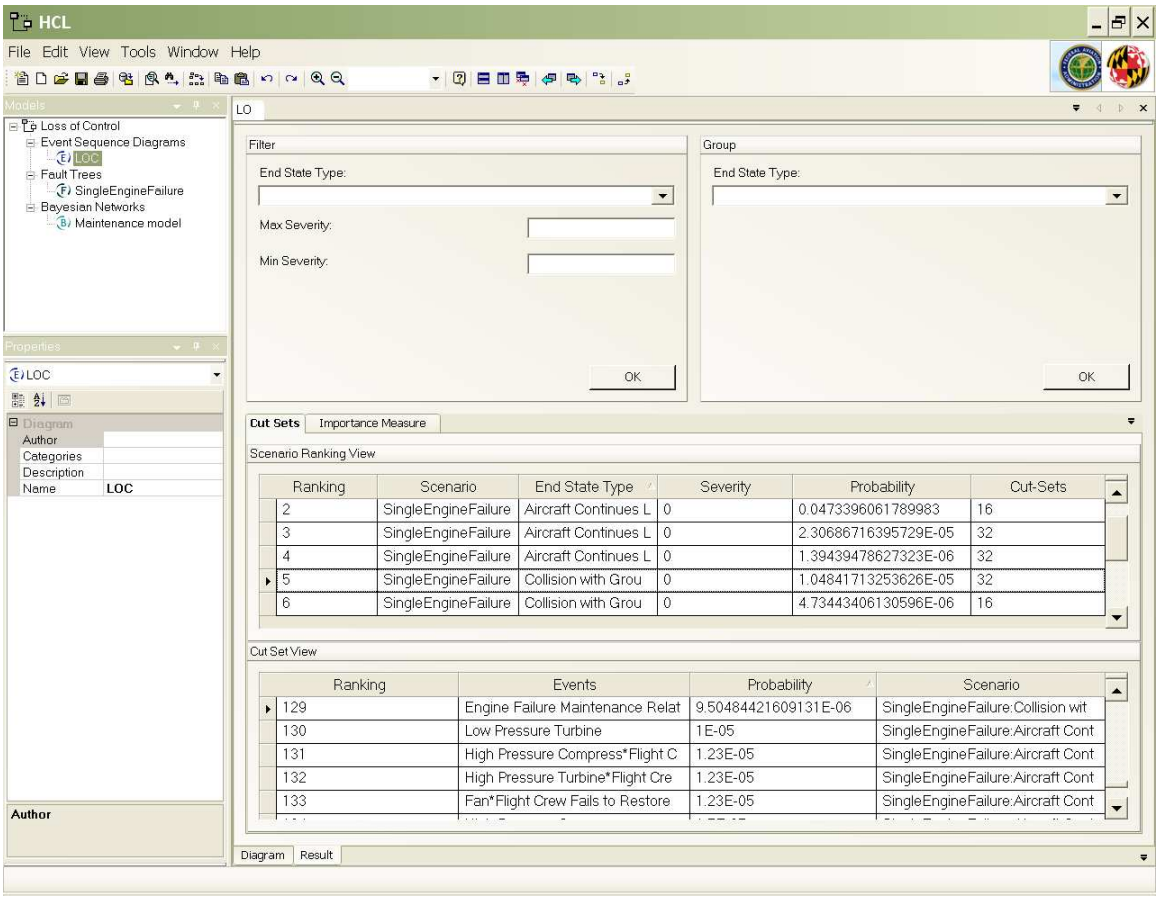


Figure 50: Event sequence diagram analysis result

9.1.2 Fault Tree Analysis

Hybrid Causal Logic Software (IRIS) has a state-of-art fault tree analysis capability. The software has a user-friendly interface to build the fault tree (see Figure 51). The user can simply drag and drop the fault tree gate and events to the diagram window. It supports move, copy and paste function. The analysis feature includes fault tree top event quantification, minimal cut set identification and quantification, importance measure, and cut set highlighting.

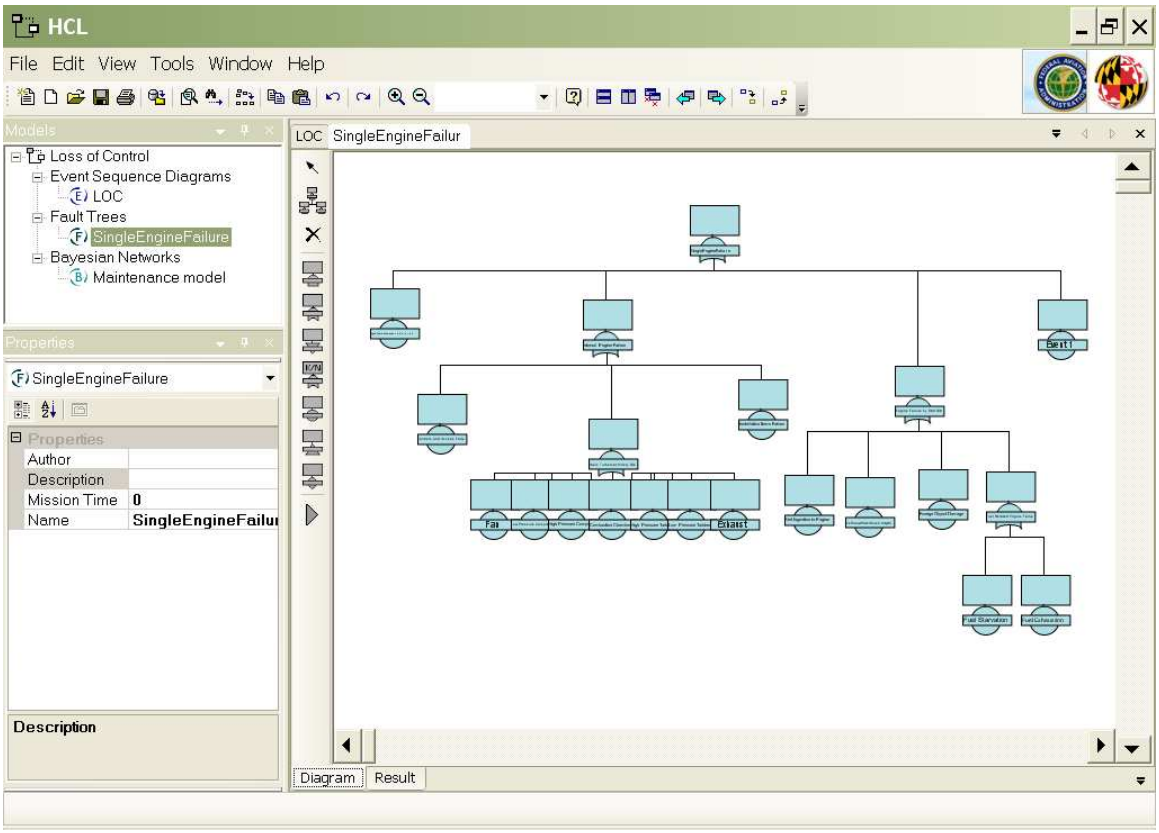


Figure 51: Fault tree analysis in Integrated Risk Information System (IRIS)

Because the Hybrid Causal Logic algorithm has adopted the Binary Decision Diagram, the fault tree analysis is fast and the result is exact compared with the traditional cut set based fault tree analysis methods.

The fault tree analysis result screen of IRIS is shown In Figure 52. The top event name, type, unavailability are presented. The minimal cut sets of the fault tree and their ranking are provided as well. The user can easily switch between the diagram and results pages.

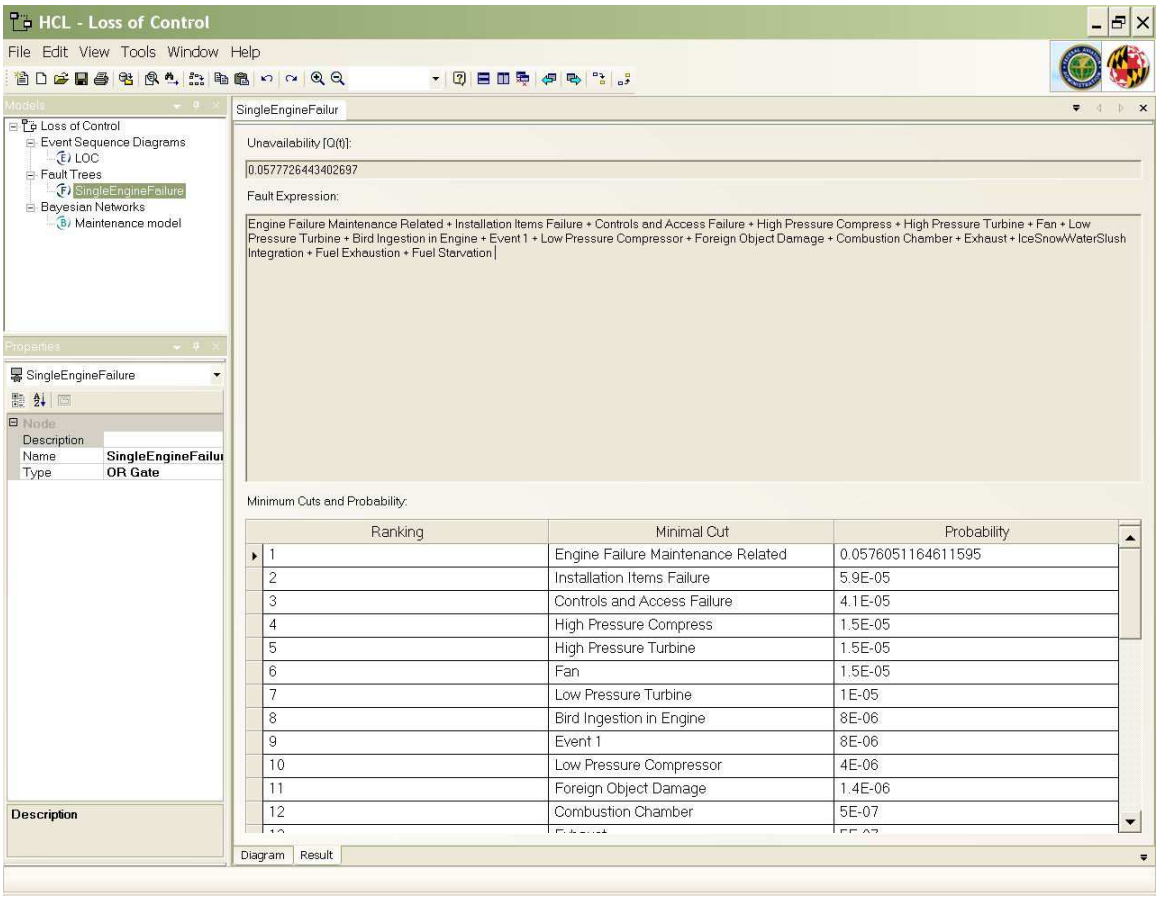


Figure 52: IRIS fault tree analysis result view

9.1.3 Bayesian Belief Network Analysis

Bayesian belief networks analysis screen (shown in Figure 53) is fully integrated into the IRIS software, making IRIS a different PRA software class compared with conventional risk analysis software.

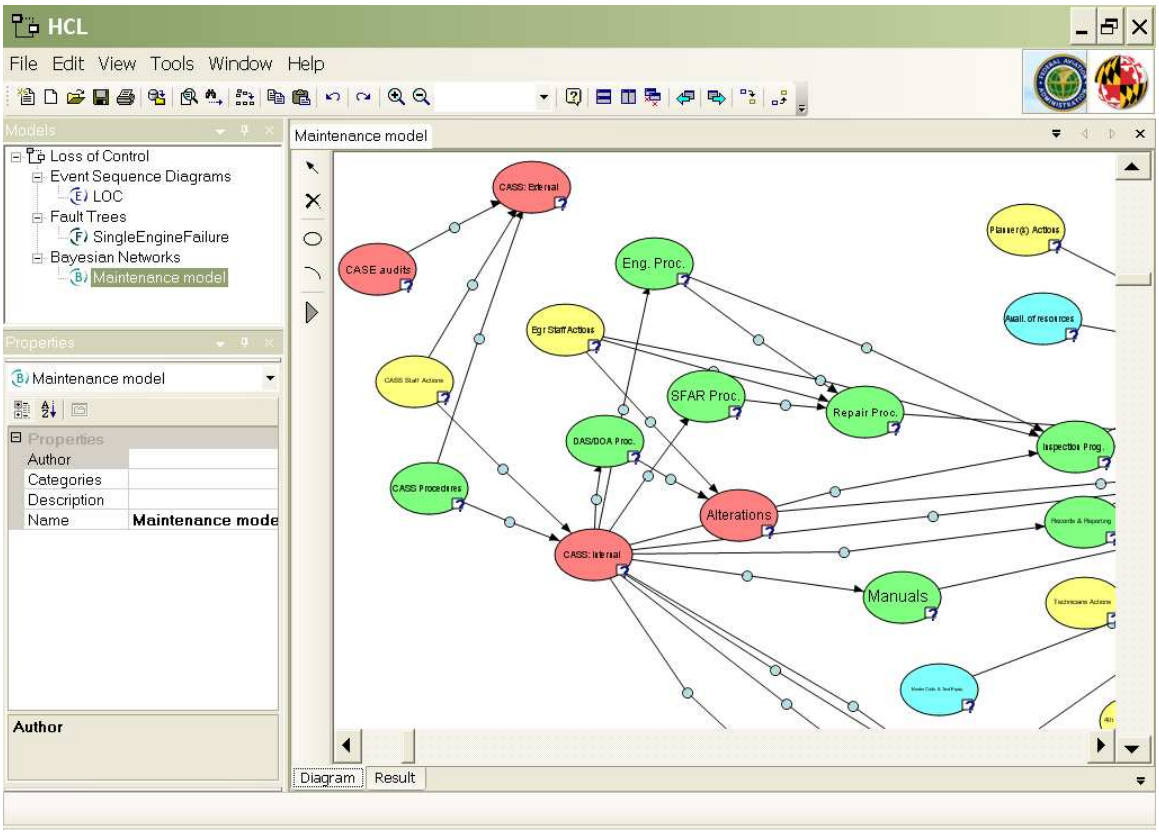


Figure 53: Bayesian belief network analysis in Hybrid Causal Logic software (IRIS)

The IRIS BBN analysis feature capability has a full inference capability. It can run inquiries for any node and the user can set evidence on any state of any node. The BBN results screen is shown in Figure 54. The node information can be seen by clicking on the node.

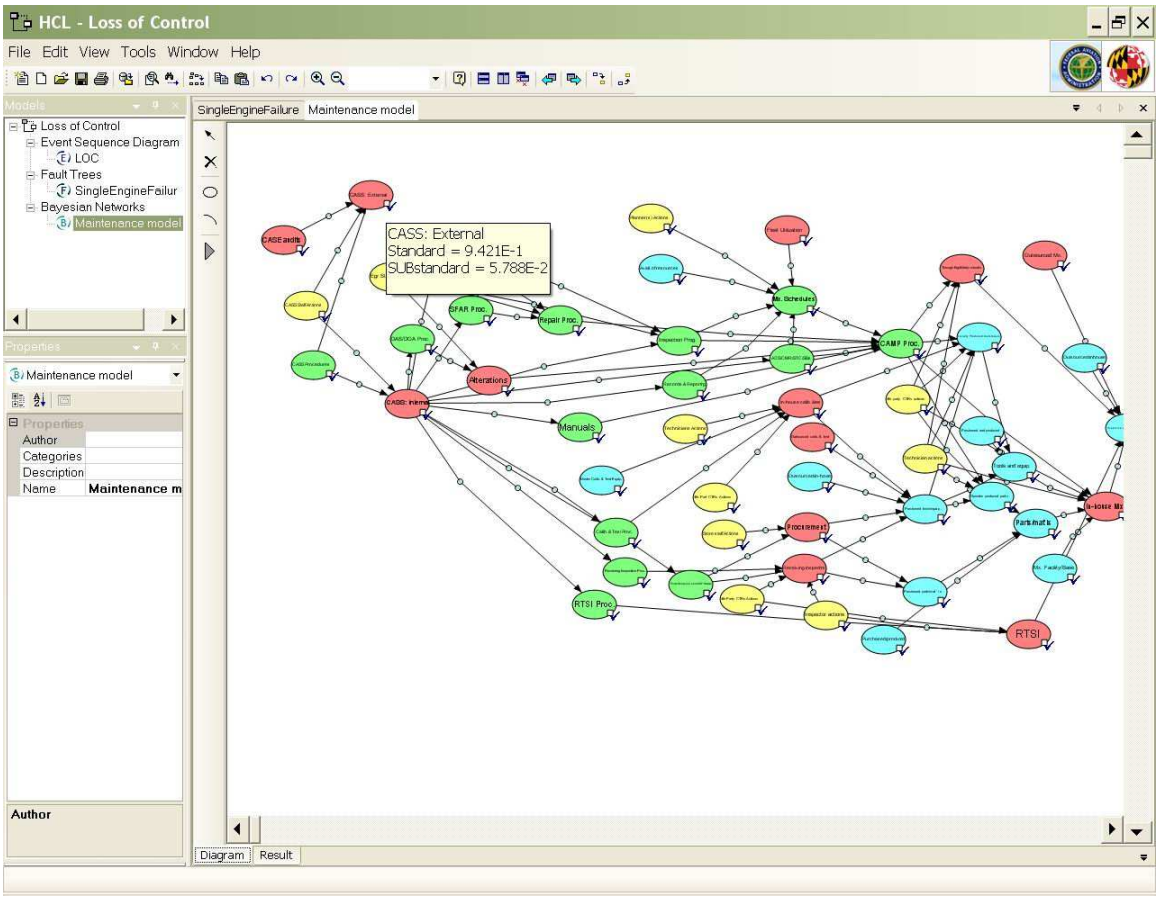


Figure 54: IRIS Bayesian belief network analysis result

The Bayesian belief analysis result can also be viewed in tabulated form (see Figure 55) which can be copied and pasted into documents.

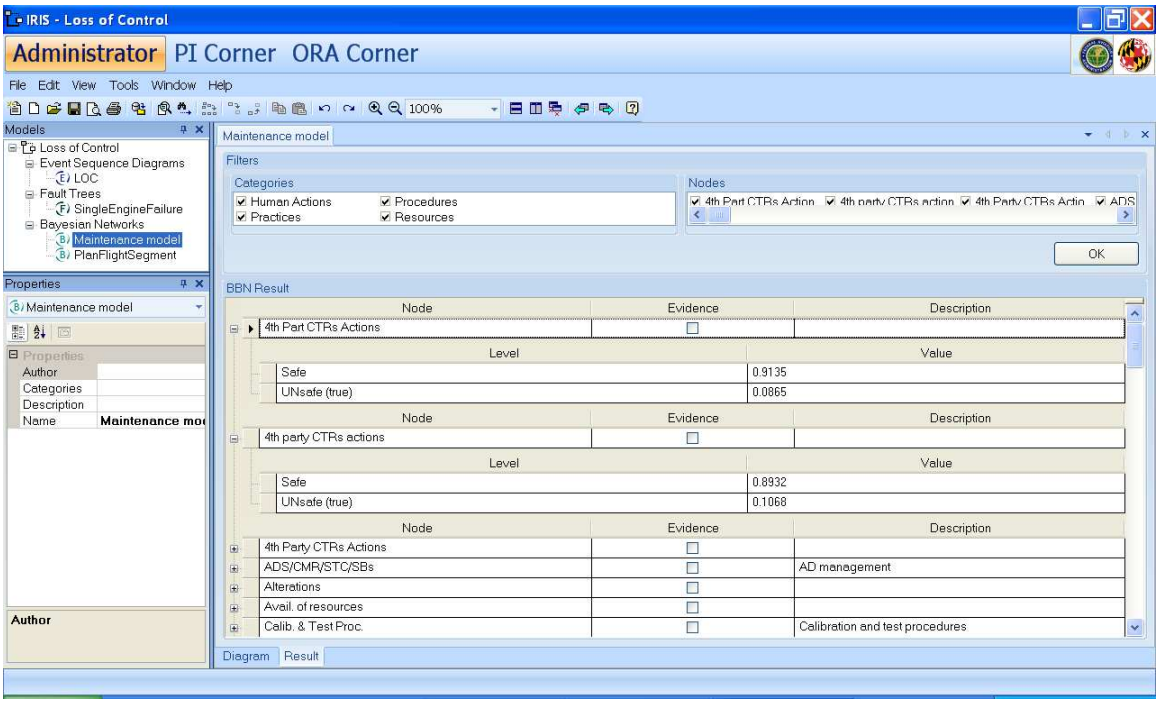


Figure 55: Tabulated Bayesian belief network analysis result

9.1.4 Integrated Hybrid Causal Logic Analysis Feature

The main feature of Integrated Risk Information System (IRIS) is the integration of the three layers of models. The user can build and link any combination of event sequence diagrams, fault trees and Bayesian belief networks to create comprehensive linked models of the system. The combined model is then solved by Hybrid Causal Logic algorithms discussed in previous chapters.

In the event sequence diagram level analysis, the fault trees and Bayesian belief networks are counted in, and the dependency is eliminated by the Hybrid Causal Logic algorithm.

In the Hybrid Causal Logic diagram analysis, the importance analysis (see Figure 56) and risk indicator provide the key information about each component or a specific set.

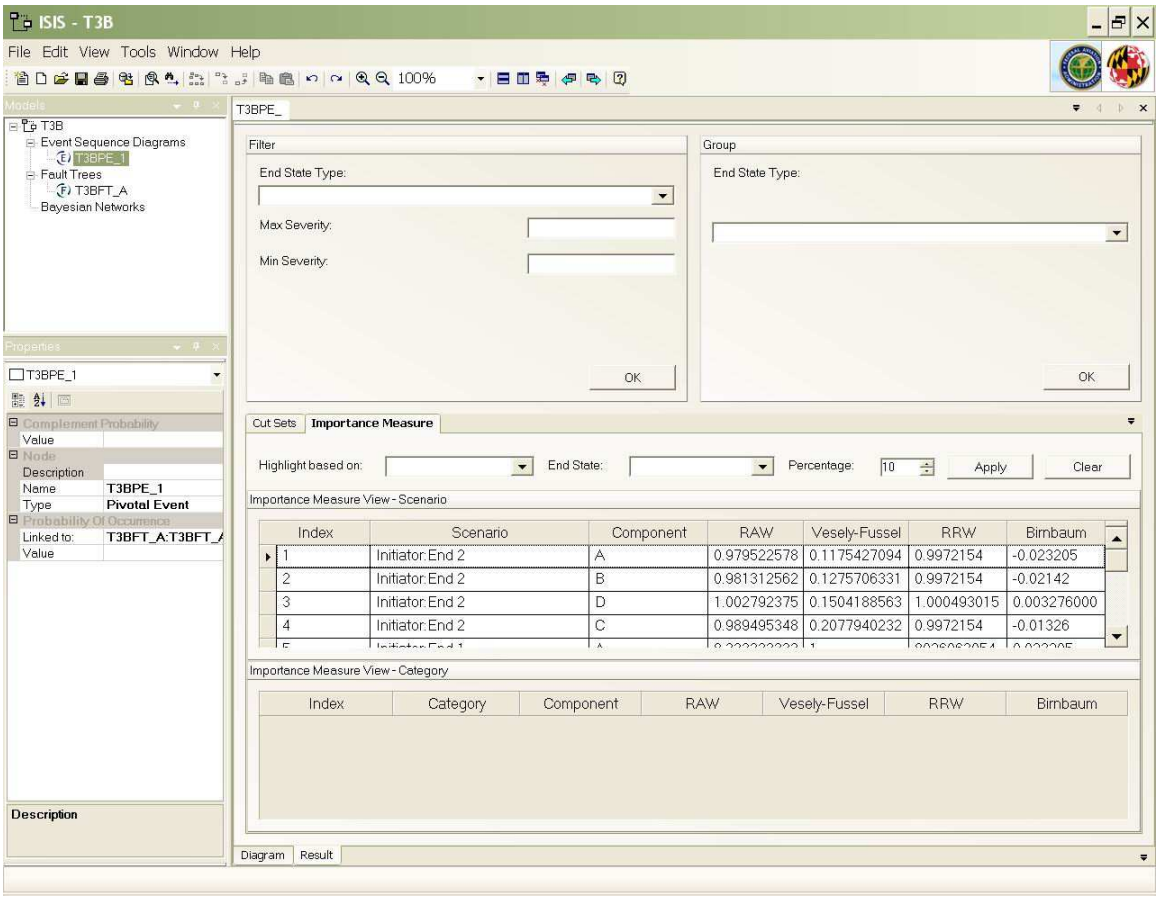


Figure 56: Importance measurement feature for linked Hybrid Causal Logic diagrams in Integrated Risk Information System (IRIS)

A path highlighting feature provides the user with a clear view of various causal paths from a given cause (BBN node or fault tree basic event) to a given end state. The following example (see Figure 57) shows the bottom-up highlighting from a basic event in the fault tree.

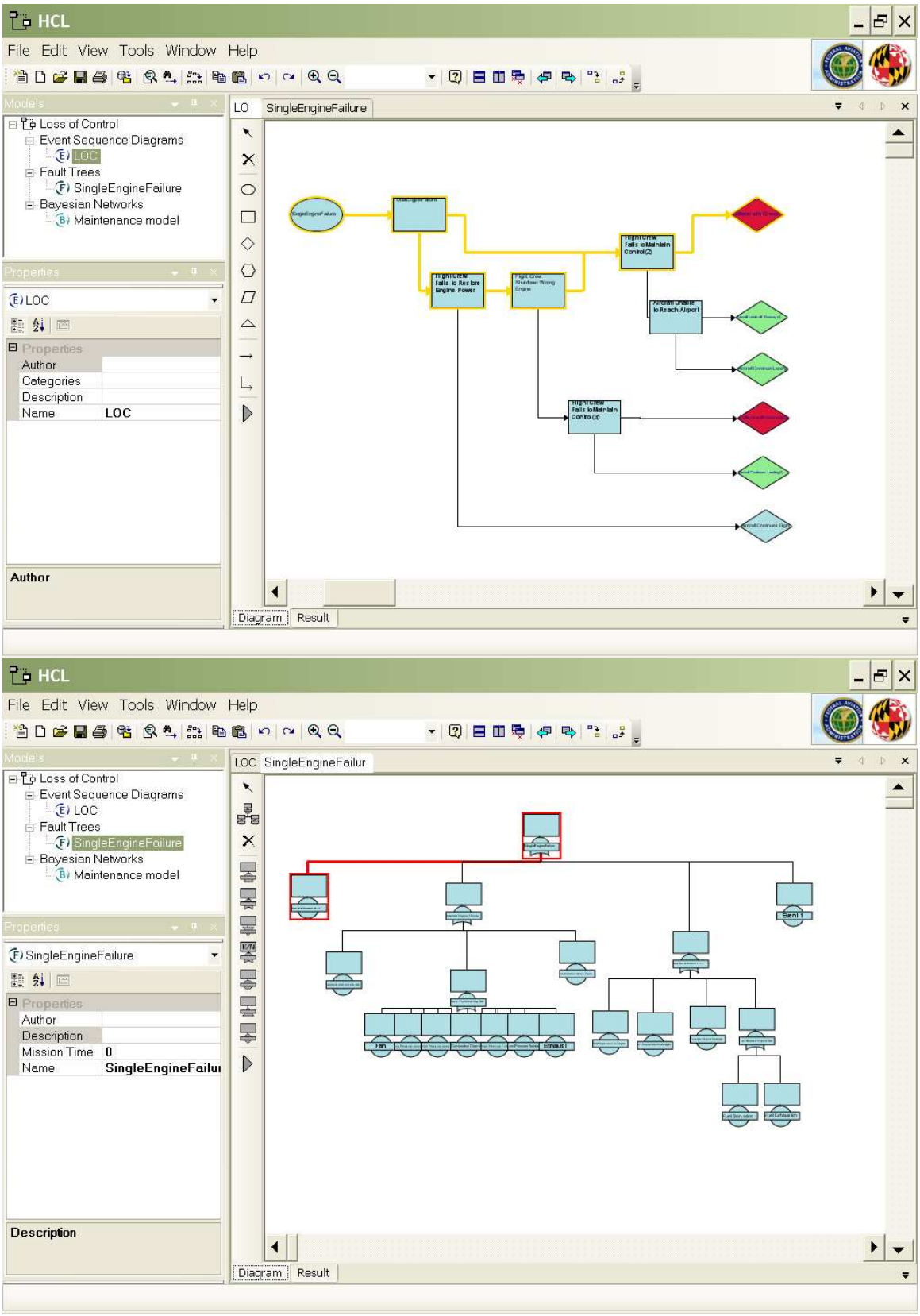


Figure 57: Causal path highlighting feature in Integrated Risk Information System (IRIS)

This highlighting feature is very helpful when the user wants to understand an accident path resulting from a specific event.

9.2 Software Testing

9.2.1 Testing Scope

The Integrated Risk Information System (IRIS) software was tested in two levels: (1) the computational engine level without Graphical User Interface (GUI), and (2) GUI test Because (IRIS) is the first software integrating the Bayesian belief network, event sequence diagram and fault tree together, the expected probability values for validating the results of the code were obtained through converting the test HCL models to Bayesian belief networks and solving it with a third party BBN algorithm. The cut set of Integrated Risk Information System (IRIS) were compared with other software, including Saphire [https://saphire.inl.gov/] and QRAS [Groen et al., 2002]. The testing has covered the following cases:

Event Sequence Diagram: The following test cases were included:

1. Single path (no pinch points). (As shown in Figure 58 when the pivotal events in event sequence diagram have several incoming path, the point where the path comes together is called pinch point. (Pinch points in event sequence diagrams may also be called middle state).
2. Multiple paths (simple pinch point, merging two sequences).
3. Multiple paths (complex pinch point, at least three sequences merged).
4. Path includes failed branches.

5. Path includes success branches.
6. Multiple event sequence diagrams, without fault trees attached to them, in one project file.
7. Multiple event sequence diagrams with fault trees attached in one project file.

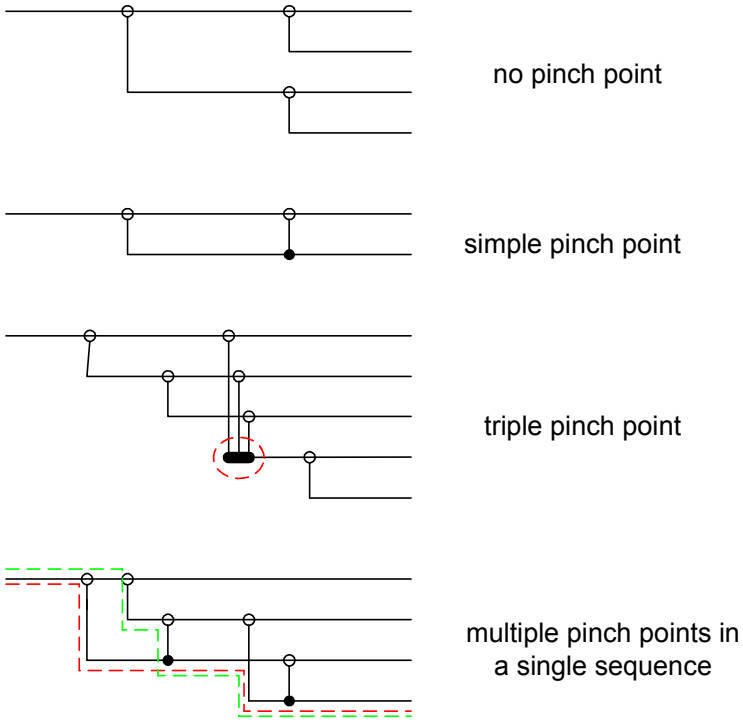


Figure 58: Pinch point configurations

Fault Tree Analysis: The following cases were tested:

1. AND Gate.
2. OR Gate.
3. NOT Gate.
4. K-out-of-N Gate.
5. Multiple instances of a basic event within a fault tree.
6. Multiple instances of a basic event across the fault trees.
7. Basic events linked to one Bayesian belief network.

8. Basic events linked to multiple Bayesian belief networks (in a single fault tree or separate fault trees).
9. Multiple basic events linked to a single Bayesian belief network (in a single fault tree or separate fault trees).

Bayesian Belief Networks: In this test category both singly connected and multiply connected BBNs were tested. Tests included were:

1. Bayesian belief networks which are polytrees.
2. Bayesian belief networks containing loops.

Numerical Ranges: In the qualitative analysis, both very small numbers and very large probabilities were tested.

1. Small basic event probabilities (< 1E-4).
2. Large basic event probabilities (> 0.1).
3. Small scenario probability (expected result) (<1E-4).
4. Large scenario probability (expected result) (>0.1).

Importance Measures: Following test coverage and criteria were met:

1. Importance measure measuring the basic event in the fault tree (linked to, or not linked to, the Bayesian belief network.).
2. Importance measure measuring the pivotal event without fault trees attached in the event sequence diagram.
3. Importance measure measuring the nodes in Bayesian belief networks.
4. Importance measure measuring the category importance in a single event sequence diagram.

5. Importance measure measuring the category importance across multiple event sequence diagram.

Cut Set Identification: Both fault tree level and scenario level cut sets were tested.

Large Scale Models: Test cases included large fault trees, large event sequence diagrams and the large Bayesian belief networks:

1. Large fault tree (More than 200 events, 100 gates)
2. Large event sequence diagram (More than 100 pivotal events)
3. Large Bayesian belief networks. (More than 50 nodes, 80 arcs and 1000 parameters)
4. Large scale HCL with large event sequence diagram, large fault tree, and large Bayesian belief networks.

Risk Indicator: In the risk indicator testing, the following types of indicators were used:

1. Fault tree basic events or pivotal events (without fault tree attached)
2. Fault tree gates or pivotal events, with fault tree attached
3. BBN nodes
4. Risk indicator corresponding to a single end state.
5. Risk indicator corresponding to an end stage category within one event sequence diagram.
6. Risk indicator corresponding to one or more end state categories within multiple event sequence diagrams.

9.2.2 Probability Computation Approach

To validate the Hybrid Causal Logic diagram probability computations in all tests, the models were converted to equivalent Bayesian belief networks and performed analysis in the GeNie software [http://genie.sis.pitt.edu/]. In the test, all the top event failure probabilities and scenario probabilities are calculated based on the exact computation; there are no rare events assumptions.

9.3 A Large Realistic Example

The following describes a full example with event sequence diagram, fault trees (see Figure 59 and Figure 60), and a large BBN (see Figure 61). The event sequence diagram and the fault tree were developed by the National Aerospace Laboratory of the Netherlands (NLR), under contract from FAA. The Bayesian belief network was developed by Hi-Tech Inc, also under the FAA contract [Eghbali and Mandalapu, 2005]. The conditional probability tables of the BBN are provided in Appendix 2.

The event sequence diagram in Figure 59 presents the different scenarios due to a single engine failure. After the initiator – single engine failure, the flight crew can take different actions such as restore engine power; maintain control, and so on. Eventually, the actions lead to four different end states: Collision with Ground, Aircraft Lands off Runway, Aircraft Continues Landing, and Aircraft Continues Flight.

The fault tree in Figure 60 models the causes of the initiator – single engine failure. It includes the internal engine failure, engine failure by external factor, engine management issue, and furthermore, the engine maintenance, which is actually detailed by a BBN (Figure 61).

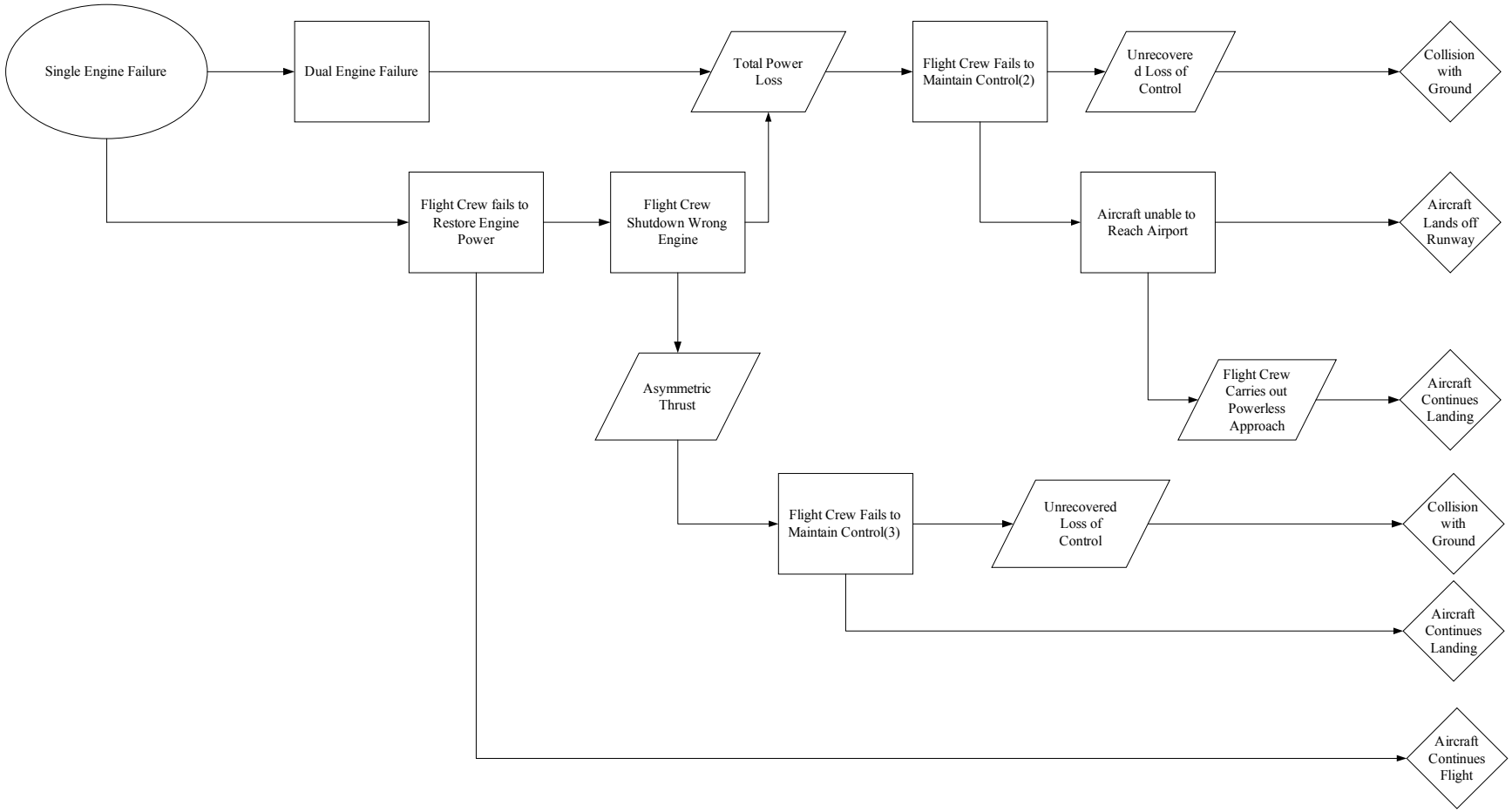


Figure 59: Event sequence diagram of the full example

In Figure 59, the asymmetric thrust is due to an engine shutdown, feathered propeller or engine in idle thrust. The pivotal event "flight crew fails to maintain control (2)" incorporates control of speed, altitude, pitch and roll. Flight crew skills related to powerless flight. The pivotal event "flight crew fails to maintain control (3)" incorporates control of speed, altitude, pitch and roll, and power management. Flight crew skills related to one engine inoperative flight.

In the end state "aircraft lands off runway", off runway means a forced landing in file or ditching. The probabilities of pivotal events are point estimated as in Table 20. The initiator is modeled by the fault tree in Figure 60.

Pivotal Event	**Probability**
Dual Engine Failure	5.50E-4
Flight Crew Fails to Restore Engine Power	8.20E-1
Flight Crew Shutdown Wrong Engine	6.70E-5
Flight Crew Fails to Maintain Control (2)	3.00E-1
Aircraft unable to Reach Airport	9.43E-1
Flight Crew Fails to Maintain Control (3)	1.00E-4

Table 20: Pivotal event probability table for event sequence diagram in Figure 59

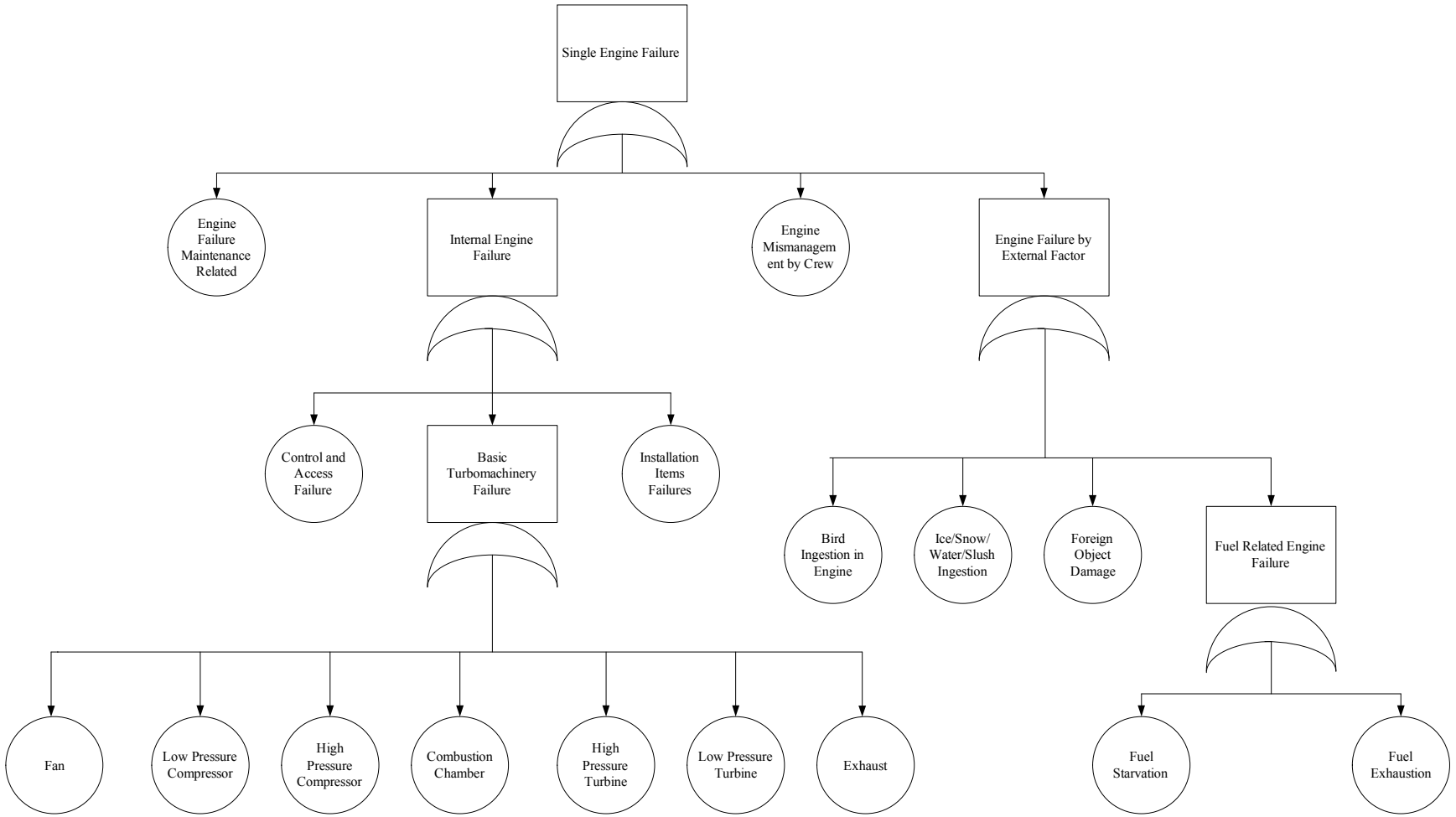


Figure 60: Fault tree of the comprehensive example

The fault tree in Figure 60 has 17 basic events. In the fault tree, the basic event – Engine Failure Maintenance Related, is modeled by the Bayesian belief networks in Figure 61. Point estimates of the probabilities of other basic events are listed Table 21.

Basic Events	**Probability**
Engine Mismanagement by Crew	8.0E-6
Controls and Access Failure	4.1E-5
Installation Items Failure	5.9E-5
Bird Ingestion in Engine	8.0E-6
Ice/Snow/Water/Slush Ingestion	3.3E-7
Foreign Object Damage	1.4E-6
Fuel Starvation	1.7E-8
Fuel Exhaustion	3.4E-8
Fan	1.5E-5
Low Pressure Compressor	4.0E-6
High Pressure Compressor	1.5E-5
Combustion Chamber	5.0E-7
High Pressure Turbine	1.5E-5
Low Pressure Turbine	1.0E-5
Exhaust	5.0E-7

Table 21: Probabilities of basic events in Figure 60

The Bayesian belief network in Figure 61 is used to model the basic event - Engine Failure Maintenance Related in Figure 60. The node - In Service Aircraft Airworthiness will be linked to the basic event - Engine Failure Maintenance Related in the fault tree. The Bayesian belief network in Figure 61 has 52 nodes and 85 links between the 58 nodes. This Bayesian belief network incorporates the procedures, the audits, the tools, the technician, the calibration, the parts and material and so on factors in one network, to model the engine failure maintenance related.

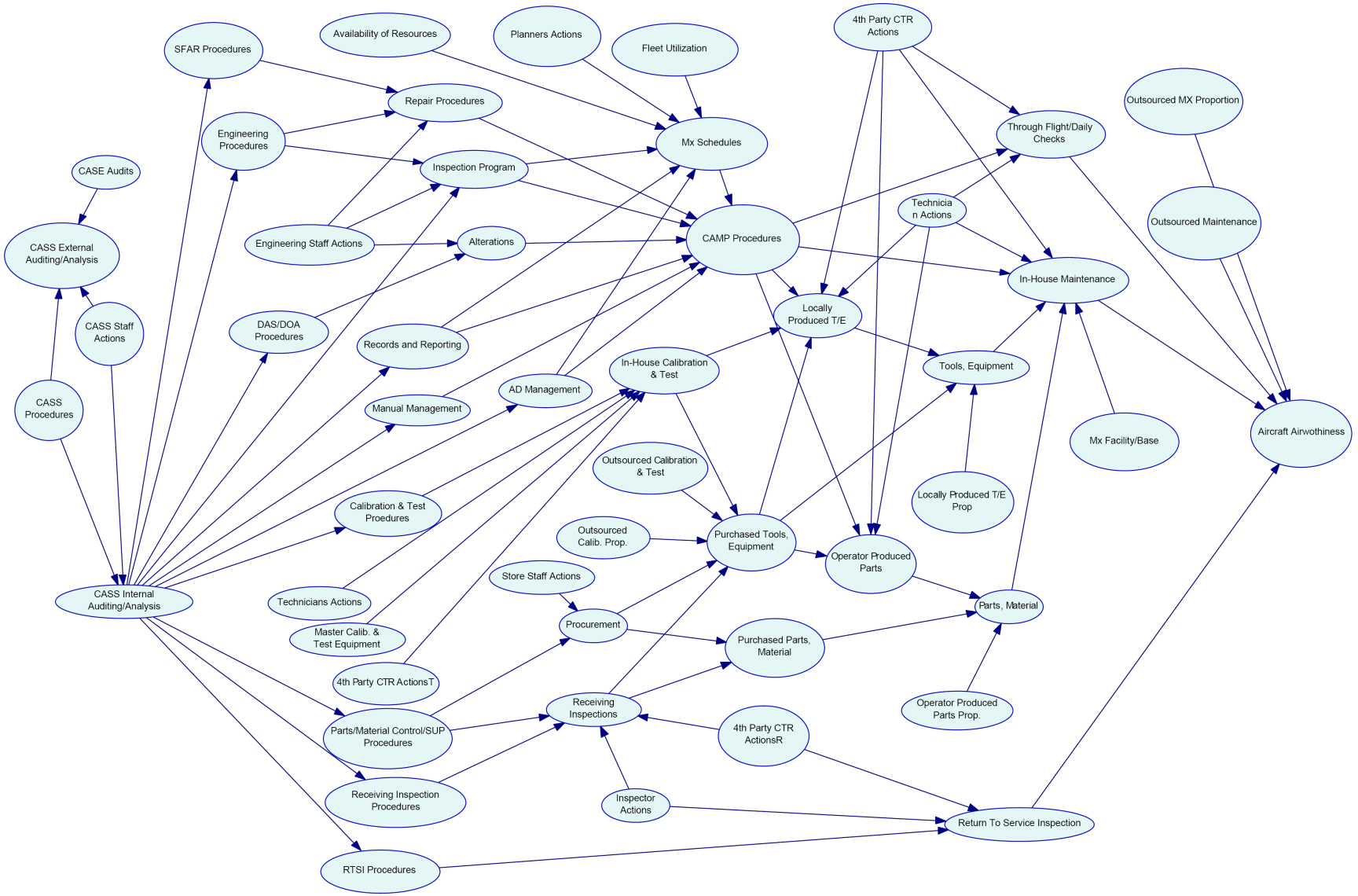


Figure 61: The Bayesian belief network of the full example [Eghbali and Mandalapu, 2005]

This example has all the three layers discussed previously; the event sequence diagram shows the scenario when a single engine failure happens, its consequence and respective probability. A fault tree is used to model the details of the initiator. Furthermore, the basic event – engine failure maintenance related in the fault tree can be modeled by the Bayesian belief network in Figure 61 by linking the basic event to the Bayesian belief network node – "In service Aircraft Airworthiness" in the Hybrid Causal Logic analyzer. The Hybrid Causal Logic algorithm is used to quantifying the whole hybrid diagram and gets the exact result.

The "In service Aircraft Airworthiness" probability is 0.057. The fault tree top event probability is 0.05777.

Scenario	**Probability**
SingleEngineFailure: Collision with Ground	1.52E-5
SingleEngineFailure: Aircraft Lands off Runway	2.31E-5
SingleEngineFailure: Aircraft Continues Flight	1.04E-2
SingleEngineFailure: Aircraft Continues Landing	4.73E-2

Table 22: Scenario probability of the full example

The analysis result of Table 22, is the same as one obtained by mapping the entire hybrid causal logic diagram to a single BBN. Table 23 provides the importance measures. The reference end state is collision with ground in the event sequence diagram.

Event Name	**Risk Achievement**	**Vesely-Fussel**

	Worth	
Dual Engine Failure	1.14E3	6.26E-1
Flight Crew Fails to Restore Engine Power	1.08	8.87E-1
Flight Crew Shutdown Wrong Engine	9.29E2	6.20E-2
Aircraft unable to Reach Airport	1.00	9.43E-1
Engine Mismanagement by Crew	6.57E4	5.25E-1
Engine Failure Maintenance Related	6.57E4	3.78E2
Controls and Access Failures	6.57E4	2.67
Installation Items Failure	6.57E4	4.53
Exhaust	6.57E4	3.29E-2
Manual Management	1.0028	4.70E-2
RTSI Procedures	1.040	3.60E-2
CASS Procedures	1.007	7.30E-2
Repair Procedures	1.007	5.00E-2
CAMP Procedures	1.06	6.60E-2
AD Management	1.007	3.50E-2
MX Schedules	1.007	4.50E-2

Table 23: Importance Measurement example for the comprehensive example

As discussed in the Chapter 6, performance indicators can be selected and analyzed using HCL models. Table 24 is an example based on the current comprehensive example. The scenario is from the initiator – single engine failure to the end state – collision with Ground. The frequency of the event is assumed to be 3 per year.

Event	Performance Indicator
Single Engine Failure	7.8E-4
Dual Engine Failure	5.1E-2
Flight Crew Fails to Restore Engine Power	4.94
Aircraft unable to Reach Airport	4.57E-5
Engine Failure Maintenance Related	7.9E-4
Engine Mismanagement by Crew	7.9E-4
Controls and Access Failure	7.9E-4
Installation Items Failures	7.9E-4
Foreign Object Damage	7.9E-4
Low Pressure Compressor	7.9E-4
High Pressure Compressor	7.9E-4
High Pressure Turbine	7.9E-4
Low Pressure Turbine	7.9E-4
Exhaust	7.9E-4
CASE Audits	4.59E-5
DAS/DOA Procedures	4.59E-5
AD Management	4.59E-5
Procurement	4.59E-5
Outsourced Maintenance	6.57E-5
In-House Maintenance	6.97E-5

Table 24: Safety performance indicators for the comprehensive example

Table 25 lists the cut sets of the full example. They were the same as those produced from the QRAS software [Groen et al., 2002].

Cut Sets Generated by QRAS	HCL Produces Same Cut Set?
Fuel Starvation*Flight Crew Fails to Maintain Control(2)*Flight Crew Fails to Restore Engine Power*Flight Crew Shutdown Wrong Engine	Yes
Fuel Exhaustion*Flight Crew Fails to Maintain Control(2)*Flight Crew Fails to Restore Engine Power*Flight Crew Shutdown Wrong Engine	Yes
Fuel Starvation*Flight Crew Fails to Restore Engine Power*Flight Crew Fails to Maintain Control(3)	Yes
Fuel Exhaustion*Flight Crew Fails to Restore Engine Power*Flight Crew Fails to Maintain Control(3)	Yes
Fuel Starvation*DualEngineFailure*Flight Crew Fails to Maintain Control(2)	Yes
IceSnowWaterSlush Integration*Flight Crew Fails to Maintain Control(2)*Flight Crew Fails to Restore Engine Power*Flight Crew Shutdown Wrong Engine	Yes
Fuel Exhaustion*DualEngineFailure*Flight Crew Fails to Maintain Control(2)	Yes
Exhaust*Flight Crew Fails to Maintain Control(2)*Flight Crew Fails to Restore Engine Power*Flight Crew Shutdown Wrong Engine	Yes
Combustion Chamber*Flight Crew Fails to Maintain Control(2)*Flight	Yes

Crew Fails to Restore Engine Power*Flight Crew Shutdown Wrong Engine	
Foreign Object Damage*Flight Crew Fails to Maintain Control(2)*Flight Crew Fails to Restore Engine Power*Flight Crew Shutdown Wrong Engine	Yes
IceSnowWaterSlush Integration*Flight Crew Fails to Restore Engine Power*Flight Crew Fails to Maintain Control(3)	Yes
Exhaust*Flight Crew Fails to Restore Engine Power*Flight Crew Fails to Maintain Control(3)	Yes
Combustion Chamber*Flight Crew Fails to Restore Engine Power*Flight Crew Fails to Maintain Control(3)	Yes
IceSnowWaterSlush Integration*DualEngineFailure*Flight Crew Fails to Maintain Control(2)	Yes
Low Pressure Compressor*Flight Crew Fails to Maintain Control(2)*Flight Crew Fails to Restore Engine Power*Flight Crew Shutdown Wrong Engine	Yes
Exhaust*DualEngineFailure*Flight Crew Fails to Maintain Control(2)	Yes
Combustion Chamber*DualEngineFailure*Flight Crew Fails to Maintain Control(2)	Yes
Foreign Object Damage*Flight Crew Fails to Restore Engine Power*Flight Crew Fails to Maintain Control(3)	Yes
Engine Mismanagement by Crew*Flight Crew Fails to Maintain Control(2)*Flight Crew Fails to Restore Engine Power*Flight Crew	Yes

Shutdown Wrong Engine	
Bird Ingestion in Engine*Flight Crew Fails to Maintain Control(2)*Flight Crew Fails to Restore Engine Power*Flight Crew Shutdown Wrong Engine	Yes
Low Pressure Turbine*Flight Crew Fails to Maintain Control(2)*Flight Crew Fails to Restore Engine Power*Flight Crew Shutdown Wrong Engine	Yes
Foreign Object Damage*DualEngineFailure*Flight Crew Fails to Maintain Control(2)	Yes
High Pressure Turbine*Flight Crew Fails to Maintain Control(2)*Flight Crew Fails to Restore Engine Power*Flight Crew Shutdown Wrong Engine	Yes
Fan*Flight Crew Fails to Maintain Control(2)*Flight Crew Fails to Restore Engine Power*Flight Crew Shutdown Wrong Engine	Yes
High Pressure Compress*Flight Crew Fails to Maintain Control(2)*Flight Crew Fails to Restore Engine Power*Flight Crew Shutdown Wrong Engine	Yes
Low Pressure Compressor*Flight Crew Fails to Restore Engine Power*Flight Crew Fails to Maintain Control(3)	Yes
Bird Ingestion in Engine*Flight Crew Fails to Restore Engine Power*Flight Crew Fails to Maintain Control(3)	Yes
Engine Mismanagement by Crew*Flight Crew Fails to Restore Engine	Yes

Power*Flight Crew Fails to Maintain Control(3)	
Low Pressure Compressor*DualEngineFailure*Flight Crew Fails to Maintain Control(2)	Yes
Controls and Access Failure*Flight Crew Fails to Maintain Control(2)*Flight Crew Fails to Restore Engine Power*Flight Crew Shutdown Wrong Engine	Yes
Low Pressure Turbine*Flight Crew Fails to Restore Engine Power*Flight Crew Fails to Maintain Control(3)	Yes
Installation Items Failure*Flight Crew Fails to Maintain Control(2)*Flight Crew Fails to Restore Engine Power*Flight Crew Shutdown Wrong Engine	Yes
Fan*Flight Crew Fails to Restore Engine Power*Flight Crew Fails to Maintain Control(3)	Yes
High Pressure Turbine*Flight Crew Fails to Restore Engine Power*Flight Crew Fails to Maintain Control(3)	Yes
High Pressure Compress*Flight Crew Fails to Restore Engine Power*Flight Crew Fails to Maintain Control(3)	Yes
Bird Ingestion in Engine*DualEngineFailure*Flight Crew Fails to Maintain Control(2)	Yes
Engine Mismanagement by Crew*DualEngineFailure*Flight Crew Fails to Maintain Control(2)	Yes
Low Pressure Turbine*DualEngineFailure*Flight Crew Fails to Maintain Control(2)	Yes

High Pressure Turbine*DualEngineFailure*Flight Crew Fails to Maintain Control(2)	Yes
High Pressure Compress*DualEngineFailure*Flight Crew Fails to Maintain Control(2)	Yes
Fan*DualEngineFailure*Flight Crew Fails to Maintain Control(2)	Yes
Controls and Access Failure*Flight Crew Fails to Restore Engine Power*Flight Crew Fails to Maintain Control(3)	Yes
Installation Items Failure*Flight Crew Fails to Restore Engine Power*Flight Crew Fails to Maintain Control(3)	Yes
Controls and Access Failure*DualEngineFailure*Flight Crew Fails to Maintain Control(2)	Yes
Installation Items Failure*DualEngineFailure*Flight Crew Fails to Maintain Control(2)	Yes
Engine Failure Maintenance Related*Flight Crew Fails to Maintain Control(2)*Flight Crew Fails to Restore Engine Power*Flight Crew Shutdown Wrong Engine	Yes
Engine Failure Maintenance Related*Flight Crew Fails to Restore Engine Power*Flight Crew Fails to Maintain Control(3)	Yes
Engine Failure Maintenance Related*DualEngineFailure*Flight Crew Fails to Maintain Control(2)	Yes

Table 25: Cut sets analysis result of the full example

The above example shows that the Hybrid Causal Logic methodology can keep the conventional Probabilistic Risk Assessment infrastructure while significantly enhancing its causal modeling and probabilistic inference capability. It can provide exact scenario probability, importance measures, performance indicators, and cut sets.

10. Research Contributions, Limitations, and Path Forward

10.1 Contributions

The objective of this research was to build a hybrid causal modeling methodology to perform probabilistic risk analysis (PRA) of complex technological systems. The PRA process for technological systems typically includes the following steps: objective and scope definition, system familiarization, identification of initiating events, scenario modeling, quantification, uncertainty analysis, sensitivity analysis, importance ranking, and data analysis. Fault trees and event trees are widely used tools for risk scenario analysis in PRAs of technological systems. This methodology is most suitable for systems made of hardware components. This research was motivated by the observation that a more comprehensive treatment of risks of technical systems would have to consider the entire environment within which such systems are designed and operated. This environment includes the physical environment, the socio-economic environment, and in some cases the regulatory and oversight environment. The technical system, supported by an organization of people in charge of its operation, is at the cross-section of these environments.

In order to develop a more comprehensive risk model for these systems, an essential step is to extend the modeling capabilities of the conventional PRA methodology to also include risks associated with human activities and organizational factors in addition to hardware and software failures and adverse conditions of the physical environment. The causal modeling should also extend to the influence of regulatory and oversight functions.

This research has offered such a methodology. It has proposed a multi-layered modeling approach so that most the appropriate techniques are applied to different individual domains of the system.

The proposed approach, called here the Hybrid Causal Logic (HCL) methodology, uses a three-layer modeling technique:

1. A model to define safety/risk context. This is done using a technique known as event sequence diagram (ESD) method that helps define the kinds of accidents and incidents that can occur in relation to the system being considered;
2. A model that captures the behaviors of the physical system (hardware, software, and environmental factors) as possible causes or contributing factors to accidents and incidents delineated by the event sequence diagrams. This is done by common system modeling techniques such as fault tress (FT); and
3. A model to extend the causal chain of events to their potential human and organizational roots. This is done using Bayesian belief networks (BBN). Bayesian belief networks are particularly useful as they do not require complete knowledge of the relation between causes and effects.

The integrated model is therefore a hybrid causal model with the corresponding sets of taxonomies and analytical and computational procedures. Since such hybrid models involve significant interdependencies, the nature of such dependencies are first determined to pave the way for developing proper algorithmic solutions of the logic model. Major achievements of this work are:

- Development of the Hybrid Causal Logic model concept and quantification algorithms;

- Development and testing of computer implementation of algorithms (collaborative work);
- Development and implementation of algorithms for HCL–based importance measures, an uncertainty propagation method the BBN models, and algorithms for qualitative-quantitative Bayesian belief networks; and
- Development and testing of the Integrated Risk Information System (IRIS) software based on HCL methodology.

10.2 Limitations and Path Foreword

The hybrid methodology introduced in this dissertation is a significant step towards more realistic modeling and quantification of risk. As the modeling and quantification is extended to even more uncertain domains via BBNs, as compared to classical PRA frameworks, the question of consistent treatment of various risk contributors and use of the analysis results for comparing different alternatives become more challenging. This is more pronounced when both qualitative and quantitative likelihood scales are used in the BBNs as suggested in Chapter 8. On this issue more work on likelihood propagation methods, assessment techniques and calibration guidelines is needed. A related point is the method for uncertainty propagation, which in this work is limited to uncertainty on quantitative measures of likelihoods. The assessment and propagation of mixed likelihoods (quantitative and qualitative) need appropriate methods, perhaps by borrowing ideas from other non-probabilistic uncertainty theories. Also while one of the HCL algorithms proposed here is extremely efficient for point estimate quantification of risk (even for vary large hybrid models), no effort was made to make the uncertainty

propagation method efficient. This is a problem that can be easily addressed by tapping into the vast array of Monte Carlo algorithms for large scale uncertainty propagation.

Appendix 1: Bayesian Belief Networks Inference Algorithm Example

1. Variable Elimination Example

In this section, the variable elimination method is demonstrated step by step using the example in Figure 62. The prior and conditional probability are in Figure 26. The steps are shown in Table 27.

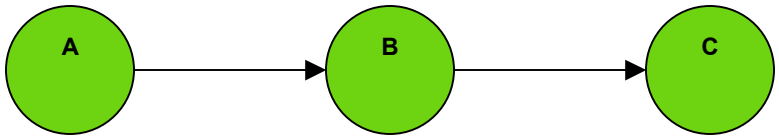


Figure 62: Variable elimination example

$$P(C) = \sum_{A,B} P(C \mid B)P(B \mid A)P(A)$$
$$= \sum_{B} P(C \mid B)\sum_{A} P(B \mid A)P(A)$$
$$= \sum_{B} P(C \mid B)f_A(A,B)$$
$$= f_B(B,C)$$

A

	P(A)
0	0.9
1	0.1

B

A	B	P(B|A)
0	0	0.9
0	1	0.1
1	0	0.4
1	1	0.6

C

B	C	P(C|B)
0	0	0.7
0	1	0.3
1	0	0.2
1	1	0.8

$P(B \mid A)P(A)$

A	B	P(B|A)P(A)
0	0	0.81
0	1	0.09
1	0	0.04
1	1	0.06

Table 26: Prior probability and conditional probability table for variable elimination example

Step 1: Eliminating A

$\sum_A P(B \mid A)P(A)$

B	$\sum_A P(B \mid A)P(A) = f_A(A,B)$
0	0.85
1	.15

Step 2: Eliminating B

$P(C \mid B) f_A(A,B)$

$f_A(A,B)$	C	$P(C \mid B) f_A(A,B)$
0	0	0.595
0	1	0.255
1	0	0.03
1	1	0.12

Step 3: Eliminating factor (A, B) to get Pr(C)

$\sum_B P(C \mid B) f_A(A,B)$

C	$\sum_B P(C \mid B) f_A(A,B)$
0	0.625
1	0.375

Table 27: Computation steps of variable elimination example

2. Conditioning Method Example

The conditioning method is shown using the example in Figure 63. The prior and conditional probability are in Table 28. The steps are shown in Table 29.

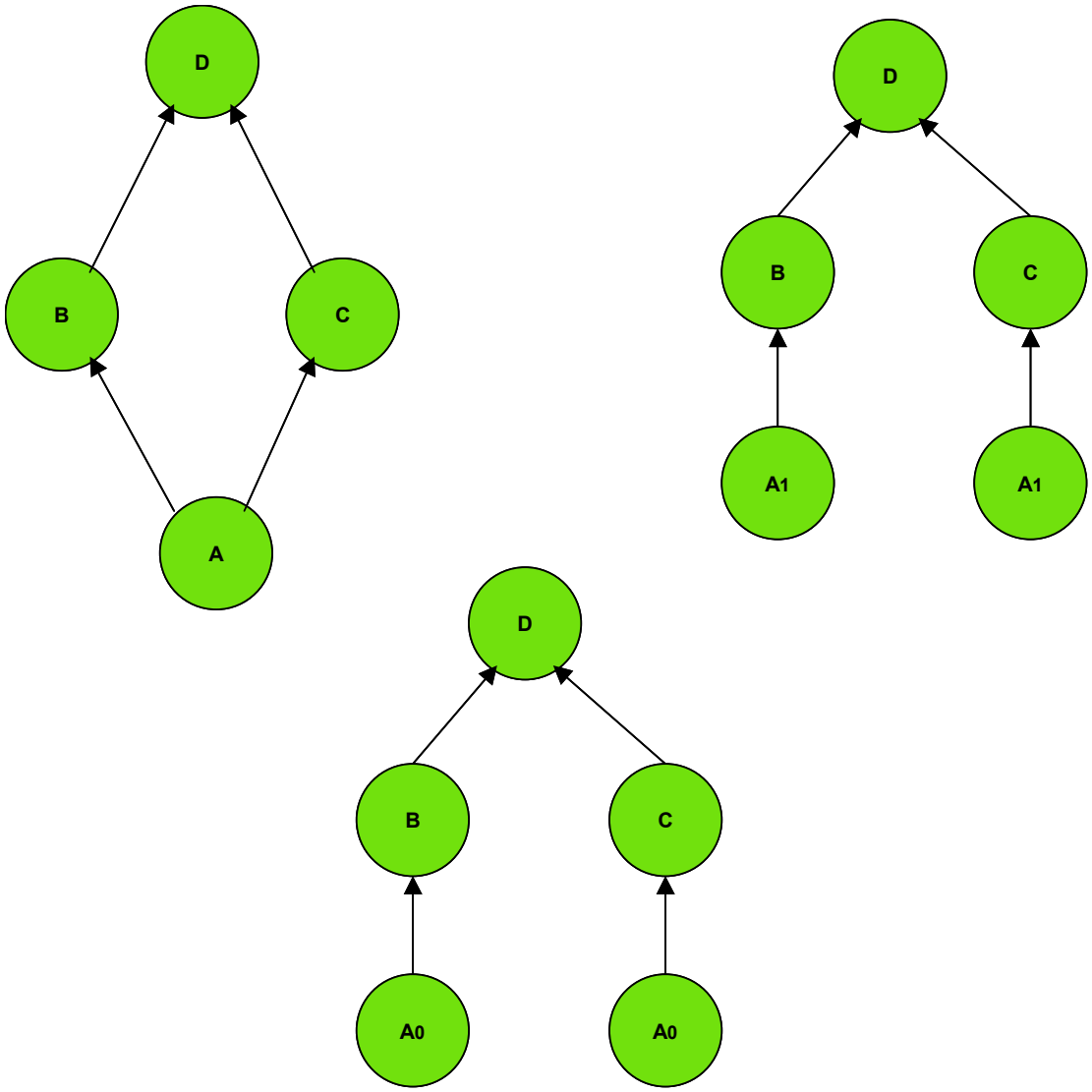


Figure 63: Conditioning algorithm example

A

	P(A)
0	0.9
1	0.1

B

A	B	P(B|A)
0	0	0.9
0	1	0.1
1	0	0.4
1	1	0.6

C

A	C	P(C|A)
0	0	0.7
0	1	0.3
1	0	0.2
1	1	0.8

D

B	C	D	P(D|B,C)
0	0	0	0.9
0	0	1	0.1
0	1	0	0.4
0	1	1	0.6
1	0	0	0.7
1	0	1	0.3
1	1	0	0.2
1	1	1	0.8

Table 28: Prior and conditional probability table for conditioning example

Step 1: Conditioning on A, first considering the left A_0 graph, eliminating A_0

$P(B \mid A_0)P(A_0)$

B	$P(B\|A_0)P(A_0)$
0	0.81
1	0.09

$P(C \mid A_0)P(A_0)$

C	$P(C\|A_0)P(A_0)$
0	0.63
1	0.27

Step 2: Eliminating factor (B,C)

$P(D \mid B,C)P(B,C)$

B	C	D	$P(D\|B,C)P(B,C)$
0	0	0	0.45927
0	0	1	0.05103
0	1	0	0.08748
0	1	1	0.13122
1	0	0	0.03969
1	0	1	0.01701
1	1	0	0.00486
1	1	1	0.01944

D	$P(D\|B,C)P(B,C)$
0	0.5913
1	0.2187

Step 3: Conditioning on A, then considering the right A_1 graph, eliminate A_1

$P(B \mid A_1)P(A_1)$

B	$P(B\|A_1)P(A_1)$
0	0.04
1	0.06

$P(C \mid A)P(A_1)$

C	$P(C\|A_1)P(A_1)$
0	0.02
1	0.08

Step 4: Eliminate factor (B,C) on the right A_1 graph

$P(D \mid B,C)P(B,C)$

B	C	D	$P(D\|B,C)P(B,C)$
0	0	0	0.00072
0	0	1	0.00008
0	1	0	0.00128

0	1	1	0.00192
1	0	0	0.00084
1	0	1	0.00036
1	1	0	0.00096
1	1	1	0.00384

D	P(D|B,C)P(B,C)
0	0.0038
1	0.0062

Step 5: Normalize to get Pr(D). It's done by combining the two networks and normalized according to the each other's prior probability

D	P(D)
0	$\frac{0.5913}{0.9} + \frac{0.0038}{0.1} = 0.695$
1	$\frac{0.2187}{0.9} + \frac{0.0062}{0.1} = 0.305$

Table 29: Computation steps of conditioning algorithm example probability table

3. Junction Tree Method Example

Junction tree algorithm is shown following the flow chart in Figure 64 by using the example in Figure 65. The prior and conditional probabilities are in Table 30. The steps are shown in Table 31.

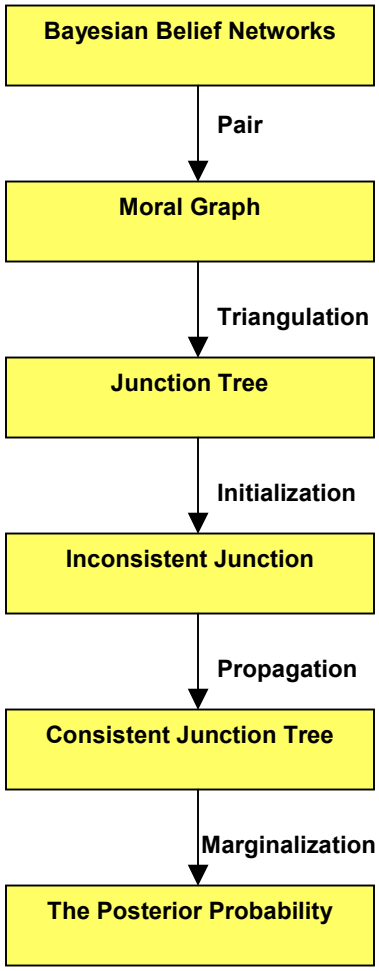


Figure 64: Junction tree algorithm flow chart

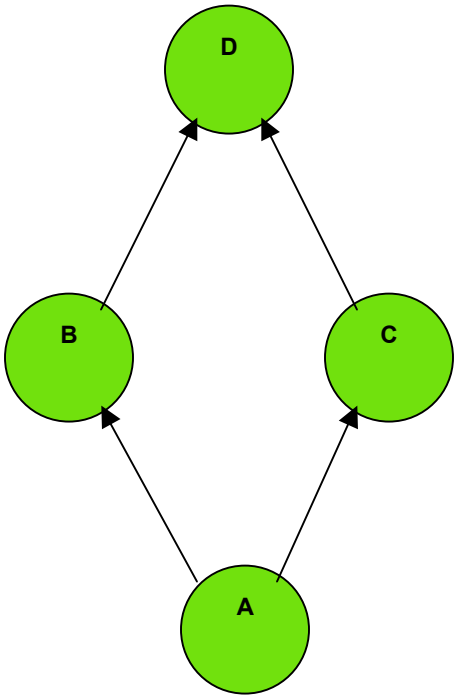


Figure 65: Junction tree algorithm example

A

	P(A)
0	0.9
1	0.1

B

A	B	P(B|A)
0	0	0.9
0	1	0.1
1	0	0.4
1	1	0.6

C

A	C	P(C|A)
0	0	0.7
0	1	0.3
1	0	0.2
1	1	0.8

D

B	C	D	P(D|B,C)
0	0	0	0.9
0	0	1	0.1
0	1	0	0.4
0	1	1	0.6
1	0	0	0.7
1	0	1	0.3
1	1	0	0.2
1	1	1	0.8

Table 30: Prior and conditional probability table for junction tree example

Step 1: Constructing the moral graph

a) Drop the direction

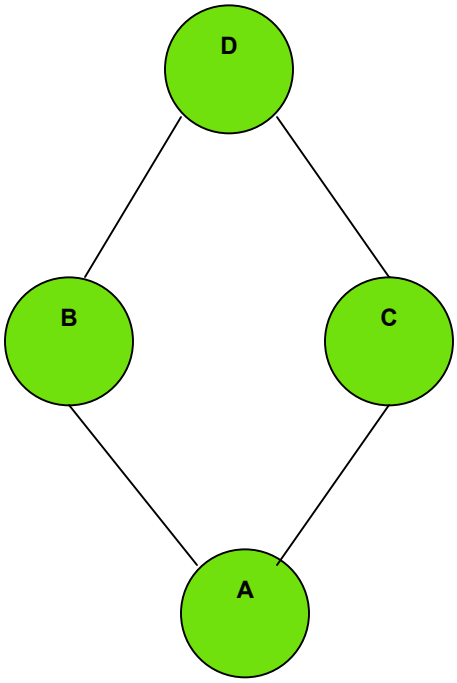


Figure 66: Removing the direction of the graph

b) Pair

For each node, identify its parents. Connect each pair of nodes by adding undirected Arcs.

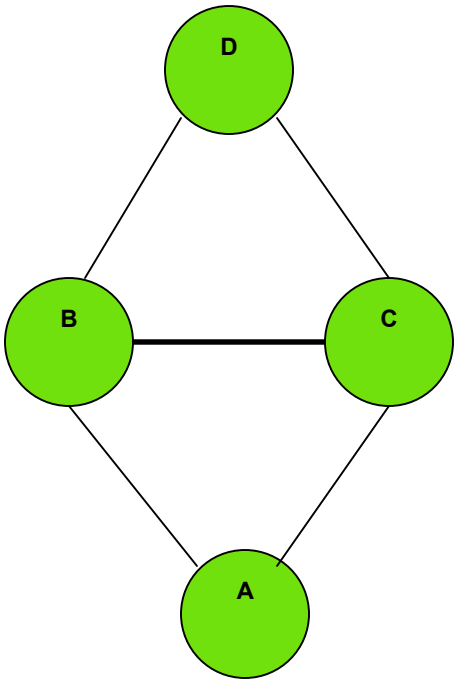


Figure 67: Pair the graph

Step 2: Triangulate the Moral Graph

An undirected graph is triangulated if and only if every cycle of length four or greater contains an edge that connects two nonadjacent nodes in the cycle.

1. This graph doesn't need any more triangulation.
2. Identifying the clique, then the junction tree is successfully built as Figure 68

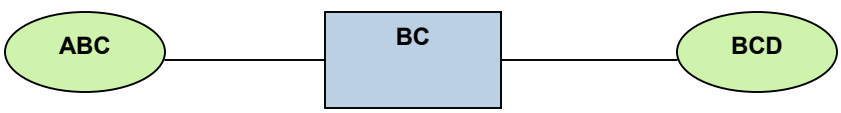


Figure 68: Junction tree built

Step 3: Initialization

A	B	C	ϕABC Initial Value
0	0	0	0.567
0	0	1	0.243
0	1	0	0.063
0	1	1	0.027
1	0	0	0.008
1	0	1	0.032
1	1	0	0.012
1	1	1	0.048

B	C	ϕBCInitial Value
0	0	1
0	1	1
1	0	1
1	1	1

B	C	D	ϕBCD Initial Value
0	0	0	0.9
0	0	1	0.1
0	1	0	0.4
0	1	1	0.6
1	0	0	0.7
1	0	1	0.3
1	1	0	0.2
1	1	1	0.8

Step 4: Global Initialization

1. Select cluster BCD
2. Collect evidence

B	C	ϕBC Updated Value (Collect Evidence)
0	0	0.575

0	1	0.275
1	0	0.075
1	1	0.075

B	C	D	ϕBCD Updated Value (Collect Evidence)
0	0	0	0.5175
0	0	1	0.0575
0	1	0	0.11
0	1	1	0.165
1	0	0	0.0525
1	0	1	0.0225
1	1	0	0.015
1	1	1	0.06

B	C	ϕBC Updated Value (Distribute Evidence)
0	0	0.575
0	1	0.275
1	0	0.075
1	1	0.075

A	B	C	ϕABC Updated Value after Distribute Evidence
0	0	0	0.567
0	0	1	0.243
0	1	0	0.063
0	1	1	0.027
1	0	0	0.008
1	0	1	0.032
1	1	0	0.012
1	1	1	0.048

Step 5: Marginalization to get the probability for all the nodes

A

	P(A)
0	0.9
1	0.1

B

	P(B)
0	0.85
1	0.15

C

	P(C)
0	0.65
1	0.35

D

	P(D)
0	0.695
1	0.305

Table 31: Junction Tree Example Computation Steps

Appendix 2: Conditional Probability Table of the Bayesian Belief Network in Chapter 9

CASE Audits

Standard	0.9445
Substandard	0.0555

Fleet Utilization

Standard	0.8595
Substandard	0.1405

Engineering Procedures

CASSInt	Standard	Substand
Standard	0.9627	0.845
Substandard	0.0373	0.155

SFAR Procedures

CASSInt	Standard	Substand
Standard	0.9623	0.8414
Substandard	0.0377	0.1586

Outsourced Calib. Prop.

Inhouse	0.5542
Outsourced	0.4458

Operator Produced Parts Prop.

Purchased	0.97
Produced	0.03

Purchased Parts, Material

Procurement	Standard		Substandard	
ReceivingInspe	Standard	Substand	Standard	Substand
Standard	0.957	0.9195	0.9195	0.8882
Substandard	0.043	0.0805	0.0805	0.1118

Operator Produced Parts

PurchasedTE	Standard								Substandard		
MechanicActio	safe				Unsafe (true)				safe		
MechanicCTRA	Safe		Unsafe (true)		Safe		Unsafe (true)		Safe		Unsafe (t
CAMPProcedur	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard
Standard	0.928	0.8847	0.8847	0.8415	0.8847	0.8415	0.8415	0.7982	0.8847	0.8415	0.8415
Substandard	0.072	0.1153	0.1153	0.1585	0.1153	0.1585	0.1585	0.2018	0.1153	0.1585	0.1585

PurchasedTE	Substandard				
MechanicActio	safe	Unsafe (true)			
MechanicCTRA	Unsafe (t	Safe		Unsafe (true)	
CAMPProcedur	Substand	Standard	Substand	Standard	Substand
Standard	0.7982	0.8415	0.7982	0.7982	0.755
Substandard	0.2018	0.1585	0.2018	0.2018	0.245

Locally Produced T/E Prop

Purchased	0.918
Locally Prod.	0.082

Purchased Tools, Equipment

OutsourcedCal	Inhouse										
ReceivingInspe	Standard								Substandard		
Procurement	Standard				Substandard				Standard		
OutsourcedCal	Standard		Substandard		Standard		Substandard		Standard		Substand
InHouseCalibT	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard
Standard	0.962	0.9323	0.962	0.9323	0.9323	0.9027	0.9323	0.9027	0.9323	0.9027	0.9323
Substandard	0.038	0.0677	0.038	0.0677	0.0677	0.0973	0.0677	0.0973	0.0677	0.0973	0.0677

OutsourcedCal	Inhouse					Outsourced					
ReceivingInspe	Substandard					Standard					
Procurement	Standard	Substandard				Standard			Substandard		
OutsourcedCal	Substand	Standard		Substandard		Standard		Substandard		Standard	
InHouseCalibT	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.9027	0.9027	0.873	0.9027	0.873	0.962	0.962	0.9323	0.9323	0.9323	0.9323
Substandard	0.0973	0.0973	0.127	0.0973	0.127	0.038	0.038	0.0677	0.0677	0.0677	0.0677

OutsourcedCal	Outsourced									
ReceivingInspe	Standard		Substandard							
Procurement	Substandard		Standard				Substandard			
OutsourcedCal	Substandard		Standard		Substandard		Standard		Substandard	
InHouseCalibT	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.9027	0.9027	0.9323	0.9323	0.9027	0.9027	0.9027	0.9027	0.873	0.873
Substandard	0.0973	0.0973	0.0677	0.0677	0.0973	0.0973	0.0973	0.0973	0.127	0.127

Locally Produced T/E

PurchasedTE	Standard										
InHouseCalibT	Standard								Substandard		
MechanicActio	safe				Unsafe (true)				safe		
MechanicCTR/	Safe		Unsafe (true)		Safe		Unsafe (true)		Safe		Unsafe (t
CAMPProcedur	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard
Standard	0.9707	0.9573	0.9573	0.944	0.9573	0.944	0.944	0.9306	0.9573	0.944	0.944
Substandard	0.0293	0.0427	0.0427	0.056	0.0427	0.056	0.056	0.0694	0.0427	0.056	0.056

PurchasedTE	Standard					Substandard					
InHouseCalibT	Substandard					Standard					
MechanicActio	safe	Unsafe (true)				safe			Unsafe (true)		
MechanicCTR/	Unsafe (t	Safe		Unsafe (true)		Safe		Unsafe (true)		Safe	
CAMPProcedur	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.9306	0.944	0.9306	0.9306	0.9173	0.9573	0.944	0.944	0.9306	0.944	0.9306
Substandard	0.0694	0.056	0.0694	0.0694	0.0827	0.0427	0.056	0.056	0.0694	0.056	0.0694

PurchasedTE	Substandard									
InHouseCalibT	Standard		Substandard							
MechanicActio	Unsafe (true)		safe				Unsafe (true)			
MechanicCTR/	Unsafe (true)		Safe		Unsafe (true)		Safe		Unsafe (true)	
CAMPProcedur	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.9306	0.9173	0.944	0.9306	0.9306	0.9173	0.9306	0.9173	0.9173	0.9039
Substandard	0.0694	0.0827	0.056	0.0694	0.0694	0.0827	0.0694	0.0827	0.0827	0.0961

Parts, Material

OperatorProdP	Purchased				Produced			
PurchasedPart	Standard		Substandard		Standard		Substandard	
OperatorProdu	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Substandard	0.9541	0.9541	0.8144	0.8144	0.9541	0.8144	0.9541	0.8144
Standard	0.0459	0.0459	0.1856	0.1856	0.0459	0.1856	0.0459	0.1856

Through Flight/Daily Checks

MechanicCTRA	Safe				Unsafe (true)			
MechanicActio	safe		Unsafe (true)		safe		Unsafe (true)	
CAMPProcedu	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.9453	0.8965	0.8965	0.8479	0.8965	0.8479	0.8479	0.7992
Substandard	0.0547	0.1035	0.1035	0.1521	0.1035	0.1521	0.1521	0.2008

Outsourced Maintenance

Standard	0.9279
Substandard	0.0721

CASS Procedures

Standard	0.9272
Substandard	0.0728

CASS Staff Actions

Safe	0.9488
Unsafe (true)	0.0512

CASS Internal Auditing/Analysis

CASSProc	Standard		Substandard	
CASSStaff	Safe	Unsafe (t	Safe	Unsafe (t
Standard	0.9518	0.8744	0.8744	0.7971
Substandard	0.0482	0.1256	0.1256	0.2029

CASS External Auditing/Analysis

CASEAudits	Standard				Substandard			
CASSProc	Standard		Substandard		Standard		Substandard	
CASSStaff	Safe	Unsafe (t	Safe	Unsafe (t	Safe	Unsafe (t	Safe	Unsafe (t
Standard	0.9524	0.8951	0.8951	0.8379	0.8951	0.8379	0.8379	0.7806
Substandard	0.0476	0.1049	0.1049	0.1621	0.1049	0.1621	0.1621	0.2194

DAS/DOA Procedures

CASSInt	Standard	Substand
Standard	0.9623	0.8414
Substandard	0.0377	0.1586

Engineering Staff Actions

Safe	0.9576
Unsafe (true)	0.0424

Repair Procedures

EngProc	Standard				Substandard			
SFARProc	Standard		Substandard		Standard		Substandard	
EngStaffActior	Safe	Unsafe (t	Safe	Unsafe (t	Safe	Unsafe (t	Safe	Unsafe (t
Standard	0.9556	0.9131	0.9131	0.8706	0.9131	0.8706	0.8706	0.8281
Substandard	0.0444	0.0869	0.0869	0.1294	0.0869	0.1294	0.1294	0.1719

Availability of Resources

Available	0.8476
NOT Available	0.1524

Planners Actions

Safe	0.9409
Unsafe (true)	0.0591

AD Management

CASSInt	Standard	Substand
Standard	0.9721	0.85
Substandard	0.0279	0.15

Records and Reporting

CASSInt	Standard	Substand
Standard	0.9494	0.7871
Substandard	0.0506	0.2129

Manual Management

CASSInt	Standard	Substand
Standard	0.9606	0.825
Substandard	0.0394	0.175

Alterations

DASProc	Standard		Substandard	
EngStaffActior	Safe	Unsafe (t	Safe	Unsafe (t
Appropriate	0.9587	0.8958	0.8958	0.8329
Inappropriate	0.0413	0.1042	0.1042	0.1671

Inspection Program

EngStaffActior	Safe				Unsafe (true)			
EngProc	Standard		Substandard		Standard		Substandard	
CASSInt	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.9616	0.9251	0.9251	0.8887	0.9251	0.8887	0.8887	0.8522
Substandard	0.0384	0.0749	0.0749	0.1113	0.0749	0.1113	0.1113	0.1478

Mx Schedules

ADManagemer	Standard										
RecordsReport	Standard										
FleetUtilization	Standard								Substandard		
ResourceAvail	Available				NOT Available				Available		
Planners	Safe		Unsafe (true)		Safe		Unsafe (true)		Safe		Unsafe (t
InspectionProg	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard
Standard	0.965	0.9451	0.9451	0.9253	0.9451	0.9253	0.9253	0.9054	0.9451	0.9253	0.9253
Substandard	0.035	0.0549	0.0549	0.0747	0.0549	0.0747	0.0747	0.0946	0.0549	0.0747	0.0747

ADManagemer	Standard										
RecordsReport	Standard					Substandard					
FleetUtilization	Substandard					Standard					
ResourceAvail	Available	NOT Available				Available			NOT Available		
Planners	Unsafe (t	Safe		Unsafe (true)		Safe		Unsafe (true)		Safe	
InspectionProg	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.9054	0.9253	0.9054	0.9054	0.8856	0.9451	0.9253	0.9253	0.9054	0.9253	0.9054
Substandard	0.0946	0.0747	0.0946	0.0946	0.1144	0.0549	0.0747	0.0747	0.0946	0.0747	0.0946

ADManagemer	Standard										Substand
RecordsReport	Substandard										Standard
FleetUtilization	Standard				Substandard						Standard
ResourceAvail	NOT Available			Available				NOT Available			Available
Planners	Unsafe (true)		Safe		Unsafe (true)		Safe		Unsafe (true)		Safe
InspectionProg	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard
Standard	0.9054	0.8856	0.9253	0.9054	0.9054	0.8856	0.9054	0.8856	0.8856	0.8657	0.9451
Substandard	0.0946	0.1144	0.0747	0.0946	0.0946	0.1144	0.0946	0.1144	0.1144	0.1343	0.0549

ADManagemer	Substandard										
RecordsReport	Standard										
FleetUtilization	Standard							Substandard			
ResourceAvail	Available			NOT Available				Available			
Planners	Safe		Unsafe (true)		Safe		Unsafe (true)		Safe		Unsafe (true)
InspectionProg	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.9253	0.9253	0.9054	0.9253	0.9054	0.9054	0.8856	0.9253	0.9054	0.9054	0.8856
Substandard	0.0747	0.0747	0.0946	0.0747	0.0946	0.0946	0.1144	0.0747	0.0946	0.0946	0.1144

ADManagemer	Substandard										
RecordsReport	Standard				Substandard						
FleetUtilization	Substandard				Standard						
ResourceAvail	NOT Available				Available				NOT Available		
Planners	Safe		Unsafe (true)		Safe		Unsafe (true)		Safe		Unsafe (t
InspectionProg	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard
Standard	0.9054	0.8856	0.8856	0.8657	0.9253	0.9054	0.9054	0.8856	0.9054	0.8856	0.8856
Substandard	0.0946	0.1144	0.1144	0.1343	0.0747	0.0946	0.0946	0.1144	0.0946	0.1144	0.1144

ADManagemer	Substandard								
RecordsReport	Substandard								
FleetUtilization	Standard	Substandard							
ResourceAvail	NOT Ava	Available			NOT Available				
Planners	Unsafe (t	Safe		Unsafe (true)	Safe		Unsafe (true)		
InspectionProg	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.8657	0.9054	0.8856	0.8856	0.8657	0.8856	0.8657	0.8657	0.8459
Substandard	0.1343	0.0946	0.1144	0.1144	0.1343	0.1144	0.1343	0.1343	0.1541

Calibration & Test Proedures

CASSInt	Standard	Substand
Standard	0.9651	0.8603
Substandard	0.0349	0.1397

Master Calib. & Test Equipment

Standard	0.962
Substandard	0.038

4th Party CTR ActionsT

Safe	0.9135
Unsafe (true)	0.0865

Technicians Actions

Safe	0.9624
Unsafe (true)	0.0376

Store Staff Actions

Safe	0.9306
Unsafe (true)	0.0694

Parts/Material Control/SUP Procedures

CASSInt	Standard	Substand
Standard	0.9701	0.8547
Substandard	0.0299	0.1453

Procurement

StoreStaffActi	Safe		Unsafe (true)	
PartsMatConSt	Standard	Substand	Standard	Substand
Standard	0.9485	0.8689	0.8689	0.7894
Substandard	0.0515	0.1311	0.1311	0.2106

Outsourced Calibration & Test

Standard	0.952
Substandard	0.048

In-House Calibration & Test

CalibTestProc	Standard								Substandard		
MasterEquip	Standard				Substandard				Standard		
TechCTRActio	Safe		Unsafe (true)		Safe		Unsafe (true)		Safe		Unsafe (t
Technicians	Safe	Unsafe (t	Safe	Unsafe (t	Safe	Unsafe (t	Safe	Unsafe (t	Safe	Unsafe (t	Safe
Standard	0.9653	0.9349	0.9349	0.9045	0.9349	0.9045	0.9045	0.8742	0.9349	0.9045	0.9045
Substandard	0.0347	0.0651	0.0651	0.0955	0.0651	0.0955	0.0955	0.1258	0.0651	0.0955	0.0955

In-House Calibration & Test

CalibTestProc	Substandard				
MasterEquip	Standard	Substandard			
TechCTRActio	Unsafe (t	Safe		Unsafe (true)	
Technicians	Unsafe (t	Safe	Unsafe (t	Safe	Unsafe (t
Standard	0.8742	0.9045	0.8742	0.8742	0.8438
Substandard	0.1258	0.0955	0.1258	0.1258	0.1562

Receiving Inspection Procedures

CASSInt	Standard	Substand
Standard	0.9594	0.8171
Substandard	0.0406	0.1829

Receiving Inspections

PartsMatConSt	Standard								Substandard		
ReceivingInspe	Standard				Substandard				Standard		
InspectorActio	Safe		Unsafe (ture)		Safe		Unsafe (ture)		Safe		Unsafe (t
InspectorCTRA	Safe	Unsafe (t	Safe	Unsafe (t	Safe	Unsafe (t	Safe	Unsafe (t	Safe	Unsafe (t	Safe
Standard	95.34	90.65	90.65	85.96	90.65	85.96	85.96	81.28	90.65	85.96	85.96
Substandard	4.66	9.35	9.35	14.04	9.35	14.04	14.04	18.72	9.35	14.04	14.04

PartsMatConSt	Substandard				
ReceivingInspe	Standard	Substandard			
InspectorActio	Unsafe (t	Safe		Unsafe (ture)	
InspectorCTRA	Unsafe (t	Safe	Unsafe (t	Safe	Unsafe (t
Standard	81.28	85.96	81.28	81.28	76.59
Substandard	18.72	14.04	18.72	18.72	23.41

Tools, Equipment

LocallyProdPro	Purchased				Locally Prod.			
PurchasedTE	Standard		Substandard		Standard		Substandard	
LocallyProduce	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.9438	0.9438	0.8285	0.8285	0.9438	0.8285	0.9438	0.8285
Substandard	0.0562	0.0562	0.1715	0.1715	0.0562	0.1715	0.0562	0.1715

4th Party CTR Actions

Safe	0.8932
Unsafe (true)	0.1068

Technician Actions

safe	0.9224
Unsafe (true)	0.0776

Mx Facility/Base

OK	0.955
NOT OK (true)	0.045

CAMP Procedures

RepairProc	Standard										
ADManagemer	Standard										
RecordsReport	Standard										
ManualManage	Standard								Substandard		
Alterations	Appropriate				Inappropriate				Appropriate		
InspectionProg	Standard		Substandard		Standard		Substandard		Standard		Substand
MXSchedules	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard
Standard	0.945	0.9228	0.9228	0.9006	0.9228	0.9006	0.9006	0.8784	0.9228	0.9006	0.9006
Substandard	0.055	0.0772	0.0772	0.0994	0.0772	0.0994	0.0994	0.1216	0.0772	0.0994	0.0994

RepairProc	Standard										
ADManagemer	Standard										
RecordsReport	Standard						Substandard				
ManualManage	Substandard						Standard				
Alterations	Appropria	Inappropriate				Appropriate			Inappropriate		
InspectionProg	Substand	Standard		Substandard		Standard		Substandard		Standard	
MXSchedules	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.8784	0.9006	0.8784	0.8784	0.8562	0.9228	0.9006	0.9006	0.8784	0.9006	0.8784
Substandard	0.1216	0.0994	0.1216	0.1216	0.1438	0.0772	0.0994	0.0994	0.1216	0.0994	0.1216

RepairProc	Standard										
ADManagemer	Standard										Substand
RecordsReport	Substandard										Standard
ManualManage	Standard		Substandard								Standard
Alterations	Inappropriate		Appropriate				Inappropriate				Appropria
InspectionProg	Substandard		Standard		Substandard		Standard		Substandard		Standard
MXSchedules	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard
Standard	0.8784	0.8562	0.9006	0.8784	0.8784	0.8562	0.8784	0.8562	0.8562	0.834	0.9228
Substandard	0.1216	0.1438	0.0994	0.1216	0.1216	0.1438	0.1216	0.1438	0.1438	0.166	0.0772

RepairProc	Standard										
ADManagemer	Substandard										
RecordsReport	Standard										
ManualManage	Standard								Substandard		
Alterations	Appropriate				Inappropriate				Appropriate		
InspectionProg	Standard		Substandard		Standard		Substandard		Standard		Substandard
MXSchedules	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.9006	0.9006	0.8784	0.9006	0.8784	0.8784	0.8562	0.9006	0.8784	0.8784	0.8562
Substandard	0.0994	0.0994	0.1216	0.0994	0.1216	0.1216	0.1438	0.0994	0.1216	0.1216	0.1438

RepairProc	Standard										
ADManagemer	Substandard										
RecordsReport	Standard						Substandard				
ManualManage	Substandard						Standard				
Alterations	Inappropriate				Appropriate				Inappropriate		
InspectionProg	Standard		Substandard		Standard		Substandard		Standard		Substand
MXSchedules	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard
Standard	0.8784	0.8562	0.8562	0.834	0.9006	0.8784	0.8784	0.8562	0.8784	0.8562	0.8562
Substandard	0.1216	0.1438	0.1438	0.166	0.0994	0.1216	0.1216	0.1438	0.1216	0.1438	0.1438

CAMP Procedures

RepairProc	Standard									Substandard	
ADManagemer	Substandard									Standard	
RecordsReport	Substandard									Standard	
ManualManage	Standard	Substandard								Standard	
Alterations	Inappropr	Appropriate				Inappropriate				Appropriate	
InspectionProg	Substand	Standard		Substandard		Standard		Substandard		Standard	
MXSchedules	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.834	0.8784	0.8562	0.8562	0.834	0.8562	0.834	0.834	0.8118	0.9228	0.9006
Substandard	0.166	0.1216	0.1438	0.1438	0.166	0.1438	0.166	0.166	0.1882	0.0772	0.0994

RepairProc	Substandard										
ADManagemer	Standard										
RecordsReport	Standard										
ManualManage	Standard						Substandard				
Alterations	Appropriate		Inappropriate				Appropriate				Inappropr
InspectionProg	Substandard		Standard		Substandard		Standard		Substandard		Standard
MXSchedules	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard
Standard	0.9006	0.8784	0.9006	0.8784	0.8784	0.8562	0.9006	0.8784	0.8784	0.8562	0.8784
Substandard	0.0994	0.1216	0.0994	0.1216	0.1216	0.1438	0.0994	0.1216	0.1216	0.1438	0.1216

RepairProc	Substandard										
ADManagemer	Standard										
RecordsReport	Standard				Substandard						
ManualManage	Substandard				Standard						
Alterations	Inappropriate				Appropriate				Inappropriate		
InspectionProg	Standard	Substandard		Standard		Substandard		Standard		Substandard	
MXSchedules	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.8562	0.8562	0.834	0.9006	0.8784	0.8784	0.8562	0.8784	0.8562	0.8562	0.834
Substandard	0.1438	0.1438	0.166	0.0994	0.1216	0.1216	0.1438	0.1216	0.1438	0.1438	0.166

RepairProc	Substandard										
ADManagemer	Standard									Substandard	
RecordsReport	Substandard									Standard	
ManualManage	Substandard									Standard	
Alterations	Appropriate				Inappropriate					Appropriate	
InspectionProg	Standard		Substandard		Standard		Substandard		Standard		Substand
MXSchedules	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard
Standard	0.8784	0.8562	0.8562	0.834	0.8562	0.834	0.834	0.8118	0.9006	0.8784	0.8784
Substandard	0.1216	0.1438	0.1438	0.166	0.1438	0.166	0.166	0.1882	0.0994	0.1216	0.1216

RepairProc	Substandard										
ADManagemer	Substandard										
RecordsReport	Standard										
ManualManage	Standard						Substandard				
Alterations	Appropria		Inappropriate				Appropriate				Inappropriate
InspectionProg	Substand	Standard		Substandard		Standard		Substandard		Standard	
MXSchedules	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.8562	0.8784	0.8562	0.8562	0.834	0.8784	0.8562	0.8562	0.834	0.8562	0.834
Substandard	0.1438	0.1216	0.1438	0.1438	0.166	0.1216	0.1438	0.1438	0.166	0.1438	0.166

CAMP Procedures

RepairProc	Substandard										
ADManagemer	Substandard										
RecordsReport	Standard		Substandard								
ManualManage	Substandard		Standard								Substand
Alterations	Inappropriate		Appropriate				Inappropriate				Appropria
InspectionProg	Substandard		Standard		Substandard		Standard		Substandard		Standard
MXSchedules	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard
Standard	0.834	0.8118	0.8784	0.8562	0.8562	0.834	0.8562	0.834	0.834	0.8118	0.8562
Substandard	0.166	0.1882	0.1216	0.1438	0.1438	0.166	0.1438	0.166	0.166	0.1882	0.1438

RepairProc	Substandard						
ADManagemer	Substandard						
RecordsReport	Substandard						
ManualManage	Substandard						
Alterations	Appropriate				Inappropriate		
InspectionProg	Standard		Substandard		Standard		Substandard
MXSchedules	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.834	0.834	0.8118	0.834	0.8118	0.8118	0.7896
Substandard	0.166	0.166	0.1882	0.166	0.1882	0.1882	0.2104

RTSI Procedures

CASSInt	Standard	Substand
Standard	0.9717	0.8633
Substandard	0.0283	0.1367

4th Party CTR ActionsR

Safe	0.9179
Unsafe (true)	0.0821

Inspector Actions

Safe	0.9632
Unsafe (ture)	0.0368

Return To Service Inspection

RTSIProc	Standard				Substandard			
InspectorCTRA	Safe		Unsafe (true)		Safe		Unsafe (true)	
InspectorActio	Safe	Unsafe (t	Safe	Unsafe (t	Safe	Unsafe (t	Safe	Unsafe (t
Standard	97.11	92.19	92.19	87.27	92.19	87.27	87.27	82.35
Substandard (t	2.89	7.81	7.81	12.73	7.81	12.73	12.73	17.65

In-House Maintenance

PartsMaterial	Substandard										
ToolsEquipmer	Standard										
MechanicCTRA	Safe								Unsafe (true)		
MechanicActio	safe				Unsafe (true)				safe		
MxFacilityBas	OK		NOT OK (true)		OK		NOT OK (true)		OK		NOT OK
CAMPProcedur	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard
Standard	0.9599	0.9298	0.9298	0.8997	0.9298	0.8997	0.8997	0.8696	0.9298	0.8997	0.8997
Substandard	0.0401	0.0702	0.0702	0.1003	0.0702	0.1003	0.1003	0.1304	0.0702	0.1003	0.1003

In-House Maintenance

PartsMaterial	Substandard										
ToolsEquipmer	Standard					Substandard					
MechanicCTRA	Unsafe (true)					Safe					
MechanicActio	safe	Unsafe (true)				safe			Unsafe (true)		
MxFacilityBas	NOT OK	OK		NOT OK (true)		OK		NOT OK (true)		OK	
CAMPProcedur	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.8696	0.8997	0.8696	0.8696	0.8396	0.9298	0.8997	0.8997	0.8696	0.8997	0.8696
Substandard	0.1304	0.1003	0.1304	0.1304	0.1604	0.0702	0.1003	0.1003	0.1304	0.1003	0.1304

PartsMaterial	Substandard										Standard
ToolsEquipmer	Substandard										Standard
MechanicCTRA	Safe	Unsafe (true)									Safe
MechanicActio	Unsafe (true)	safe				Unsafe (true)					safe
MxFacilityBas	NOT OK (true)	OK		NOT OK (true)		OK		NOT OK (true)		OK	
CAMPProcedur	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard
Standard	0.8696	0.8396	0.8997	0.8696	0.8696	0.8396	0.8696	0.8396	0.8396	0.8095	0.9298
Substandard	0.1304	0.1604	0.1003	0.1304	0.1304	0.1604	0.1304	0.1604	0.1604	0.1905	0.0702

PartsMaterial	Standard										
ToolsEquipmer	Standard										
MechanicCTRA	Safe						Unsafe (true)				
MechanicActio	safe			Unsafe (true)			safe				
MxFacilityBas	OK	NOT OK (true)		OK		NOT OK (true)		OK		NOT OK (true)	
CAMPProcedur	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.8997	0.8997	0.8696	0.8997	0.8696	0.8696	0.8396	0.8997	0.8696	0.8696	0.8396
Substandard	0.1003	0.1003	0.1304	0.1003	0.1304	0.1304	0.1604	0.1003	0.1304	0.1304	0.1604

PartsMaterial	Standard										
ToolsEquipmer	Standard					Substandard					
MechanicCTRA	Unsafe (true)					Safe					
MechanicActio	Unsafe (true)					safe			Unsafe (true)		
MxFacilityBas	OK		NOT OK (true)		OK		NOT OK (true)		OK		NOT OK
CAMPProcedur	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard
Standard	0.8696	0.8396	0.8396	0.8095	0.8997	0.8696	0.8696	0.8396	0.8696	0.8396	0.8396
Substandard	0.1304	0.1604	0.1604	0.1905	0.1003	0.1304	0.1304	0.1604	0.1304	0.1604	0.1604

PartsMaterial	Standard								
ToolsEquipmer	Substandard								
MechanicCTRA	Safe	Unsafe (true)							
MechanicActio	Unsafe (t	safe				Unsafe (true)			
MxFacilityBas	NOT OK	OK		NOT OK (true)		OK		NOT OK (true)	
CAMPProcedur	Substand	Standard	Substand	Standard	Substand	Standard	Substand	Standard	Substand
Standard	0.8095	0.8696	0.8396	0.8396	0.8095	0.8396	0.8095	0.8095	0.7794
Substandard	0.1905	0.1304	0.1604	0.1604	0.1905	0.1604	0.1905	0.1905	0.2206

Outsourced MX Proportion

Inhouse	0.5313
Outsourced	0.4687

Aircraft Airwothiness

ThroughFlightC	Standard										
OutsourcedMx	Standard								Substandard		
RTSI	Standard				Substandard (true)				Standard		
InHouseMx	Standard		Substandard		Standard		Substandard		Standard		Substand
OutsourceMXP	Inhouse	Outsourc	Inhouse	Outsourc	Inhouse	Outsourc	Inhouse	Outsourc	Inhouse	Outsourc	Inhouse
Standard	0.9521	0.9521	0.8932	0.9521	0.8932	0.8932	0.8342	0.8932	0.9521	0.8932	0.8932
Substandard	0.0479	0.0479	0.1068	0.0479	0.1068	0.1068	0.1658	0.1068	0.0479	0.1068	0.1068

ThroughFlightC	Standard						Substandard				
OutsourcedMx	Substandard						Standard				
RTSI	Standard	Substandard (true)					Standard			Substandard (true)	
InHouseMx	Substand	Standard		Substandard			Standard		Substandard		Standard
OutsourceMXP	Outsourc	Inhouse	Outsourc	Inhouse	Outsourc	Inhouse	Outsourc	Inhouse	Outsourc	Inhouse	Outsourc
Standard	0.8932	0.8932	0.8342	0.8342	0.8342	0.8932	0.8932	0.8342	0.8932	0.8342	0.8342
Substandard	0.1068	0.1068	0.1658	0.1658	0.1658	0.1068	0.1068	0.1658	0.1068	0.1658	0.1658

ThroughFlightC	Substandard									
OutsourcedMx	Standard		Substandard							
RTSI	Substandard (true)		Standard		Substandard (true)					
InHouseMx	Substandard		Standard		Substandard		Standard		Substandard	
OutsourceMXP	Inhouse	Outsourc	Inhouse	Outsourc	Inhouse	Outsourc	Inhouse	Outsourc	Inhouse	Outsourc
Standard	0.7753	0.8342	0.8932	0.8342	0.8342	0.8342	0.8342	0.7753	0.7753	0.7753
Substandard	0.2247	0.1658	0.1068	0.1658	0.1658	0.1658	0.1658	0.2247	0.2247	0.2247

Table 32: Conditional Probability Table of the Bayesian belief network in chapter 9.3 [Eghbali and Mandalapu, 2005]

References

[Ahrens and Dieter] J H Ahrens and U Dieter, Computer Methods for Sampling from Gamma, Beta, Poisson and Binomial Distributions, Computing, Volume 12, 1974, pages 223 - 246.

[Ahrens and Dieter] J H Ahrens and U Dieter, Generating Gamma Variates by a Modified Rejection Technique, Communications of the ACM, Volume 25, Number 1, January 1982, pages 47 - 54.

[Akers, 1978] S.B. Akers, *Binary Decision Diagrams*, IEEE Trans. Computer, C-27, 6(Aug.), 509-516, 1978

[Arnborg et al., 1987] S. Arnborg, D. G. Corneil, A. Proskurowski, *Complexity of findings embedding in a k-tree*. SIAM J. Alg. Disc. Meth., 8(2), 277–284, 1987

[Bartlett and Andrews, 1999] L. M. Bartlett and J. D. Andrews, *Efficient Basic Event Ordering Schemes for Fault Tree Analysis*, Quality and Reliability Engineering International, 5: 95–101 (1999)

[Bartlett and Andrews, 2001] L.M. Bartlett, J.D. Andrews, *An ordering heuristic to develop the binary decision diagram based on structural importance*, Reliability Engineering and System Safety 72 (2001) 31-38

[Bertele and Brioschi, 1972] U. Bertel`e, F. Brioschi, *Nonserial dynamic programming*, Vol. 91 of mathematics in Science and Engineering. Academic Press., 1972

[Boussiou, 1996] M. Boussiou, *An ordering heuristic for building binary decision diagrams from fault trees*, Proc. ARMS'96, Las Vegas, NV, January 1996, pp. 208–214

[Bouissou et al., 1997] M. Bouissou, F. Bruyere, A. Rauzy, *BDD Based Fault-Tree Processing: A Comparison of Variable Ordering Heuristics*. Proceedings of ESREL '97 Conference, August 1997.

[Bobbio et al., 1999] A. Bobbio, L. Portinale, M. Minichino, E. Ciancamerla. *Comparing fault trees and Bayesian networks for dependability analysis*. Proceedings of the 18th International Conference on Computer Safety, Reliability and Security, SAFECOMP99, vol. 1698, 1999. p. 310–22.

[Bobbio et al., 2001] A. Bobbio, L. Portinale, M. Minichino, and E. Ciancamerla. *Improving the analysis of dependable systems by mapping Fault Trees into Bayesian Networks*. Reliability Engineering and System Safety 71 (3), 249–260, (2001)

[Bobbio et al., 2003] A. Bobbio, G. Franceschinis, R. Gaeta, L. Portinale, M. Minichino, and E. Ciancamerla. *Sequential application of heterogeneous models for the safety analysis of a control system: a case study*. Reliability Engineering and System Safety 82 (3), 269–280 (2003).

[Bolling and Wegener, 1996] B. Bolling, I. Wegener, *Improving the Variable Ordering of OBDDs is NP- Complete*, IEEE Transactions on Computers, VOL. 45, NO. 9, September 1996

[Bryant, 1987] R. Bryant, *Graph Based Algorithms for Boolean Function Manipulation*, IEEE Transactions on Computers, 35(8), 677-91 (1987)

[Brace et al., 1990] K. Brace, R. Rudell, & R. Bryant, *Efficient Implementation of a BDD Package*, Paper presented at the 27th ACM/IEEE Design Automation Conference, 1990, IEEE 0738

[Bryant, 1992] R. Bryant, *Symbolic Boolean Manipulation with Ordered Binary-Decision Diagram*, ACM Computing Surveys, Vol. 24, No. 3 September 1992

[Che, Neapolitan, Kenevan and Evens, 1993] P. Che, R.E. Neapolitan, J. Kenevan and M. Evens, An implementation of a method for computing the uncertainty in inferred probabilities in belief networks, in D. Heckerman and A. Mamdami, eds., Uncertainty in Artificial Intelligence (Kaufmann, San Mateo, CA, 1993) 293-300

[Chow and Liu 1968] C.K. Chow and C.N. Liu. Approximating Discrete Probability Distribution with Dependence Trees. *IEEE Trans. On Information Theory* 14(3):462-467. IEEE Press, Piscataway, NJ, USA 1968

[Cowell et al., 1999] R. G. Cowell, A. P. Dawid, S. L. Lauritzen and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems***.** Springer-Verlag. 1999.

[Darwiche, 1995] A. Darwiche, *Conditioning algorithms for exact and approximate inference*, Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, 99-107, 1995.

[Darwiche, 1995] A. Darwiche. *Conditioning methods for exact and approximate inference in causal networks*. Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, Montreal, Morgan Kauffman, August 1995.

[Darwiche, 2001] A. Darwiche, *Recursive conditioning*. In Artificial Intelligence Journal. Vol 125, No 1-2, pages 5-41, 2001.

[David Heckman] Probabilistic similarity networks. *Networks*, 20: 607-636, 1990

[Dechter, 1996] R. Dechter. *Bucket Elimination: a Unifying Framework for Probabilistic Inference*. In proceedings of the 12th Conference on Uncertainty in Artificial Intelligence, PP 211-219, 1996

[Demster 1977] A.P. Dempster, N.M. Laird, and D.B. Rubin. *Maximum likelihood from incomplete data via the em algorithm*. Journal of the Royal Statistical Society, Series B (Methololigical), 39: 1-38, 1977

[Diez, 1996] F.J. Diez, *Local Conditioning in Bayesian Networks*, Artificial Intelligence 87 (1996)1-20

[Dugan et al., 1990] J. B. Dugan, S. J. Bavusoand, M. A. Boyd, *Fault Trees and Sequence Dependencies*, Proc. Reliability and Maintainability Symp., PP286-293, 1990

[Dugan and Pai, 2002] J. B. Dugan and G. J. Pai, "Estimate the Software Reliability in the Absence of Data", NASA OSMA SAS 2002

[Eghbali and Mandalapu, 2005] H. Eghbali, S. Mandalapu, *14 CFR Part 121 AIR CARRIERS' MAINTENANCE OPERATIONS CAUSAL MODEL*, FAA Report, March 2005

[Epstein and Rauzy, 2005] S. Epstein, A. Rauzy, Can we trust PRA? Reliability Engineering and System Safety 88 (2005) 195–205

[Fertig and Breese, 1990] K.W. Fertig and J.S. Breese, Interval influence diagrams, in H. Henrion, R.D. Shachter, L.N. Kanal and J.F. Lemmer, eds., Uncertainty in Artificial Intelligence 5 (North-Holland, Amsterdam, 1990) 149-161

[Fujita et al., 1988] M. Fujita, H. Fujisawa, N. Kawato *Evaluation and improvements of Boolean comparison method based on binary decision diagrams*. IEEE Transactions on Computer Aided Design (Conference), November 1988. p. 2-5.

[Fussel 1975] J. Fussel, How to hand-calculate system reliability characteristics, IEEE Transactions Reliability 1975, R24(3)

[Groen et al., 2002] F. J. Groen, C. S. Smidts, A. Mosleh, , *QRAS - The Quantitative Risk Assessment System,* Reliability Engineering & System Safety, Volume 91, Issue 3, March 2006, Pages 292-304

[Groth 2007] Katrina Groth, *IRIS Integrated Risk Information System Volume 1: User Guide.* University of Maryland, College Park, MD, USA; 2007.

[Hardy 1889] G.F. Hardy. Letter *Insurance Record* (Reprinted in Transactions of Faculty of Actuaries, Vol. 8 1920

[Henley and Kumamoto, 1991]E. Henley, H. Kumamoto, *Reliability Engineering and Risk Assessment*. Englewood Cliffs, NJ.: Prentice-Hall, 1981(Second version in 1991)

[Hewitt and Savage 1995] E. Hewitt and L.J. Savage. "Symmetric Measures on Cartesian Product" *Transactions of the American Mathematical Society*, Vol. 80 1995

[Horvitz, Breese, Heckerman, Hovel, and Rommelse] The Lumiere project: Bayesian user modeling for inferring the goals and needs of software USERS. In proceeding of the 14th Conf. on Uncertainty in AI, pages 256-265, 1998

[Jeffreys 1939] H. Jeffreys, Theory of Probability, Clarendon Press, Oxford

[Jeffreys 1961]Jeffreys H. Theory of probability, 3rd ed. London: Oxdord University Press, 1961

[Jensen et al., 1990] F. V. Jensen, S. Lauritzen and K.G. Olesen. *Bayesian updating in causal probabilistic networks by local computation*. Computational Statistics Quarterly 4:269-282, 1990.

[Jensen, 2001] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer. 2001.

[Johnson 1932] W.E. Johnson, "Probability: the Inductive and Deductive Problem" Mind, Vol. 49

[Kim and Pearl, 1983] J. Kim and J. Pearl, *A computational model for combined causal and diagnostic reasoning in inference systems*. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83), pp. 190-193, Karlsruhe, Germany. Morgan Kaufmann, 1983

[Kim and Seong, 2002] M. C. Kim and P. H. Seong, R*eliability graph with general gates: An intuitive and practical method for system reliability analysis*, Reliability Engineering & System Safety 78(3) (2002), pp. 239-246.

[Kullback and Leibler 1951] S. Kullback and R.A. Leibler. On Information and Sufficiency. *Annals of Mathematical Statistics* 22:79-86. Institute of Mathematical Statistics, Hayward, CA, USA 1951

[Laruritzen, 1996] S. Lauritzen. *Graphical Models*. Oxford. 1996

[Lauritzen and Spiegelhalter, 1988] S. L. Lauritzen and D. J. Spiegelhalter. *Local computations with probabilities on graphical structures and their applications to expert systems*, Proceedings of the Royal Statistical Society, Series B., 50, 154-227, 1988

[Lauritzen and Spiegelhalter 1990] S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert system. In G. Shafer and J. Pearl, editors, Reading in Uncertain Reasoning, Pages 415-448. Kaufmann, San Mateo, CA 1990

[Lee 2001] B.H. Lee, Using Bayes belief networks in industrial FMEA modeling and analysis, Reliability and Maintainability Symposium, 2001. Proceedings, pp 7-15, 2001

[Lerner 2001] U. Lerner, E. Segal, D. Koller, Exact inference in networks with discrete children of continuous parents, in: J. Breese, D. Koller (Eds.), Uncertainty in Artificial Intelligence, Vol. 17, Morgan Kaufmann, San Francisco, CA, 2001, pp. 319–328.

[Lee, 1959] C. Y. LEE, *Representation of Switching Circuits by Binary-Decision Programs*, Bell system Tech J. 38, 4(July), 985-999, 1959

[Li and D'Ambrosio, 1994] Z. Li and B.D. D'Ambrosio. *Efficient Inference in Bayes Networks as a Combinatorial Optimization Problem*. International Journal of Approximate Reasoning, 11, 55-81, 1994

[Malhotra and Trivedi, 1994] M. Malhotra and K. Trivedi, *Power-Hierarchy of Dependability Model Types*, IEEE Trans. Reliability, vol. 43, no. 2, pp. 493-502, Sept. 1994.

[McEliece et al., 1998] R.J. McEliece, D.J.C. MacKay, and J.-F. Cheng, *Turbo decoding as an instance of Pearl's "belief propagation" algorithm*. IEEE Journal on Selected Areas in Communications, vol. 16, no. 2, pp. 140--151, Feb. 1998.

[Minato et al., 1991] S. Minato, N. Ishiura, S. Yajima, *On variable ordering of binary decision diagrams for the application of the multi-level logic synthesis*. Proceedings of the European Conference on Design Automation, 1991. p. 50-4.

[Modarres, et al., 1999] M. Modarres, M. Kaminskiy, V. Krivtsov, *Reliability Engineering and Risk Analysis: A Practical Guide,* CRC Press 1999

[Mosleh et al., 2002] A. Mosleh, P. Rutledge, F. J. Groen, *Quantitative risk assessment system (QRAS) for space mission PRA,* European Space Agency, (Special Publication) ESA SP, n 486, 2002, p 101-107

[Mosleh and Dias, 2003] Mosleh, A., Dias, A, "Towards an Integrated Framework for Aviation Hazard Analysis," University of Maryland Report, Dec. 2003.

[Mosleh, Dias, Eghbali and Fazen, 2004] A. Mosleh, A. Dias, G. Eghbali, K. Fazen, *An integrated framework for identification, classification, and assessment of aviation systems hazard*, Proc. PSAM 7, p2384-2389, Berlin, 2004.

[Mosleh, Groth 2006] A. Mosleh, K. Groth, *Integrated Risk Information System Software Manual*, 2006

[Mosleh, Wang, Groen 2006] A. Mosleh, C. Wang, F. Groen, Hybrid Causal Logic, US patent application, 2006

[Neapolitan, 1990] R. Neapolitan, *Probabilistic Reasoning in Expert Systems*. J. Wiley, (1990).

[Neapolitan and Kenevan, 1991] R.E. Neapolitan and J.R. Kenevan, Investigations of variances in belief networks, in Uncertainty in Artificial Intelligence (North-Holland, Amsterdam, 1991) 232-240

[Nilsson 1998] D. Nilsson, *Finding the M most probable configurations in probabilistic expert systems*, Statistics and Computing 8 (2) (1998) 159–173.

[Pai and Dugan, 2001] G.J. Pai and J.B. Dugan, "Enhancing Software Reliability Estimation Using Bayesian Networks and Fault Trees", 12th IEEE International Symposium on Software Reliability Engineering (Fast Abstract Track), Nov. 2001

[Pearl, 1982] J. Pearl, *Reverend Bayes on inference engines: A distributed hierarchical approach*. In Proceedings of the National Conference on Artificial Intelligence (AAAI-82), pages 133-1, Pittsburgh, Pennsylvania. Morgan Kaufmann

[Pearl, 1986-1] J. Pearl. *A constraint-propagation approach to probabilistic reasoning*. In: L. N. Kanal and J. F. Lemmer (eds), Uncertainty in Artificial Intelligence, Amsterdam, NorthHolland, 357-370, 1986.

[Pearl, 1986-2] J. Pearl. *Fusion, propagation and structuring in belief networks*. *Artificial Intelligence*, Vol. 29, No. 3, 241-288, September 1986.

[Pear, 2001] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann. 1988, revised second printing 2001

[Rai et al., 1995] S. Rai, M. Veeraraghavan, and K. Trivedi, *A Survey on Efficient Computation of Reliability Using Disjoint Products Approach*, Networks, Vol. 25, No 3, PP147-163, 1995

[Russell and Norvig, 2003] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall. 1995, second edition 2003

[Peot and Shachter, 1991] M. A. Peot and R. D Shachter, *Fusion and propagation with multiple observations in belief networks*, Artificial Intelligence, 48(3), 299-318, 1991.

[Rauzy, 1993] A. Rauzy, *New Algorithms for Fault Trees Analysis*, Reliability Engineering and System Safety 40(1993), 203-211

[Rauzy and Dutit, 1997] A. Rauzy &Y. Dutit, *Exact and Truncated Computations of Prime Implicants of Coherent and Non-coherent Fault Trees within Aralia*, Reliability Engineering and System Safety 58 (1997) 127-144

[Russell and Norvig 2003] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Second Edition, Prentice Hall, 2003

[Shachter et al., 1994] R. D. Shachter, S. K. Anderson, and P. Szolovits. *Global conditioning for prababilistic inference in belief networks*. Proceedings of the Uncertainty in AI Conference 1994, San Francisco, CA:Morgan Kaufmann, 514—522, 1994.

[Shachter et al., 1990] R. D. Shachter, B. D'Ambrosio, and B. A. Del Favero. *Symbolic Probabilistic Inference in Belief Networks*. In Proc. Conf. on Uncertainty in AI, PP 126-131, 1990

[Shenoy and Shafer, 1990] P. Shenoy and G. Shafer. *Axioms for Probability and Belief-Function Propagation*. In Shachter, R. et al., eds., Uncertainty in AI, volume 4, 169-198, 1990.

[Sinnamon and Andrews, 1997] R. M. Sinnamon, and J. D. Andrews, *Improved Accuracy in Quantitative Fault Tree Analysis*, Quality and Reliability Engineering International, Vol 13, 1997, PP 299-309

[Sinnamon and Andrews, 1997] R. M. Sinnamon, and J. D. Andrews, *Improved Efficiency in Qualitative Fault Tree Analysis*, Quality and Reliability Engineering International, Vol 13, 1997, PP285-292

[Solano-Soto and Sucar, 2001] J. Solano-Soto and L. Sucar (2001). *A methodology for reliable system design*. In Lecture Notes in Computer Science, Volume 2070, pp. 734–745. Springer.

[Spiegelhalter 1991] D.J. Spiegelhalter, A unified approach to imprecision and sensitivity of beliefs in expert systems, MCR Biostatistics Unit, Cambridge 1991

[Tarjan and Yannakakis, 1984] R. E. Tarjan, M. Yannakakis, S*imple linear time algorithm to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs*. SIAM J. Comput., 13, 566–579., 1984

[Torres-Toledano and Sucar, 1998] J. Torres-Toledano and L. Sucar (1998). *Bayesian networks for reliability analysis of complex systems*. In Lecture Notes in Artificial Intelligence 1484. Springer Verlag.

[Veeraraghavan and Trivedi, 1994] M. Veeraraghavan and K.S. Trivedi, *Combinatorial Algorithm for Performance and Reliability Analysis Using Multistate Models*, IEEE Trans. Computers, vol. 43, no. 2, pp. 229-234, Feb.1994.

[Weiss, 2000] Y. Weiss,, *Correctness of Local Probability Propagation in Graphical Models with Loops*, Neural Computation 12, 1-41(2000)

[Whitworth 1897] W.A. Whitworth. *DCC Exercises in Choice and Chance* (Reprinted 1965, Hafner, New York)

[Whittaker 1990] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. J. Wiley & Sons, Chichester, United Kingdom 1990

[Wood, 1985] A. P. Wood, *Multistate Block Diagrams and Fault Trees*, IEEE Trans. Reliability, vol. 34, pp. 236-240, 1985.

[Yu et al., 1994] K. Yu, I. Koren, and Y. Guo, *Generalized Multistate Monotone Coherent Systems*, IEEE Trans. Reliability, vol. 43, no. 2, pp. 242-250, 1994.

[Zabell 1982] S.L. Zabell. W.E. Johnson's "Sufficientness Postulate", *The Annals of Statistics*, Vol. 10, No. 4

[Zang et al., 2003] X. Zang, D.Wang, H. Sun, and K. Trivedi, *A BDD-Based Algorithm for Analysis of Multistate Systems with Multistate Components*, IEEE Trans. Computers, Vol. 52, No 12, PP.1608-1618, Dec. 2003

[Zhang and Poole, 1994] N. L. Zhang and D. Poole. *A simple approach to Bayesian network computations*. In Proc. of the Tenth Canadian Conference on Artificial Intelligence, pages 171--178, 1994

[Zhang and Poole, 1996] N. L. Zhang and D. Poole. *Exploiting Causal Independence in Bayesian Network Inference*. Journal of Artificial Intelligence Research, 5:301-328, 1996