Reasoning with Incomplete Probabilistic Knowledge: The RIP Algorithm for de Finetti's Fundamental Theorem of Probability

by

Tracy S. Myers

B.A., University of California, Irvine (1989)

Submitted to the Department of Brain and Cognitive Sciences
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1995

Signature of Author .....
Department of Brain and Cognitive Sciences
May 15, 1995

Certified by .....
Gordon Kaufman, Ph.D.
Professor of Management
Alfred P. Sloan School of Management
Thesis Supervisor

Accepted by .....
Professor Gerald E. Schneider
Chairman, Graduate Committee, Department of Brain and Cognitive Sciences

Reasoning with Incomplete Probabilistic Knowledge: The RIP Algorithm for de Finetti's Fundamental Theorem of Probability

by

Tracy S. Myers

Submitted to the Department of Brain and Cognitive Sciences
on May 15, 1995, in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Abstract

De Finetti's Fundamental Theorem of Probability (FTP) addresses the question of probabilistic inference when the specification of a probability distribution over the sample space is incomplete. The FTP provides a conceptually simple mechanism to compute bounds on the probability of events, and provides necessary and sufficient conditions for probability assessments to be coherent. Lad, Dickey, and Rahman (1990) restated de Finetti's theorem in terms of linear programs providing a paradigm for coherent probability assessment and inference. In practice the linear programs prescribed by the FTP are intractable even for small problems using conventional linear-programming algorithms. Constructing and solving the linear programs prescribed by the FTP is an *NP*–complete problem. The RIP algorithm is a column–generation derivation of the simplex method for linear programming. It solves the master linear program prescribed by the FTP without explicitly constructing it. Instead the RIP algorithm maintains a current basis for the master linear program and employs a related mixed–integer program to generate the column to enter the basis at each iteration. The related integer program is constructed such that 1) the set of feasible solutions is exactly the set of columns of the master linear program; and 2) the optimal solution to the related integer program is the column to enter the basis. For moderately large practical problems the RIP algorithm is an effective computational paradigm for the theoretical framework established by de Finetti's fundamental theorem of probability including Lad, Dickey, and Rahman's extensions. The algorithm provides practical tools for parsing a specification of probability assessments, constructing the linear and mixed–integer programs, and solving the system for the desired probability bounds.

Thesis Supervisor: Gordon Kaufman, Ph.D.
Title: Professor of Management
Alfred P. Sloan School of Management

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank all of the people who have helped me during the course of my education. Most of all my wife Tina has patiently taken on far more than her fair share of things while I have be absorbed in this project. Likewise my daughter Elisabeth who can't quite understand why it takes so long to write a paper has missed many adventures with me since I have become immersed in writing. My dog Sadie who worries until I get home has suffered many days without a long walk in the woods. Finally I would like to thank my parents who encouraged me every step of the way.

I have benefited from contact with many colleagues throughout my education. William Faust and William Banks at Pomona college kindled my interest in the science of minds and brains. At UCI Myron Braunstein provided support and encouragement while I was an undergraduate at UCI. I matured tremendously as a scientist while working in his laboratory. I would like to thank my advisor Gordon Kaufman who never let sloppy mathematics creep into my thesis. Gordon has been of tremendous help in bringing closure to this project. Gordon's excellent feedback has greatly improved my writing and his many ideas were critical to the completion of this work. Likewise many key ideas in this thesis are due directly or indirectly to Rob Freund who always seems to have a clever way to represent just about anything in a linear program. I would like to thank Michael Jordan and Molly Potter for their service on my committee and for being the point of contact with the department of Brain and Cognitive Sciences. Jan Ellertsen, Eleanor Bonsaint, Greta Buck, Robin Nahmias-Fincke, and Pat Claffey constantly helped me get things done, and went out of their way to assist me while I have been off campus. This work has also benfited greatly from many e-mail discussions with Frank Lad over the years. Frank patiently provided detailed explanations for my many questions and provided me with several advance drafts of his text. Portions of this research were conducted while I author was supported by an Office of Naval Research Graduate Fellowship.

Trademarks and copyrights The RIP algorithm was implemented using CPLEX optimization software. CPLEX and the CPLEX callable library are trademarks of CPLEX Optimization, Inc (CPLEX Inc, 1994). The MATHEMATICA software package was also used for some of the optimizations and for producing the 3-D surface graphs. MATHEMATICA is a trademark of Wolfram Research, Inc (Wolfram Research Inc., 1992). The system parser and auxiliary tools were written in PERL v4.0x which was developed by Larry Wall who retains the copyright. PERL is distributed under the GNU license. The MATLAB toolkit was also used. MATLAB is a trademark of The MathWorks, Inc.

All of the computations were performed on Sun SPARC workstations running the SunOS v4.1.x version of UNIX. The RIP algorithm was implemented in C++ using the SunSoft SunPro C++ v3.x compiler and the SunPro development environment. Sun, Sparc, Sparcstation, SunOS, SunPro, and SunSoft are trademarks of Sun Microsystems, Inc. UNIX is a trademark of X/Open, Inc.

Models and data used in this report The source of the systems studied in this report are cited where they are introduced and discussed. These systems provide a concrete context for examining the FTP. The focus of this research is the FTP and the RIP algorithm and not on accurate modeling of any of these particular systems. In most of the examples, the logical structure of the system was derived from the source while the probability data was specifically constructed for the example. In those cases where probability data was reported in a form that could be applied to the present purposes that data was used. In other cases the examples derived from the literature did not report appropriate probability data for the purposes of this work. *The probability assessments used in these examples were constructed to illustrate interesting features of the fundamental theorem of probability and the* RIP *algorithm rather than to model accurately the actual systems.* Where actual data is used, it is specifically noted and the source cited.

In particular most of the fault trees are based on Norman Rassmussen's seminal nuclear reactor safety study (United States Nuclear Regulatory Commission, 1974). The probability assessments used in these examples were constructed specifically for the examples and are not based on the data reported in the study. The actual failure rates for many of the events in these fault trees can range from 1×10^{-9} to 1×10^{-4} per year. The extreme range of these values leads to examples which fail to showcase the interesting features of the FTP.

Likewise Henley and Kummamoto (1981) did not provide any reliability data for

the pumping system example. The PROSPECTOR example is based on a critique of rule–based expert systems in Neopolitan's (1990) text on causal networks.

Chapter 1

Introduction

Knowing ignorance is strength.
Ignoring knowledge is sickness.
— Lao Tsu, *Tao Te Ching*, 6th century B.C.

1.1 Probability assessment and inference

One of the most difficult problems in the development of probabilistic expert systems is specifying the necessary probabilities. The problem of probability assessment has been studied from many perspectives including psychology, decision-analysis, and artificial-intelligence. We can assign paradigms for the study of probability assessment into two broad classes, *prescriptive* and *descriptive*. Prescriptive or *normative* analysis of probabilistic assessment addresses the question of how an assessor should assess uncertain events if he or she is to adhere to the axioms of subjective probability. Descriptive analysis addresses how assessors actually behave in assessing probabilities of uncertain events. The goal of this thesis is to take a prescriptive result, de Finetti's fundamental theorem of probability, and develop it into a useful tool for supporting an assessor's efforts to comply with the prescriptions of subjective probability axioms in realistic problem settings. The value of the fundamental theorem of probability as an assessment tool is limited by the difficult computational problems it engenders. Here efficient algorithms for implementing the fundamental theorem of probability are developed, and its utility as an assessment tool is critically examined.

De Finetti's fundamental theorem of probability (FTP) employs the law of total probability to determine bounds on the probability of an uncertain event when this probability cannot be directly computed from the assessments asserted by the as-

sessor. The theorem is designed to provide these bounds when the specification of a probability distribution over the (finite) sample space which contains the event is incomplete. These bounds are determined by *logical relations* among events in the sample space as well as by the (possibly incomplete) *appraisal* or *assessment* of probabilities of events contained within the sample space. The FTP is a natural tool for policing coherence of a priori assignments of probabilities to events. In particular, it provides necessary and sufficient conditions for coherent assignment of probabilities for any event in a finite sample space.

Recent work by Lad, Dickey, and Rahman (1990,1991) has made it clear that the FTP applied to events defined by a finite partition of any sample space can be interpreted as a linear programming problem, or in the case of conditional probability assessments, as a linear-fractional problem. If we construe the FTP as a linear programming problem, then we immediately benefit from extensive knowledge of the behavior of systems of linear inequalities. For instance, the simplex algorithm for linear programming guarantees that we will either find the optimal solution or determine that the linear program is infeasible in finite time. The work of Lad, Dickey, and Rahman is primarily of conceptual value because the FTP prescribes linear programming problems which are intractable by conventional linear programming algorithms such as the simplex method. If we were to assess as few as 100 events, then the linear program prescribed by the FTP could have more decision variables than there are elementary particles in the universe!

Because de Finetti's theorem affords such a conceptually simple device for computing coherent bounds on the probability of events, we are motivated to develop procedures which can effectively solve the linear programming problems prescribed by the theorem. In fact we will exploit the specific form of the linear programs resulting from the FTP and techniques for representing the logic of complex systems to develop an algorithm that can surmount the computational difficulties of conventional algorithms.

The methods developed in this thesis have been successfully applied to problems with 36 dichotomous states. This leads to a sample space with as many as $2^{36} > 6.8 \times 10^{10}$ events. Problems of this magnitude can be comfortably solved on a typical mid-range UNIX workstation. The full range of implementation issues concerning the robustness of the algorithm with respect to graceful error recovery, degenerate solutions, ill-conditioned floating point arithmetic, other similar exceptional cases, or computational resource optimization are not addressed here. Some of these issues, detecting and handling ill-conditioned floating point arithmetic for example,

involve implementing well-known standard procedures in the current context. Other issues beyond the scope of this thesis, such as minimizing computational resource requirements and handling exceptional cases, are interesting in their own right. When these implementation issues are resolved, the capacity for solving systems with very large numbers of dichotomous states will be greatly enhanced.

The reader should recognize that even with as few as 12 dichotomous states, novel algorithms are necessary. Standard linear programming algorithms require inordinate computational resources even for these small problems [1]. Moreover, standard techniques for large-scale numerical computation, particularly sparse representation of problem data, offer no help[2].

In order to overcome the limitations of conventional linear programming algorithms, it was necessary to develop an algorithm for constructing and optimizing the linear programs that requires computational resources proportional to the number of dichotomous states of interest rather than proportional to the size of the sample space. The FTP yields linear programming problems of a very special form. An adaptation of the simplex algorithm to this specific form needs access only to a small subset of the problem data during much of its operation. In fact it requires access to the entire problem data set only when it calculates a cost statistic for each of the variables in the problem. The computation of the cost statistic in the conventional simplex algorithm is done in the RIP algorithm by solving a *related integer problem* (RIP). This problem is a mixed-integer linear program constructed in such a way that its optimal solution is the needed cost statistic. This two–step procedure makes an almost impossible computation possible.

Several simple examples of small dimension set the stage for the analysis of more realistic complex problems. These small examples provide a concrete illustration of how the FTP interacts with the specified problem logic. Chapter 2 discusses the mathematical foundation for the FTP and Lad, Dickey, and Rahman's extensions of it. Chapter 2 also illustrates how even fairly small examples quickly overwhelm both conventional LP algorithms and the assessor's ability to manually construct the FTP problems from the system logic. Chapter 3 begins with a close look at the linear programs dictated by the FTP and exactly what makes them intractable, then develops an algorithm for this class of problems. Chapter 4 discusses the issues related to implementing the algorithm including enhancements that improve performance. In

[1] See example (2.25) in chapter 2.

[2] Sparse matrix representation actually increases the storage requirements since the FTP problems have constraint matrices that often exceed 50% or even 75% density.

particular Chapter 4 discusses automated methods for constructing and solving the linear problems for the FTP directly from a specification of problem logic. It is at least as difficult a task to construct the linear programs as it is to solve them. Conventional methods for constructing these problems are overwhelmed even more quickly than are the algorithms for optimizing them. Finally chapter 5 discusses basic limitations of the paradigm, addresses the scalability of the algorithm, and suggests avenues of future research.

1.2 Probability models

Let $\Omega = \{\mathsf{e}_1, \ldots, \mathsf{e}_\eta\}$ be a finite sample space composed of η atomic events e_i such that $\mathsf{e}_i \cup \mathsf{e}_j = \emptyset$ for distinct $i, j \in \{1, 2, \ldots, \eta\}$ and $\bigcup_{i=1}^{\eta} \mathsf{e}_i = \Omega$. Let x_i be a generic event in 2^Ω (i.e., a union of atomic events in Ω) and $\neg\mathsf{x}_i$ be its complement[3]. In order to represent Boolean relations in terms of ordinary arithmetic it will be convenient to associate an indicator function I_{x_i} with x_i

$$(1.1) \qquad I_{\mathsf{x}_i} = \begin{cases} 1 & \text{if } \mathsf{x}_i \text{ obtains;} \\ 0 & \text{if } \neg\mathsf{x}_i \text{ obtains} \end{cases}$$

and further to define elliptically $\mathsf{x}_i = I_{\mathsf{x}_i}$; namely $\mathsf{x}_i = 1$ if $I_{\mathsf{x}_i} = 1$ and $\mathsf{x}_i = 0$ if $I_{\mathsf{x}_i} = 0$.

An event x_i partitions Ω into 2 subsets, x_i and $\neg\mathsf{x}_i$. Often we would like to partition Ω into some finite number of subsets. This is accomplished with a *propositional variable* $\mathsf{a}_\mathcal{X}$ which maps a set of propositions $\mathcal{X} = \{\mathsf{x}_1, \ldots, \mathsf{x}_N\}$ into a partition of Ω,

$$\mathsf{a}_\mathcal{X} = \{\mathsf{x}_1 = x_1, \ldots, \mathsf{x}_N = x_N \,|\, x_i \in \{0, 1\}\}.$$

A function $P : 2^\Omega \to [0, 1]$ such that

(a) $P(\Omega) = 1$,

(b) $P(\mathsf{x}_i) \geq 0$ for all $\mathsf{x}_i \in 2^\Omega$, and

(c) $P(\mathsf{x}_i \cup \mathsf{x}_j) = P(\mathsf{x}_i) + P(\mathsf{x}_j)$ if $\mathsf{x}_i \cap \mathsf{x}_j = \emptyset$,

is a *probability measure* on Ω, and $P(\mathsf{x}_i)$ is called the *probability* of x_i. Let the class of probability measures on Ω be $\mathcal{P}$.

[3]Depending on the context we will also use the term *Boolean proposition* for event. Events and Boolean propositions are denoted by lower-case sans-serif type (e.g., x and y). Vectors of events or Boolean propositions are denoted by upper-case sans-serif type (e.g., X and Y).

In the preceding discussion, it was shown how propositions can be defined in terms of a sample space Ω. In practical applications one is typically faced with the converse situation, namely one has some propositions or events of interest and would like to define a sample space on these propositions. Any vector of N Boolean propositions $\mathsf{X}^T = [\mathsf{x}_1, \ldots, \mathsf{x}_N]^T$ can be used to construct a sample space

$$\Omega_{\mathsf{X}} = \{\chi_1 \wedge \ldots \wedge \chi_N | \chi_i \in \{\mathsf{x}_i, \neg \mathsf{x}_i\}\}. \tag{1.2}$$

If there are N Boolean propositions of interest, then there are 2^N elements in Ω_{X}. Each x_i is a generic event in Ω_{X}. Each joint event e in Ω_{X} corresponds to a distinct assignment of truth-values to the propositions in X. Thus, e is a possible state of the system under study and is often referred to as *possible world*[4]. The sample space Ω_{X} is the enumeration of all possible truth–valuations of the propositions in X or the set of all possible worlds. De Finetti used the term *realm* of X to refer to Ω_{X} and the term *constituent* to refer an atomic event e in Ω_{X}. For finite $\Omega_{\mathsf{X}} = \{\mathsf{e}_1, \ldots, \mathsf{e}_\eta\}$ constructed as in (1.2) we can enumerate a table where the columns correspond to the atomic events and the rows correspond to the generic events $\mathsf{x}_i \in \mathsf{X}$.

(1.3)

	e_1	e_2	$\cdots$	$\mathsf{e}_{\eta-1}$	e_η
x_1	1	1	$\cdots$	0	0
x_2	1	1	$\cdots$	0	0
$\vdots$					
x_N	1	0	$\cdots$	1	0

This table enumerates the occurrence or non-occurrence of each of the generic events x_i in each of the atomic events e_j. We can view (1.3) as a matrix that expresses generic events as linear combinations of atomic events. De Finetti coined the term *realm matrix* for (1.3). The columns of (1.3) correspond to atomic events in Ω_{X} that can be arranged in a vector $\mathsf{E} = [\mathsf{e}_1, \ldots, \mathsf{e}_\eta]^T$ which de Finetti called a *constituent vector.*

Given a set of propositions $\mathcal{X} = \{\mathsf{x}_1, \ldots, \mathsf{x}_N\}$, denote the closure of $\mathcal{X}$ under finite conjunctions and negation by $\mathcal{L}_{\mathcal{X}}$. The set $\mathcal{L}_{\mathcal{X}} = \{\mathsf{s}_1, \mathsf{s}_2, \ldots\}$ is called a *language* for $\mathcal{X}$ and the s_i are called *sentences* of $\mathcal{L}$.

[4]In general the concepts addressed in this paper have analogous terms in statistics, logic, and in de Finetti's development of probability. For example *joint event*, *possible world*, and *constituent* each refer to a possible state of the system under study. De Finetti's deviation from the standard vocabulary of probability theory was in fact quite deliberate and is one reason his work has not received wider attention. Where possible the terms most relevant to the context and in common usage will be employed (e.g. *joint event* vs *constituent*).

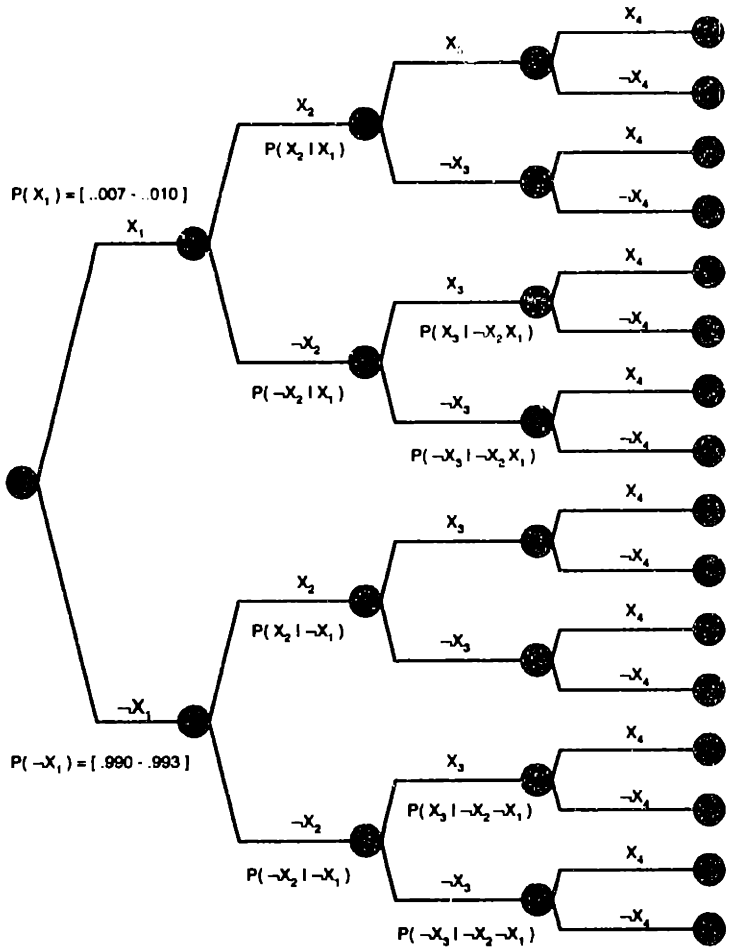

Figure 1.1: The probability tree for the safety system example.

Let P_{X} be a probability distribution for Ω_{X}. Then P_{X} is called a *joint probability distribution* for X. The distribution P_{X} is a *complete probabilistic model* for X, because any question about the probability of the generic events x_i is determined by their joint distribution; however, it is not practical to assess or work with the joint distribution directly. If there are 24 generic events of interest then their joint distribution contains $2^{24} > 16.7 \times 10^6$ atomic events in P_{X}. Depending on the inferences one wants to draw and the logical and stochastic structure of the domain, it may be necessary to specify probabilities on only a subset of the sample space.

(1.4) EXAMPLE: **A Safety System Example.** Consider a small part of a safety system. There are 4 generic events, $\mathsf{x}_1, \ldots, \mathsf{x}_4$ as shown in Table 1.1. If we desire to answer any probability question about $\mathsf{x}_1, \ldots, \mathsf{x}_4$, then we would need to make $2^4 = 16$ assessments. On the other hand if we are only interested in the probability that the system will fail, $P(\mathsf{x}_4)$, then how many assessments must we make? The system will fail just in case

Event	Description	Relation
x_1	Sprinkler Head Clogs	—
x_2	Sprinkler Head Bursts	—
x_3	Insufficient Water Pressure	$x_3 \leq 1 - x_2$
x_4	The System Fails	$x_4 = x_1 \vee x_2 \vee x_3$ or equivalently $x_4 = 1 - (1 - x_1)(1 - x_2)(1 - x_3)$

Table 1.1: The safety system example. (Adapted from Dickey, 1991.)

one or more of x_1, x_2, or x_3 occurs. If we cannot assess $P(x_4)$ directly, then we can economically calculate $P(x_4)$ indirectly using

$$P(\neg x_4) = 1 - P(x_4). \tag{1.5}$$

Figure 1.1 shows a probability tree for the 4 generic events. The path at the bottom of the figure corresponds to $P(\neg x_4)$, and we need to make only 3 assessments of the form

$$P(\neg x_4) = P(\neg x_1)P(\neg x_2|\neg x_1)P(\neg x_3|\neg x_2\neg x_1) \tag{1.6}$$

to calculate $P(x_4)$.

In example (1.4) the number of assessments needed to determine an answer to our question was quite small. This is not true in general, and in a worst case scenario we would be forced to assess every element in the joint distribution. Notice that it is logically possible either to assess the joint distribution directly or to assess it indirectly via a set of conditional probabilities as in (1.6). Direct assessment requires an assessor to assign probabilities to terminal nodes (leaves) of the probability tree in Figure 1.1. Alternatively conditional probability assessment corresponds to assigning conditional probabilities to graph edges in Figure 1.1. Human experts are usually more comfortable making conditional probability assessments (Heckerman, 1991; Pearl, 1988; Neopolitan, 1990).

1.2.1 Lotteries and subjective probability

De Finetti derived probability theory from a few simple axioms concerning lotteries and a definition of rational behavior. The discussion follows de Finetti (1964) and Lad (1993). In this framework probabilities are subjective and correspond to the measure of belief an *agent* or *assessor*, $\mathcal{A}$, associates with some generic event $x \in 2^{\Omega}$. An

explicit assumption about the agent in de Finetti's axiomatization is that the agent is *risk-neutral* which is to say that for the agent utility equals expected monetary value[5].

For $\mathcal{A}$ de Finetti defines probability as follows.

(1.7) DEFINITION: Let $\mathbf{x}$ be an event defined for some sample space Ω. Then the probability of $\mathbf{x}$, $P(\mathbf{x})$, is the fair price for $\mathcal{A}$ of a lottery ℓ which pays \$1 if $\mathbf{x}$ obtains and \$0 otherwise.

Let $P(\mathbf{x})$ be the fair price of the lottery ℓ in (1.7), then $P(\mathbf{x})S_{\mathbf{x}}$ is the fair price for $\mathcal{A}$ of a lottery which pays $\$S_{\mathbf{x}}$ if $\mathbf{x}$ obtains and \$0 otherwise for any scalar $S_{\mathbf{x}}$. The scaling factor $S_{\mathbf{x}}$ is called the *stakes* of the lottery ℓ. The $\mathcal{A}$ contributes $\$P(\mathbf{x})S_{\mathbf{x}}$ of the stakes while the "bookie" contributes the remaining $\$(1-P(\mathbf{x}))S_{\mathbf{x}}$. The *net gain* for $\mathcal{A}$ of a lottery ℓ is the difference between the stakes of the lottery and the ticket price.

Suppose that events $\mathbf{e}_1,\ldots,\mathbf{e}_n$ partition Ω, and let $P(\mathbf{e}_i)$ be $\mathcal{A}$'s probability assessment for $\mathbf{e}_i$. Fixing stakes $S_{\mathbf{e}_1},\ldots,S_{\mathbf{e}_n}$ for the n possible cases yields n lotteries such that the net gain for $\mathcal{A}$ is given by

$$g_k = S_k - \sum_{i=1}^{n} P(\mathbf{e}_i)S_{\mathbf{e}_i} \tag{1.8}$$

if $\mathbf{e}_k$ obtains. The expression for g_k is the difference between the bet won and the n paid outlays. Treating the $S_{\mathbf{e}_k}$'s as unknowns leads to a system of n equations with determinant

$$\begin{vmatrix} 1-P(\mathbf{e}_1) & -P(\mathbf{e}_2) & \ldots & -P(\mathbf{e}_n) \\ -P(\mathbf{e}_1) & 1-P(\mathbf{e}_2) & \ldots & -P(\mathbf{e}_n) \\ \vdots & \vdots & \ldots & \vdots \\ -P(\mathbf{e}_1) & -P(\mathbf{e}_2) & \ldots & 1-P(\mathbf{e}_n) \end{vmatrix} = 1-(P(\mathbf{e}_1)+\cdots+P(\mathbf{e}_n)). \tag{1.9}$$

If this determinant is non–zero, then the system of equations can be solved for arbitrary $g_{\mathbf{e}_1},\ldots,g_{\mathbf{e}_N}$ and a clever bookie can set the stakes such that each individual lottery is fair for $\mathcal{A}$ and jointly $\mathcal{A}$ is certain to suffer a net loss. Such a set of lotteries is called a *Dutch book.*

[5]The assumption that the agent is risk-neutral in de Finetti's early works has been widely criticized. The commentary in Smith (1961) provides an interesting historical reference for various perspectives on this issue including the views of D. R. Cox, D. V. Lindley, and I. J. Good. Ramsey (1964) developed an axiomatization of subjective probability without the restriction to risk-neutral assessors. In his classic tome Savage (1954) tackled this problem in a particularly unique way. Keeney and Raiffa (1976) provide a thorough treatment of multi–attribute utility theory for agents who are not risk-neutral.

The *Dutch book theorem* and the *converse Dutch book theorem* show that $\mathcal{A}$'s assessments adhere to the probability calculus if and only if no dutch book can be constructed against $\mathcal{A}$ (cf. Resnik, 1987; Skyrms, 1986).

(1.10) DEFINITION: An assessor who obeys the laws of probability (i.e., no Dutch book can be made against the assessor) is *coherent.* Otherwise the assessor is said to be *incoherent.*

A coherent assessor is also said to be *normative* (i.e. the assessor obeys the laws or *norms* of the probability calculus).

The definition of conditional probability requires the notion of a *conditional lottery.* Let x and y be generic events.

(1.11) DEFINITION: A conditional lottery $\ell_{\mathsf{x}|\mathsf{y}}$ is a lottery with 3 possible outcomes:

(a) If x and y both occur then the payoff is \$1.

(b) If $\neg\mathsf{x}$ and y both occur then the payoff is \$0.

(c) If $\neg\mathsf{y}$ occurs then the lottery is cancelled.

Using conditional lotteries de Finetti defines conditional probability as follows.

(1.12) DEFINITION: The *conditional probability* of x given y, $P(\mathsf{x}|\mathsf{y})$, is the fair price for $\mathcal{A}$ of the conditional lottery (1.11).

We can deduce the standard definition of conditional probability

$$P(\mathsf{x} \wedge \mathsf{y}) = P(\mathsf{x}|\mathsf{y})P(\mathsf{y}) \tag{1.13}$$

from (1.12). Based on (1.7) and (1.12) the following are fair bets for $\mathcal{A}$.

(1.14)

Lottery	*Stakes*	*Outcome*	*Payoff*	*Net gain*
$\ell_{\mathsf{x}\wedge\mathsf{y}}$	$P(\mathsf{x}\wedge\mathsf{y})S_{\mathsf{x}\wedge\mathsf{y}}$	$\mathsf{x}\wedge\mathsf{y}$	$S_{\mathsf{x}\wedge\mathsf{y}}$	$S_{\mathsf{x}\wedge\mathsf{y}} - P(\mathsf{x}\wedge\mathsf{y})S_{\mathsf{x}\wedge\mathsf{y}}$
		$\neg(\mathsf{x}\wedge\mathsf{y})$	0	$-P(\mathsf{x}\wedge\mathsf{y})S_{\mathsf{x}\wedge\mathsf{y}}$
ℓ_{y}	$P(\mathsf{y})S_{\mathsf{y}}$	y	S_{y}	$S_{\mathsf{y}} - P(\mathsf{y})S_{\mathsf{y}}$
		$\neg\mathsf{y}$	0	$-P(\mathsf{y})S_{\mathsf{y}}$
$\ell_{\mathsf{x}|\mathsf{y}}$	$P(\mathsf{x}|\mathsf{y})S_{\mathsf{x}|\mathsf{y}}$	$\mathsf{x}\wedge\mathsf{y}$	$S_{\mathsf{x}|\mathsf{y}}$	$S_{\mathsf{x}|\mathsf{y}} - P(\mathsf{x}|\mathsf{y})S_{\mathsf{x}|\mathsf{y}}$
		$\neg\mathsf{x}\wedge\mathsf{y}$	0	$-P(\mathsf{x}|\mathsf{y})S_{\mathsf{x}|\mathsf{y}}$
		$\neg\mathsf{y}$	annulled	0

There are three possible outcomes and net gains:

$$\begin{aligned}
\mathsf{x} \wedge \mathsf{y} &: (1 - P(\mathsf{x} \wedge \mathsf{y}))S_{\mathsf{x}\wedge\mathsf{y}} + (1 - P(\mathsf{y}))S_{\mathsf{y}} + (1 - P(\mathsf{x}|\mathsf{y}))S_{\mathsf{x}|\mathsf{y}} = g_{\mathsf{x}\wedge\mathsf{y}} \\
\mathsf{x} \wedge \neg\mathsf{y} &: -P(\mathsf{x} \wedge \mathsf{y})S_{\mathsf{x}\wedge\mathsf{y}} + (1 - P(\mathsf{y}))S_{\mathsf{y}} - P(\mathsf{x}|\mathsf{y})S_{\mathsf{x}|\mathsf{y}} = g_{\mathsf{y}} \\
\neg\mathsf{y} &: -P(\mathsf{x} \wedge \mathsf{y})S_{\mathsf{x}\wedge\mathsf{y}} + -P(\mathsf{y})S_{\mathsf{y}} = g_{\mathsf{x}|\mathsf{y}}
\end{aligned}$$

Equivalently,

$$\begin{bmatrix} 1 - P(\mathsf{x} \wedge \mathsf{y}) & 1 - P(\mathsf{y}) & 1 - P(\mathsf{x}|\mathsf{y}) \\ -P(\mathsf{x} \wedge \mathsf{y}) & 1 - P(\mathsf{y}) & P(\mathsf{x}|\mathsf{y}) \\ -P(\mathsf{x} \wedge \mathsf{y}) & -P(\mathsf{y}) & 0 \end{bmatrix} \begin{bmatrix} S_{\mathsf{x}\wedge\mathsf{y}} \\ S_{\mathsf{y}} \\ S_{\mathsf{x}|\mathsf{y}} \end{bmatrix} = \begin{bmatrix} g_{\mathsf{x}\wedge\mathsf{y}} \\ g_{\mathsf{y}} \\ g_{\mathsf{x}|\mathsf{y}} \end{bmatrix}. \tag{1.15}$$

The determinant of the above matrix is

$$\begin{vmatrix} 1 - P(\mathsf{x} \wedge \mathsf{y}) & 1 - P(\mathsf{y}) & 1 - P(\mathsf{x}|\mathsf{y}) \\ -P(\mathsf{x} \wedge \mathsf{y}) & 1 - P(\mathsf{y}) & P(\mathsf{x}|\mathsf{y}) \\ -P(\mathsf{x} \wedge \mathsf{y}) & -P(\mathsf{y}) & 0 \end{vmatrix} = P(\mathsf{x}|\mathsf{y}) - P(\mathsf{y})P(\mathsf{x} \wedge \mathsf{y})$$

If this determinant is non-zero, then we can solve (1.15) for arbitrary $[g_{\mathsf{x}\wedge\mathsf{y}}, g_{\mathsf{y}}, g_{\mathsf{x}|\mathsf{y}}]^T$, and by the converse Dutch book theorem, $\mathcal{A}$'s assessments do not satisfy the probability calculus. In other words a clever bookie can set the stakes $[S_{\mathsf{x}\wedge\mathsf{y}}, S_{\mathsf{y}}, S_{\mathsf{x}|\mathsf{y}}]^T$ such that $\mathcal{A}$ considers each of the three bets fair and jointly $\mathcal{A}$ is certain to suffer a net loss.

The foregoing discussion illustrates the principle of *Bayesian conditionalization.* Suppose $\mathcal{A}$ asserts that $P(\mathsf{x}) = p$ and $P(\mathsf{x}|\mathsf{y}) = q$. Subsequently $\mathcal{A}$ learns that y is true (i.e., $P(\mathsf{y}) = 1.0$. Then the principle of conditionalization states that in the light of this new information $\mathcal{A}$ is coherent just in case $\mathcal{A}$ asserts $P(\mathsf{x}) = q$ (i.e., $P(\mathsf{x}|\mathsf{y})$ before learning that y is true) (cf. Howson & Urbach, 1989).

An important consequece of (1.13) is that $P(\mathsf{x} \wedge \mathsf{y})$ is a linear function of $P(\mathsf{y})$ if $P(\mathsf{x}|\mathsf{y})$ is fixed. If $\mathcal{A}$ asserts that $P(\mathsf{x}|\mathsf{y}) = q$, then by (1.13),

$$P(\mathsf{x} \wedge \mathsf{y}) = qP(\mathsf{y}).$$

This fact allows conditional assessments of the form $P(\mathsf{x}|\mathsf{y}) = q$ to be represented in the the linear paradigm of the FTP.

For de Finetti, probability was a special case of the more generalized notion of *prevision*[6]. A *random variable* $\tilde{x}$ is a real–valued function defined on a sigma–field of subsets of Ω.

[6]The term *prevision* has also been translated from the original Italian text into English as *foresight.*

(1.16) DEFINITION: Let Ω be a finite sample space and $\tilde{x}$ be an random variable defined on Ω. Then the prevision of $\tilde{x}$, $P(\tilde{x} = X)$, is the fair price for $\mathcal{A}$ of a lottery ℓ which pays $\$X_i$ if $\mathbf{e}_i$ obtains.

If $\tilde{x} : \Omega \to \{0, 1\}$, then $\tilde{x}$ is an event and $P(\tilde{x} = X)$ is the probability of $\tilde{x}$. The realm of a random variable $\tilde{x}$ is simply the range of $\tilde{x}$. Thus, for a rational $\mathcal{A}$ prevision corresponds to mathematical expectation. Probability is a special case in which $\tilde{x}$ is an event, so that like mathematical expectation prevision, $P(\cdot)$, is a linear operator.

Prevision is a particularly good example of the contrast between de Finetti's theory of probability and Kolmogorov's axiomatization which has become the standard treatment of probability (e.g., Ross, 1985;1988; Hogg & Craig, 1978). For the Kolmogorov axiomatization, the sample space Ω is fundamental and everything else is defined relative to the sample space. De Finetti held quite the opposite view. For him the sample space is defined relative to the events and random quantities of interest to the particular agent. In the subjective framework a random variable, or in de Finetti's preferred terms a *random quantity*, is the result of an operationally well-defined measurement. That is the uncertainty in regards to values that $\tilde{x}$ may assume arises from the agent's lack of knowledge of which particular value of $\tilde{x}$ may obtain. The uncertainty is resolved once the measurement is made (de Finetti, 1964; Lad, 1993). For example $\tilde{x}$ might be the number of spots showing on the top faces of a pair of standard six-sided dice after a *particular* role. Before the dice are rolled, $P(\tilde{x}) = 7$ if $\mathcal{A}$ believes the dice are fair. After the dice are rolled $\tilde{x} = k$ for some $k \in \{2, 3, \ldots, 12\}$ and for a rational $\mathcal{A}$, $P(\tilde{x}) = k$.

1.3 Incomplete Probabilistic Models

Suppose that our knowledge consists of a set of assessments, $\mathcal{K}$. In many practical applications this knowledge is insufficient to determine a unique joint probability distribution $P_{\mathcal{K}}$ over Ω. In this case an assessor is said to have an *incomplete probabilistic model*, and there is a class of distributions $\mathcal{P}_{\mathcal{K}} \subseteq \mathcal{P}$ which are consistent with the assessments $\mathcal{K}$. If the assessments are inconsistent then $\mathcal{P}_{\mathcal{K}}$ is the empty set.

1.3.1 Ampliative inference and the MAXENT principle

Given an incomplete probabilistic model $\mathcal{P}$ one might be tempted to just *pick* some $P^* \in \mathcal{P}$ and thus gain the benefits of a complete probability distribution. Rather than arbitrarily picking P^*, one might resort to some rational criteria to determine

which P^* is the appropriate choice. A simple example will provide a concrete context for discussion.

(1.17) EXAMPLE: **A wheel of fortune**[7].

> Suppose that a wheel of fortune has been partitioned into three sectors, giving payoffs of \$1, \$2, and \$3. No direct information about the relative sizes of the sectors is available, but it is known that the expectation of the payoff for the wheel is \$1.50.

Consider the following question.

(1.18) In the absence of any other information about the wheel, what probability distribution represents fair odds on the three sectors?

Figure 1.2 shows two wheels (i.e., probability distributions) with the stated payoffs and average winnings of \$1.50. In fact a continuum of wheels with payoffs of \$1, \$2, and \$3 and average winnings of \$1.50 can be constructed by finding distinct solutions, p_1, p_2, p_3, to the following equations which reflect our knowledge that the expectation of the wheel's payoff is \$1.50.

(1.19)

$$\begin{aligned} 1p_1 + 2p_2 + 3p_3 &= 1.50 \\ p_1 + p_2 + p_3 &= 1 \\ p_1, p_2, p_3 &\geq 0 \end{aligned}$$

These equations define a closed convex set of distributions $\mathcal{P}_K$ consistent with the information given.

Example (1.17) illustrates what has been called the *ampliative inference problem* because a unique solution is not implied by the probability calculus (Klir & Folger, 1988).

Solving the ampliative inference problem by appealing to some doctrine in addition to the probability calculus illustrates one approach to handling incomplete probabilistic knowledge. The most famous rule for ampliative inference is known as MAXENT which advises the assessor to pick the distribution satisfying constraints such as in (1.19) which has maximum entropy (cf. Jaynes, 1957;1968;1979). The

[7]This example and the accompanying figure are adapted from Myers and Osherson (1992).

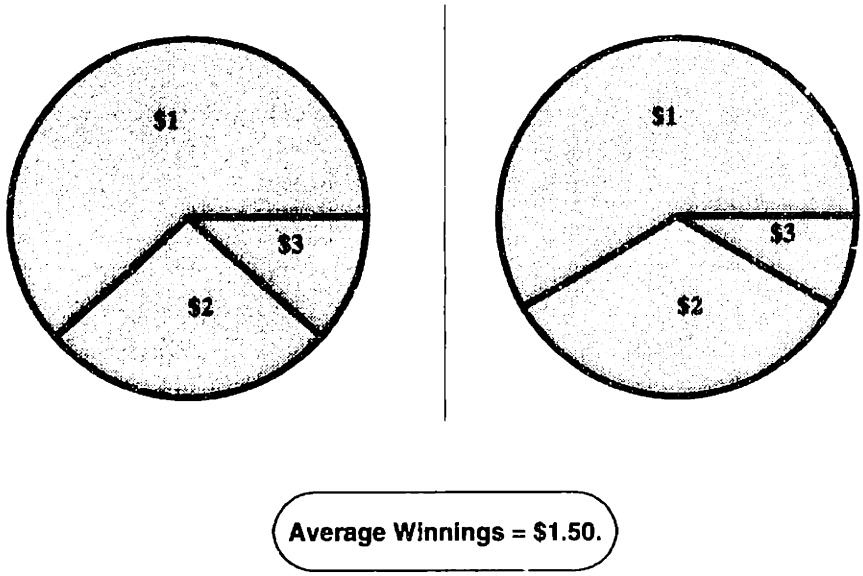

Figure 1.2: Two distinct wheels of fortune with average winnings of $1.50 and payoffs of $1, $2, and $3.

MAXENT principle can be explained in the context of example (1.17) as follows. For each distribution $[p_1, p_2, p_3]$ over the three sectors in the wheel of fortune, there is a number $-\sum_{i=1}^{3} p_i \log_2 p_i$, called the *entropy* of $[p_1, p_2, p_3]$. Given a closed convex class of distributions as in example (1.17), it can be shown that there is a unique distribution with maximum entropy. The MAXENT principle advises the assessor to choose this distribution as a rational solution to (1.17). The wheel on the left in Figure 1.2 depicts the maximum entropy distribution for example (1.17).

Ampliative inference using a doctrine such as MAXENT has strong intuitive appeal. Ampliative inference yields a complete probabilistic model. In addition there are a number of persuasive normative arguments for MAXENT as the ampliative inference principle of choice (cf. Shore & Johnson, 1980; Rosenkrantz, 1977; Williams, 1980). Jaynes (1968) goes as far as arguing that MAXENT is an extension of Bayesian conditionalization. Likewise there are a number of practical considerations such as the existence of algorithms for effectively computing the MAXENT distribution in many different kinds of problems (e.g., Cheeseman, 1983; Goldman & Rivest, 1988; Tsao, Fang, & Lee, 1992).

On the other hand there are a number of serious problems with MAXENT in particular and any such principle in general. Myers and Osherson (1992) found that MAXENT is not an adequate descriptive model. Seidenfeld (1986) presents critical counter–examples to Jaynes' argument that MAXENT is a generalization of Bayesian

conditionalization showing that an assessor performing ampliative inference using MAXENT is coherent only in limited circumstances. One might argue that a better principle is needed; however, Osherson, Shafir, and Smith (1993) show that *any* principle of ampliative inference either contradicts probabilistic information already in hand, or when applied iteratively to new information generates successive distributions which are not related by conditionalization. An assessor relying on an ampliative inference doctrine is guaranteed to be incoherent. Thus both the intuitive appeal and computational economy of ampliative inference are tempered by its questionable behavior vis-a-vis conditionalization.

1.3.2 Causal Networks

Example (1.4) illustrates how a complete joint probability distribution could be obtained directly by assessing probabilities directly for each of the outcomes in the joint probability distribution (i.e., the terminal nodes in Figure 1.1), or by assessing an equivalent set of conditional probabilities (i.e., the graph edges in Figure 1.1). Formally, the relationship is given by the *chain rule.*

(1.20)

$$\begin{aligned} & P(\mathsf{a}_1\mathsf{a}_2\ldots\mathsf{a}_N) \\ &= P(\mathsf{a}_N|\mathsf{a}_{N-1}\mathsf{a}_{N-2}\ldots\mathsf{a}_1)P(\mathsf{a}_{N-1}|\mathsf{a}_{N-2}\mathsf{a}_{N-3}\ldots\mathsf{a}_1)\ldots P(\mathsf{a}_2|\mathsf{a}_1)P(\mathsf{a}_1) \end{aligned}$$

The left-hand side of (1.20) is the joint event characterized by a terminal node in Figure 1.1. The right-hand side of the equation is the product of probabilities on the path from the root node to the terminal node. The chain rule shows how an assessor could specify a complete probability distribution for Ω either by assessing joint events directly or by assessing the equivalent conditional events on the right hand side of the chain rule. In principle the number of assessments is the same in either case. In practice there exist conditional independence relationships in the domain which lead to an equivalent expression for the right hand side of (1.20) involving fewer assessments.

Causal networks take the approach of specifying a complete probability distribution via conditional distributions (Pearl, 1988)[8]. Consider the chain rule (1.20) and

[8]There exists an extensive literature on causal networks and their application in artificial intelligence. Both Pearl (1988) and Neapolitan (1990) provide thorough introductions to the subject.

suppose that we construct a directed-acyclic graph (DAG) G such that there is a 1:1 mapping between nodes in G and propositional variables in (1.20). Denote the set of parents of node a_i by $c(\mathsf{a}_i)$ and the ancestors of a_i by $a(\mathsf{a}_i)$. Suppose G is such that

$$\begin{aligned} P(\mathsf{a}_i|c(\mathsf{a}_i)) &= P(\mathsf{a}_i|a(\mathsf{a}_i)) \\ &= P(\mathsf{a}_i|\mathsf{a}_{i-1}\ldots\mathsf{a}_1). \end{aligned}$$

Then (1.20) reduces to the following equation.

(1.21)

$$\begin{aligned} &P(\mathsf{a}_1\mathsf{a}_2\ldots\mathsf{a}_N) \\ &= P(\mathsf{a}_N|\mathsf{a}_{N-1}\mathsf{a}_{N-2}\ldots\mathsf{a}_1)P(\mathsf{a}_{N-1}|\mathsf{a}_{N-2}\mathsf{a}_{N-3}\ldots\mathsf{a}_1)\ldots P(\mathsf{a}_2|\mathsf{a}_1)P(\mathsf{a}_1) \\ &= P(\mathsf{a}_N|c(\mathsf{a}_N))P(\mathsf{a}_{N-1}|c(\mathsf{a}_{N-1}))\ldots P(\mathsf{a}_2|c(\mathsf{a}_2))P(\mathsf{a}_1|c(\mathsf{a}_1)) \end{aligned}$$

To construct G we follow a straightforward procedure (Pearl, 1988).

(a) Let $c(\mathsf{a}_1) = \emptyset$,

(b) For each a_i, $2 \leq i \leq N$, let $c(\mathsf{a}_i)$ be the smallest subset of $a(\mathsf{a}_i)$ such that $P(\mathsf{a}_i|c(\mathsf{a}_i)) = P(\mathsf{a}_i|a(\mathsf{a}_i))$.

The set of edges in G is $E = \{(\mathsf{a}_i, \mathsf{a}_j)|\mathsf{a}_i \in c(\mathsf{a}_j)\}$.

A simple example from Neapolitan (1990) shows how (1.21) can simplify the assessment of the joint distribution. If we have a vector of ten binary propositional variables $\mathsf{A}_{10} = [\mathsf{a}_1, \ldots, \mathsf{a}_{10}]^T$, then the joint distribution of these variables has $2^{10} = 1024$ outcomes. Suppose we have a casual network (P, G) for X_{10} where G is shown in Figure 1.3. Figure 1.3 shows that a_1 has no ancestors, a_2 has one parent, and each of the remaining nodes $\mathsf{a}_3, \ldots, \mathsf{a}_{10}$ has 2 parents. The number of assessments necessary to compute the joint distribution using (1.21) is quite small in comparison to direct assignment of a joint probability distribution. The marginal distribution for the node with no ancestors, a_2, requires 2 assessments [9]. The conditional distribution for the node with one parent, a_2, requires 4 assessments, and each of the 8 conditional distributions for the nodes with 2 parents, $\mathsf{a}_3, \ldots, \mathsf{a}_{10}$ requires 8 assessments. Thus we

Lauritzen and Spiegelhalter (1988) contains a particularly lively commentary on the subject of causal networks and the role of probability in artificial intelligence.

[9]Since the assessments for each node are a probability distribution and are constrained to sum to one, the degrees of freedom for each node is one less than the number of assessments for the node. While 70 assessments are required, overall only 60 need to be specified by the assessor.

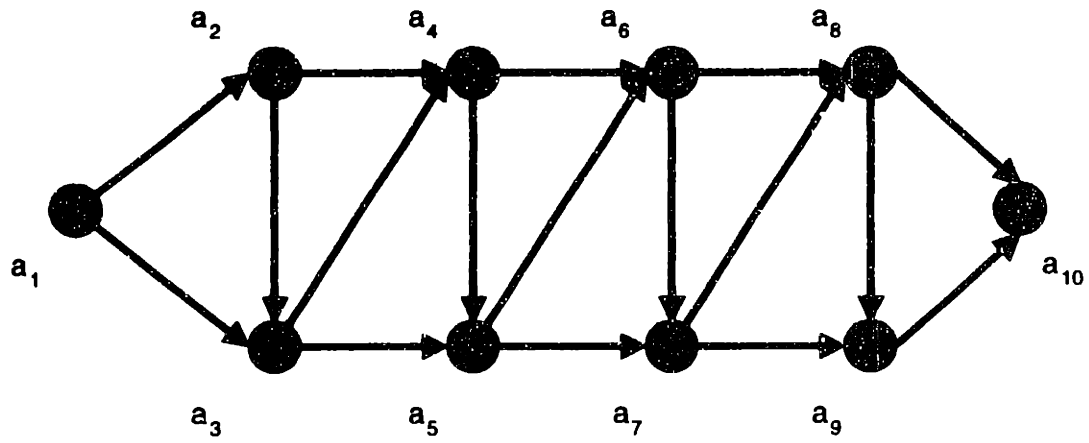

Figure 1.3: The graph for a simple causal network.

would need to make only $2 + 4 + 8 \times 8 = 70$ assessments instead of 1024. The extent to which the structure of the domain exhibits this kind of conditional independence structure determines the economy of the model in terms of the necessary assessments. While the reduction in the number of required assessments can be quite substantial, obtaining the required assessments can still be a formidable task. In the Pathfinder system developed by Heckerman (1991), over 75,000 probability assessments were required. All 75,000 probability judgements were solicited from a Stanford Medical School pathologist over the course of several *years*! In many cases the cost of obtaining and validating such a large number of judgements from an expert would be prohibitive.

There has been some recent progress mitigating the high cost of model construction and assessment. Neural networks and algorithms for extracting causal models from empirical databases have shown promise in some cases (Neal, 1992; Neapolitan, 1990). Another alternative is to extrapolate the necessary probability assessments from a modest number of expert assessments using a descriptive model of the assessor and an extrapolation algorithm (Osherson, Smith, Shafir, Myers, & Stob, 1993). Henrion (1989) has suggested sensitivity analysis techniques for deciding which assessments can be rough and which assessments need to be precise. Among the most promising developments in this area are *similarity networks* and *partitions* (Heckerman, 1991). Similarity networks and partitions are a mechanism for constructing a large causal network (the global network) from a number of smaller simpler causal networks (local networks). The probability distributions assessed in the local networks are conditioned on fewer events than in the global network. Thus, experts are usually more comfortable making assessments in the local networks. Heckerman used similar-

ity networks and partitions to reconstruct Pathfinder. These techniques reduced the number of assessments required by the Pathfinder system to approximately 14,000 from 75,000 while increasing diagnostic accuracy. Heckerman's techniques apply only to causal networks with a specific topology in which there is exactly one root node (i.e., a node where $c(\mathbf{a}_i) = \emptyset$).

The domain of the Pathfinder system, diagnosis of lymph-node disease, is quite narrow. The system encompasses roughly 60 diseases with 100 manifestations (symptoms, disease findings, etc.). Henrion (commentary in Lauritzen & Spiegelhalter, 1986) has pointed out some of the difficulties in constructing causal networks in complex domains such as internal medicine. The INTERNIST-1/QMR expert system addresses over 600 diseases with more than 4000 manifestations (Miller, McNeil, Challinor, Massarie, & Myers, 1986). Some manifestations have 150 or more possible causes. Assessing *one* such conditional distribution would require 2^{151} assessments if all the propositional variables were binary!

Cooper (1990) has shown that the problem of computing the probabilities of a causal network in the light of new evidence is *NP*-hard in general. However, there is a linear-time algorithm for networks which are singly-connected (Pearl, 1988). For non-singly connected networks, the complexity of the updating process is exponential in the number of nodes in the largest clique in the DAG. Fortunately for many practical models the size of the largest clique is modest. For example the largest clique in the medical expert system MUNIN has 4 nodes (Andreassen, Woldbye, Falck, & Andersen, 1987). So while causal networks promise normative inference and provide an economical expression of a complete joint probability distribution, they require an extensive model construction/validation process and can involve intractable computations if the network topology is not singly-connected.

1.4 Logic and probability

In the 19th century there existed two broad schools of thought concerning the relationship between logic and arithmetic. Like many philosophical debates of that age, the proponents tended to be on one or the other side of the English channel. Boole (1847,1854) articulated the position that the truths of logic are truths of finite integer arithmetic. The contrasting view held by Frege (1980) among others is that the theorems of arithmetic are the theorems of some first-order logical model. De Finetti's work fundamentally rests on the earlier work of Boole who developed both the arithmetic representation of logic and the groundwork for the fundamental

theorem of probability (Boole 1847,1854). Boole derived the following bounds on the probability of the union and intersection of events $x_1, \ldots, x_N$ which are known as the *Boole inequalities.*

(1.22)

$$\begin{aligned} P(x_1 \vee \ldots \vee x_N) &\leq P(x_1) + \ldots + P(x_N) \\ P(x_1 \wedge \ldots \wedge x_N) &\geq P(x_1) + \ldots + P(x_N) - (N-1) \end{aligned}$$

Furthermore, using Fourier–Motzkin elimination Boole (1854) leveraged his results in arithmetic representation of logical functions to compute linear inequalities for the bounds on the probability of an event given the probabilities of some other events (Hailperin, 1965). Thus, without the benefit of subsequent advances in linear algebra, Boole managed to derive a precursor to de Finetti's FTP. The principle contribution of de Finetti was his precise specification of the FTP in terms of necessary and sufficient conditions for a set of assessments to be coherent in a general context.

Given an arbitrary Boolean expression s on propositions $x_1, \ldots, x_N$, is there an assignment of truth values to $x_1, \ldots, x_N$ such that s is true? This question is known as the satisfiability problem of propositional logic (*NSAT*). One method for determining whether or not s is satisfiable is to generate a *semantic tree* for s (Winston, 1992). To construct the semantic tree for s, the root node corresponds to x_1. The tree branches on the truth value of x one branch asserts x the other branch asserts $\neg x$. The second level in the tree branches on x_2 and so on. Each path in the tree corresponds to a unique assignment of truth-values to the propositions $x_1, \ldots, x_N$. Paths are closed if the partial assignment of truth values on the path is inconsistent with s. Each complete path that remains open satisfies s. The tree is a complete enumeration of the possible worlds that satisfy s or equivalently the atomic events in Ω for s. Constructing a semantic tree in this fashion is one method for explicitly constructing the realm matrix for $x_1, \ldots, x_N$. For arbitrary s the satisfiability problem is *NP*-complete (Hopcroft & Ulman, 1979). Thus the FTP is a generalization of the satisfiability problem, and Fagin and Halpern (1989) showed that the FTP is equivalent to *NSAT*[10]. Georgakopoulos, Kavvadias, and Papadimitriou (1988) further showed that probabilistic equivalent of *2SAT* is *NP*–hard. The *2SAT* problem of propositional logic is a very restricted form of *NSAT* which is solvable in linear time.

[10]**The generalization of the satisfiability problem (*NSAT*) is known as the *probabilistic satisfiability problem* (*PSAT*).**

(1.23) DEFINITION: Let f be a function, $f : \{0,1\}^N \rightarrow \{0,1\}$, then f is called a *Boolean* or *logical* function.

Boole (1847,1854) also showed that we can numerically evaluate an arbitrary Boolean function on N propositions, $x_1, \ldots, x_N$. For example $x_1 \wedge x_2$ can be represented by $x_1 \times x_2$. The value of $x_1 \times x_2$ is 1 just in case $x_1 \wedge x_2$ is true. Boole further showed how a pair of *linear* inequlities could achieve the same result. Logical tautologies and logical impossibilities are simply constant functions with range $\{1\}$ and range $\{0\}$ respectively. For example the logical contradiction $x \wedge \neg x{=}0$ if $x = 1$ or if $x = 0$.

Ultimately the philosophical tide turned in Frege's direction leading to the development of modern logic and the theory of computation (cf. Ebbinghaus, Flum, & Thomas, 1984; Mendelson, 1987). Ironically modern digital computers evaluate program logic using mechanisms derived from Boole's methods.

1.4.1 Logic and numerical optimization

The relationship between mathematical programming and logic has an interesting history. Karp proved that integer programming is *NP*-complete by a reduction of integer programming to the satisfiability problem of propositional logic *NSAT* (cf. Hopcroft & Ulman, 1979). Jeroslow (1989) in an epic monograph describes his extensive research on representing logical inference by mixed-integer programming[11]. Among other advances Jeroslow demonstrated that numerical optimization techniques are competitive alternatives to systems based on production rules and symbolic constraint propagation (e.g., Davis-Putnam family of algorithms). Furthermore his work formalized what practitioners in operations research had been implicitly doing all along. Unfortunately this work is not well known among the operations research and decision analysis communities.

Blair, Jeroslow, and Lowe (1986) developed a recursive algorithm for converting general well-formed formulae of the propositional calculus into conjunctive-normal form. Formulae in conjunctive-normal form can be economically translated and represented in a mixed-integer program. Sentences in conjunctive normal form directly correspond to equivalent sets of simultaneous linear inequality constraints. The Blair, Jeroslow, and Lowe algorithm is linear in the number of atomic propositions in the

[11]Oddly enough Jeroslow was aware of Nils Nilsson's work and cites several of Nilsson's papers regarding theorem proving and logic. He did not seem to be aware of Nilsson's (1986) probabilistic logic paper or the relevance of his own work to this problem. Jeroslow much like Savage died suddenly and prematurely of a heart attack in 1988.

sentence. It achieves this level of performance by introducing new atomic propositions into the transformed clause. Other algorithms for converting an arbitrary sentence of propositional logic to conjunctive normal form without introducing additional atomic propositions require super-exponential space in the worst case.

1.4.2 Logic and probability

The dichotomous states (i.e., generic events) of an assessment problem generally have some *logical structure* which is to say that the set of realizable states of the system is a proper subset of the Cartesian product of these states $\mathsf{x}_1 \times \mathsf{x}_2 \times \ldots \times \mathsf{x}_N$ (i.e., the sample space Ω). In practice our knowledge of the system is such that there is a set of logical functions $\mathcal{B} = \{f_1, \ldots, f_M\}$ such that $\mathbf{e}$ is a realizable state of the system only if $f_i(\mathbf{e}) = 1$ for each $f_i \in \mathcal{B}$.

Figure 1.1 illustrates the impact of logical structure on the realizable outcomes. The set of terminal nodes in the figure corresponds to the Cartesian product of $\{\mathsf{x}_1, \mathsf{x}_2, \mathsf{x}_3, \mathsf{x}_4\}$. The black nodes in this set are the realizable outcomes. De Finetti used the term *realm* to denote the subset of the sample space consisting of the realizable events. The grey nodes in Figure 1.1 are not realizable states of the safety system because they are incompatible with one or more of the logical restrictions specified for the safety system.

Logical structure is a powerful tool in probability assessment providing a rich language to express properties of the problem and to drastically reduce the number of realizable events in the sample space. For a given set of N dichotomous states, there are 2^{2^N} distinct possible logical relations on the N states. For example if $N = 2$, then the set of Boolean functions is given by the following.

(1.24)

Event	x	y	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
$\mathbf{e}_1$	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$\mathbf{e}_2$	0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
$\mathbf{e}_3$	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
$\mathbf{e}_4$	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

In (1.24) f_8 is the familiar Boolean "or" function, f_7 corresponds to "xor", and f_2 is the "and" function. The problem logic dictates constraints on the relative likelihood of problem states. If x and y are constrained to satisfy f_7, then $\mathbf{e}_2$ and $\mathbf{e}_3$ are the only realizable states of the system. Similarly if the system is characterized by f_{14} (the occurrence of x implies the occurrence of y) then $P(\mathsf{x}) \leq P(\mathsf{y})$. In the safety system example (1.4), an expert assessing positive probability to any of the system states

corresponding to a gray node in Figure 1.1 is guaranteed to be incoherent. Alone logical structure is not sufficient. It identifies the set of realizable outcomes, but says little about the likelihood of them, at most providing a relative ordering of the likelihood of the propositions. The FTP captures the interaction between the logical and probabilistic structure of a problem, and the implications of this interaction on the class of coherent probability distributions over the sample space.

1.4.3 Probability assessment: Fault trees

One method used to assess the reliability of a complex mechanical system such as a nuclear reactor is *fault tree analysis* [12]. Fault tree analysis involves a bottom-up modeling of the ways in which a system can fail. This is achieved by constructing *fault trees* for the system. A fault tree is Boolean logic diagram that specifies the failure logic of the system under consideration. Because of their explicit logical structure they are a useful illustration of the relationship between the Boolean and probabilistic aspects of a probabilistic model. The hierarchical logical structure of fault trees makes them an ideal test suite for the RIP algorithm implementation because probability bounds on even moderately large fault trees can be computed by hand if the set of probability assessments is not too complicated.

Fault trees specify higher-order failures of a system as logical functions of lower order failure events. The root of a fault tree is the failure event of interest, typically whether or not the system fails or does not. A mechanical system can be viewed as a collection of components organized into a hierarchy of subsystems. For example consider a nuclear reactor sprinkler system designed to cool the reactor core in the event of an emergency. The root event of the fault tree for this system is whether or not the system provides sufficient cooling to the reactor core. The basic failure events, the leaves of the fault tree, are the state of the individual components such as valves, pumps, and pipes. The leaf events are called *basic failure events* or *elementary events.* Intermediate nodes are logical functions of the leaves of the tree and other intermediate failures. The intermediate nodes correspond to the subsystems of the system, and the logical functions which define them mirror the failure structure of the system (Frankel, 1988).

In fault tree analysis the elementary failures are are presumed to be readily quantifiable either by an expert's subjective judgement or using reliability data(Frankel, 1988; United States Nuclear Regulatory Commission, 1981). The assessments for the

[12]Appendix A provides a quick and dirty tutorial on fault tree diagrams.

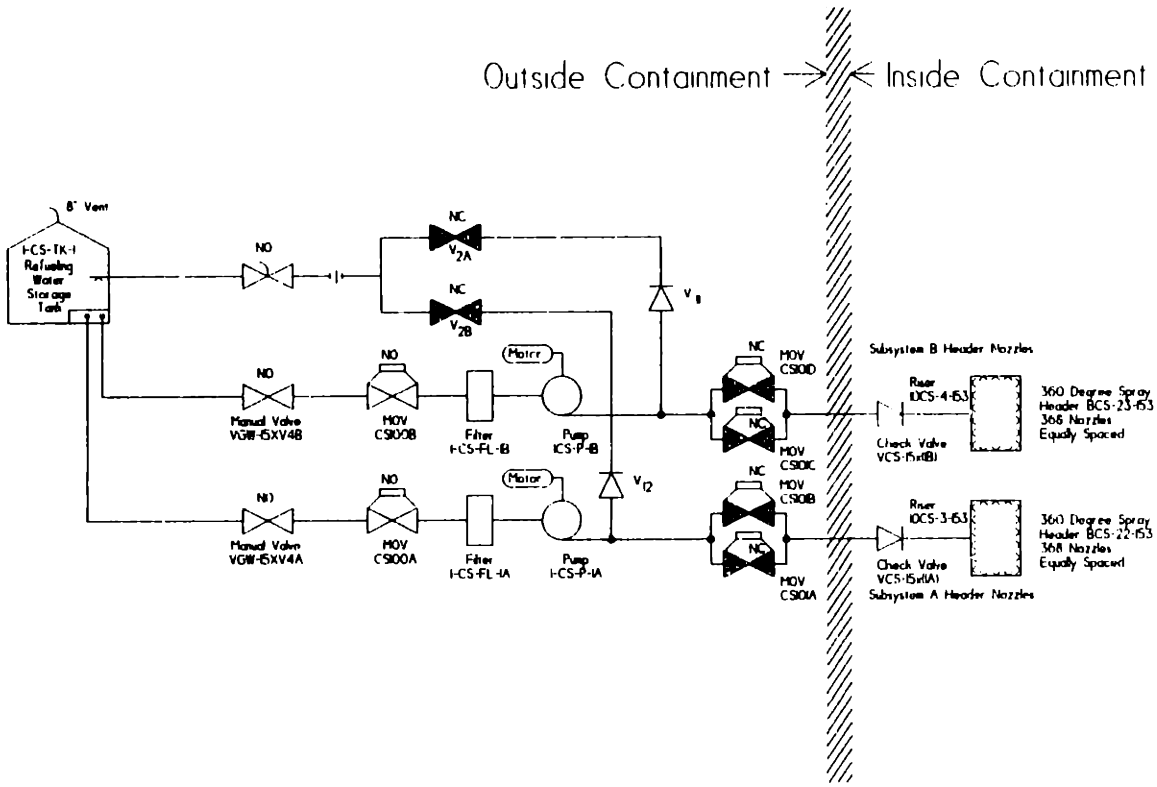

Figure 1.4: The logic diagram of the CSIS safety system in a nuclear reactor.

elementary events are used to compute recursively estimates for the non-elementary events. A fault tree with assessments attached to its leaves is an incomplete probabilistic model. Fault tree analysis proceeds by making assumptions about the probabilistic structure of the system in order to force the joint probability distribution of the events to be coherent and complete. The most commonly used assumption is independence of elementary events(cf United States Nuclear Regulatory Commission, 1981, Karimi, Rasmussen, & Wolf, 1980). While assuming independence greatly simplifies the computations required to generate an estimate for the root failure, it has serious implications for the accuracy of the resulting estimate. In many systems the most likely causes of system failure are *common-mode failures* (Henley & Kummamoto, 1981; Pages & Gondran, 1986) A common-mode failure is a set of elementary failures jointly sufficient to cause system failure where *the elementary events in the set are not stochastically independent.* In one safety system for a nuclear reactor the class of common-mode failures was more than an order of magnitude more likely than the next most likely class of failures(United States Nuclear Regulatory Commission, 1976). Hence a major effort in reliability analysis is identifying and quantifying common-mode failures (Heissing, Rasmussen, & Mak, 1982).

Figure 1.4 shows the flow graph of the CSIS safety system in a nuclear reactor.

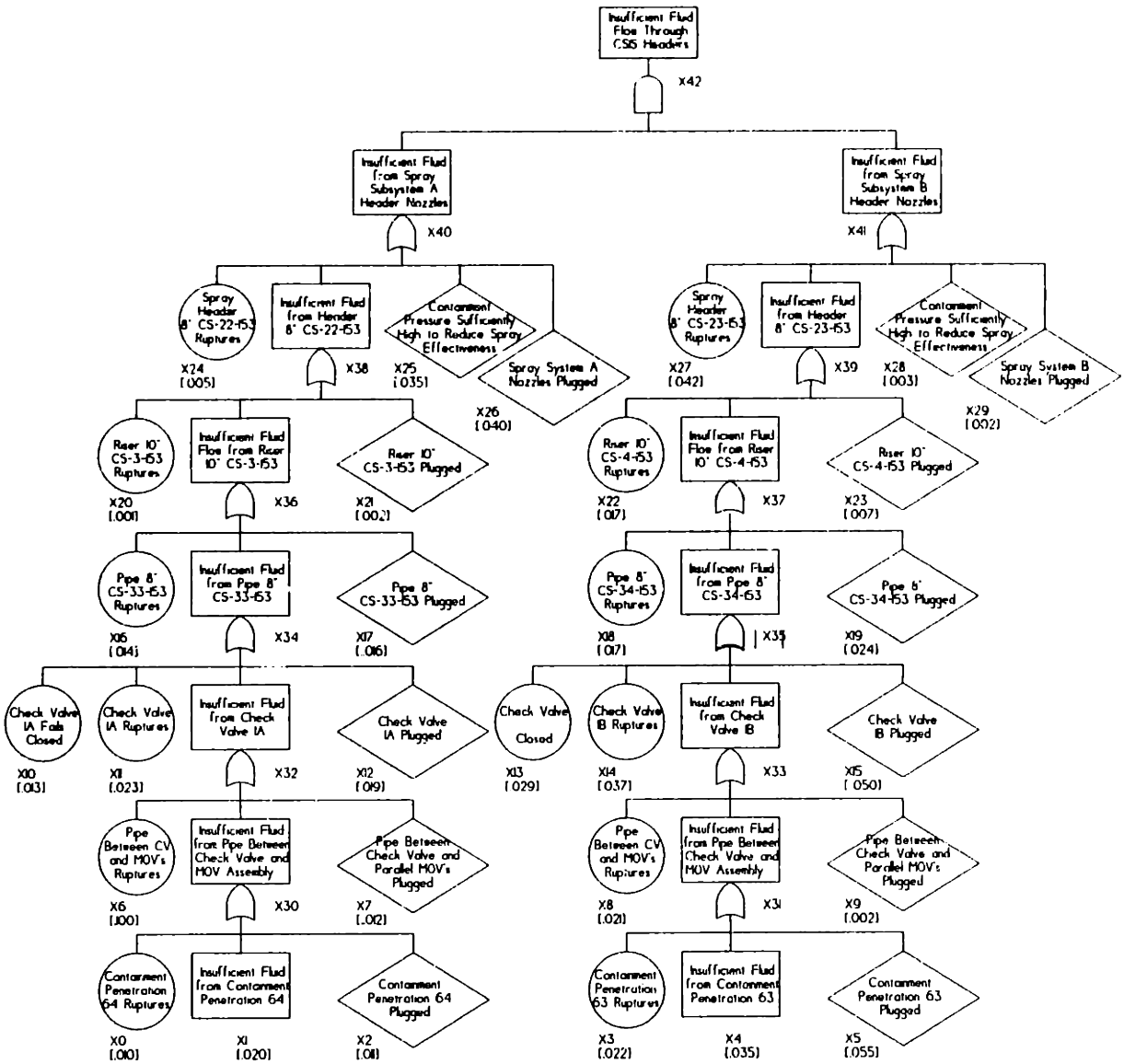

Figure 1.5: The top events in the fault tree for the CSIS system shown in Figure 1.4.

The safety system example (1.4) considered a small piece of the CSIS system. In a reactor accident the system pumps solution from the storage tank and injects it into the reactor containment vessel. The solution serves to control the temperature of the containment environment and to absorb radiation. As such it is a critical component in the accident control procedures. The system components are pipes, motorized pumps, valves, filters, spray-nozzles, and associated control equipment. There are thousands of components in the system. Ignoring passive parts such as pipes which have failure rates several orders of magnitude less than active components such as valves and pumps, there are still over a hundred components whose failure could contribute to a serious accident (United States Nuclear Regulatory Commission, 1975). These components are mutually dependent on each other in many ways. There are *logical dependencies* in which the success or failure of component $\mathbf{x}$ is a logical relation of some set of components $\mathcal{X}$. In the CSIS system, the nozzles which spray the solution into the containment fail if any of the upstream components fail to provide solution in sufficient quantity at sufficient pressure to the nozzles. An example of this is $\mathbf{x}_{23} \rightarrow \mathbf{x}_{42}$ in Figure 1.5. There are also *stochastic dependencies* where the relationship is not deterministic but rather probabilistic. Consider the failure of one of the 368 spray nozzles due to clogging. Since the remaining 367 nozzles are subject to the same environmental conditions (e.g., manufacturing processes, construction materials, contaminated solution, maintenance procedures, etc.), the failure of one of the nozzles makes the failure of the others more likely. Specifying the logical dependencies of the components in the CSIS is tedious but straightforward. Documenting system failure logic is standard practice for complex systems with high potential for public risk such as nuclear reactors or hazardous materials processing facilities.

In contrast specifying the probabilistic dependencies is a difficult task indeed. In addition to the large number of assessments required, many of the assessments are difficult for the expert to make. Experts develop a particular manner in which they apply their knowledge (Baron, 1988). The expert system may approach the problem from an entirely different perspective. The human expert may be required to assess events which are unfamiliar or have extremely small probabilities (von Winterfeldt & Edwards, 1986). For instance passive failures in nuclear power plants have been assigned probabilities of order 10^{-9} per reactor per year (United States Nuclear Regulatory Commission, 1975). Such assessments are subject to well-known cognitive biases (Kahneman, Slovic, & Tversky, 1982). Consider the safety system example (1.4) again. The event that the system fails $\mathbf{x}_4$ is a disjunction of $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$. People routinely underestimate the probability of disjunctions and overestimate the

probability of conjunctions (Tversky & Kahneman, 1974).

1.5 Probabilistic logic

Early expert systems often eschewed the probability calculus in favor of other mechanisms for representing uncertainty which were considered to be more tractable or expressive (e.g., Shortlife and Buchanan, 1975; Shafer, 1979; Szolovits & Pauker, 1978). The most well-known of these early systems, MYCIN, was a rule-based system for diagnosing bacterial infections (Buchanan & Shortlife, 1984). Heckerman (1986) later showed that a probabilistic interpretation could be given to the uncertainty calculus of MYCIN. Under this interpretation the inference methods used in MYCIN implicitly embody very strong assumptions about the sample space, assumptions which are very unlikely to hold in practice.

Concerns about the various ad hoc approaches to representing uncertainty in expert systems led Duda, Hart, and Nilsson (1976) to develop Bayesian methods for probabilistic inference in rule-based expert systems. These methods were first displayed in the PROSPECTOR expert system for oil exploration (Duda, Hart, Nilsson, & Sutherland, 1978).

The set of rules in a rule–based expert system describes a directed graph. Table 1.2 shows a small subset of the rules in PROSPECTOR[13]. To distinguish the "if ...then ..." construction of the expert sytem rules from logical implication, we will use the symbol $\rightsquigarrow$ to denote the "if ...then ..." relationship. The rules in Table 1.2 describe the graph shown in Figure 1.6. Each generic event corresponds to a vertex in the graph. An arc is drawn from the event in the "if" clause to the event in the "then" clause for every rule in the database. Inference in an inference network is unidirectional, and the graph described by the rules must be acyclic. The user of the PROSPECTOR system is interested in the likelihood of the propositions that are terminal nodes (i.e., the hypotheses) in the inference network, in this case that there is a massive sulfide deposit present (y). The user is asked to determine the likelihood of propositions that are "roots" in the network (i.e., the evidence), $x_1, \ldots, x_3$. Thus, the arcs in an inference network represent evidential information flow, and the set of possible inferences are pre–determined.

Rule-based systems like MYCIN and PROSPECTOR have a number of appealing

[13]The PROSPECTOR example is based on Neapolitan's (1990) comparison of rule-based expert systems and causal networks. Neopolitan's example concern's the PROSPECTOR system as described in Duda, Hart, and Sutherland (1978).

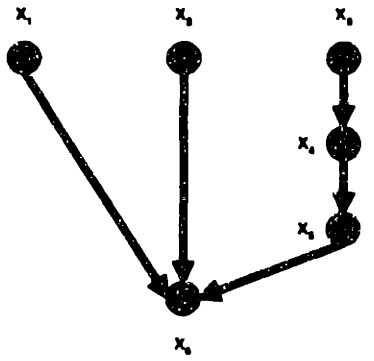

Figure 1.6: The DAG determined by the inference rules in Table 1.2.

Rule	*Description*
$x_1 \rightsquigarrow x_6$	If barite is overlying sulfide, then there are massive sulfide deposits.
$x_2 \rightsquigarrow x_6$	If galena, sphalerite, or chalcopyrite fill cracks in rhyolite or dacit then there are massive sulfide deposits.
$x_3 \rightsquigarrow x_4$	If there are bleached rocks then there is a reduction process.
$x_4 \rightsquigarrow x_5$	If there is a reduction process then there are clay minerals.
$x_5 \rightsquigarrow x_6$	If there are clay minerals then there massive sulfide deposits.

Table 1.2: Some simplified rules from PROSPECTOR.

features. Inference is accomplished via a modular, local propagation mechanism. Duda, Hart, & Nilsson (1976) developed the method of odds-likelihood ratios. The *odds* of a proposition y are

$$o(x) = \frac{P(y)}{P(\neg y)}. \tag{1.25}$$

Likewise the *conditional odds* of a propositions y given x are

$$o(y|x) = \frac{P(y|x)}{P(\neg y|x)}. \tag{1.26}$$

Further let

$$\lambda = \frac{P(x|y)}{P(x|\neg y)} \tag{1.27}$$

and

$$\bar{\lambda} = \frac{P(\neg x|y)}{P(\neg x|\neg y)}. \tag{1.28}$$

Then

$$o(y|x) = \lambda o(y) \tag{1.29}$$

and

$$o(y|\neg x) = \bar{\lambda} o(y) \tag{1.30}$$

Assessments for λ and $\bar{\lambda}$ are stored with each rule of the form $x \rightsquigarrow y$. If it is learned that x is true, then the updated odds of y can be easily computed using (1.29) and (1.30). This leads to a very efficient method for propagating new information. The procedure simply walks through the arcs in the graph from the initial node x until all paths to terminal nodes are exhausted, computing new values of (1.29) and (1.30) at each encountered node. This procedure is local because the each node is updated using only the λ values of its parents in the network. Unfortunately this procedure is simple and local only in networks which are trees and only when the evidence x is certain. The odds-likelihood method also incorporates the assumption that the evidence is conditionally independent given the hypothesis and its negation. In the PROSPECTOR example this assumption corresponds to the following assertions for all distinct x_i and x_j.

$$\begin{aligned} P(x_i) &= P(x_i|x_j) \\ P(x_i) &= P(x_i|\neg x_j) \end{aligned} \tag{1.31}$$

This assumption has been widely criticized both because it rarely holds in practice[14] (Szolovits & Pauker, 1978) and because it can lead to situations where no probability updating can take place (Pednault, Zucker, & Muressan, 1981). Heckerman and Horvitz (1986,1987) showed how the odds-likelihood mechanism breaks down when there are multiple causes for a given hypothesis or when the evidence is correlated. In the case of multiple causes the number of arcs in the inference network grows dramatically as does the assessor's job of assigning λ and $\bar{\lambda}$ to each arc.

Reflecting on his own work and on other popular formalisms for representing uncertainty, Nilsson (1986) was led to once again consider the problem of probabilistic inference. Nilsson looked to logic as a standard for inductive inference. Logical or *deductive* inference is *sound* in the sense that only valid conclusions can be deduced from a set of premises. Logic is also *complete* in the sense that all valid conclusions can be deduced from a set of premises using the inference mechanisms[15]. In this vein Nilsson undertook to generalize *modus ponens* to a probabilistic context without compromising sound inference and without implicit or unacknowledged probabilistic assumptions. Nilsson's probabilistic generalization of modus ponens has become known as the *probabilistic entailment* problem and his generalization of logic to encompass probability has become known as *probabilistic logic* (Nilsson, 1986;1993).

Suppose that we have a set of propositions of interest $\mathbf{X}_N = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, then the set of well-formed formulas consisting of the closure of $\mathbf{X}_N$ under finite conjunctions, and negations is a language $\mathcal{L}_{\mathbf{X}_N}$[16]. A knowledge-base in $\mathcal{L}$ is a set $\mathcal{K} = \{(\mathbf{s}_1, p_1), \ldots, (\mathbf{s}_N, p_N)\}$ where $\mathbf{s}_i \in \mathcal{L}$ and $p_i \in [0,1]$ is a probability assessment for $\mathbf{s}_i$. We might have $\mathcal{L} = \{\mathbf{x}_1, \mathbf{x}_1 \rightarrow \mathbf{x}_2, \mathbf{x}_2\}$ and $\mathcal{K} = \{(\mathbf{x}_1, .5), (\mathbf{x}_1 \rightarrow \mathbf{x}_2, .95)\}$ [17]. Determining the probability of $\mathbf{x}_2$ from $\mathcal{K}$ is Nilsson's (1986) probabilistic entailment problem. In this case the knowledge-base $\mathcal{K}$ is not a complete probabilistic model; it implies only that $P(\mathbf{x}_2)$ lie in an interval.

---

[14]Consider for example *fever* and *body ache* which are highly correlated in the diagnosis of bacterial infections.

[15]As AI researchers are keenly aware the soundness and completeness of propositional and first order logic is a theoretical tenet and not an algorithm for efficient inference. Propositional logic is *decidable* which means that there is a well defined procedure to determine in finite time whether or not an argument is valid. First order logic is not decidable. Practical deduction systems compromise completeness for computational efficiency. Expert systems are *incomplete* by design. They trade computational efficiency for completeness and are able to draw only a limited subset of the possible conclusions from a given set of premises.

[16]Our interest is typically restricted to a small finite subset of $\mathcal{L}$. For convenience we will often speak of $\mathcal{L} = \{\mathbf{s}_1, \ldots, \mathbf{s}_m\}$ where the $\mathbf{s}_i$'s are the subset of sentences in $\mathcal{L}$ in which we are interested ignoring the remaining part of $\mathcal{L}$.

[17]The symbol "$\rightarrow$" in the sentence $\mathbf{x}_1 \rightarrow \mathbf{x}_2$ denotes the logical conditional and is defined by the

$$
\begin{array}{rcccl}
P(x_1 \rightarrow x_2) + P(x_1) - 1 & \leq & P(x_2) & \leq & P(x_1 \rightarrow x_2) \\
0 & \leq & P(x_2) & \leq & .95
\end{array}
\tag{1.32}
$$

Among the various discoverers of the FTP, Nilsson's unique contribution was addressing the computational aspects of the problem. Nilsson made several suggestions which he divided into exact approaches suitable for small problems and approximations that could be used for problems too large to enumerate feasibly the joint distribution. Unlike de Finetti who embraced an incomplete probabilistic model, Nilsson ultimately sought to extract a unique joint probability distribution. For small problems Nilsson suggested using a maximum entropy distribution consistent with the knowledge–base rather than settle for the incomplete model provided by the knowledge–base.

For large problems where it is infeasible to enumerate the entire joint distribution, Nilsson suggested several ways of approximating the exact solution either by partitioning the problem or by approximating the realm matrix. Let $\mathbf{s}^*$ be the sentence for which we want to compute bounds and suppose that $\mathcal{K}$ could be partitioned into $\mathcal{K}_*$ and $\mathcal{K}^*$ such that no proposition in $\mathbf{s}^*$ occurs in $\mathcal{K}_*$. Then the bounds on $\mathbf{s}^*$ could be computed from $\mathcal{K}^*$ alone. Nilsson goes on to suggest several ways in which this partitioning could be approximated, for example, find a set of sentences $\{\mathbf{s}_1, \ldots, \mathbf{s}_k\}$ for a partition $\{\mathcal{K}_1, \ldots, \mathcal{K}_k\}$ of $\mathcal{K}$ where $\mathbf{s}_i$ encapsulates (is sufficient for) $\mathcal{K}_i$. Finding $\{\mathbf{s}_1, \ldots, \mathbf{s}_k\}$ is impractical unless there is some known structure to $\mathcal{K}$ which can be used to advantage.

Finding a surrogate matrix $\mathbf{R}'$ for the true realm matrix $\mathbf{R}$ was Nilsson's other suggested approach. Instead of enumerating all the column vectors in $\mathbf{R}$, enumerate only those vectors "close" to $P(\mathbf{X})$. The approximation $\mathbf{R}$ is constructed as follows

(1.33) (a) Construct a Boolean vector $\mathbf{r}$ from $P(\mathbf{X})$ where $r_i = 1$ if $P(x_i) \geq \frac{1}{2}$ and $r_i = 0$ otherwise.

---

following truth table.

x_1	x_2	$x_1 \rightarrow x_2$	$\neg x_1 \vee x_2$
1	1	1	1
0	1	1	1
1	0	0	0
0	0	1	1

The truth table shows that $x_1 \rightarrow x_2$ is equivalent to $\neg x_1 \vee x_2$ The logical conditional is often denoted by the "horseshoe", $x_1 \supset x_2$. Intuitively the logical conditional corresponds to the "if ... then" statement of mathematical theorems. One way of proving such a theorem is by contradiction. To prove a theorem by contradiction, we show that $x_1 \rightarrow x_2$ is logically true by showing that $x_1 \wedge \neg x_2$ is logically false (i.e., a contradiction). Of course by de Morgan's laws, $x_1 \wedge \neg x_2$ is logically equivalent to $\neg(\neg x_1 \vee x_2)$.

(b) If $y = 1$ is consistent with $\mathbf{r}$, then append $[\mathbf{r}, 1]^T$ to $\mathbf{R}'$. If $y = 0$ is consistent with $\mathbf{r}$, then append $[\mathbf{r}, 0]^T$ to $\mathbf{R}'$.

(c) For $p_i \in [\frac{1}{2}-\delta, \frac{1}{2}+\delta]$, flip the polarity of the corresponding components in $\mathbf{r}$ one at a time, two at a time repeating the above steps for each new $\mathbf{r}'$ until $\mathbf{R}'$ is as large as computational resources reasonably allow.

The matrix $\mathbf{R}'$ produced by (1.33) approaches $\mathbf{R}$ as the neighborhood around $\frac{1}{2}$ is increased. This procedure produces an approximation to the convex hull of $\mathbf{R}$ using vectors that are "close" to $P(\mathbf{X})$. Once $\mathbf{R}'$ is constructed, then small matrix methods are applied. This heuristic provides a computationally efficient mechanism for constructing an approximation. It's most significant advantage is that the size of $\mathbf{R}'$ can be decided upon a priori which gives a very good estimate of the computational cost of the approximation. Once an approximation is computed, improving the approximation is a matter of adding more vectors to the existing $\mathbf{R}'$ and re-computing the approximation. The downside of this technique is that there is no way to determine how good the approximation is without solving the problem exactly. Also, the choice of δ is arbitrary and scenarios with clustered assessments are problematic. A small change in the size of the δ–neighborhood about $\frac{1}{2}$ could drastically increase/decrease the size of $\mathbf{R}'$. Nilsson cites a number of arguments in support of a well–behaved approximation, but did not systematically study its behavior.

1.5.1 INFERNO

Nilsson's work is primarily of a theoretical nature. Similar concerns about strong ad hoc assumptions made implicitly in rule-based inference systems led Quinlan (1983) to develop the INFERNO system. The system makes no implicit assumptions about probabilistic structure on the domain, and is based upon an incomplete probabilistic model. INFERNO like the FTP, computes bounds on probabilities of events in the domain. If sufficient information is provided to the system, it will generate point probabilities in some instances. Quinlan cites two primary contributions of INFERNO (1) inferences drawn by INFERNO are guaranteed to be correct; and (2) the consistency of the assessments is validated by INFERNO. INFERNO represents uncertain knowledge using the relations in Table 1.3 which also shows the interpretation for each relation.

INFERNO computes bounds on the probability of events in the domain using a numerical constraint propagation network (cf. Winston, 1992). Each of the relations corresponds to a set of propagations constraints on the probability of the events in

Relation	*Interpretation*
y enables x with strength q	$P(\mathrm{x}\|\mathrm{y}) \geq q$
y inhibits x with strength q	$P(\neg\mathrm{x}\|\mathrm{y}) \geq q$
y requires x with strength q	$P(\neg\mathrm{y}\|\neg\mathrm{x}) \geq q$
y unless x with strength q	$P(\mathrm{y}\|\neg\mathrm{x}) \geq q$
y negates x	$\mathrm{y} = \neg\mathrm{x}$
y conjoins $\mathrm{x}_1 \ldots, \mathrm{x}_N$	$\mathrm{y} = \mathrm{x}_1 \wedge \cdots \wedge \mathrm{x}_N$
y conjoins independent $\mathrm{x}_1 \ldots, \mathrm{x}_N$	$\mathrm{y} = \mathrm{x}_1 \wedge \cdots \wedge \mathrm{x}_N$ and $P(\mathrm{x}_i\mathrm{x}_j) = P(\mathrm{x}_i)P(\mathrm{x}_j)$ if $i \neq j$
y disjoins $\mathrm{x}_1 \ldots, \mathrm{x}_N$	$\mathrm{y} = \mathrm{x}_1 \vee \cdots \vee \mathrm{x}_N$
y disjoins independent $\mathrm{x}_1 \ldots, \mathrm{x}_N$	$\mathrm{y} = \mathrm{x}_1 \vee \cdots \vee \mathrm{x}_N$ and $P(\mathrm{x}_i\mathrm{x}_j) = P(\mathrm{x}_i)P(\mathrm{x}_j)$ if $i \neq j$
y disjoins exclusive $\mathrm{x}_1 \ldots, \mathrm{x}_N$	$\mathrm{y} = \mathrm{x}_1 \vee \cdots \vee \mathrm{x}_N$ and
$\mathrm{x}_1 \ldots, \mathrm{x}_N$ mutually exclusive	$P(\mathrm{x}_i\mathrm{x}_j) = 0$ if $i \neq j$

Table 1.3: The INFERNO relations.

the relation. The system represents the interval in which the probability of an event x must lie by computing a lower bound $t(\mathsf{x})$ on the probability of x and a lower bound $f(\mathsf{x})$ on the probability of $\neg\mathsf{x}$. Thus,

$$t(\mathsf{x}) \leq P(\mathsf{x}) \leq 1 - f(\mathsf{x}). \tag{1.34}$$

The constraints are straightforward derivations from the elementary relation

$$\max P(\mathsf{x}_i) \leq P(\mathsf{x}_1 \vee \cdots \vee \mathsf{x}_N) \leq P(\mathsf{x}_1) + \cdots + P(\mathsf{x}_N). \tag{1.35}$$

The right hand side of (1.35), is one of Boole's inequalities (1.22).

For example the "enables" relation

(1.36) y enables x with strength q

has propagation constraints

(1.37)

$$\begin{aligned} t(\mathsf{x}) &\geq qt(\mathsf{y}) \\ t(\mathsf{y}) &\geq 1 - \frac{1 - f(\mathsf{x})}{q}. \end{aligned}$$

Similarly the "conjoins" relation

(1.38) y conjoins $\mathsf{x}_1, \ldots, \mathsf{x}_N$

has propagation constraints

(1.39)

$$\begin{aligned} t(\mathsf{y}) &\geq 1 - (1 - t(\mathsf{x}_1)) + \cdots + (1 - t(\mathsf{x}_N)) \\ f(\mathsf{y}) &\geq f(\mathsf{x}_i) \\ t(\mathsf{x}_i) &\geq t(\mathsf{y}) \\ f(\mathsf{x}_i) &\geq f(\mathsf{y}) - \sum_{j \neq i} (1 - t(\mathsf{x}_i)). \end{aligned}$$

A propagation mechanism can be built by implementing the propagation constraints as production rules. Each production rule evaluates a set of constraints such as (1.37). If the left-hand side is less than the right hand side (i.e, the constraint is violated), then the new value of the left-hand side (i.e., $t(\mathsf{x})$ or $f(\mathsf{x})$) is the value of

the right-hand side. Once we have updated $t(\mathsf{x})$ or $f(\mathsf{x})$, then for all relations in the database involving x we trigger the appropriate production rule. Since INFERNO is a constraint propagation network, new evidence never weakens the bounds on $P(\mathsf{x})$. The probability values always reflect the strongest constraints known to the system. Weaker constraints are redundant.

An important feature of INFERNO is that the propagation contstraints are weaker than the correponding relations in Table 1.3. The relations imply the constraints, but the converse is not true (Quinlan, 1983). The propagation mechanism also prevents cycling. If propagation is initiated by a change to the the bounds on x, then the bounds on x cannot be further updated by the resulting propagation. This prevents the system from entering an infinite propagation chain, and so again prevents the system from achieving the tightest possible bounds (Quinlan, 1983).

The INFERNO system is both conceptually and computationally simple. The propagation constraints are simple derivations from the interpretation of the relations in Table 1.3 using (1.35). The simplicity of the propagation constraints in combination with the limits on cyclic propagation makes INFERNO a fast and lean system. Quinlan presents a case-study involving a system with 13 logically independent generic events involving 27 INFERNO relations. The entire case study required less than 4 CPU seconds on a VAX 11/780[18]! Thus, INFERNO provides a great deal of bang for the computational buck. It gives inferences that are guaranteed to be valid at a very reasonable computational cost, but it does so by drawing inferences which are weaker than the available information permits. As such INFERNO will be a baseline against which to benchmark the RIP algorithm.

1.6 An Assessment and Inference Paradigm

A method for processing incomplete assignments of probability to a complex system should have the following desirable properties; the method should

(1.40) (a) Validate or invalidate the coherence of the assigned probabilities.

(b) should allow us to impute coherent bounds on the probability of events that have not been directly assessed by the expert.

[18]The computational economy of INFERNO is especially impressive considering the relative horsepower of the VAX 11/780 to current computers. For purposes of comparison, a current mid-range RISC workstation such as the author's Silicon Graphics workstation is roughly 150 times faster than the VAX 11/780 used by Quinlan. The author's current equivalent to the departmental size minicomputer used by Quinlan, an eight CPU Silicon Graphics Onyx, is more than 1000 times faster.

(c) Allow for efficient revision of bounds in light of additional information which may come in the form of

i. an expert's revision of judgements about the probability of events initially appraised,

ii. additional probability assignments to events not initially appraised, or

iii. the occurrence of an event or its complement.

(d) Not contradict Bayesian conditionalization.

(e) Be computationally tractable for realistic problems of moderate to large size.

(f) Not be based on ad hoc or unreasonable assumptions about the probabilistic structure assigned to uncertain events.

Pearl (1990b) argues that the following three principles characterize Bayesian reasoning:

(1.41) (a) reliance on complete probabilistic models of the domain;

(b) willingness to incorporate subjective judgements as an expedient substitute for empirical data; and

(c) the use of conditionalization as the primary mechanism for updating beliefs in the light of new observations.

Pearl follows these principles with the following advice.

> To fully appreciate the role of the Bayesian approach in AI, one should consider situations in which we do not have complete information to form a probability model. In such situations the Bayesian strategy encourages the agent to complete the model by making reasonable assumptions, rather than abandoning the useful device of conditionalization. (Pearl, 1990b, p 382).

Certainly Pearl's advice is sound in principle. If a complete probabilistic model is within reach, then the advantages of a complete probabilistic model are worth the modest effort required. Unfortunately Pearl's criteria are not feasible in all circumstances. If the available knowledge is not sufficient to make a complete probabilistic model feasible, then it is not necessary to abandon probability theory altogether. Under these circumstances de Finetti's fundamental theorem of probability (FTP)

can be the foundation for an assessment and inference paradigm having the desired properties in (1.40).

The contrast between the perspectives of de Finetti and Pearl on this issue is seen throughout the field of inductive inference and drives the development of new methods in the field. Inference networks and ampliative inference can be viewed as attempts to develop a rational method of constructing a complete probabilistic model when one's knowledge is incomplete. Causal network algorithms provide a tractable mechanism for inference in a complete model. The research presented here is an initial step at providing a similar enabling mechanism for the de Finetti perspective.

This paradigm lies in the intersection of cognitive science, artificial intelligence, and operations research. As a method of inductive inference, the FTP belongs to the domain of artificial intelligence and cognitive science. The computational mechanisms used by the FTP include linear programming, mixed-integer programming, and the representation of Boolean logic by integer programs; hence the FTP is a part of operations research as well. The relationship between Boole's arithmetic representation of propositional logic and the probability calculus was independently discovered/rediscovered by a number of individuals during the 20th century. In particular components of de Finetti's fundamental theorem of probability have appeared in several equivalent forms in the work of Boole (1847), de Finetti (1974a,1974b), Good (1950), Hailperin (1965,1976), Nilsson (1986,1993), Quinlan (1983), and Smith (1961) among others. Among these perspectives the work of de Finetti, Nilsson, and Quinlan stand out. Nilsson and Quinlan focus on the computational aspects of inductive inference in the absence of a complete probabilistic model. De Finetti and Quinlan both firmly embraced incomplete probabilistic models.

De Finetti's fundamental theorem of probability provides a computational framework for coherent probability assessment and inference. Nilsson (1986) independently developed a probabilistic logic which is equivalent to de Finetti's theorem in the context of probabilistic entailment. Quinlan's (1983) INFERNO system can be viewed as a local approximation to probabilistic logic and the FTP (Pearl, 1988; Nilsson, 1993). In this paper we apply the FTP to problems of assessment and inference. The FTP is the foundation for an interactive assessment paradigm which is evaluated against (1.40). Rough opinions can be used to jump start the process. Subsequently the paradigm provides a natural mechanism for refining these initial coarse opinions and an inference scheme for computing probability bounds on events not initially assessed.

Chapter 2

The Fundamental Theorem of Probability

2.1 Prelude

The underlying logic of de Finetti's FTP can be explained by a simple example. If we have a finite set of generic events X, then we can easily enumerate a sample space Ω_{X} for the events in X (Nilsson, 1986). For example, if $\mathsf{X} = \{x_1, x_2, x_1 \rightarrow x_2\}$, then the set of consistent truth values is given by the following table.

(2.1)

	e_1	e_2	e_3	e_4
x_1	1	1	0	0
$x_1 \rightarrow x_2$	1	0	1	1
x_2	1	0	1	0

Individually the columns of (2.1) are atomic events $\mathbf{e}_i \in \Omega$ or possible worlds (i.e., a realizable state of the system being modeled). The table in (2.1) can be viewed as a realm matrix for X expressesing the generic events of X in terms of the atomic events in which they are true.

(2.2)

$$\begin{bmatrix} x_1 \\ x_1 \rightarrow x_2 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3 \\ \mathbf{e}_4 \end{bmatrix}$$

$$\mathsf{X} = \mathbf{RE}$$

Equation (2.2) yields

$$\mathsf{x}_1 \rightarrow \mathsf{x}_2 = \mathsf{e}_1 + \mathsf{e}_3 + \mathsf{e}_4$$

where "+" is interpreted as a logical "or".

If we have a probability assessment for each e_i arranged in a vector

$$\mathbf{q}^T = [P(\mathsf{e}_1), P(\mathsf{e}_2), P(\mathsf{e}_3), P(\mathsf{e}_4)]^T,$$

then the probability of the generic events is given by a linear equation.

(2.3)

$$\begin{bmatrix} P(\mathsf{x}_1) \\ P(\mathsf{x}_1 \rightarrow \mathsf{x}_2) \\ P(\mathsf{x}_2) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} P(\mathsf{e}_1) \\ P(\mathsf{e}_2) \\ P(\mathsf{e}_3) \\ P(\mathsf{e}_4) \end{bmatrix}$$

$$P(\mathsf{X}) = \mathbf{R}\mathbf{q}$$

where $\mathbf{q} \geq 0$ and $\mathbf{1}^T\mathbf{q} = 1$. This result is a simple consequence of the law of total probability and the fact that the atomic events in Ω are disjoint and exhaustive.

In practical applications it is uncommon to be given a joint probability distribution $\mathbf{q}$ over Ω. Let $\mathsf{X} = [\mathsf{x}_1, \ldots, \mathsf{x}_{N+k}]^T$ be a vector of generic events in 2^Ω. Typically we have available probability assessments for some subset $\mathsf{X}_N = [\mathsf{x}_1, \ldots, \mathsf{x}_N]^T$ of the events in X, and desire to use these probability assessments to draw inferences about other generic events in X. Assessing probabilities for the events in X_N does not in general determine a unique probability distribution $\mathbf{q}$ over Ω, and indeed the set of such assessments may or may not be coherent. A vector of coherent but logically incomplete probability assessments $P(\mathsf{X}_N)$, yields a closed convex set of joint distributions for Ω. Specifically, constraints such as (2.3) define a convex polytope or equivalently a *simplex* of feasible distributions (de Finetti, 1974a; Lad, Dickey, & Rahman, 1990).

Specifically suppose that $\mathsf{X} = [\mathsf{X}_N^T, \mathsf{y}]^T = [\mathsf{x}_1, \ldots, \mathsf{x}_N, \mathsf{y}]^T$ is a set of $N+1$ events, and that we are given a vector of probability assessments $P(\mathsf{X}_N) = [P(\mathsf{x}_1), \ldots, P(\mathsf{x}_N)]^T$ for the events in X_N. Let $\mathbf{R}_\mathsf{X}$ be a realm matrix for X, $\mathsf{E}^T = [\mathsf{e}_1, \ldots, \mathsf{e}_\eta]^T$ be the corresponding vector of atomic events, and $\mathbf{r}_\mathsf{y}^T$ be the row in $\mathbf{R}_\mathsf{X}$ corresponding to y. We wish to determine a sub-interval of $[0, 1]$ in which the probability of y must lie subject to the constraints imposed by assignment of probabilities to $P(\mathsf{X}_N)$. Ideally we would like to solve

(2.4)

$$\begin{aligned} P(\mathsf{X}_N) &= \mathbf{R}_N\mathbf{q} \\ \mathbf{q} &\geq 0 \\ \mathbf{1}^T\mathbf{q} &= 1 \end{aligned}$$

for $\mathbf{q}$ where $q_i = P(\mathbf{e}_i)$ and $\mathbf{R}_N$ is the matrix obtained by deleting the row corresponding to y in $\mathbf{R}$. Then we could obtain the probability of y by computing $P(\mathsf{y}) = \mathbf{r}_\mathsf{y}^T\mathbf{q}$. Since the constraints are underdetermined in general, we can use linear programming to obtain bounds on $P(\mathsf{y})$.

2.1.1 De Finetti's fundamental theorem of probability.

The informal discussion in the previous section illustrates how questions about probabilities of events can be recast as linear programs. We are led to ask under what circumstances can we use the methods of linear programming to answer questions about the probability of events? De Finetti's fundamental theorem of probability and its extensions address this question. The fundamental theorem and its extensions provide necessary and sufficient conditions for applying linear programming algorithms to problems of probabilistic inference (de Finetti, 1974a; Lad, Dickey, & Rahman, 1990).

While the fundamental theorem in its original form was presented by de Finetti (1974a), its roots date back as far as the work of Boole in the 19th century(Kyburg & Smokler, 1964). Lad, Dickey, and Rahman have extended de Finetti's basic theorem in a number of interesting ways, and are responsible for explicitly couching the theorem in terms of a linear programming problem.

(2.5) **The Fundamental Theorem of Probability** (Lad, Dickey, & Rahman, 1990). Let $\mathsf{X}_N = [\mathsf{x}_1, \ldots, \mathsf{x}_N]^T$ be a vector of events, $P(\mathsf{X}_N) = [p_1, \ldots, p_N]^T = \mathbf{p}_N$ be a vector of proability assessments for X_N, and x_{N+1} be any other event. The event vector $\mathsf{X}_{N+1} = [\mathsf{x}_1, \ldots, \mathsf{x}_N, \mathsf{x}_{N+1}]^T$ generates a sample space $\Omega_{\mathsf{X}_{N+1}}$ of size $\eta(N+1) \leq 2^{N+1}$. Let $\mathsf{E}_{N+1} = [\mathbf{e}_1, \ldots, \mathbf{e}_{\eta(N+1)}]^T$ be a vector of atomic events of $\Omega_{\mathsf{X}_{N+1}}$. Then there is an $(N+1) \times \eta(N+1)$ matrix $\mathbf{R}_{N+1}$ such that

$$\mathsf{X}_{N+1} = \mathbf{R}_{N+1}\mathsf{E}_{N+1}.$$

Denote the first N rows of $\mathbf{R}_{N+1}$ by $\mathbf{R}_N$ and the $(N+1)$-st row by $\mathbf{r}_{N+1}$.

Event	Description	Relation	Probability
x_1	Sprinkler Head Clogs	—	[.007,.010]
x_2	Sprinkler Head Bursts	—	.005
x_3	Insufficient Water Pressure	$x_3 \leq 1 - x_2$	.015
x_4	The System Fails	$x_4 = 1 - (1 - x_1)(1 - x_2)(1 - x_3)$	

Table 2.1: Probability assessments for the safety system example.

(a) Then an extended vector of probability assessments, $P(X_{N+1}) = [\mathbf{p}_N^T, p_{N+1}]^T$, for X_{N+1} is coherent just in case $l \leq p_{N+1} \leq u$ where l and u are determined by the following linear programming problems:

(2.6) Find the minimum l and the maximum u of

$$\mathbf{r}_{N+1}\mathbf{q}$$

subject to

i. $\mathbf{R}_N\mathbf{q} = \mathbf{p}_N$,

ii. $\mathbf{q}^T\mathbf{1} = 1$

iii. and $\mathbf{q} \geq \mathbf{0}$.

(b) The probability vector $P(X_N) = \mathbf{p}_N$ is coherent just in case the feasible region for the LP problems (2.6) is non-empty.

The non-negativity constraint is explicit in the standard form of LP problems, and it will be implicit whenever an LP problem is presented in what follows.

(2.7) EXAMPLE:

The Safety System Example. Recall the safety system from example (1.4). We have a vector of events

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}.$$

The events $x_1, \ldots, x_4$ are not mutually logically independent. The values of x_3 and x_2 are restricted by the logical relation $x_2 \rightarrow \neg x_3$ which prohibits both x_2 and x_3 from being true. In addition the value of x_4 is strictly

determined by the values of x_1, x_2, and x_3. The realm matrix for X is given by

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

and the vector of $\mathbf{e} \in \Omega$ is

$$\mathsf{E} = \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3 \\ \mathbf{e}_4 \\ \mathbf{e}_5 \\ \mathbf{e}_6 \end{bmatrix} = \begin{bmatrix} (x_1 \neg x_2 x_3 x_4) \\ (\neg x_1 \neg x_2 x_3 x_4) \\ (x_1 x_2 \neg x_3 x_4) \\ (\neg x_1 x_2 \neg x_3 x_4) \\ (x_1 \neg x_2 \neg x_3 x_4) \\ (\neg x_1 \neg x_2 \neg x_3 \neg x_4) \end{bmatrix}$$

The columns of $\mathbf{R}$ denote realizable joint events such as $(\neg x_1 x_2 \neg x_3 x_4)$. Notice that the joint event $(x_1 x_2 x_3 x_4)$ is not represented in either E or $\mathbf{R}$ because it violates the logical restriction prohibiting x_2 and x_3 from both occurring. If we define $\mathbf{r}_i$ to be the ith row in $\mathbf{R}$, then the marginal probabilities of $P(x_1) \ldots, P(x_4)$ are given by

$$P(x_i) = \sum_{i=1}^{6} \mathbf{r}_i \mathbf{q}_i \tag{2.8}$$

which expresses $P(x_i)$ as a sum of the probability of joint events via the law of total probability. For example,

$$\begin{aligned} P(x_1) &= P(x_1 \neg x_2 x_3 x_4) + P(x_1 x_2 \neg x_3 x_4) + P(x_1 \neg x_2 \neg x_3 x_4) \\ &= q_1 + q_3 + q_5 \end{aligned}$$

Table 2.1 shows the safety system events and gives marginal probability assessments for the events x_1, x_2, and x_3. Let $\mathbf{R}_3$ be the matrix consisting of the rows in $\mathbf{R}$ corresponding to x_1, x_2, and x_3 and let $\mathbf{r}_{x_4}$ be the row in $\mathbf{R}$ corresponding to x_4. Then we can compute bounds on x_4 given the assessments shown in Table 2.1 by solving the following LP problems.

(2.9) Find the extrema of

$$\mathbf{r}_{x_4} \mathbf{q}$$

subject to

$$\mathbf{R}_3\mathbf{q} = P(\mathbf{X}_3),$$

the normalization constraint $\mathbf{q}^T\mathbf{1} = 1$, and the non-negativity constraint $\mathbf{q} \geq \mathbf{0}$.

Solving the programming problems (2.9) using the probabilities given in Table 2.1 yields $.02 \leq P(\mathbf{x}_4) \leq .03$.

Linear programming algorithms lead to joint distributions on $\mathbf{R}$ that minimize/maximize the probability of $\mathbf{x}_4$ subject to the constraints dictated by the assignment of probabilities to $P(\mathbf{x}_1), P(\mathbf{x}_2), P(\mathbf{x}_3)$.

2.1.2 Interval Assessments

By stating the FTP in the framework of linear programming, we can leverage the extensive work in linear programming to better understand the FTP. The basic form of the FTP allows us to compute bounds on an event $\mathbf{x}_{N+1}$ given a coherent vector of assessments $P(\mathbf{X}_N)$. Suppose that instead of a vector of point assessments , $P(\mathbf{X}_N)$, we have a set of assessments of the form $l_i \leq P(\mathbf{x}_i) \leq u_i$. The linear programming framework for the FTP extends quite naturally to encompass this situation (Lad, Dickey, and Rahman 1990).

The normalization and the the non-negativity constraints on $\mathbf{q}$ restrict the feasible region of the vectors $\mathbf{q}$ to the convex hull of $\mathbf{R}_{\mathbf{X}_{N+1}}$. The image of the convex hull of $\mathbf{R}_{\mathbf{X}_{N+1}}$ under the transformation matrix $\mathbf{R}_{N+1}$ is a convex polytope in $(N+1)$–dimensional space. In the case of (2.5), specifying a value for $P(\mathbf{x}_i)$ has the consequence of reducing the dimension of the polytope by 1. The FTP can be extended to compute bounds on $P(\mathbf{x}_{N+1})$ where the constraints specified are of the form

$$l_i \leq P(\mathbf{x}_i) \leq u_i$$

for $1 \leq i \leq N$. Extending (2.5) to allow input of a partial probability assertion of the form

$$\mathbf{l}_N \leq P(\mathbf{X}_N) \leq \mathbf{u}_N$$

is done by replacing the N linear equality constraints

$$\mathbf{R}_N\mathbf{q} = \mathbf{p}_N$$

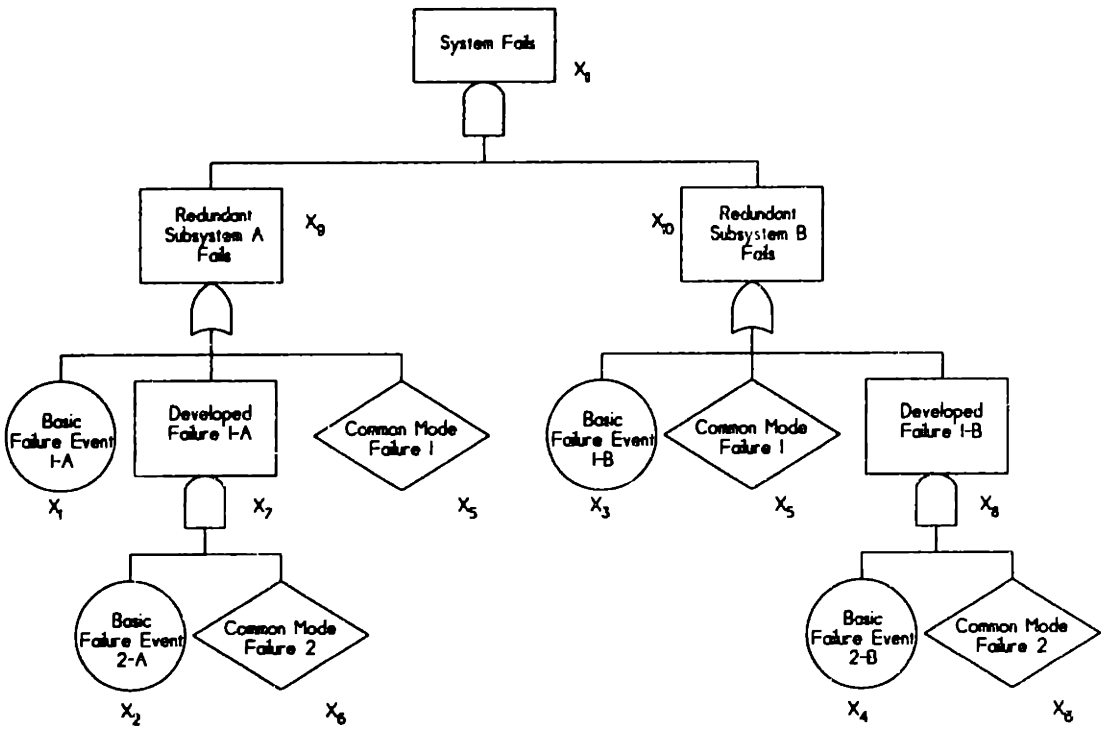

Figure 2.1: A simple fault tree.

in the statement of (2.5) with the N pairs of linear inequality constraints

$$\mathbf{l}_N \leq \mathbf{R}_N\mathbf{q} \leq \mathbf{u}_N.$$

The objective function can be specified so as to determine coherent bounds for $P(\mathsf{x}_i)$ with respect to any of the events x_i. The bounds determined by $[l_i, u_i]$ in this case are the projection of ith dimension of the convex polytope determined by the partial probability assertion. Note that the condition $\mathbf{l}_N \leq P(\mathbf{X}_N) \leq \mathbf{u}_N$ is a necessary but not sufficient condition for $P(\mathbf{x}_N)$ to be coherent.

(2.10) EXAMPLE: **A simple fault tree: Scenario 1.** Figure 2.1 shows a simple fault tree with 4 elementary and 2 common-mode failures. The fault tree represents a system consisting of two parallel redundant subsystems. The entire system fails (x_{11}) just in case *both* parallel subsystems fail (x_9 and x_{10}). Common-mode Failure 1 (x_5) appears in both parallel subsystems and is sufficient to cause the entire system to fail. Common-mode Failure 2 also appears in both parallel subsystems, but is not a sufficient cause for the system to fail. Common-mode Failure 2 will cause the system to fail just in case both Basic Failure 2-A and Basic Failure 2-B occur as well.

Suppose we have assessments for the elementary events $\mathsf{x}_1, \ldots, \mathsf{x}_6$ in the fault tree, and we would like to compute bounds on the probabilities of

the compound events $x_7, \ldots, x_{11}$. The fundamental theorem of probability (FTP) can be used to compute necessary bounds on $x_7, \ldots, x_{11}$ from whatever set of relevant assessments are available. It is not necessary in this case to make any assumptions regarding probabilistic dependencies among the events. Table 2.2 shows the probability assessments used in the following examples and the corresponding bounds determined by the FTP for the compound events.

Given the assessments shown in Table 2.2 for $x_1, \ldots, x_6$, the FTP can be applied to compute bounds on $x_7, \ldots, x_{11}$. The realm matrix $\mathbf{R}_X$ for the event vector X is a 11×64 matrix.

$\mathbf{R}_X =$

$$\left[\begin{array}{l}
1111111111111111111111111111111100000000000000000000000000000000 \\
1111111111111111000000000000000011111111111111110000000000000000 \\
1111111100000000111111110000000011111111000000001111111100000000 \\
1111000011110000111100001111000011110000111100001111000011110000 \\
1100110011001100110011001100110011001100110011001100110011001100 \\
1010101010101010101010101010101010101010101010101010101010101010 \\
\hline
1010101010101010000000000000000010101010101010100000000000000000 \\
1010000010100000101000001010000010100000101000001010000010100000 \\
1111111111111111111111111111111111011101110111011001100110011001 \\
1111111111011001111111111101100111111111110110011111111111101100 \\
1111111111011001111111111101100111011101110111001100110011001100
\end{array}\right]$$

The submatrix consisting of the first 6 rows of $\mathbf{R}_X$, shown above the horizontal line, correspond to the 2^6 events in the sample space for $x_1, \ldots, x_6$ generated as in (1.2). Rows 7 through 11, below the horizontal line, correspond to the generic events $x_7, \ldots, x_{11}$ which are logical functions of $x_1, \ldots, x_6$. Let $X_6 = [x_1, \ldots, x_6]^T$ and let $\mathbf{p} = [p_1, \ldots, p_6]^T$ be the vector of probability assessments for X_6.

The LP problem consists of linear system $\mathbf{Aq} = \mathbf{b}$ of constraints and an objective vector $\mathbf{r}$. To define the constraint matrix $\mathbf{A}$ for the LP problem take rows 1 through 6 in $\mathbf{R}_X$ (i.e., the ones for which we have assessments). The normalization constraint is incorporated by appending a row of ones to this matrix yielding the constraint matrix $\mathbf{A}$ for the LP problem.

$\mathbf{A} =$

$$\begin{bmatrix}
11111111111111111111111111111100000000000000000000000000000000 \\
11111111111111110000000000000001111111111111110000000000000000 \\
11111111000000001111111100000000111111110000000011111111000000000 \\
11110000111100001111000011110000111100001111000011110000111100000 \\
11001100110011001100110011001100110011001100110011001100110011001100 \\
10101010101010101010101010101010101010101010101010101010101010101010 \\
11111111111111111111111111111111111111111111111111111111111111111111
\end{bmatrix}$$

The events $x_7, \ldots, x_{10}$ are logical functions of $x_1, \ldots, x_6$ and were not assessed. They are not included in the LP matrix because the constraints they impose would be redundant. Further let the objective vector $\mathbf{r}$ for the LP be the row in $\mathbf{R_X}$ corresponding to x_{11}

$$\mathbf{r} = [11111111111011001111111111101100111011101110111011001100110011001100]$$

and let the right-hand-side vector be $\mathbf{b} = [\mathbf{p}^T, 1]^T$. Then to compute bounds on $P(x_{11})$ given $\mathbf{p}$ we solve the following LP problem.

(2.11) Find the minimum l and the maximum u of

$$\mathbf{rq}$$

subject to

$$\mathbf{Aq} = \mathbf{b}.$$

Then $l \leq P(x_{11}) \leq u$. Bounds on $x_7, \ldots, x_{10}$ are computed by substituting the appropriate vector for each of $x_7, \ldots, x_{10}$ for x_{11} in turn. Table 2.2 shows the bounds computed for $x_7 \ldots, x_{11}$.

The FTP generalizes to handle expectation assessments and linear relations(Lad, Dickey, Rahman, 1990).

2.2 Conditional Probability

2.2.1 Conditional Assessments

Surprisingly the LP framework can incorporate some types of conditional assessments as well as the marginal assessments discussed above. Suppose that we have the assessment $P(x|y) = q$. Then $P(xy)$ is a linear function of $P(y)$ for fixed $P(x|y) = q$:

Scenario	Assessments		Computed Bounds
	Marginal	Conditional	
1	$P(x_1) = .010$		$P(x_7) = [0.0, .020]$
	$P(x_2) = .020$		$P(x_8) = [0.0, .020]$
	$P(x_3) = .010$		$P(x_9) = [.030, .060]$
	$P(x_4) = .020$		$P(x_{10}) = [.030, .060]$
	$P(x_5) = .030$		$P(x_{11}) = [.030, .060]$
	$P(x_6) = .050$		
2(a)	$P(x_1) = .010$	$P(x_2\|x_6) = .300$	$P(x_7) = [.015, .015]$
	$P(x_2) = .020$	$P(x_4\|x_6) = .200$	$P(x_8) = [.010, .010]$
	$P(x_3) = .010$		$P(x_9) = [.030, .055]$
	$P(x_4) = .020$		$P(x_{10}) = [.030, .050]$
	$P(x_5) = .030$		$P(x_{11}) = [.030, .050]$
	$P(x_6) = .050$		
2(b)	$P(x_1) = .010$	$P(x_1\|x_6) = .200$	$P(x_7) = [0.0, .020]$
	$P(x_2) = .020$	$P(x_3\|x_6) = .100$	$P(x_8) = [0.0, .020]$
	$P(x_3) = .010$		$P(x_9) = [.030, .060]$
	$P(x_4) = .020$		$P(x_{10}) = [.030, .060]$
	$P(x_5) = .030$		$P(x_{11}) = [.030, .055]$
	$P(x_6) = .050$		
2(c)	$P(x_1) = .010$	$P(x_1\|x_3) = P(x_3\|x_1) = .020$	$P(x_7) = [0.0, .020]$
	$P(x_2) = .020$	$P(x_2\|x_4) = P(x_4\|x_2) = .100$	$P(x_8) = [0.0, .020]$
	$P(x_3) = .010$	$P(x_1\|x_2) = P(x_2\|x_1) = .200$	$P(x_9) = [.030, .056]$
	$P(x_4) = .020$		$P(x_{10}) = [.030, .056]$
	$P(x_5) = .030$		$P(x_{11}) = [.030, .044]$
	$P(x_6) = .050$		
3	$P(x_1) = .010$	$P(x_2\|x_4) = .020$	$P(x_7) = .001$
	$P(x_2) = .020$	$P(x_2\|x_6) = .020$	$P(x_8) = .001$
	$P(x_3) = .010$	$P(x_4\|x_6) = .020$	$P(x_9) = [.030, .041]$
	$P(x_4) = .020$		$P(x_{10}) = [.030, .041]$
	$P(x_5) = .030$		$P(x_{11}) = [.030, .041]$
	$P(x_6) = .050$		$P(x_7\|x_6) = P(x_8\|x_6) = .020$
			$P(x_9\|x_6) = P(x_{10}\|x_6) = [0.020, .820]$
			$P(x_{11}\|x_6) = [0.0, .820]$

Table 2.2: The assessments and bounds determined by the FTP for the simple fault tree.

(2.12) $$P(xy) = qP(y).$$

Subtracting $P(x|y)$ from both sides of (2.12),

$$\begin{aligned} P(xy) &= P(x|y)P(y), \\ P(xy) - P(x|y) &= P(x|y)P(y) - P(x|y), \end{aligned}$$

and

$$-P(x|y) = P(x|y)P(y) - P(x|y) - P(xy)$$

so that

$$P(x|y) = P(xy) + (1 - P(y))P(x|y).$$

(2.13) $$P(x|y) = P(xy) + (1 - P(y))P(x|y)$$

is linear constraint of the proper form for the FTP. If we have an assessment of the form $P(x|y) = q$ de Finetti calls the random variable

$$\tilde{x}_{x|y} = xy + (1 - y)P(x|y)$$

the *conditional quantity* of x given y. When no ambiguity can arise we will abbreviate $\tilde{x}_{x|y}$ as $(x|y)$. Figure 2.2 shows the realm of the $\tilde{x}_{x|y}$. The large dots show the points of the joint realm of $[x, y, (x|y)]^T$ for the case where $P(x|y) = 1/2$. The small dots are the points in the Cartesian product of $\mathbf{R}_x \times \mathbf{R}_y \times \mathbf{R}_{(x|y)}$ that are not in the joint realm. The figure shows the realm of $(x|y)$ where $P(x|y) = 1/2$.

(2.14) EXAMPLE: **A simple fault tree: Scenario 2.**

(a) Suppose we have assessments for $x_1, \ldots, x_6$ as above and assessments for $P(x_2|x_6)$ and $P(x_4|x_6)$. Then we can compute bounds on x_{11} as before by appending the row vectors for the conditional quantities

i. $(x_2|x_6) = x_2x_6 + (1 - x_6)P(x_2|x_6)$ and

ii. $(x_4|x_6) = x_4x_6 + (1 - x_6)P(x_4|x_6)$

to the realm matrix $\mathbf{R}_X$ and by adding the corresponding linear equality constraints

i. $(x_2|x_6) = P(x_2|x_6)$ and

ii. $(x_4|x_6) = P(x_4|x_6)$

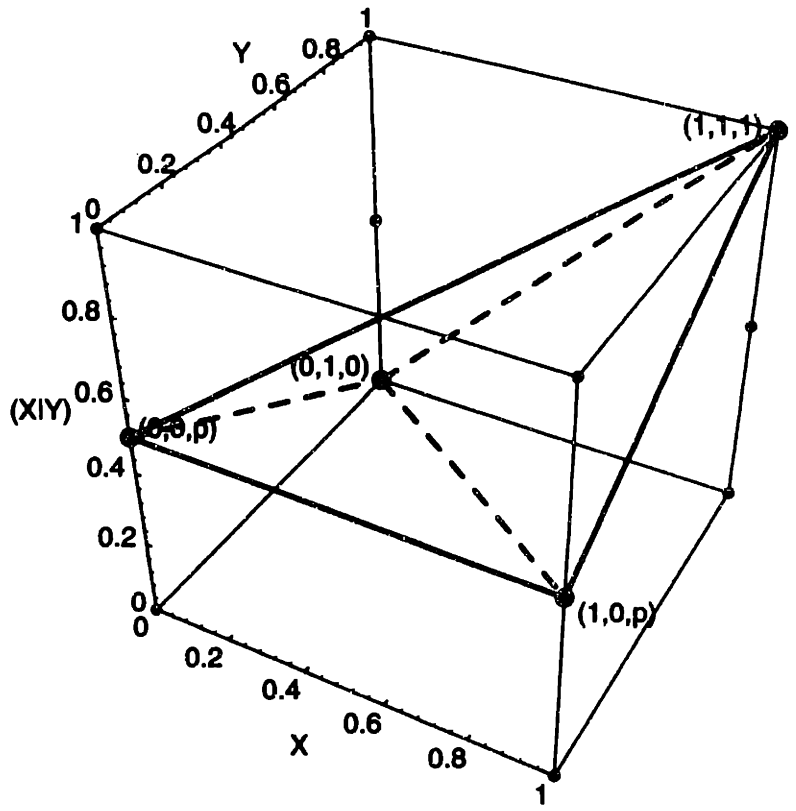

Figure 2.2: The realm of the conditional event (x|y).

to the LP problems (2.11).

The effects of making these additional assessments on the probability bounds of the compound events is shown in Table 2.2. For example, the bounds on the root event x_{11} are reduced from [.03, .06] to [.03, .05]. In particular, the interval computed for x_7 and x_8 both contain just a single point. This is expected since

$$P(x_7) = P(x_2x_6) = P(x_2|x_6)P(x_6)$$

and

$$P(x_8) = P(x_4x_6) = P(x_4|x_6)P(x_6).$$

These additional assessments fix each of the terms in the expressions for $P(x_7)$ and $P(x_8)$ just shown.

(b) Suppose we have assessments $P(x_1|x_6) = .2$ and $P(x_2|x_6) = .1$ in addition to the marginal assessments for $x_1, \ldots, x_6$. The effects of making these additional assessments on the probability bounds of the compound events are shown in Table 2.2. The additional assessments reduce the bounds on x_{11} from [.03, .06] to [.03, .055].

(c) Suppose that in addition to the marginal assessments for $x_1, \ldots, x_6$ we have the following conditional assessments:

i. $P(x_1|x_3) = P(x_3|x_1) = 0.02$,

ii. $P(x_2|x_4) = P(x_4|x_2) = 0.1$, and

iii. $P(x_1|x_2) = P(x_3|x_4) = 0.2$.

Table 2.2 shows the effects of these additional assessments on the probability bounds of the compound events. In particular the additional assessments have the effect of tightening the bounds on x_{11} to $[.03, .044]$.

2.2.2 Computing bounds on conditional events

The FTP in the form given by (2.5) can be used to compute bounds only on the probability of marginal events. It can be generalized to compute bounds on conditional events(Lad, Dickey, & Rahman, 1991).

(2.15) THEOREM: **The Extended Fundamental Theorem of Probability.** Let $X_N = [x_1, \ldots, x_N]^T$ be a vector of N generic events and let $P(X_N) = p_N$ be a vector of probability assessments for X_N. Let x_{N+2} and x_{N+1} be any other generic events. Define x_{N+3} to be the conjunction of x_{N+1} and x_{N+2},

$$x_{N+3} = x_{N+1}x_{N+2},$$

and let $X_{N+3} = [x_1, \ldots, x_N, x_{N+1}, x_{N+2}, x_{N+3}]^T$. Let $\mathbf{R}_{X_{N+3}}$ be the realm matrix for X_{N+3} and E be a vector of the atomic events in $\Omega_{X_{N+3}}$. Then X_{N+3} is representable as

$$X_{N+3} = \mathbf{R}_{N+3}E.$$

Let $\mathbf{R}_N$ be the matrix composed of the N rows in $\mathbf{R}_{N+3}$ corresponding to X_N, and denote the final three rows of $\mathbf{R}_{N+3}$ by r_{N+1}, r_{N+2}, and r_{N+3} respectively.

(a) Then any further assessment of conditional probability $P(x_{N+1}|x_{N+2})$ coheres with $P(X_N)$ just in case $P(x_{N+1}|x_{N+2}) \in [l, u]$ where l and u are the extrema of the following non–linear programming problems:

(2.16) Find the minimum l and the maximum u of

$$\frac{\mathbf{r}_{N+3}\mathbf{q}}{\mathbf{r}_{N+2}\mathbf{q}}$$

subject to

i. $\mathbf{R}_N\mathbf{q} = \mathbf{p}_N$,

ii. $\mathbf{q}^T\mathbf{1} = 1$, and

iii. $\mathbf{q} \geq \mathbf{0}$.

(b) When the feasible region is non–empty, this non–linear programming problem will yield finite extreme value solutions just in case $\mathbf{r}_{N+2}^T\mathbf{q}$ is strictly positive for all vectors $\mathbf{q}$ in the feasible region.

(c) The feasible region is non–empty if and only if $P(\mathsf{X}_N)$ is coherent.

The non–linear programming problems (2.16) can be reduced to LP by use of the method of *linear fractional programming* (cf. Whittle, 1982). This is of great practical importance here because the largest non–linear problems that can be routinely solved are orders of magnitude smaller than the size of linear problems that can be routinely solved. The RIP algorithm like the simplex algorithm for LP exploits the property of linear problems known as the *fundamental theorem of linear programming.* The fundamental theorem of linear programming guarantees that the optimal solution if it exists is contained in a finite set of extreme points in the feasible region[1]. This unique feature of linear programs is the basis for both conventional and column–generation derivatives of the simplex procedure such as the RIP algorithm.

(2.17) **Linear Fractional Problem.** Find the vector $\mathbf{x}$ that maximizes

$$\frac{\mathbf{c}^T\mathbf{q}}{\mathbf{d}^T\mathbf{q}}$$

such that

$$\mathbf{A}\mathbf{q} = \mathbf{b}$$

where $\mathbf{q} \geq \mathbf{0}$ and $\mathbf{d}^T\mathbf{q} > 0$ for all feasible $\mathbf{q}$.

To solve (2.17) it is necessary to transform the problem. Let

(2.18) $$\mathbf{w} = \frac{\mathbf{q}}{\mathbf{d}^T\mathbf{q}}$$

[1]This issue is discussed in detail in section 3.3.1 in chapter 3.

and let

$$\theta = \frac{1}{\mathbf{d}^T\mathbf{q}}. \tag{2.19}$$

Then we have

$$\mathbf{q} = \frac{\mathbf{w}}{\theta}. \tag{2.20}$$

Applying (2.18) and (2.19) to (2.17) we obtain an equivalent LP problem.

(2.21) **Transformed Linear Fractional Problem.** Find the vector $\mathbf{w}$ that maximizes

$$\mathbf{c}^T\mathbf{w}$$

such that

$$\mathbf{A}\mathbf{w} - \mathbf{b}\theta = \mathbf{0}$$

and

$$\mathbf{d}^T\mathbf{w} = 1$$

where $\mathbf{w} \geq \mathbf{0}$ and $\theta > 0$.

In order for the transformed problem (2.17) to have the same feasible region and objective value as (2.21) it is necessary that $\theta > 0$.

(2.22) EXAMPLE: **A simple fault tree: Scenario 3.** Suppose that we have the following assessments:

(a) $P(\mathbf{x}_i)$ for $1 \leq i \leq 6$.

(b) $P(\mathbf{x}_2|\mathbf{x}_4)$,

(c) $P(\mathbf{x}_2|\mathbf{x}_6)$, and

(d) $P(\mathbf{x}_4|\mathbf{x}_6)$

as given in Table 2.2. We desire to compute bounds on $P(x_{11}|x_6)$. In this context define the event vector

$$X_{11|6} = \begin{bmatrix} x_1 \\ \vdots \\ x_6 \\ x_{2|4} \\ x_{2|6} \\ x_{4|6} \\ x_{11} \\ x_{12} \end{bmatrix} = \begin{bmatrix} X_9 \\ x_{11} \\ x_{12} \end{bmatrix}$$

where X_{12} is the logical conjunction of x_{11} and x_6, namely $x_{12} = x_{11}x_6$. Denote the realm matrix corresponding to $X_{11|6}$ by $\mathbf{R}_{X_{11|6}}$ and let $\mathbf{p} = [p_1, \ldots, p_6, p_{2|4}, p_{2|6}, p_{4|6}]^T$ be the vector of probability assessments X_9. Then define $\mathbf{A}$ as follows

$$\mathbf{A} = \begin{bmatrix} \mathbf{R}_{X_9} \\ \mathbf{1} \end{bmatrix}$$

where $\mathbf{R}_{X_9}$ is the first 9 rows of $\mathbf{R}_{X_{11|6}}$ and $\mathbf{1}$ is a length 64 vector of ones. Taking the corresponding rows of $\mathbf{R}_{X_{11|6}}$ define

$$\mathbf{r}_{x_{11}} = [1111111111111111111111111111111111111011101110111011001100110011001]$$

and

$$\mathbf{r}_{x_{12}} = [1010101010101010000000000000000010101010101010100000000000000000].$$

Further let $\mathbf{b} = [\mathbf{p}^T, 1]^T$. Then $P(x_{11}|x_6)$ must lie in the interval $[l, u]$ where l and u are the extreme values obtained from solving the following non-linear programming problems:

(2.23) **Parent Linear Fractional Program.** Find the extrema of

$$\frac{\mathbf{r}_{x_{12}}\mathbf{q}}{\mathbf{r}_{x_6}\mathbf{q}}$$

subject to

$$\mathbf{Aq} = \mathbf{b}$$

and

$$\mathbf{q} \geq \mathbf{0}$$

The above non-linear programming problem can be solved by linear fractional programming. The non-linear problem is transformed into a linear programming problem as follows.

Define

$$\mathbf{w} = \frac{\mathbf{q}}{\mathbf{r}_{x_6}^T \mathbf{q}},$$

and

$$\theta = \frac{1}{\mathbf{r}_{x_6}^T \mathbf{q}}.$$

Note that

$$\mathbf{x} = \frac{\mathbf{w}}{\theta},$$

and then

(2.24) **Transformed Linear Fractional Program.** Find the extrema of

$$\mathbf{r}_{x_{12}}^T \mathbf{w}$$

subject to

(a) $\mathbf{Aw} - \mathbf{b}\theta = \mathbf{0}$,

(b) $\mathbf{r}_{x_6}^T \mathbf{w} = 1$,

(c) and $\mathbf{w} \geq \mathbf{0}$ and $\theta > 0$.

Solving (2.24) yields $P(x_{11}|x_6) \in [0.0, .82]$.

2.2.3 Computational Considerations

Unfortunately, the associated LP problems associated with the FTP quickly become intractable as the number of events under consideration becomes large. The number of decision variables in the LP problems is exponential in the number of events of interest. A problem involving 24 events could yield an LP problem with $2^{24} > 16.7 \times 10^6$ decision variables.

(2.25) EXAMPLE: **A Pumping System**[2]. Figure 2.3 shows the fault tree for a pumping system which has 12 elementary events and 10 compound events. The probability intervals for the compound events were computed using the marginal probabilities of the elementary events shown in the figure. Arguably the fault tree in figure 2.3 is not much more complicated than the tree shown in Figure 2.1, but the realm matrix for the pumping system tree is a 22×4048 matrix. Computing the bounds on $\mathbf{x}_{22}$ took more than 38.04 hours of CPU time and 27 megabytes of virtual memory on a Sun 4/370 using the standard LP routine in Mathematica[3].

In the following chapters we present an efficient column–generation algorithm to solve the LP problems efficiently and exactly even when N is large.

[2]Adapted from Henley & Kumamoto, 1981, p. 71

[3]The linear programming routines in the optimization toolbox for Matlab v 3.0 could not solve this problem at all in the available virtual memory on the same machine (64 megabytes).

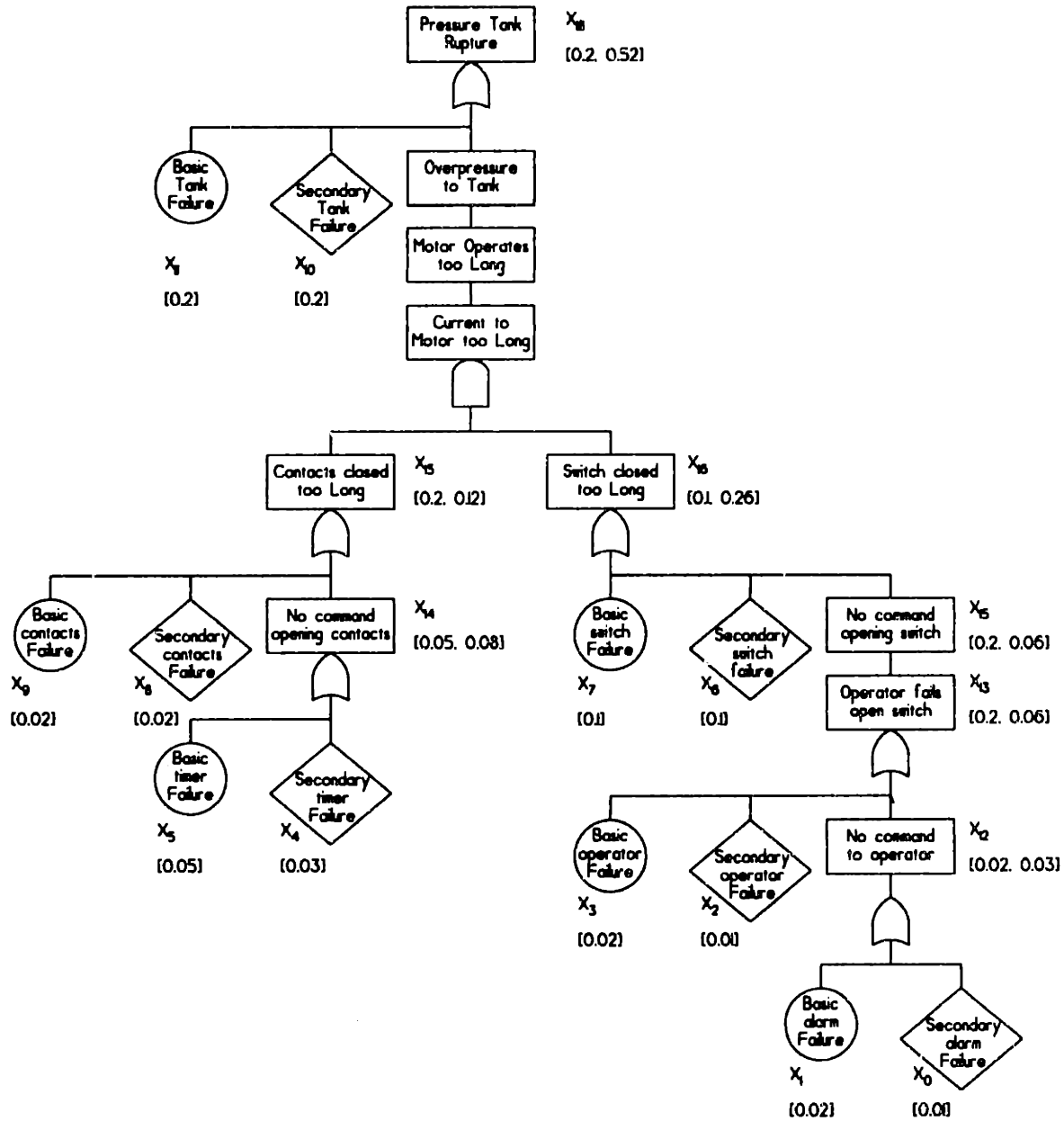

Figure 2.3: The pumping system fault tree.

Chapter 3

The RIP Algorithm

Here we examine the computational aspects of linear programming with realm matrices, beginning with a precise statement of the linear programming problems that need to be solved. Once we have specified the class of linear programming problems that must be solved, we examine properties of the revised simplex method that render it intractable when applied to the class of problems that interest us. Then we develop an algorithm that, by exploiting specific features of this class of problems, is computationally tractable. For the sake of clarity, we will limit the discussion in this section to the case of marginal probabilities. Chapter 4 shows how the the RIP algorithm extends naturally to include conditional probabilities; however, including conditional probabilities in the current discussion would needlessly complicate and obscure the main features of the RIP algorithm. If the advantage of developing a specialized algorithm for solving this class of problems is in doubt, the RIP algorithm solved the pumping system example (2.25) from the preceding chapter in 5.44 CPU seconds on a SUN Sparcstation 10! [1].

3.1 The LP problems for the FTP

Any LP problem can be stated in the following standard form (Luenberger, 1984).

(3.1) Find

$$\min_{\mathbf{q}} \mathbf{c}^T\mathbf{q}$$

[1]CPU times are combined user and system times in SunOS 4.1.x.

subject to

$$\mathbf{Aq} = \mathbf{b}$$

and $q_1, \ldots, q_n \geq 0$.

Suppose we have a vector of $N+1$ events

$$\text{(3.2)} \qquad \mathsf{X} = \begin{bmatrix} \mathsf{x}_1 \\ \vdots \\ \mathsf{x}_N \\ \mathsf{y} \end{bmatrix} = \begin{bmatrix} \mathsf{X}_N \\ \mathsf{y} \end{bmatrix}$$

and a vector of assessments $\mathbf{p}_N$ for X_N. If we want to compute bounds on $P(\mathsf{y})$, then the FTP counsels us to solve an LP problem where

$$\text{(3.3)} \qquad \mathbf{A} = \begin{bmatrix} \mathbf{R}_N \\ \mathbf{1} \end{bmatrix}$$

and

$$\text{(3.4)} \qquad \mathbf{b} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} p_1 \\ \vdots \\ p_N \\ 1 \end{bmatrix}$$

where $\mathbf{R}_N$ and X_N are as defined in (2.4). Similarly $\mathbf{c}^T$ is the row in $\mathbf{R}$ corresponding to the target event, y. Using the FTP to construct an LP representation of our probability model yields the **master LP problem.**

(3.5) **Master LP Problem.** Find

$$\min_{\mathbf{q}} \mathbf{r}_{N+1}^T \mathbf{q}$$

subject to

$$\mathbf{Ax} = \mathbf{b}$$

or equivalently

$$\begin{bmatrix} \mathbf{R}_N \\ 1 \end{bmatrix} \mathbf{q} = \begin{bmatrix} \mathbf{p}_N \\ 1 \end{bmatrix}$$

Recall that a realm matrix for $N+1$ events can have up to 2^{N+1} columns. To represent and solve (3.5) directly would involve storing and operating on a $N \times 2^{N+1}$ matrix (greater than 2.2 billion columns if $N = 32$). Thus, we are motivated to find an economical method for representing and solving this class of LP problems.

If the LP is feasible, the optimal solution, $\mathbf{q}_*$, to (3.5) is a 2^{N+1}-ary vector. The fundamental theorem of linear programming assures us that at most N components of $\mathbf{q}_*$ are non-zero (cf Luenberger, 1984). This fact is central to both the simplex algorithm for linear programming and the RIP algorithm for (3.5). If $N = 32$ then $\mathbf{q}_*$ has more than 2.2 billion components. However, the fundamental theorem of linear programming guarantees that at most 32 of them are non-zero. Corresponding to each non-zero component in $\mathbf{q}_*$ is a column of $\mathbf{A}$.

3.2 Representing realm matrices

If we are to be able to work with large event vectors, then we need an efficient representation of the realm matrix. In this context an efficient representation would be one that is proportional in size to N rather than 2^N, can be effectively constructed from the event vector in time and space proportional to N, and that supports the kinds of operations that we will need to perform on the realm matrix. Figures 3.1 and 3.3 show the realm of a disjunction, $\mathsf{z} = \mathsf{x} \vee \mathsf{y}$, and a conjunction, $\mathsf{z} = \mathsf{x} \wedge \mathsf{y}$, respectively. The large dots are the points in the joint realm of $[\mathsf{x}, \mathsf{y}, \mathsf{z}]^T$. The small dots are the points in $\mathbf{R}_\mathsf{x} \times \mathbf{R}_\mathsf{y} \times \mathbf{R}_\mathsf{z}$ not in the joint realm. Figure 3.2 shows the realm of $\mathsf{z} = \mathsf{x} \vee \mathsf{y}$ and the two planes

(3.6)

$$\begin{aligned} z &= x + y \\ z &= \frac{1}{2}(x + y). \end{aligned}$$

Figure 3.4 shows the realm of $\mathsf{z} = \mathsf{x} \wedge \mathsf{y}$ and the two planes

(3.7)

$$\begin{aligned} z &= \frac{1}{2}(x + y) \\ z &= \frac{1}{2}(x + y - 1). \end{aligned}$$

In both cases the simplex formed by the planes and the faces of the unit cube contains exactly the realm of z. In other words $\mathbf{R}_z$ is characterized by a set of linear inequalities in each case. It turns out that for Boolean relations, this is true in general. It is also the case for conditional quantities as well. Moreover, as will be developed in subsequent sections, this representation leads to efficient algorithms for the operations we wish to perform on realm matrices in conjunction with the FTP.

(3.8) EXAMPLE: **The Safety System Example.** Consider once again the safety system example (1.4) and (2.7). We have an event vector with four components

$$\mathsf{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}.$$

Recall that the components of X are restricted by

$$x_2 \leq 1 - x_3, \tag{3.9}$$

$$x_4 = 1 - (1 - x_1)(1 - x_2)(1 - x_3), \tag{3.10}$$

and that, as reflected in (3.10), x_4 is the union of x_1, x_2, and x_3. These constraints can be expressed in terms of linear inequalities and integer restrictions $x_i \in \{0, 1\}$. The constraint (3.9) is already linear:

$$x_2 + x_3 \leq 1.$$

Constraint (3.10) is equivalent to a pair of linear inequalities

$$x_4 \leq x_1 + x_2 + x_3 \leq 3x_4. \tag{3.11}$$

We derive (3.11) by expanding (3.10)

$$x_4 = x_1 + x_2 + x_3 - x_1x_2 - x_1x_3 - x_2x_3 + x_1x_2x_3.$$

Observe that $x_1x_2x_3 \leq x_1x_2$ which implies

$$x_4 \leq x_1 + x_2 + x_3. \tag{3.12}$$

Likewise $x_i \leq x_4$ for $1 \leq i \leq 3$, thus

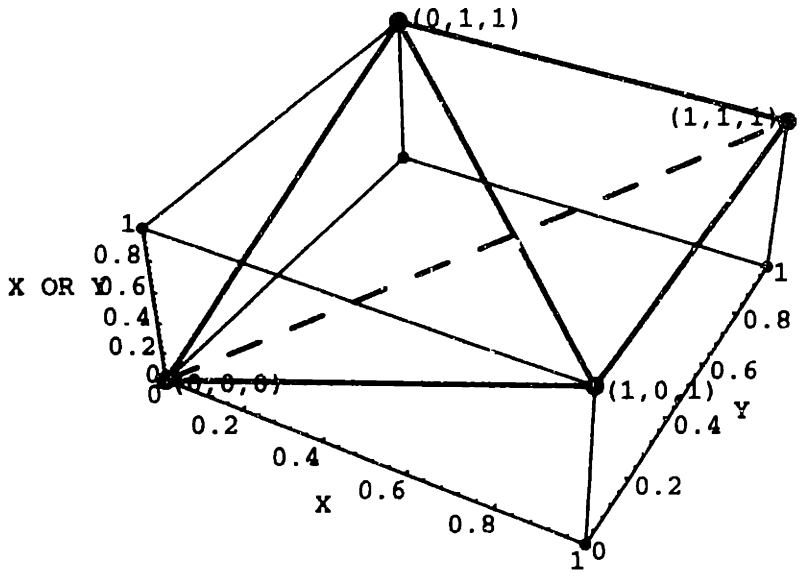

Figure 3.1: The realm of a logical disjunction of two events.

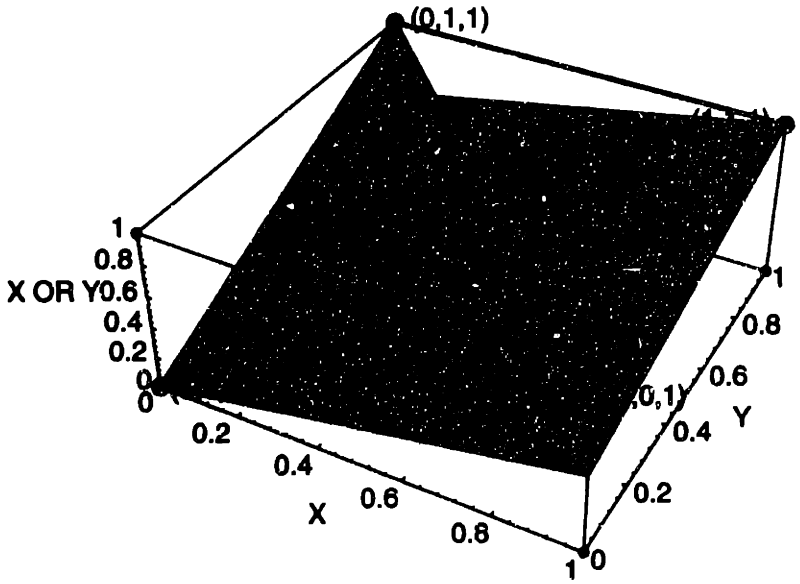

Figure 3.2: The constraints used to represent a disjunction.

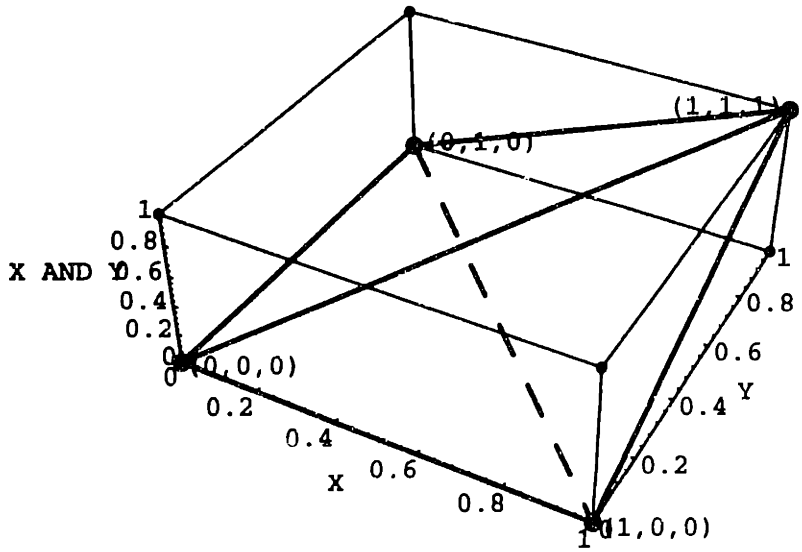

Figure 3.3: The realm of a logical conjunction of two events.

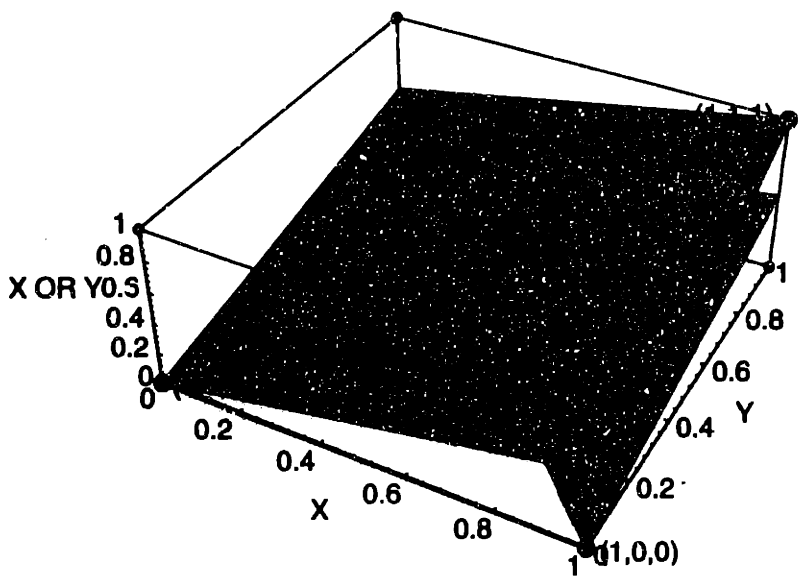

Figure 3.4: The constraints used to represent a conjunction.

(3.13) $\quad x_1 + x_2 + x_3 \leq 3x_4.$

The above linear inequalities can be combined into a linear matrix inequality

(3.14) $\quad \mathbf{Mz} \leq \mathbf{g}$

which characterizes the columns of $\mathbf{R}$ (possible states of the world) as follows. Set

$$\begin{aligned}\mathcal{R} &= \{\mathbf{z}|\ \mathbf{z} \text{ is a column vector of } \mathbf{R}\} \\ &= \{\mathbf{z}|\mathbf{Mz} \leq \mathbf{g}\} \\ &= \left\{\mathbf{z}\left|\begin{bmatrix} 0 & 1 & 1 & 0 \\ -1 & -1 & -1 & 1 \\ 1 & 1 & 1 & -3 \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\right.\right\}.\end{aligned}$$

The above example illustrates how Boolean functions can be represented by linear inequality integer constraints. If we desire to handle any arbitrary Boolean function then it is sufficient that we be able to represent disjunction and negation (cf Ebbinghaus, Flum, & Thomas, 1984;Mendelson, 1987). In addition to Boolean relations, the FTP can accommodate conditional probability assessments, and arbitrary linear relations, The FTP also requires the normalization constraint. Thus it is necessary that each of these classes of constraints be representable as a system of linear inequalities.

Let

(3.15) $\quad \mathsf{X}_N = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}$

be an event vector such that if x_i is a relation on other events in X_N then x_i is one of the following:

(a) a disjunction of events in X_N,

(b) a conjunction of events in X_N,

(c) a negation of an event in X_N,

(d) a conditional quantity on events in X_N,

(e) a linear relation on events in X_N, or

(f) a normalization constraint.

We would like to have an algorithm to construct a system of linear inequalities $\mathbf{M}_{\mathbf{X}_N}\mathbf{z} \leq \mathbf{g}_{\mathbf{X}_N}$ from the event vector $\mathbf{X}_N$ such that a vector $\mathbf{z}$ which satisfies $\mathbf{M}_{\mathbf{X}_N}\mathbf{z} \leq \mathbf{g}_{\mathbf{X}_N}$ is a column vector of $\mathbf{R}_{\mathbf{X}_N}$.

(3.16) PROPOSITION: The pair $\left(\mathbf{M}_{\mathbf{X}_N}, \mathbf{g}_{\mathbf{X}_N}\right)$ is a matrix function with domain $\mathbf{X}_N$. For a given $\mathbf{X}_N$, the vector $\mathbf{z}^T$ is a column of $\mathbf{R}_{\mathbf{X}_N}$ if and only if

$$\mathbf{z} \in \left\{\mathbf{z} | \mathbf{M}_{\mathbf{X}_N}\mathbf{z} \leq \mathbf{g}_{\mathbf{X}_N}, z_i \in \mathbf{R}_{\mathbf{x}_i}\right\}$$

for some $\mathbf{M}_{\mathbf{X}_N}$ and $\mathbf{g}_{\mathbf{X}_N}$.

We can construct a system of inequalities $\mathbf{M}_{\mathbf{X}_N}\mathbf{z} \leq \mathbf{g}_{\mathbf{X}_N}$ that satisfies (3.16) by iterating through the events in $\mathbf{X}_N$. For each $\mathbf{x}_i$ in $\mathbf{X}_N$ we add to the system $\mathbf{M}_{\mathbf{X}_N}\mathbf{z} \leq \mathbf{g}_{\mathbf{X}_N}$ the appropriate inequalities.

(3.17) DEFINITION: If $\mathbf{X}_N$ is an event vector as in (3.15), then for every $\mathbf{x}_i \in \mathbf{X}_N$, $(\mathbf{M}_{\mathbf{X}_N}, \mathbf{g}_{\mathbf{X}_N})$ is such that

(a) *Conjunction Case.*

$$\mathbf{x}_i = \bigwedge_{j=1}^{n} \mathbf{x}_{k_j}$$

if and only if

i. $\mathbf{x}_{k_1} + \cdots + \mathbf{x}_{k_n} - n\mathbf{x}_i \geq 0$

ii. $\mathbf{x}_{k_1} + \cdots + \mathbf{x}_{k_n} - \mathbf{x}_i \leq n - 1$

are in $(\mathbf{M}_{\mathbf{X}_N}, \mathbf{g}_{\mathbf{X}_N})$.

(b) *Disjunction Case.*

$$\mathbf{x}_i = \bigvee_{j=1}^{n} \mathbf{x}_{k_j}$$

if and only if

i. $\mathbf{x}_{k_1} + \cdots + \mathbf{x}_{k_n} - \mathbf{x}_i \geq 0$

ii. $\mathbf{x}_{k_1} + \cdots + \mathbf{x}_{k_n} - n\mathbf{x}_i \leq 0$

are in $(\mathbf{M}_{\mathbf{X}_N}, \mathbf{g}_{\mathbf{X}_N})$.

(c) *Negation Case.*

$$\mathsf{x}_i = \neg \mathsf{x}_j$$

if and only if

$$\mathsf{x}_i + \mathsf{x}_j = 1$$

is in $(\mathbf{M}_{\mathsf{X}_N}, \mathbf{g}_{\mathsf{X}_N})$.

(d) *Conditional Quantity Case.*

$$\mathsf{x}_i = (\mathsf{x}_j | \mathsf{x}_k)$$

if and only if

i. $\mathsf{x}_i = (1 - \mathsf{x}_k) P(\mathsf{x}_j | \mathsf{x}_k) + \omega$

ii. $\omega \leq \mathsf{x}_j$

iii. $\omega \leq \mathsf{x}_k$

iv. $\omega \geq \mathsf{x}_j + \mathsf{x}_k - 1$

v. $\omega \in [0, 1]$

are in $(\mathbf{M}_{\mathsf{X}_N}, \mathbf{g}_{\mathsf{X}_N})$.

(e) *Linear Relation Case.*

$$\mathsf{x}_i \leq \sum_{j=1}^{n} a_j \mathsf{x}_{k_j}$$

if and only if

$$\mathsf{x}_i \leq \sum_{j=1}^{n} a_j \mathsf{x}_{k_j}$$

is in $(\mathbf{M}_{\mathsf{X}_N}, \mathbf{g}_{\mathsf{X}_N})$.

(f) *Normalization Case.* x_i is a normalization event if and only if

$$\mathsf{x}_i = 1$$

is in $(\mathbf{M}_{\mathsf{X}_N}, \mathbf{g}_{\mathsf{X}_N})$.

Definition (3.17) is the basis for a linear-time algorithm for representing a realm matrix $\mathbf{R}_{\mathsf{X}_N}$ by a system of linear inequalities, $(\mathbf{M}_{\mathsf{x}_N}, \mathbf{g}_{\mathsf{x}_N})$. For the current purpose we will limit discussion to logical relations within the scope of (3.17). The procedure extends naturally to well–formed formulae in conjunctive normal form (Blair,Jeroslow, & Lowe, 1985; Jeroslow, 1989). For each relation in X_N, at most 4 inequalities are introduced into$(\mathbf{M}_{\mathsf{x}_N}, \mathbf{g}_{\mathsf{x}_N})$. Thus, given an event vector of N events the conversion

algorithm would output a matrix expression of the form (3.14) where $\mathbf{M}_{x_N}$ is no larger than $4N \times N$ and $\mathbf{g}$ is no larger than $4N$.

3.3 The RIP Algorithm

This section discusses a column-generation algorithm for solving the LP problems associated with the FTP. The RIP algorithm is a modification of the revised simplex method. The key to the effectiveness of the RIP algorithm is that the matrix $\mathbf{A}$ as defined in (3.8) is never represented directly. Instead (3.17) is used to construct a system of linear inequalities that characterizes the column vectors of $\mathbf{A}$. Second, during each iteration the standard simplex algorithm calculates the reduced cost coefficient for each of the non-basic columns of $\mathbf{A}$. The column with the minimum reduced cost is brought into the basis. The search for the vector with the most negative reduced cost at each iteration is done by employing a related mixed-integer programming problem. The size of the mixed-integer programming problem is much smaller than the size of the primary linear programming problem. If in the master LP problem $\mathbf{A}$ has dimensions $N \times 2^N$ and 2^N decision variables, then the related IP problem has N decision variables. Figure 3.5 shows a diagram of an iteration of the RIP algorithm.

3.3.1 Relevant features of the simplex algorithm

There are several features of the simplex algorithm for linear programming which are exploited in the RIP algorithm. The following discussion is based on Luenberger (1984).

(3.18) DEFINITION: **Basic Solution.** Given a set of m simultaneous linear equations in n variables,

$$\mathbf{Ax} = \mathbf{b},$$

let $\mathbf{B}$ be any non-singular $n \times n$ submatrix consisting of columns of $\mathbf{A}$. Then if all $n - m$ components of $\mathbf{x}$ not associated with $\mathbf{B}$ are set to zero, the solution to the resulting set of equations is a *basic solution* of $\mathbf{Ax} = \mathbf{b}$ with respect to basis $\mathbf{B}$. The components of $\mathbf{x}$ associated with columns of $\mathbf{B}$ are called basic variables. A basic solution, $\mathbf{x}$, is called a *basic feasible solution* if $\mathbf{x} \geq \mathbf{0}$.

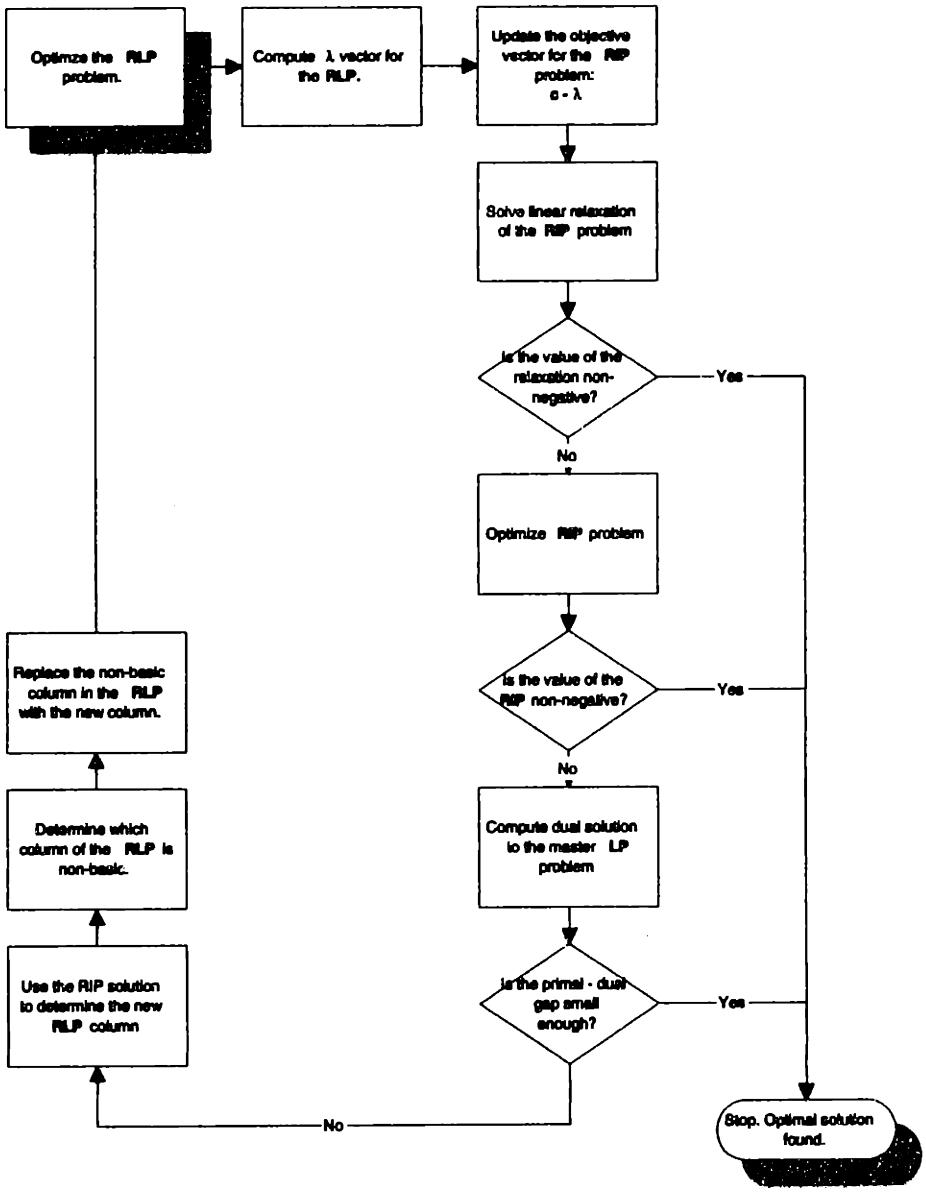

Figure 3.5: An iteration of the RIP column-generation algorithm.

For a linear program (3.1) a basic feasible solution that is optimal is called an *optimal basic feasible solution.* Central to the theory and practice of linear programming is the following theorem.

(3.19) THEOREM: **The fundamental theorem of linear programming.** Given a linear program (3.1) where $\mathbf{A}$ is an $m \times n$ matrix of rank m,

(a) if there is a feasible solution, there is a basic feasible solution;

(b) if there is an optimal feasible solution, there is an optimal basic feasible solution;

Thus, solving a linear program reduces to a search of the space of basic feasible solutions. The simplex algorithm is an efficient iterative procedure for a search of this space. Given a basic feasible solution $\mathbf{B}$, each iteration of the simplex procedure finds another basic feasible solution $\mathbf{B}'$ with a lower objective value or concludes that $\mathbf{B}$ is optimal. The basis $\mathbf{B}$ and the basis $\mathbf{B}'$ differ in exactly one column.

Given a basis $\mathbf{B}$ for (3.1), the simplex procedure determines which of the non-basic columns would be advantageous to enter the basis by calculating the *reduced cost* for each non-basic column

$$r_j = c_j - \boldsymbol{\pi}^T \mathbf{a}^{(j)} \tag{3.20}$$

where $\mathbf{a}^{(j)}$ is the jth column of $\mathbf{A}$, c_j is the jth objective coefficient, and $\boldsymbol{\pi}$ is the vector of dual prices for $\mathbf{B}$. Any column of $\mathbf{A}$ with $r_j < 0$ will yield a lower objective value. If no column has a negative reduced cost, then the current basis, $\mathbf{B}$ is optimal. Typically the column with the most negative reduced cost is chosen as the column to enter the basis. The simplex procedure then determines which column should be replaced in $\mathbf{B}$ to yield $\mathbf{B}'$ so as to maintain feasibility.

3.3.2 The related LP problem and the related IP problem

We outline the standard revised simplex method first, and then contrast it with our integer programming algorithm. An iteration of the standard revised simplex algorithm consists of the following steps (cf. Sedgewick, 1983).

(3.21) **Revised Simplex Method.** Given a basic feasible solution,

(a) Calculate the reduced cost vector $\mathbf{r}$. If $\mathbf{r} \geq 0$, the current basis is optimal.

(b) Determine the vector to enter the basis.

(c) Determine the vector to leave the basis.

(d) Update the basis and solution.

If the realm matrix for X_N can be represented by (3.16) and if the reduced cost (3.20) of a column $a^{(j)}$ can be represented by a linear function, $\mathbf{f}$ of $a^{(j)}$ then (3.21)(a,b) could be construed as a *mixed-integer programming problem.* Given a basis, $\mathbf{B}$, the remaining steps in a simplex iteration (3.21)(c,d) operate only on $\mathbf{B}$ and are tractable for large N. Moreover, using a mixed-integer program brings the well developed theory and algorithms for integer programming to bear on our problem of probabilistic inference.

Let $\mathbf{c}^T$ be the row in $\mathbf{R}$ corresponding to the event for which we want to compute bounds, and let $\mathbf{b} = [\mathbf{p}_N^T, 1]^T$. Then we have have the following linear programming problem and its dual.

(3.22) **Master LP Problem (MLP).** Find

$$\min_{\mathbf{x}} \mathbf{c}^T\mathbf{x}$$

subject to

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

or equivalently

$$\begin{bmatrix} \mathbf{z}_1 & \dots & \mathbf{z}_{\eta(N)} \\ 1 & \dots & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{p}_N \\ 1 \end{bmatrix}$$

(3.23) **Dual of the Master LP Problem.** Find

$$\max_{\boldsymbol{\pi}} \boldsymbol{\pi}^T\mathbf{b}$$

subject to

$$\boldsymbol{\pi}^T\mathbf{A} \geq \mathbf{c}^T.$$

Thus

$$\boldsymbol{\pi}^T\mathbf{a}^{(j)} \geq c_j$$

is the dual constraint imposed by the jth column of $\mathbf{A}$. The RIP algorithm proceeds as follows. Given a basis $\mathbf{B}$ and its inverse $\mathbf{B}^{-1}$, the column of $\mathbf{A}$ with the most negative reduced cost is the solution to a mixed-integer programming problem.

Let the dual prices for the current basis $\mathbf{B}$ be $\boldsymbol{\pi} = [\boldsymbol{\pi}^T, \theta]^T$. We distinguish θ, the coefficient in $\boldsymbol{\pi}$ corresponding to the normalization constraint as it plays an important role in a subsequent stage of the algorithm. The column of $\mathbf{A}$ with the most negative reduced cost is given by the following integer programming problem.

(3.24) **Related IP Problem (RIP).** Find

$$\min_{\mathbf{z}} \left(\mathbf{f}^T \mathbf{z} - \boldsymbol{\pi}^T \mathbf{z} - \theta \right)$$

subject to

$$\mathbf{M}\mathbf{z} \leq \mathbf{g}$$

and $z_i \in \mathbf{R}_{\mathbf{x}_i}$

The objective function of the RIP must be $c_j - \boldsymbol{\pi}^T \mathbf{a}^{(j)}$ for all vectors in the feasible region of the RIP. The vector of dual prices $\boldsymbol{\pi}$ is determined by the current basis $\mathbf{B}$, and thus is constant for a given iteration. We need c_j to be a linear function of $\mathbf{a}^{(j)}$ which would make the RIP objective function $(\mathbf{f} - \boldsymbol{\pi})\mathbf{a}^{(j)}$. It turns out that the solution to this problem is quite straightforward. If the target event $\mathbf{y}$ is one of the forms in (3.17), then we can add $\mathbf{y}$ as a variable in the RIP problem. Then $\mathbf{f}$ has a particularly simple form. If $\mathbf{y}$ is the kth variable in the RIP then

$$(3.25) \qquad f_i = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases}$$

Thus, $\mathbf{f}$ is a binary vector that simply picks out the correct variable in the RIP, namely the variable corresponding to c_j. An additional benefit is that we need the objective coefficient in the RLP for the column to enter the basis anyway and in this way we can get it directly from the RIP solution vector instead of having to compute it separately. The only complication is that we give up the one–to–one correspondence between columns of the RLP and solution vectors of the RIP. This complication is minor and only requires that we maintain the correct mapping between the variables in the RIP problem and coefficients in the RLP constraints. Thus, for each iteration of the RIP algorithm, we construct a new objective vector for the RIP using the current dual prices.

It is necessary that $\mathbf{f}^T\mathbf{z} - \boldsymbol{\pi}^T\mathbf{z} - \theta$ be the reduced cost of $\mathbf{z} \in \mathbf{A}$. In order for (3.24) to be a linear integer problem, the objective vector must linear. In order to accomplish this, we assume that the component c_j of $\mathbf{c}^T$ corresponding to the column $\mathbf{z}$ of $\mathbf{A}$ is given by the linear function $\mathbf{f}^T\mathbf{z}$. That this can be done, and the specific form that $\mathbf{f}$ will take are discussed in Chapter 4. The optimal solution $[\mathbf{z}^T, 1]$ to (3.24) is the vector to enter the basis. The RIP algorithm then proceeds as in the standard revised simplex method. A standard branch-and-bound IP algorithm can be used to solve the RIP problem (cf. Bradley, Hax, & Magnanti, 1977). Although integer programming is NP-complete, in practice the branch-and-bound procedure has been shown to be an effective algorithm for many practical problems so long as N is not too large, roughly $N \leq 60$.

In the course of solving (3.24), the branch-and-bound IP algorithm provides valuable information about the master LP problem (3.22). The branch-and-bound algorithm searches for the optimal integer solution to (3.24) by repeatedly subdividing the set of feasible solutions until the optimal solution is determined. The branch-and-bound algorithm initiates the search for the optimal IP solution by solving a linear relaxation of (3.24).

(3.26) **Linear Relaxation of the Related IP Problem.** Find

$$\min_{\mathbf{z}} \left(\mathbf{f}^T\mathbf{z} - \boldsymbol{\pi}^T\mathbf{z} - \theta\right)$$

subject to

$$\mathbf{Mz} \leq \mathbf{g}$$

The solution to (3.26) gives a lower bound on the objective function value for the optimal integer solution to (3.24) (cf. Bradley, Hax, & Magnanti, 1977). Let $\ell_{\mathbf{ip}}$ be the value of the objective function for the optimal integer solution to (3.24), and let $\ell_{\mathbf{lp}}$ be the value of the objective function for the optimal solution to (3.26). Then

$$(3.27) \qquad \ell_{\mathbf{lp}} \leq \ell_{\mathbf{ip}} \leq \mathbf{f}^T\mathbf{z} - \boldsymbol{\pi}^T\mathbf{z} - \theta$$

for all feasible $\mathbf{z}$ (cf. Bradley, Hax, & Magnanti, 1977). If the branch-and-bound algorithm determines that $\ell_{\mathbf{lp}} \geq 0$ then our current solution to the master LP problem must be optimal(i.e., there is no column of $\mathbf{A}$ with negative reduced costs). Moreover, the IP algorithm yields a sequence of feasible integer solutions as it searches for the optimal integer solution. Given a feasible integer solution $\mathbf{z}'$ with cost

$$\mathbf{f}^T\mathbf{z}' - \boldsymbol{\pi}^T\mathbf{z}' - \theta,$$

we can use ℓ_{lp} to decide whether to terminate the search. If

$$(\mathbf{f}^T\mathbf{z}' - \boldsymbol{\pi}^T\mathbf{z}' - \theta) - \ell_{\mathrm{lp}} \leq \epsilon$$

and

$$(\mathbf{f}^T\mathbf{z}' - \boldsymbol{\pi}^T\mathbf{z}' - \theta) \leq 0,$$

then we let $\mathbf{z}'$ be the column to enter the basis; otherwise we continue the branch-and-bound search.

Surprisingly the optimal IP solution provides a lower bound on the master LP problem (3.22). We have the following theorem.[2]

(3.28) THEOREM: Let $\boldsymbol{\pi}' = [\boldsymbol{\pi}', \theta']^T$ be the current dual prices for the master LP and $\ell_{\mathrm{lp}}{}'$ be the lower bound on the corresponding IP problem (3.24). Then a feasible objective value for (3.23) is

$$\mathbf{b}^T \begin{bmatrix} \boldsymbol{\pi}' \\ \theta' \end{bmatrix} + \ell_{\mathrm{lp}}{}'$$

Proof. For every feasible $[\mathbf{z}, 1]^T$ and $c_i = \mathbf{f}^T\mathbf{z}$ in (3.24),

$$\mathbf{f}^T\mathbf{z} - \boldsymbol{\pi}'^T\mathbf{z} - \theta' \geq \ell_{\mathrm{lp}}{}'$$

or

$$c_i - \boldsymbol{\pi}'^T\mathbf{z} - \theta' \geq \ell_{\mathrm{lp}}{}'$$

or

$$c_i \geq \boldsymbol{\pi}'^T\mathbf{z} + \theta' + \ell_{\mathrm{lp}}{}'.$$

Thus,

$$\mathbf{A}^T \begin{bmatrix} \boldsymbol{\pi}' \\ \theta' \end{bmatrix} \leq \mathbf{c}$$

and $[\boldsymbol{\pi}', \theta' + \ell_{\mathrm{lp}}{}']^T$ is dual-feasible with objective value $\mathbf{b}^T[\boldsymbol{\pi}', \theta' + \ell_{\mathrm{lp}}{}']^T$. ■

Thus each simplex iteration provides us with a solution to the dual of (3.22). We can use the dual solution and the current primal solution as bounds on the optimal objective value of the master LP problem quitting when the bounds are sufficiently tight for the purpose at hand.

[2]The theorem and proof are due to Rob Freund.

Chapter 4

Implementing the RIP Algorithm

This chapter discusses issues related to implementing the RIP algorithm, including the nature of the system specification, representing the system logic in terms of linear inequalities, constructing the RIP and RLP problems, and solving the principle problem.

A flow chart of the RIP algorithm is shown in Figure 4.1. The figure illustrates the structure of the RIP algorithm as a series of discrete processes. The algorithm inputs a system specification, determines the RIP and RLP problems, finds a feasible basis for the problem, and searches for an optimal basis. The specification of the system events, logic and probability assessments are discussed in Section 4.1. The procedures for computing the RLP and RIP problems from the system specification are discussed in Section 4.2. Finally Section 4.4 covers the procedures involved in finding a feasible solution and in computing an optimal solution.

The RIP algorithm naturally divides into two modules, a system parser and a system solver. Dividing the RIP algorithm in this way enables the mechanisms of parsing the system logic to be segregated from those related to optimizing the system. Furthermore this allows for a well-defined interface to both modules. For example the current implementation of the system parser outputs the RLP and RIP problems as MPS files. Because these are a standard input format for LP problems, the system solver can be easily ported to a different LP optimizer. This also allows the solver to jump directly into phase II if a feasible basis for the system is already available. This is a simple matter of indicating to the system solver to read the advanced basis MPS file instead of the artificial variables MPS file.

The system parser reads a system specification as input. From the system specification the parser determines the set of events in the system and generates a schema for the RLP and the RIP problems based on the system logic. The parser encapsulates

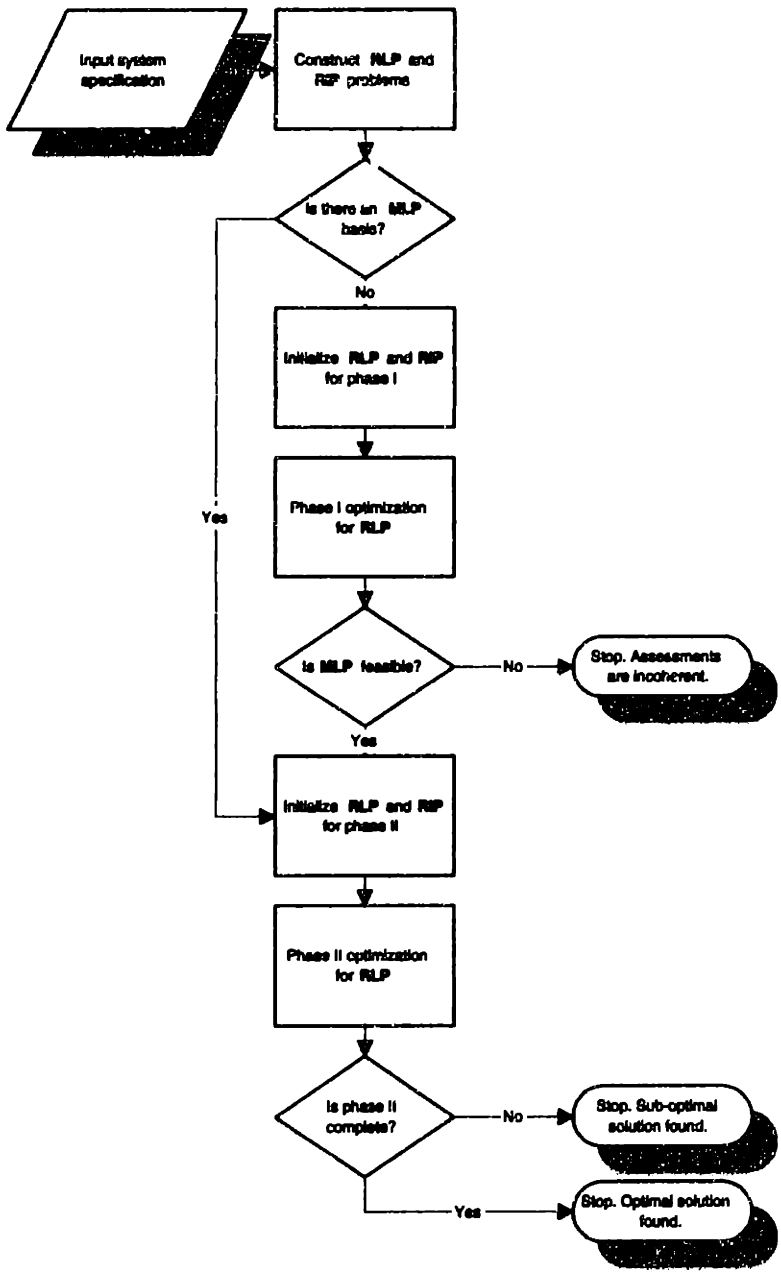

Figure 4.1: The RIP algorithm.

the mechanisms related to manipulating the logical structure of the system.

The system solver takes as input a RIP schema, a RLP schema, and a representation of the system events. The event representation includes whether or not each of the events is assessed, which events are logical relations of other events, and which event is the target. The information about which events are assessed and which are logical relations is used only to compute the mapping of events to decision variables in the RIP and the RLP problems. Thus, the system solver does not need to maintain any representation of the system logic.

The system solver module encapsulates the mechanisms for optimizing an FTP system. The system solver implements the procedures for creating the related problems, running an iteration of the RIP algorithm, computing the dual solution, and testing for optimality. Another concrete advantage of this structure is that the modifications required to extend the solver to linear fractional programming are relatively moderate because the solver is not tied to the system logic.

4.1 The system description

The FTP operates on a realm matrix representation of the system logic. The realm matrix is an enumeration of all of the realizable valuations of the events in the specification. Each column of the realm matrix is a realizable joint event on the events in the system. The realm matrix for a given system is determined by the system logic. The RIP algorithm implements the FTP by representing the realm matrix implicitly as a set of linear inequalities corresponding to the system logic. The RIP algorithm requires that

(a) the system logic;

(b) the system event probabilities; and

(c) the parameters to the RIP algorithm

be specified for the FTP problem. In addition the current implementation of the RIP algorithm requires that each logical relation be either

(4.1) (a) a negation of some other event in the system,

(b) a conjunction of other events in the system,

(c) a disjunction of other events in the system,

(d) a conditional quantity defined in terms of other events in the system, or

(e) a linear relation among other events in the system.

These restrictions introduce more events than is strictly necessary, but simplifies the processing of the logic substantially without limiting expressibility (Jeroslow, 1989; Ebbinghaus, Flum, & Thomas, 1984). In particular this limitation entails a simple mapping between events in the system and constraints in the RIP problem, and allows the use of a non-recursive parser to construct the RLP and RIP problems from the system specification.

(4.2) EXAMPLE: **The Safety System.** Figure 4.2 shows the system specification for the safety system example (1.4). The set of events has been expanded from example (1.4) in order to satisfy the requirements of the current implementation. The correspondence between the events in Example (1.4) and Figure 4.2 is shown in Table 4.1. The safety system in example (1.4) has 4 events. There are two logical restrictions in the system:

(a) At most one of x_2 and x_3 can be true.

(b) The event x_4 is true if any other event x_1, x_2, or x_3 is true.

These restrictions are captured in the systems specification in Figure 4.2 as follows:

(a) The restriction on x_2 and x_3 requires introducing 1 additional variable into the RIP specification. The event Y3 is a linear relation on x_2 and x_3.

(b) Event x_4 is a disjunction of other events in the system, and thus is directly representable.

4.2 The related problems

In order to implement the FTP we must have an algorithm for constructing the RLP and the RIP from the system specification. This is accomplished by iterating through each of the events in the system. Each event corresponds to zero or more constraint equations in the RLP and RIP. For each event we have a logical relation (possibly null) on other events in the system and an assessment interval (possibly null). Figure

Event Specification	Example	Description
Y0	x_1	Sprinkler head clogs
Y1	x_2	Sprinkler head bursts
Y2	x_3	Insufficient water pressure
Y3	–	$x_2 + x_3 \leq 1$
Y4	x_3	The system fails.

Table 4.1: The correspondence between the events in the Safety System example and the RIP system specification.

```
1     @CONFIGURATION_SECTION          // begin system config.  sect.
2         @BEGIN                      // mark the start of a section
3           @NAME=safety              // the system name
4           @BASISFILE=@NULL          // a basis for the system, if avail.
5           @LPFILE=safety-rlp.mps    // the RLP for the system,
6           @MIPFILE=safety-rip.mps   // the RIP for the system,
7           @MAXPIVOTSI=10            // allowable pivots phase I & II
8           @MAXPIVOTSII=10           //
9           @TOLERANCE=.000001        //
10          @EPSILON=.001             // exit condition on primal/dual gap
11          @NEVENTS=4                // # of events in the system
12          @TEV=3                    // index of target event (from 0)
13          @SENSE=MIN                // min or max?
14        @END                        // mark the end of a section
15    @LOGIC_SECTION                  // begin logical spec.
16        @BEGIN
17          Y0  @NULL
18          Y1  @NULL
19          Y2  @NULL
20          Y3  @LINEAR(Y1+Y2 <= 1 )
23          Y4  @OR(Y1,Y2,Y3)
24        @END
25    @PROBABILITY_SECTION            // begin probability spec.
26        @BEGIN
27          Y0  .007  .010            // prob interval for Y0,Y1,Y2
28          Y1  .005  .005
29          Y2  .015  .015
30          Y3  @NULL  @NULL           // no prob spec'ed for
31          Y4  @NULL  @NULL           // Y3 - Y4
34        @END
```

Figure 4.2: The system specification for the safety system.

4.3 shows an overview of the process for constructing the RLP and RIP problems from a system specification.

Suppose we have a vector of events, X_N as in (3.15). An assessment for an event $\mathsf{x}_i \in \mathsf{X}_N$ is a pair $v = (l_{\mathsf{x}_i}, u_{\mathsf{x}_i})$ such that l_{x_i} and u_{x_i} are either real numbers or null. A *system specification* is a triple

$$\mathcal{S} = (\mathsf{X}_N, v_{\mathsf{x}_N}, \mathsf{y})$$

where $v_{\mathsf{X}_N} = [v_{\mathsf{x}_1}, \ldots, v_{\mathsf{x}_N}]^T$ is a vector of assessments for X_N.

4.2.1 The related linear problem

The related linear problem, RLP for a system specification, $\mathcal{S}$, is based on (3.22). Actually there is a distinct RLP for each iteration of the RIP algorithm. Recall that the vector decision variables for the master linear problem (3.22) is $\mathsf{E} = [\mathsf{e}_1, \ldots, \mathsf{e}_\eta]$ the vector of atomic events in Ω_X. The RLP is derived from the master linear problem by representing only those decision variables in the current basis $\mathbf{B}$. Thus, the RLP problem is given by

(4.3) **Related Linear Problem.** Find

$$\min_{\mathbf{q}} \mathbf{rq}$$

subject to

$$[\mathbf{B} \,|\mathbf{a}^*]\,\mathbf{q} = \mathbf{b}$$

where $\mathbf{B}$ is a basis for (3.22), $\mathbf{a}^*$ is any column in the constraint matrix for (3.22) not in $\mathbf{B}$, and $\mathbf{r}$ are the coefficients in $\mathbf{r}_{N+1}$ corresponding to the columns of $\mathbf{B}$ and $\mathbf{a}^*$.

The optimal solution to (4.3) is a basic feasible solution to (3.22). Suppose $\mathbf{a}'$ is a possibly sub-optimal feasible solution to the RIP problem (3.24) having a negative objective value. Then let the new RLP problem be

(4.4) Find

$$\min_{\mathbf{q}} \mathbf{rq}$$

subject to

$$[\mathbf{B} \,|\mathbf{a}']\,\mathbf{q} = \mathbf{b}$$

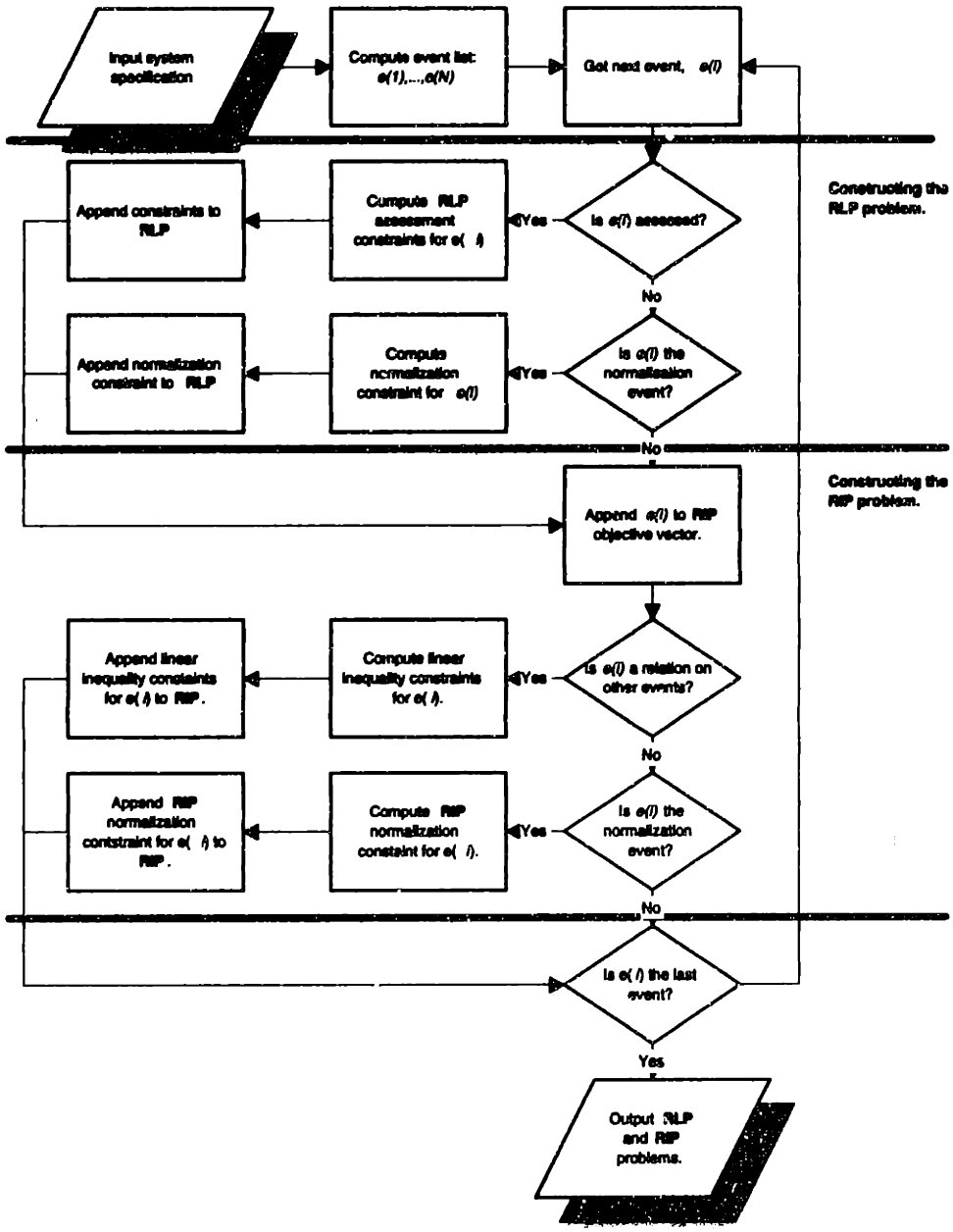

Figure 4.3: Building the RIP and RLP problems, an overview.

The basis $\mathbf{B}'$ for the optimal solution to (4.4) will include the column $\mathbf{a}'$ and some column of $\mathbf{B}$ will be non-basic. The objective value of (4.4) will be less than (4.3). There are two advantages to representing the current basic feasible solution to (3.22) by (4.3).

(a) After replacing the non-basic column $\mathbf{a}^*$, with $\mathbf{a}'$, (4.3) can be optimized with a single simplex iteration. Thus, we can directly use one of the efficient implementations of the simplex algorithm that are available.

(b) Efficient LP implementations use a matrix factorization with a rank one update (order N complexity) rather than explicitly finding the inverse of the current basis during each simplex iteration (order N^3 complexity). Furthermore, sophisticated LP implementations such as CPLEX can tolerate changing a non-basic column without disturbing the factorization. This avoids an expensive refactorization of the basis during each iteration (order N^3 complexity). Periodically the basis can be explicitly refactored in order to prevent excessive accumulation of floating–point errors.

4.2.2 The related integer problem

The related integer problem, RIP for a system specification, $\mathcal{S}$, is given by (3.24). Given a system specification, $\mathcal{S} = \left(\mathsf{X}_N, \upsilon_{\mathsf{X}_N}, \mathsf{y}\right)$, iteratively construct (3.24) as follows. For each $\mathsf{x}_i \in \mathsf{X}_N$ and y, append to $(\mathbf{M}, \mathbf{g})$ the appropriate inequalities given in (3.17). Finally append the normalization constraint $\mathsf{x}_{\text{norm}} = 1$. The vector of decision variables for the RIP problem is

$$[\mathsf{X}_N, \mathsf{y}, \mathsf{x}_{\text{norm}}]^T = [\mathsf{x}_1, \ldots, \mathsf{x}_N, \mathsf{y}, \mathsf{x}_{\text{norm}}]^T. \tag{4.5}$$

The objective vector for the RIP problem is given by

$$\mathbf{f} - [\boldsymbol{\pi}^T, \theta]. \tag{4.6}$$

We let

$$\mathbf{f} = [f_1, \ldots, f_{N+2}] \tag{4.7}$$

where

$$f_i = \begin{cases} 0 & i \neq N+1 \\ 1 & i = N+1 \end{cases}$$

Note the slight adjustment to the definition of the objective vector in (4.6) compared to (3.24). We have included both the normalization event, x_{norm} and the target event y as decision variables in the RIP. Thus, we assume that y is an event subject to the same conditions as the events in X_N. In turn we have defined $\mathbf{f}$ in such a manner as to pick out the appropriate coefficient of $\mathbf{z}$, namely the one corresponding to the target event y. Thus, $c_j = \mathbf{f}^T\mathbf{z}$.

In the same manner that the RLP is a series of related problems, there is a distinct RIP for each iteration of the RIP algorithm. Unlike the RLP the constraints in the problem, $(\mathbf{M}, \mathbf{g})$ remain fixed. The objective vector for the RIP depends on the dual prices for the RLP $\boldsymbol{\pi}$, which are specific to a particular basis for the master problem (3.22). Thus with each iteration we compute the dual prices for the RLP and then compute the new objective vector for the RIP using (4.6). The dual prices for the RLP are

$$\boldsymbol{\pi} = \begin{bmatrix} \pi_{\mathsf{x}_1} \\ \vdots \\ \pi_{\mathsf{x}_N} \\ \pi_{\mathsf{x}_{\text{norm}}} \end{bmatrix} \tag{4.8}$$

where π_{x_i} is the dual price for the constraint in the RLP corresponding to x_i. Let

$$\boldsymbol{\pi} = [\pi_1, \ldots, \pi_{N+2}]^T \tag{4.9}$$

where

$$\pi_i = \begin{cases} \pi_{\mathsf{x}_i} & 1 \leq i \leq N \\ \pi_{\mathsf{x}_{\text{norm}}} & i = N+2 \\ 0 & \text{otherwise} \end{cases}$$

Then the components in the dual prices for the RLP will map to the correct decision variable in the RIP.

Once we have a representation of the RLP and RIP problems, then these representations must be converted into a form suitable for input into an optimization package such as CPLEX. The standard format for linear programs is the MPS file format (CPLEX Inc, 1994). The system parser produces an MPS file representing the RIP and the RLP which are taken as input by the RIP implementation.

It is straightforward to parse a system specification such as in Figure 4.2 to extract the set of events in the system, the target event, the system logic, the set of assessments, and parameters to the algorithm. To construct the RLP and RIP problems

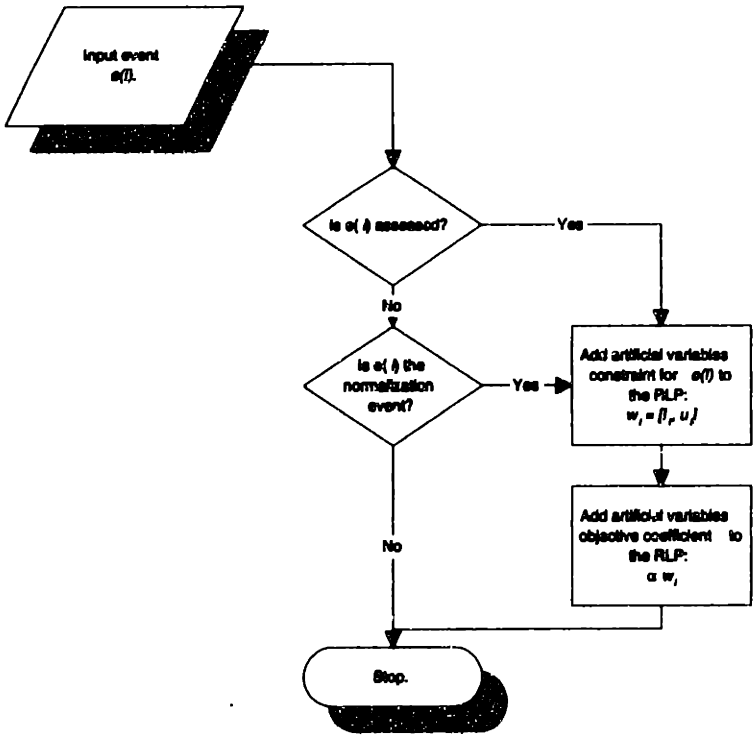

Figure 4.4: Constructing the RLP problem.

from the specification, the program iterates through the list of events. For each event x_i the program computes the equations in the constraint matrix for both the RLP and RIP problems that correspond to the event. The constraint equations for y_i are then appended to the set of constraint equations for the appropriate problem, RLP or RIP. Figures 4.4 and 4.5 show the mapping of event logic into constraint equations of the RLP and RIP problems.

4.3 Linear fractional problems

Solving systems using the extended fundamental theorem of probability (2.15) requires that a linear fractional problem be solved. Section 2.2.2 in chapter 2 showed how to transform the linear fractional problem into an equivalent linear problem. The RIP algorithm requires a number of modifications to accommodate linear fractional programming (LFP). The system solver itself requires only a moderate amount of additional program logic to correctly optimize LFP problems. However, the system logic parser requires substantial modification in order to be capable of generating both straight LP and the transformed LFP schema.

First the system specification grammar must be extended to include an event class for the *target conditional*, `@TCOND()`. A target conditional event in the system

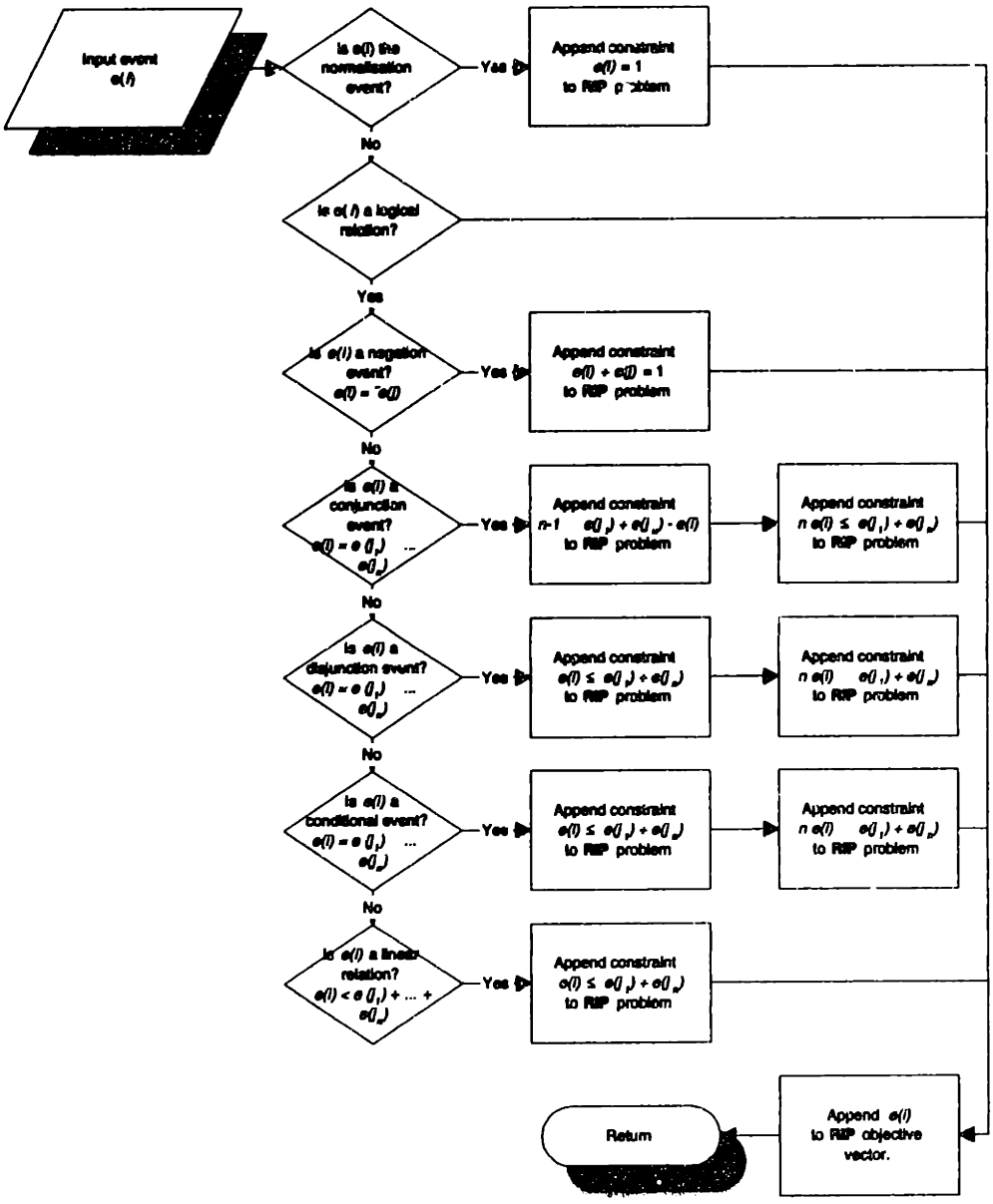

Figure 4.5: Constructing the RIP problem.

specification

(4.10) Y @TCOND(X1,X2)

introduces the target $P(y) = P(x_i|x_2)$ where x_1 and x_2 are events specified to the system.

The system parser must generate the transformed LFP. When the parser encounters a conditional target x given y, it must introduce a new event z into the system representing the conjunction of the these two events (i.e., the numerator in the LFP objective function). Subsequently z is treated exactly as the target event in the non-LFP case. The parser must introduce θ as a decision variable in the RIP problem which includes transforming the right-hand side of system into the appropriate column of coefficients r_θ. Finally the parser must introduce the constraint for y (i.e., the denominator in the LFP objective function). This adds significantly to the complexity of the system parser since it must manage the mapping of these new and transformed events between the RLP and the RIP problems.

By comparison the modifications to the system solver are much more local. Linear fractional programming introduces an additional variable θ into the RLP problem. The corresponding vector of constraint coefficients r_θ is the negation of the right hand side vector in (2.16). For the RLP r_θ must be distinguished because it is not a feasible solution for the RIP problem. Thus, r_θ must never be replaced by another column in the RLP or else it would be permanently lost. Thus, r_θ is tagged as not replaceable, but it can be pivoted into or out of the RIP basis as necessary by the optimizer.

The system logic parser generates the initial RLP and RIP problems for the RIP-algorithm. The RIP problem includes the additional constraints for the event z which is the conjunction of the events x and y in the target assessment $P(x|y)$. Aside from the modification to prevent the r_θ column from being replaced, the basic structure of a RIP iteration remains untouched. The conditions for phase I and II optimality are undisturbed as is the basic mechanism for updating the RLP and RIP problems at each iteration. The system solver only needs to maintain the proper mapping of system events to decision variables in the RLP and RIPwhich is only slightly different from the non-LFP case.

4.4 Solving the problems

4.4.1 Computational considerations

Eliminating redundant constraints from the RLP The RLP problem stated in (4.3) may have redundant constraints. If $x_i \in X_N$ and $x_i = [0.0, 1.0]$, then there is a constraint in the RLP for x_i which is redundant with respect to the normalization constraint. In practice such redundant constraints will arise when events are introduced into the problem to represent logical relations which are not directly representable in (3.15). In the CSIS fault tree example from Figure 1.5, $x_{42} = x_{40} \wedge x_{41}$ which in turn are logical functions of other events. Unless there is an explicit non–trivial assessment for x_{40} and x_{41}, then the constraint introduced into the RLP problem for them will be redundant. Let

$$x_N = \begin{bmatrix} x_0 \\ \vdots \\ x_{29} \\ x_{30} \\ \vdots \\ x_{41} \end{bmatrix}. \tag{4.11}$$

Constructing the RLP according to (4.3) gives an LP with 42 rows and 43 columns, but there are explicit assessments only for $x_0, \ldots, x_{29}$. Thus, there are 12 redundant constraints in the RLP which can be removed. The RIP problem still requires these events because they are the link between the assessed events, $x_0, \ldots, x_{29}$, and the target event x_{41}. Thus, an event $x_i \in X_N$ is a constraint in the RLP problem just in case it has a non-trivial assessment.

Taming an aggressive LP optimizer Sophisticated LP codes such as CPLEX use an array of performance enhancing techniques during optimization that significantly reduce the number of iterations required to find an optimal solution. For instance, CPLEX will tolerate solutions which are slightly infeasible. The RIP algorithm has an unfortunate interaction with this behavior. The optimizer will find a solution during iteration i which is infeasible, but within the tolerance limits. Iteration $i+1$ replaces the one non-basic column with a new one. The optimizer in this context may then re-scale the infeasibilities, and conclude that the problem is infeasible. This behavior is especially prevalent when degenerate solutions are encountered. Because the LP has N rows and $N+1$ columns the optimizer's ability to find a feasible basis is tightly

constrained.

One solution to this problem is to keep a larger number of extra columns around. The current implementation keeps N extra columns. The additional columns can significantly reduce the number of iterations required to find the optimal solution while only marginally increasing the cost of each iteration. The effect of this modification can be quite dramatic. Solving a particular set of assessments for the CSIS system required more than 5000 iterations when only one extra column was retained. The system was solvable only for a narrow subset of the possible coherent assessments. Retaining N additional columns in the RLP reduced the number of iterations for this case from over 5000 to less than 700!

In principle the set of degenerate solutions for an LP problem has measure zero. On the other hand degenerate solutions are not at all infrequent in practice, and given the very restricted form of the LP problems related to the FTP, degenerate solutions are frequently encountered. A second benefit to keeping more non-basic columns in the RLP is to handle degenerate solutions, particularly when the phase I optimal solution is degenerate and there are artificial variables in the basis at level zero. In this case, the artificial variable can be removed from the basis, by manually pivoting into the basis any of the N non-basic columns which has a non-zero entry in the same row as the non-zero component of the artificial variable. The non-basic column will enter at a zero level. If the artificial variable is the only non-zero entry in that row, then the row is redundant and the artificial variable can remain in the basis at a zero level indefinitely.

4.5 Applying the RIP algorithm

A number of examples of the FTP have been presented throughout this paper. These illustrative examples were first computed using the linear programming facilities in MATHEMATICA(Wolfram Research, Inc., 1991). These examples showcased the limitations of a standard simplex algorithm for these problems, and served as a test suite to validate the implementation of the RIP algorithm. Reviewing these examples and contrasting the computational requirements of the off–the–shelf simplex method and the RIP algorithm will highlight the advantages of the RIP algorithm.

A note regarding performance The comparison of the off–the–shelf LP routines in MATHEMATICA with the RIP algorithm is something of a straw–man. While the MATHEMATICA routine is very robust, it is just one routine in a general purpose

mathematical toolkit. Moreover, the MATHEMATICA system because of its breadth introduces significant overhead both with respect to time and space resources. A highly–optimized dedicated LP package package such as CPLEX would certainly yield much better results. A well–behaved 5000 column LP even a dense one, is well within the limits of CPLEX on a mid–range RISC workstation [1]. On the other hand, while CPLEX might perform n times faster or be able to solve problems which are n times larger, even CPLEX is no match for an LP with more than 1 billion columns.

The performance figures reported in this chapter are based on the CPU times reported by the operating system[2]. These figures can vary substantially between identical runs of a process depending on such factors as machine load, network load, paging, etc. The times reported are from one run of the given computation. Thus, the solution times are a merely a general indicator of the time requirements of the computations for this class of problems. They are not a definitive benchmark for a broad class of problems.

All computations were run on SUN SPARC workstations running SunOS v4.1.x. Because of licensing restrictions, the analyses were run on several different machines. The MATHEMATICA results were obtained on a Sun 4/370 workstation. All computations involving CPLEX (i.e., the RIP algorithm) were run on a Sun Sparcstation 10. The SPECmark rating of the Sparcstation 10 is roughly 3 times that of the 4/370.

4.6 The fault tree suite

The CSIS fault tree system. Recall the fault tree for the CSIS system shown in figure 1.5. The fault tree has 30 basic failure events and 12 non–basic failure events. Figure 1.5 shows marginal probability assessments for the 30 basic failure events. The realm matrix for the fault tree contains more than $2^{30} \geq 1.07 \times 10^9$ columns. Thus the standard LP algorithm was not applied to this system.

The results of applying the RIP algorithm to the CSIS system are shown in Table 4.2. The bounds for each of the non–basic failure events were computed using the sub–tree for each intermediate event. For example the sub–tree for x_{30} contains the

[1] Ideally the baseline for comparing a "standard" LP algorithm to the RIP algorithm would have been to use CPLEX instead of MATHEMATICA. Time constraints prevented this comparison. The power of a system like MATHEMATICA is evident in the roughly 25 line program to run the pumping system example system which took about an hour to write and verify.

[2] All times are based on CPU times reported by the SunOS `time(1)` utility. The MATHEMATICA results based on the `Timing[]` facility in MATHEMATICA include user time only. The RIP times include both user and system times.

Target	*Events*	*Realm*	*Pivots*	*Bounds*	*Terminating condition*
x_{30}	5	8	7 (5:2)	0.610	RIP solution non-negative
			7 (5:3)	0.641	RIP relaxation non-negative
x_{31}	5	8	7 (5:2)	0.410	RIP solution non-negative
			7 (5:3)	0.500	RIP relaxation non-negative
x_{32}	8	32	13 (11:2)	0.753	RIP solution non-negative
			16 (11:5)	0.753	RIP relaxation non-negative
x_{33}	8	32	13 (11:2)	0.410	RIP solution non-negative
			18 (11:7)	0.523	RIP relaxation non-negative
x_{34}	12	128	35 (33:2)	0.610	RIP solution non-negative
			43 (33:10)	0.808	RIP relaxation non-negative
x_{35}	12	128	28 (26:2)	0.410	RIP solution non-negative
			38 (26:12)	0.639	RIP relaxation non-negative
x_{36}	15	1024	53 (51:2)	0.610	RIP solution non-negative
			66 (51:15)	0.838	RIP relaxation non-negative
x_{37}	15	1024	34 (32:2)	0.410	RIP solution non-negative
			51 (32:19)	0.680	RIP relaxation non-negative
x_{38}	18	4096	144 (142:2)	0.610	RIP solution non-negative
			94 (77:17)	0.841	RIP relaxation non-negative
x_{39}	18	4096	54 (52:2)	0.410	RIP solution non-negative
			72 (52:20)	0.704	RIP relaxation non-negative
x_{40}	22	32768	168 (166:2)	0.610	RIP solution non-negative
			181 (166:15)	0.921	RIP relaxation non-negative
x_{41}	22	32768	102 (100:2)	0.410	RIP solution non-negative
			132 (100:32)	0.751	RIP relaxation non-negative
x_{42}	44	1.07×10^9	696 (487:209)	0.020	RIP solution non-negative
			553 (487:66)	0.751	RIP solution non-negative

Table 4.2: The results of applying the RIP algorithm to the CSIS fault trees.

Target	*Events*	*Realm*	*Pivots*	*Bounds*	*Dual*	*Terminating condition*
Pumping system fault tree						
x_{18}	20	4096	30 (28:2)	0.200		RIP solution non-negative
			45 (28:17)	0.520		RIP solution non-negative
Simple fault tree — Scenario 1.						
x_{11}	12	64	13 (11:2)	0.030		RIP solution non-negative
			17 (11:6)	0.060	0.702	RIP solution non-negative
Simple fault tree — Scenario 2.						
x_{11}	12	64	14 (12:2)	0.030		RIP solution non-negative
			17 (12:5)	0.520		RIP solution non-negative
Simple fault tree — Scenario 3.						
$(x_{11}\|x_6)$	19	64	22 (13:9)	0.060		RIP solution non-negative
			22 (13:9)	0.820		RIP solution non-negative

Table 4.3: The results of applying the RIP algorithm to the fault trees.

events $x_0 - x_2$. For each event listed, the two rows correspond to the computations for the lower and the upper bound respectively. For each non–basic event Table 4.2 shows the number of events in the RIP system specification and number of columns in the realm matrix. The number of columns in the realm matrix is 2^N where N is the number of basic failure events in the subtree for the given non–basic event. The *Pivots* column shows both the total number of pivots and the breakdown of phase I vs phase II. The final column lists the condition which determined that the optimal solution had been found.

The entire case was required 270 seconds of CPU time to compute all of the bounds shown in Table 4.2. This figure includes both the time required to solve the systems plus the overhead introduced by the UNIX shell script used to sequentially run each of the cases shown in the table. The RIP algorithm can be configured to use an advanced basis so that it can be jump started directly into phase II. This was not done in this case. Thus, the times reflect finding the same basis twice, once for the lower bound and again for the upper one. Of the 270 CPU seconds, 214 CPU seconds were spent solving the root event of the fault tree, x_{42}. The x_{42} case has 30 basic failure events compared to the next largest cases, x_{40} and x_{41} which have 15 basic failure events which each required 10-15 CPU seconds to complete.

The pumping fault tree system. The fault tree for the pumping system is shown in figure 2.3. This system has 12 basic failure events and 6 non-basic failure events. This is the largest problem that could be run using the LP routines in MATHEMATICA. The MATHEMATICA computations required 38.04 hours of CPU time. The problem was too large for the optimization toolbox in MATLAB v3.x to optimize in a 64MB virtual address space. The RIP algorithm required 5.44 CPU seconds!

The simple fault tree system. The simple fault tree examples used to develop the FTP in Chapter 2 were originally run using MATHEMATICA. As part of the test suite for the RIP algorithm, these examples were run on the RIP algorithm. The last 4 rows of Table 4.3 present the results of the RIP algorithm for two of the scenarios from chapter 2. The entry for $\mathbf{x}_{11}$ corresponds to example (2.10). The rows for $(\mathbf{x}_{11}|\mathbf{x}_6)$ correspond to example (2.22). Both of these examples required less than $\frac{1}{2}$ second of CPU time compared to just over 20 CPU seconds for the equivalent MATHEMATICA calculations. The entire Scenario 3 case in Table 2.2 required 244 CPU seconds for the MATHEMATICA implementation compared to just under 2 CPU seconds for the RIP implementation.

For the set of problems in the fault tree suite, in only one case was the dual solution better than the trivial bounds $0 \leq \mathbf{x} \leq 1$.

4.7 Inferno

In this we examine the FTP and the RIP algorithm in the context of the INFERNO semantics for inference networks. Section 4.7.1 presents a small example which shows in great detail how the FTP and the RIP algorithm are applied to a small inference network.

The INFERNO semantics provide a mechanism for translating inference network models into logical system specifications which can be solved by using the FTP. The inference networks illustrate some of the computational issues that must be addressed in a robust implementation of the RIP algorithm, particularly with respect to degenerate linear-programs.

4.7.1 Prospector

Recall the PROSPECTOR example from Chapter 1. The FTP can be applied to the example without making any of the implicit problematic assumptions made in the

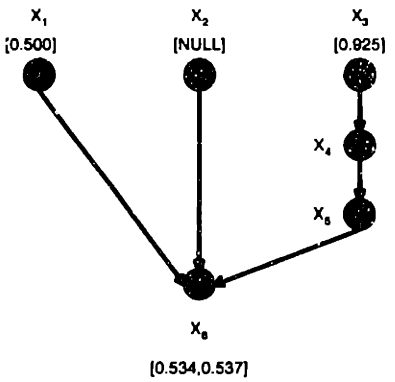

Figure 4.6: The FTP assessments for the PROSPECTOR example 1.2.

inference network implementation. The DAG defined by the rules is shown in Figure 4.6. Table 4.4 gives an interpretation of the inference rules embodied in the network using the INFERNO semantics [3]. The assessments related to the links in Figure 4.6 are shown in Table 4.4. Recall that an inference network separates knowledge about the domain from knowledge related to a specific case. The rule–base contains domain knowledge that generalizes across cases. The rule–base is implemented in INFERNO semantics by pairs of enables/requires relations. Case–specific information is input by the user of the inference network who assigns probabilities to the propositions that are root (evidence) nodes in the network. The system then computes the probability of the terminal (hypothesis) nodes by propagating the implications of the values at the evidence nodes through the network to the hypothesis nodes.

Suppose our user assigns probabilities to x_1 and x_2 as shown in Figure 4.6. The user is unable to make any assessment to x_2 other than the trivial assessment $0 \leq P(\mathsf{x}_2) \leq 1$. The RIP algorithm system specification corresponding to these assessments is shown in Figure 4.7.

Applying the RIP algorithm The RIP algorithm is implemented in two steps. The first step involves submitting the system specification to a system logic parser which parses the system logic and constructs problem files in MPS format for the RIP and RLP problems suitable for input into CPLEX. The MPS file format is a standard file format for specifying linear and integer–linear programs (CPLEX Optimization Inc, 1994). The system parser is written in PERL v4.x, an interpreted language designed for processing arbitrary text files. The flexible regular expression capabilities of PERL

[3]The probabilities in Table 4.4 and Figure 4.6 were constructed specifically for this example and are *not* derived from the actual PROSPECTOR system or the Neopolitan's (1990) critique of rule–based expert systems.

Rule	*Relations*	Interpretation
$x_1 \rightsquigarrow x_6$	x_1 enables x_6 with strength 0.750	$P(x_6\|x_1) \geq 0.750$
	x_6 requires x_1 with strength 0.900	$P(\neg x_6\|\neg x_1) \geq 0.900$
$x_2 \rightsquigarrow x_6$	x_2 enables x_6 with strength 0.830	$P(x_6\|x_2) \geq 0.830$
	x_6 requires x_2 with strength 0.900	$P(\neg x_6\|\neg x_2) \geq 0.900$
$x_3 \rightsquigarrow x_4$	x_3 enables x_4 with strength 0.900	$P(x_4\|x_3) \geq 0.900$
	x_4 requires x_3 with strength 0.900	$P(\neg x_4\|\neg x_3) \geq 0.900$
$x_4 \rightsquigarrow x_5$	x_4 enables x_5 with strength 0.883	$P(x_5\|x_4) \geq 0.883$
	x_5 requires x_4 with strength 0.900	$P(\neg x_5\|\neg x_4) \geq 0.900$
$x_5 \rightsquigarrow x_6$	x_5 enables x_6 with strength 0.727	$P(x_6\|x_5) \geq 0.727$
	x_6 requires x_5 with strength 0.900	$P(\neg x_6\|\neg x_5) \geq 0.900$

Table 4.4: Interpretation of PROSPECTOR rules.

make it an ideal candidate for parsing system logic. Parsing the system specification in Figure 4.7 required just under 12 CPU seconds on a SUN 4/330 workstation. Optimizing the system required 1.90 CPU seconds on a SUN Sparcstation 10 to compute both the upper and lower bound. The RIP algorithm required 9 phase I iterations to find a basis. To find the upper bound required 2 phase II pivots. The phase I basis was optimal for the lower bound. The objective values are 0.534 and 0.550 for the lower and upper bound respectively.

Increasing the probability of each of the "requires" relations in Table 4.4 from 0.900 to 0.925 causes the bounds on $P(x_6)$ to converge to the neighborhood of 0.535. With all the "requires" relations set at 0.925, CPLEX finds a feasible basis in phase I. In the first iteration of phase II the optimizer finds optimal solutions with infeasibilities just below the optimizer's default threshold of infeasibility. This suggests that the set of assessments is borderline inconsistent.

```
@CONFIGURATION_SECTION
  @BEGIN
    @NAME=p1.l
    @BASISFILE=@NULL
    @LPFILE=p1.l.rlp.mps
    @MIPFILE=p1.l.rip.mps
    @MAXPIVOTSI=50
    @MAXPIVOTSII=50
    @TOLERANCE=.000001
    @EPSILON=.001
    @VERBOSE=0
    @NEVENTS=22
    @TEV=5
    @SENSE=MIN
  @END
@LOGIC_SECTION
  @BEGIN
    X1  @NULL
    X2  @NULL
    X3  @NULL
    X4  @NULL
    X5  @NULL
    X6  @NULL
    X7  @NOT(X1)
    X8  @NOT(X2)
    X9  @NOT(X3)
    X10  @NOT(X4)
    X11  @NOT(X5)
    X12  @NOT(X6)
    X13  @COND(X6,X1)
    X14  @COND(X12,X7)
    X15  @COND(X6,X2)
    X16  @COND(X12,X8)
    X17  @COND(X4,X3)
    X18  @COND(X10,X9)
    X19  @COND(X5,X4)
    X20  @COND(X11,X10)
    X21  @COND(X6,X5)
    X22  @COND(X12,X11)
  @END
⋮
```

```
@PROBABILITY_SECTION
  @BEGIN
    X1  0.500  0.500
    X2  @NULL  @NULL
    X3  0.925  0.925
    X4  @NULL  @NULL
    X5  @NULL  @NULL
    X6  @NULL  @NULL
    X7  @NULL  @NULL
    X8  @NULL  @NULL
    X9  @NULL  @NULL
    X10  @NULL  @NULL
    X11  @NULL  @NULL
    X12  @NULL  @NULL
    X13  0.750  1.000
    X14  0.900  1.000
    X15  0.830  1.000
    X16  0.900  1.000
    X17  0.900  1.000
    X18  0.900  1.000
    X19  0.883  1.000
    X20  0.900  1.000
    X21  0.727  1.000
    X22  0.900  1.000
  @END
```

Figure 4.7: The system specification for the PROSPECTOR example.

Chapter 5

Conclusions

5.1 The RIP algorithm

The paradox of the FTP is that such a conceptually simple device generates such profoundly complex computations. The RIP algorithm addresses this issue in a fashion that mirrors the conceptual elegance of the FTP. Like the FTP the RIP algorithm is appealing for its conceptual simplicity. And just like the FTP the RIP algorithm involves a number of complexities in practice.

The goal of the current project has been to take the conceptually elegant framework of the FTP and develop it into a useful assessment and inference paradigm. The FTP as developed by de Finetti (1974a,1974b) provided a framework for understanding the interaction between the logical and the probabilistic structure of a system. The RIP algorithm for the FTP preserves both the fidelity of the FTP regarding the probability calculus without making any implicit probabilistic assumptions or arbitrarily limiting the set of inferences that may be drawn. De Finetti embraced incomplete probability models which he called *partial prevision assertions*, but did not fully exploit the connection of the FTP and the theory of linear programming. Nilsson (1986;1993) communicated the FTP to a much wider audience, and addressed the computational implications of the FTP. Nilsson's focus on the computational aspects of the FTP led to a more precise characterization of the relationship between the FTP and standard baselines of computational complexity such as *NSAT* (Fagin & Halern, 1989; Georgakopoulos, Kavvadias, and Papadimitriou, 1988). Nilsson like de Finetti did not fully exploit the connection with linear programming to address interval assessments and conditional probability. The work of Lad, Dickey, and Rahman (1990;1991) fully leveraged the connection with linear programming to greatly

increase the scope of the FTP to encompass conditional probability. Lad, Dickey, and Rahman did not address the computational aspects of the problem. In the meantime practitioners in operations research have developed powerful numerical techniques such as integer programming for representing complex system logic and performing logical inference (Jeroslow, 1989).

The goal of this project has been to unify these diverse threads, and in particular embrace both inference in an incomplete probability model and the linear programming framework. In their discussion of Nilsson's work Frisch and Haddawy (1994) observe that the most significant problem with the FTP is characterizing the set of atomic events. The RIP algorithm draws on techniques from operations research to effectively resolve this issue for a large class of interesting problems. Similarly Nilsson's probabilistic logic and Quinlan's (1983) INFERNO are restricted to marginal probability assessment and inference. The RIP algorithm gracefully handles conditional probability for a restricted class of assessments and inferences. The RIP algorithm also extends Nilsson's probabilistic logic by allowing interval assessments.

Once a feasible basis has been found, each iteration of the RIP algorithm generates a bound on the objective value of the MLP. This bound is an effective tool for evaluating the cost of continuing the optimization compared to the potential improvement in the objective value. The bound is particularly cheap to compute since each of the terms in (3.28) is required during each iteration for some other reason.

The RIP algorithm simultaneously exploits the linear programming paradigm fully and addresses the computational complexity of the FTP. The elegance of the RIP algorithm is in its reduction of a single overwhelming problem into many smaller individually tractable problems. Moreover, this reduction is to well-understood problems in a standard form. By reducing the problem in this way, the RIP algorithm immediately benefits from the highly–developed theory of how to solve systems of linear equations. The RIP algorithm is able to use what has become standard technology, and further advances in linear optimization and representation are of immediate benefit to the RIP algorithm. A key advantage of this is that the RIP algorithm depends only indirectly through a well–defined interface on the algorithm used to optimize the RLP and RIP problems.

The above discussion emphasized the structure of the RIP algorithm as an example of the practice in mathematics of solving a hard problem of interest by reducing it to a number of small problems in a standard form. The RIP algorithm can also be viewed as an example of the similar practice in computer science of partitioning a difficult computation into many small modular sub–problems. From a software

engineering perspective the RIP algorithm cuts the enormous problem prescribed by the FTP into a number of cleanly defined modules. A *system logic parser* maps an arbitrary specification of logical and probabilistic relationships among uncertain events to a system of linear inequalities that respects the specified relationships. The *linear problem formatter* translates the representation into an industry standard form. Finally the *system optimizer* optimizes the system using the services of a *linear optimization module*. These modules have a well-defined role in the algorithm and each can be replaced with a more sophisticated or robust solution as necessary.

From the perspective of cognitive science and decision analysis, the FTP offers a paradigm for coherent probabilistic assessment and inference. The RIP algorithm delivers on the promise of the FTP for many practical situations. It does so in a particularly unique manner for cognitive science, by highlighting a connection with the practical-minded operations research community and the vast array of tools and techniques they have developed. In some areas of cognitive science, particularly motor control and computational vision, there is significant awareness among cognitive scientists of relevant operations research work. This thesis is an example of how vital the connection can be.

5.2 The FTP paradigm

5.2.1 World view

What makes probabilistic inference so hard? One reason is that unlike logic, probability is neither *modular* nor *monotonic*. The following is a valid logical inference.

$$\begin{array}{l} x_1 \\ x_1 \rightarrow x_2 \\ \hline x_2 \end{array} \qquad (5.1)$$

In (5.1) the premises are the sentences above the horizontal line and the conclusion is the sentence below. This inference is an example of *modus ponens*. If our knowledge consists of $x_1 \rightarrow x_2$ and we learn that x_1 is true, then we can conclude that x_2 is true. Logical inference is modular in this regard because *nothing* else in our database can affect this deduction. If our knowledge is consistent, then inference (5.1) is valid regardless of any other knowledge we possess. Furthermore logical inference is monotonic because once no additional knowledge will ever require the retraction of the conclusion in (5.1). Probabilistic reasoning lacks these properties, and the inferences are necessarily weaker. A well known example makes this point clear (Pearl, 1990a).

(5.2)

If my lawn is wet then it rained last night.
My lawn is wet.

It rained last night.

The above argument would lead us to increase the likelihood of it having rained after observing the wet lawn. While this looks innocuous, we would quickly unplug an expert system which reasoned as follows.

(5.3)

My sprinkler was on.
If the sprinkler was on, then the grass is wet
If the grass is wet then it rained last night.

It rained last night.

In logical reasoning, the truth-value of a formula is a characterization of the truth-values of the sub-formulae. The sub–formulae are *sufficient* for the formula as a whole. Modularity and monotonicity are the result. The above example clearly shows how this is not the case in probabilistic inference. The uncertainty of a formula is not a composition of the likelihood of the sub-formulas, but is a specification of the possible worlds in which the statement is true (Pearl 1990a). This problem is guaranteed to snare a rule-based system attempting to treat probabilistic reasoning as a generalization of logical inference because the rules in a rule–based system are evaluated locally without regard to the global context.

Unlike rule–based expert systems the FTP paradigm treats the logical structure of the system as constraints on the realizable outcomes in the sample space. The probability assessments act as constraints on the class of feasible distributions for realizable events in the sample space. The logical constraints and assessments have global scope and the FTP computes all inferences that fall within this global scope. Probability assessments such as $P(\mathsf{x} \vee \mathsf{y}) = p_{\mathsf{x}\vee\mathsf{y}}$ corresponds to a constraint on the probability of the joint events in Ω where $\mathsf{x} \vee \mathsf{y}$ is true. This "world view" contrasts with that of Bayesian rule–based expert systems and causal networks in which new knowledge is used to compute an updated posterior distribution from a prior distribution. The FTP like INFERNO does not temporally parse information into prior and posterior distributions. The FTP views new assessments merely as additional constraints on the class of probability distributions consistent with the given logical structure and probabilistic assessments. Additional assessments will never weaken the bounds on an event unless some other assessment is explicitly removed from the database. Assessments which correspond to weaker constraints are redundant and have no impact on the bounds of the target event.

Section 1.6 in chapter 1 listed several measures by which to judge the utility of the FTP as an assessment paradigm. How well does the FTP stand up to these criteria (1.40)?

Clearly the FTP in principle satisfies (1.40)(a,b). The RIP algorithm demonstrates that the computational demands of the FTP are not overwhelming for at least moderately large ($N \leq 60$) practical inference problems.

Efficiency Regarding (1.40)(c), again the FTP provides this capability in principle. As a global approach to probabilistic inference, the efficiency of the FTP is arguable depending on one's perspective on what is meant by *efficient*. The utility of the FTP in the context of the RIP algorithm depends on (1) the cost of the RLP and RIP optimizations, and (2) the fact that the FTP is based on a global optimization of the system.

The RIP algorithm is based on two exponential–time algorithms, the simplex method for linear programming and the branch–and–bound method for mixed–integer programming. In this light the RIP algorithm and the FTP could be judged quite inefficient indeed. There exist polynomial–time algorithms for linear programming (e.g., Karmarkar's method (cf. Strang, 1988)) and on average the simplex method is competitive with them. Mixed–integer programming is an *NP*–complete problem, so we must evaluate the efficiency of the FTP with this in mind. Like the simplex algorithm, the branch–and–bound algorithm on average performs well on a variety of practical problems, albeit on relatively small problems compared to the LP problems that can be solved by the simplex method (less than 100 vs greater than 10^4 decisions variables). The efficiency of the branch–and–bound algorithm with respect to the scalability of the RIP algorithm is further discussed in Section 5.2.3.

A consequence of the fact that the RIP algorithm is a global optimization is that the entire system must be optimized from scratch each time new assessments are made or new information is acquired. In some cases an advanced basis can be used to jump start the optimization (e.g., when finding the upper bound after the lower bound has been found), but this still corresponds to optimizing the entire system. In contrast each node in a singly–connected causal network can be updated locally given a set of input messages from its ancestors in the graph. Depending on the network topology, incremental evidence can very economical in such a system. Only the nodes in the path of the propagation of the updating need to be recalculated. On the other hand the FTP requires that the full cost of optimizing the entire system be paid each time new assessments are made.

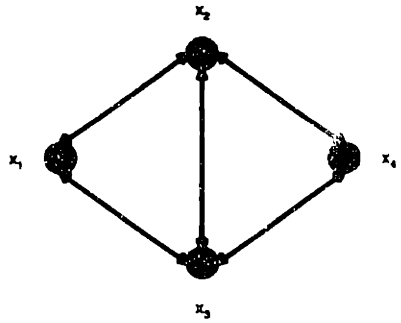

Figure 5.1: A simple communications network.

Once again the tradeoff between model development and model execution costs becomes critical. The cost of model development for an expert system which repeatedly solved the same problem schema varying only the parameters of the particular case would be a worthwhile investment. The cost of global optimization of a system that would only be solved a handful of times would compete well against the cost of building and validating an elaborate model.

5.2.2 Expressiveness of the FTP

A network example Consider a communications network such as the internet or a telephone network[1]. This physical network consists of arcs and nodes. The physical topology corresponds to a logical structure among uncertain events. In the phone network, a call between two phones (i.e., two nodes) can be placed only if the switches in the phone network can be configured so as to provide a circuit linking the two phones. Similarly two computers on the internet can communicate only if the network routers can find an operational path between the two nodes. In both cases there are redundant potential paths between nodes to reduce the likelihood that a single failure could disconnect the network.

Figure 5.1 shows a simple network with four nodes. Let x_i be the event that node i is functioning properly. For this example we will restrict attention to the nodes and assume that the links are perfectly reliable. Thus, the system is characterized by a 4–ary state vector,

[1]This example was suggested by Gordon Kaufman.

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{5.4}$$

and the system can be in any one of 16 states.

(5.5)

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}	e_{15}	e_{16}
x_1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
x_2	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
x_3	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
x_4	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Denote the event "Node x_i can communicate with node x_j" by $x_i \rightleftharpoons x_j$ where

$$x_i \rightleftharpoons x_j = \begin{cases} 1 & x_i = x_j = 1 \\ 0 & \text{otherise} \end{cases} \tag{5.6}$$

If we are interested the event $x_1 \rightleftharpoons x_4$, then $x_1 \rightleftharpoons x_4 = e_1 + e_3 + e_5 + e_7$. If all we know about the network is a vector of assessments $P(X)$, then there is very little we can say about $P(x_1 \rightleftharpoons x_4)$. The FTP would yield the following bounds.

(5.7)

$$\begin{aligned} P(x_1 \rightleftharpoons x_4) &\leq \min(P(x_1), P(x_4)) \\ P(x_1 \rightleftharpoons x_4) &\geq \max(0, P(x_1) + P(x_4) - 1) \end{aligned}$$

These bounds would be quite weak unless the $P(x_1)$ and $P(x_4)$ were very close to 1 since for the information specified so far $x_1 \rightleftharpoons x_4$ is equivalent to the conjunction of x_1 and x_4. A common property of communications networks is that they are often characterized by cliques. The network in Figure 5.1 has two cliques x_1, x_2, x_3 and x_2, x_3, x_4. Further suppose that

(5.8)

$$\begin{aligned} P(x_1 | x_2 x_3 x_4) &= P(x_1 | x_2 x_3) \\ P(x_4 | x_1 x_2 x_3) &= P(x_4 | x_2 x_3) \end{aligned}$$

Then by the chain rule (1.20) and (5.8) $P(x_1 x_2 x_3 x_4)$ is a product of conditional probabilities

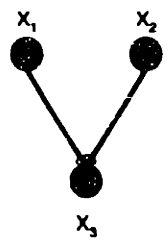

Figure 5.2: A causal network with binary propositional variables.

(5.9)

$$\begin{aligned} P(x_1x_2x_3x_4) &= P(x_1x_2x_3)P(x_4|x_2x_3) \\ &= \frac{P(x_1x_2x_3)P(x_2x_3x_4)}{P(x_2x_3)} \end{aligned}$$

Unfortunately (5.8) is a non-linear constraint on the class of distributions $\mathcal{P}$ and cannot be represented in the FTP paradigm. Conditional independence assessments such as (5.8) are a very powerful device in inductive inference as illustrated in causal networks. Extending the FTP paradigm to encompass assessments such as (5.8) either natively or via an approximation would be a major advance.

Weakness of bounding methods The contrast between paradigms that support inference using a complete joint probability distribution and paradigms that support inference without requiring a complete distribution has been emphasized throughout this work. An ideal application of the FTP paradigm would be as an assessment tool supporting the assessment of probabilities for a causal network for example.

Suppose there are three propositional variables of interest x_1, x_1, and x_3. If we need to assess $P(x_3|x_1x_2)$ then we would like to use empirical data or enable our expert to be able to make some "simple" assessments such as $P(x_1)$, $P(x_2)$, $P(x_3)$, $P(x_3|x_1)$, and $P(x_3|x_2)$ which we could use to constrain the expert's assessment of more complicated assessments such as $P(x_3|x_1x_2)$.

Figure 5.2 shows a causal network where the propositional variables are binary. The expert must provide assessments for $P(x_3|x_1x_2)$. Suppose the assessor makes the following assessments.

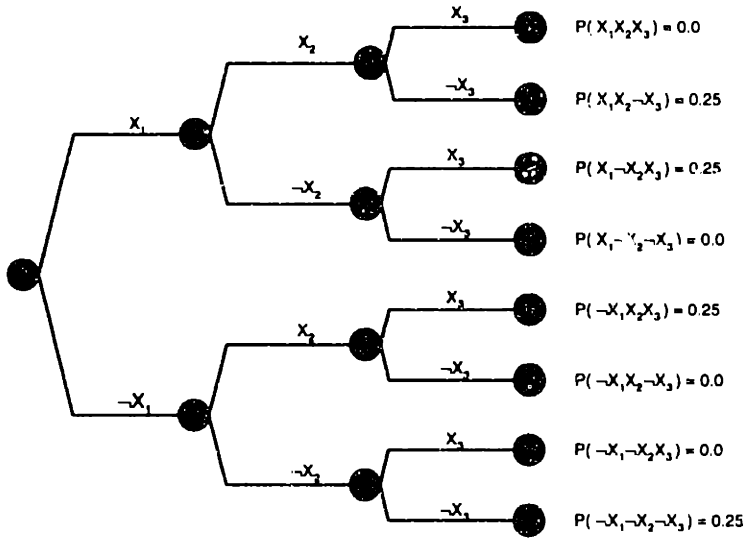

Figure 5.3: A possible joint probability distribution for the causal network.

$$
\begin{aligned}
& P(\mathbf{x}_1) = P(\mathbf{x}_1) - P(\mathbf{x}) = 0.50 \\
& P(\mathbf{x}_i|\mathbf{x}_j) = 0.25 \\
& P(\mathbf{x}_i|\neg\mathbf{x}_j) = 0.25 \\
& P(\mathbf{x}_i \wedge \mathbf{x}_j) = 0.25 \\
& P(\mathbf{x}_i \vee \mathbf{x}_j) = 0.25
\end{aligned}
\tag{5.10}
$$

Unfortunately the promise of the FTP for this task is severely limited by the probability calculus (Neopolitan, 1990). Two distinct joint probability distributions for $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ are shown in Figures 5.3 and 5.4. It is easy to verify on these probability trees that these distributions satisfy (5.10). For the distribution in Figure 5.3, $P(\mathbf{x}_3|\mathbf{x}_1\mathbf{x}_2) = 0.0$, while $P(\mathbf{x}_3|\mathbf{x}_1\mathbf{x}_2) = 1.0$ in the distribution shown in Figure 5.4. The probability calculus imposes no constraints on $P(\mathbf{x}_3|\mathbf{x}_1\mathbf{x}_2)$ given (5.10). Thus the FTP is unlikely to be useful assessment tool to support developing causal networks.

Stochastic independence and conditional assessments The above network example illustrated a particular limitation of the expressiveness of the FTP paradigm. One of the most important and ubiquitous assessments is *stochastic independence*. An assessor asserts that two events $\mathbf{x}$ and $\mathbf{y}$ are stochastically independent if

$$P(\mathbf{xy}) = P(\mathbf{x})P(\mathbf{y}). \tag{5.11}$$

The independence assessment (5.11) is a specific case of a general class of assessments that are non-linear.

$$P(\mathbf{x}|\mathbf{y}) \geq P(\mathbf{y}). \tag{5.12}$$

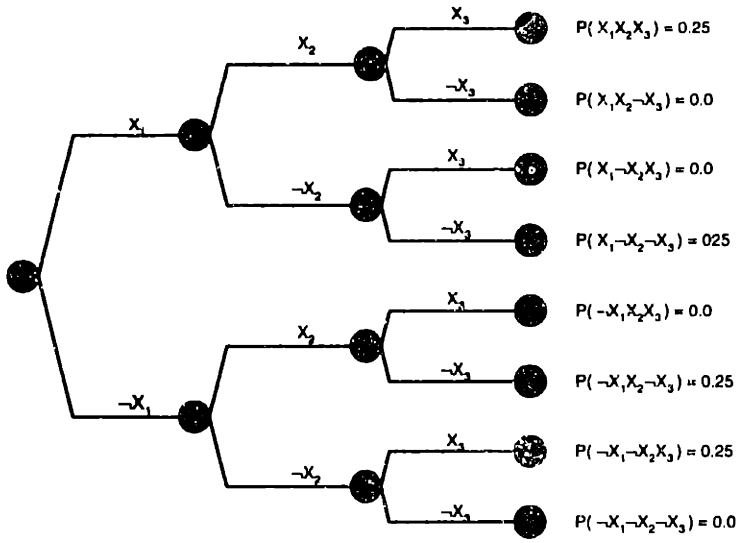

Figure 5.4: A second possible joint probability distribution for the causal network.

Both (5.11) and (5.11) are linear for fixed $P(\mathbf{y})$ which is the case developed in Chapter 2. The above assessments imposes a non–linear constraint on the class $\mathcal{P}_{\mathcal{K}}$ of coherent distributions, and the linear techniques developed here cannot be used. Stochastic independence is a particular example of a general class of non–linear constraints on the class of distributions $\mathcal{P}$ that an assessor may assert.

5.2.3 Scalability

An important characteristic of any algorithm is *scalability* which refers to the behavior of the algorithm with increasing problem size. The FTP is *NP*–hard in principle, but like integer programming we expect on average to be able to solve moderately large systems within the constraints of the computational resources available. The branch–and–bound algorithm comes to mind as the most likely candidate for a limiting factor in the scalability of the RIP algorithm. If the RIP problem optimization dominates the algorithm and given that the branch–and–bound algorithm is the least scalable sub-process in the algorithm, then the RIP algorithm would be very unlikely to scale to systems much larger than are presented here. Test results from the fault tree test suite are very encouraging with respect to the difficulty of the RIP problem optimizations.

To get an estimate of the relative cost to optimize the RIP problems relative to the RLP problems, the RIP algorithm was run against the CSIS system in a "debugging mode" which causes the system solver to write an MPS file for both the RLP and the RIP problem during each iteration. Both the set of RIP files and the set of RLP files were fed to the CPLEX optimizer as a batch. This allows us to get a rough idea

of the proportion of the computations spent optimizing the LP problems versus the MIP problems. The RLP problems have 62 variables and 31 constraints while the RIP problems have 44 variables and 27 constraints for the CSIS system. The optimizer required 54.7 CPU seconds for the RLP optimizations (LP) compared to 129.08 CPU seconds for the set of RIP problems (MIP). Thus, the RIP algorithm spent roughly twice as much time solving the RIP problems as it did solving the RLP problems [2]. This result indicates the computations were not dominated by the branch–and–bound optimizations as might be expected. Similar results were obtained for the other examples, and the 2 to 1 ratio for the CSIS system was the worst case. If the RIP algorithm was spending 90% of the time optimizing the RIP problems or if the ratio of the time spend optimizing the RIP problems versus the RLP problems grew dramatically then we would have immediate cause to doubt the ability of the algorithm to scale to larger problems. Thus, the current suite of examples does not indicate any immediate problems regarding the integer optimization for problems of moderate size. A more systematic study of the integer problems in the RIP algorithm is warranted.

5.3 Epilogue

5.3.1 Technical commentary on the implementation

A sophisticated system parser A limit of the current implementation is that the system parser is non-recursive. The system parser cannot directly parse an event that is an arbitrary Boolean function of other events in the system. The parser requires that the user specify the event as a conjunction or disjunction of other events in the system. For example system parser cannot parse the specification

$$y = (x_1 \vee x_2) \wedge (x_2 \vee x_3) \tag{5.13}$$

into the appropriate RIP problem constraints even though (5.13) is in conjunctive normal form. The current parser requires that additional atomic propositions be introduced into the system. For (5.13),

(5.14)

$$y_1 = x_1 \vee x_2$$

[2] A newer release of the CPLEX optimizer (v 3.0) required roughly 60 CPU seconds to solve each of the RIP and the RLP set of problems.

$$y_2 = x_2 \vee x_3$$
$$y = y_1 \wedge y_2$$

Parsing this class of specification requires a recursive–descent parser which while straightforward to implement is beyond the scope of the current project.

The CPLEX optimizer One of the most important design decisions made was to use the CPLEX optimizer which represents the state of the art in linear optimization software. The CPLEX optimizer uses various techniques to reduce the number of iterations and to accelerate the computation of each iteration. While these techniques dramatically improve the performance of the optimizer for most problems, they can lead to difficulties in the current application.

The CPLEX optimizer does not expect to be employed as a module in a column–generation algorithm. Rather CPLEX expects that the entire problem matrix is available. As discussed in Chapter 4 the optimizer will scale the data so that it will tolerate slightly infeasible solutions. Changing a non–basic column in such a problem and re-optimizing it can lead to a re-scaling of the problem data (including the measure of the problem infeasibility) and the conclusion that the problem is infeasible. This scaling behavior is quite beneficial in typical LP problems where there are many more columns than rows. In this case if the optimizer concludes at some point that the current basis is infeasible, it reverts to phase I while retaining as much of the current basis as possible and finds a new feasible basis. This can be a very successful strategy for reducing the number of iterations in problems where there are many more columns than rows. The number of non–basic columns is very restricted in the RIP algorithm and the optimizer can get into a situation where after re–scaling it cannot construct a feasible basis from the columns currently in the RLP.

In a similar vein the CPLEX optimizer complicates the problem of handling a degenerate basis. The optimizer does not provide an interface to manually pivot columns into the basis. Thus if phase I of the RIP algorithm terminates with an artificial variable in the basis at a zero level, there is no interface to force CPLEX to replace it with a legitimate column.

The benefit of using a sophisticated package such as CPLEX is obvious in the solution performance. The optimizer employs a number of mechanisms to accelerate optimization such as matrix factorization and data scaling. The optimizer implements a number of sophisticated strategies for detecting and handling ill–conditioned floating point arithmetic. Equally important the optimizer includes a number of facilities for

reporting problem statistics, parameters for controlling optimization heuristics, and interfaces to industry standard data formats.

Degenerate problems The LP problem matrix for the FTP problems consists almost entirely of binary coefficients. The tightly constrained range of the coefficients can lead to highly degenerate problems. The current implementation of the RIP algorithm was not designed to handle degenerate problems in a robust manner. As discussed in the above section CPLEX does not provide an interface to manually pivot a column into the basis. Thus repairing a degenerate phase I basis would require the pivot operation to be done outside of the CPLEX mechanisms.

One possible strategy for handling this problem takes advantage of the enormous number of columns in the MLP. Given the number of columns, there will usually be many suitable choices for the column to enter the basis on a given iteration. Constraints can be added to the RIP problem to prevent a degenerate pivot from being chosen. This stategy works only if a suitable column exists.

Let $\mathbf{b}$ be the right hand side vector for the MLP $\mathbf{B}$ be the current basis, and let $\overline{\mathbf{b}} = \mathbf{B}^{-1}\mathbf{b}$. Let $\mathbf{A}$ be the set of non–basic columns of the MLP, and let $\overline{\mathbf{A}} = \mathbf{B}^{-1}\mathbf{A}$. If $\mathbf{a}$ is the column from $\overline{\mathbf{A}}$ to enter the basis, then a degenerate pivot occurs if there is some coefficient b_i of $\overline{\mathbf{b}}$ such that the ith component a_i of $\overline{\mathbf{a}}$ is positive. Let β be the set of indices of zero components in $\overline{\mathbf{b}}$. It can be guaranteed that no degenerate pivot will occur if $a_i \leq 0$ for all $i \in \beta$. This can be accomplished by adding an extra constraint to the RIP problem for each i in β.

$$\left(\mathbf{B}^{-1}\mathbf{z}\right)_i \leq 0 \tag{5.15}$$

where $\mathbf{z}$ is the vector of decision variables for the RIP. The resulting RIP may be infeasible if no column $\mathbf{z}$ of the MLP satisfies these additional constraints. This mechanism has not yet been implemented in the RIP algorithm.

5.4 Future research

Decomposition The RIP algorithm pays a significant penalty for any unnecessary events in the system. Each event added to a system potentially doubles the search space for the RIP problem. A new piece of evidence requires that the entire system be solved from scratch. The fault tree examples are a good illustration of systems that can be decomposed in certain circumstances.

The CSIS system fault tree has 30 logically independent basic failure events. Solving directly for x_{42} involved optimizing an LP involving roughly 10^9 columns. However, knowledge of the bounds on x_{40} and x_{41} is sufficient for determining the bounds on x_{42}. If the bounds on x_{40} and x_{41} are known then solving for x_{42} is an optimization of a 4 column system rather than a *billion* column system. To a large extent the logical hierarchy of the CSIS system fault tree is a deliberate artifact of the modeling process introduced by the analysts to simplify thinking about system reliability.

It is this purity that makes the unadorned fault trees such a useful test suite. While perfect logical hierarchies are uncommon, most real world systems are to at least some extent characterized as a collection of interacting subsystems. Thus, a promising avenue for future work would be to explore decomposition techniques such as Dantzig–Wolf decomposition (Luenberger, 1988).

Representing finite random variables Lad, Dickey, and Rahman (1990) extended the FTP to include finite random variables. The extension is straightforward; however, the size of the realm matrix of such problems grows enormously. A problem with 10 events has a realm matrix with at most $2^{10} = 1024$ columns. A problem with 9 events and one random variable with 10 possible values could have a realm matrix with as many as $2^9 \times 10 = 5120$ columns. A problem with 8 events and 2 random variables with 10 values each could have as many as $2^8 \times 10 \times 10 = 256,000$ columns. Currently the RIP algorithm is is limited to binary events. A robust and efficient RIP algorithm for finite random variables would greatly increase the scope of problems that could be solved. The size of the RLP problem is not affected by this generalization. The RIP problem bears the full weight of this generalization. One important consequence of this generalization would be that exchangeable events could be represented elegantly and efficiently (Lad, 1993).

Appendix A

Fault trees

Figure A.1 shows the symbols used to represent the events in fault trees. The leaves of the tree are elementary events. They are represented by circles or diamond symbols. (The distinction between circles and diamonds is not relevant to our discussion.) *Compound events* are represented by rectangles. They are *logical functions* of other elementary or compound events in the tree. The logical functions are represented by *logic gates*. Logic gates represent Boolean functions of their input events which assign a truth value to their output event. Logic gates come in two flavors AND and OR . Figure A.2 shows two simple fault trees. The tree on the left shows an OR gate. The root event occurs if any one of the elementary failures X, Y, or Z occurs. The OR gate characterizes the logic of systems where every component must function in order for the system to function. The tree on the right shows an AND gate. The root event occurs only if every one of the elementary failures X, Y, and Z occurs. The AND gate characterizes the logic of redundant systems where every component must fail in order for the system to fail. It is worth noting that the AND and OR gates with logical negation are sufficient to characterize any Boolean function/relation on

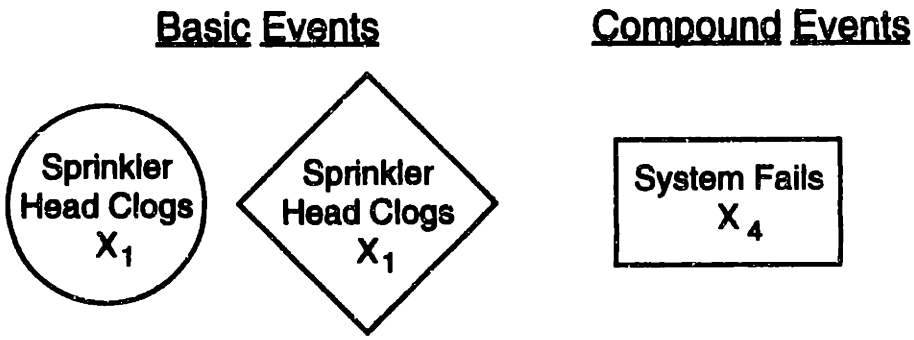

Figure A.1: The symbols used to represent events in fault trees.

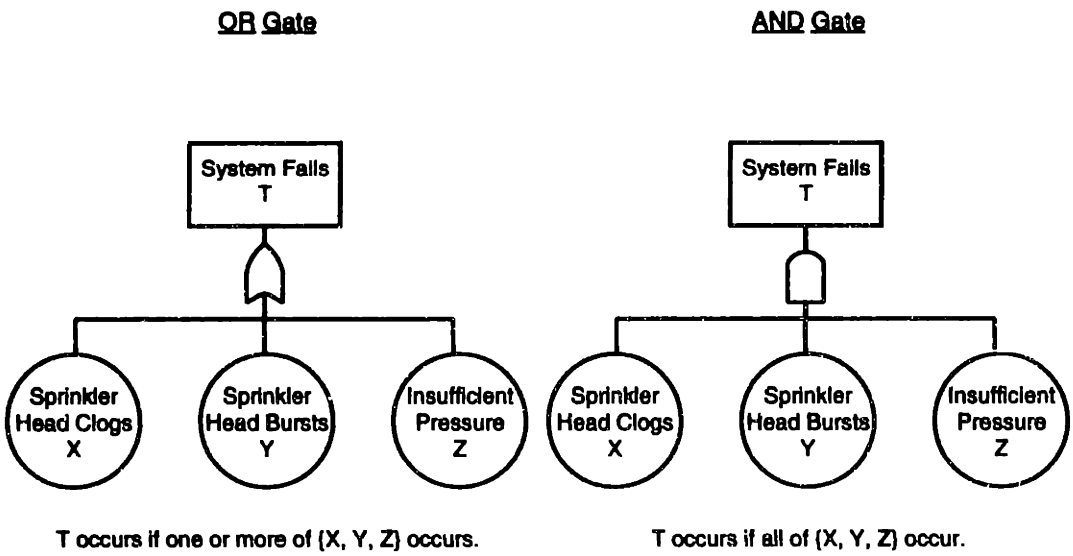

Figure A.2: An illustration of the logic gates used in fault trees.

the input events.

References

Andreasson, S. M., Woldbye, B., Falck, B., & Andersen, K. (1987). MUNIN – A causal probabilistic network for interpretation of electromyographic findings. *Proceedings of the 10th International Joint Conference on Artificial Intelligence.* Milan, Italy.

Baron, J. (1988). *Thinking and Deciding.* Cambridge, U.K.: Cambridge University Press.

Blair, C. E., Jeroslow, R. G., & Lowe, J. K. (1985). Some results and experiments in programming techniques for propositional logic. *Computers and Operations Research*, **13**, 633-645.

Boole, G. (1854). *An Investigation of the Laws of Thought on which Are Founded the Theories of Logic and Probabilities.* Cambridge, UK: MacMillan.

Boole, G. (1847). *The Mathematical Analysis of Logic.* Cambridge, UK: MacMillan.

Bradley, S. P., Hax, A. C., & Magnanti, T. L. (1977). *Applied Mathematical Programming.* Reading, MA: Addison-Wesley.

Buchanan, B. G. & Shortlife, E. H., (1984). *Rule-based Expert Systems.* Reading, MA: Addison-Wesley.

Cheesman, P. (1983). A method of computing generalized bayesian probability values for expert systems. In *Proceedings Eighth International Joint Conference on Artificial Intelligence,* Karlsruhe, Germany. Los Angeles: William Kaufman.

Cooper, G. F. (1990). The computational complexity of probabilistic inference using belief networks. *Artificial Intelligence,* **42**, 393-405.

CPLEX Optimization Inc. (1994). *Using the CPLEX Callable Library.* Incline Village, NV: CPLEX Optimization Inc.

Dickey J. (1991). Probability assessment and inference by interactive computational use of the fundamental theorem of probability. Paper presented at SBIES conference. Boston, MA.

Duda, R., Hart, P., & Nilsson, N. J. (1976). *Subjective bayesian methods for rule based systems.* Technical report TR-124, Artificial Intelligence Center, SRI International, Menlo Park, CA.

Duda, R., Hart, P., Nilsson, N. J., & Sutherland, G. L. (1978). Semantic network representation in rule based inference systems. In D. A. Waterman & F. Hayes-Roth (Eds.) *Pattern-directed inference systems.* New York: Academic Press.

Duda, R. & Hart, P. (1973). *Pattern Classification and Scene Analysis.* New York: Wiley.

Ebbinghaus, H.D., Flum, J., & Thomas W. (1984). *Mathematical Logic.* New York: Springer-Verlag.

Fagin, R. & Halpern, J. Y. (1989). Uncertainty, belief, and probability. *Proceedings of the IJCAII Eleventh International Conference on Artificial Intelligence.* Menlo Park, CA: AAAI Press.

Finetti, B. de. (1964). Foresight: Its logical laws, its subjective sources. In *Studies in Subjective Probability.* H. Kyburg & H. Smokler (Eds.), New York: Wiley.

Finetti, B. de. (1974b). *Theory of Probability.* (vol. 1) A. Machi & A. Smith (trans.) New York: Wiley Interscience.

Finetti, B. de. (1974b). *Theory of Probability.* (vol. 2) A. Machi & A. Smith (trans.) New York: Wiley Interscience.

Fishoff, B., Slovic, P., & Lichtenstein, S. (1978). Fault trees: Sensitivity of estimated failure probabilities to problem representation. *Journal of Experimental Psychology: Human Perception and Performance, 4,* 330-344.

Frankel, E. G. (1988). *Systems Reliability and Risk Analysis.* Boston: Kluwer.

Frege, G. (1980). *The Foundations of Arithmetic.* (J. L. Austin trans.) Evanston, ILL: Northwestern University Press.

Frisch, A. M. & Haddawy (1994). Anytime deduction for probabilistic logic. *Artificial Intelligence,* **69,** 93-122.

Gärdenfors, P. & Sahlin, N. (1988). *Decision, Probability, and Utility.* Cambridge, U.K.: Cambridge University Press.

Georgakopoulos, G., Kavvadias, D., & Papadimitriou, C. H. (1988). Probabilistic satisfiability. *Journal of complexity,* **4,** 1-11.

Goldman, S. & Rivest, R. (1988). A non-iterative maximumn entropy algorithm. In P. Bonissone et al (eds.) *Uncertainty in Artificial Intelligence.* Amsterdam: Elsevier.

Good, I. J. (1950). *Probability and the Weighting of Evidence.* London: Griffen.

Hailperin, T (1965). Best possible inequalities for the probability of a logical function of events. *American Math Monthly,* **72,** 343-359

Hailperin, T (1976). *Boole's Logic and Probability: A Critical Exposition from the Standpoint of Contemporary Algebra and Probability Theory.* Amsterdam: North-Holland.

Heckerman, D. (1986). Probalistic interpretations for MYCIN's certainty factors. In L. Kanal & J. Lemmer (Eds.), *Uncertainty in Artificial Intelligence.* New York: North-Holland.

Heckerman, D. & Horvitz, E. J. (1986). The myth of modularity in rule-based systems. In L. Kanal & J. Lemmer (Eds.) *Uncertainty in Artificial Intelligence 2.* New York: North-Holland.

Heckerman, D. & Horvitz, E. J. (1987). On the expressiveness of rule-based systems for reasoning with uncertainty. *Proceedings of the AAAI Sixth National Conference on Artificial Intelligence.* Menlo Park, CA: AAAI Press.

Heckerman, D. (1991). *Probalistic Similarity Networks.* Cambridge, MA: MIT Press.

Heising, C., Rasmussen, N., & Mak, C. (1982). Common Cause Analysis: A Review and Extension of Existing Methods. MIT Energy Laboratory Report No. MIT-EL 82-038.

Henley, H. J. & Kumamoto, H. (1981). *Reliability Engineering and Risk Assessment.* Englewood Cliffs, NJ:Prentice-Hall.

Henrion, M. (1989). Some practical issues in constructing belief networks. In L. N. Kanal, T. S. Levitt, & J. F. Lemmer (Eds.) *Uncertainty in artificial intelligence 3.* Amsterdam: Elsevier.

Hogg, R. V., & Craig, A. T. (1978). *Introduction to Mathematical Statistics.* New York: Macmillan.

Hopcroft, J. E. & Ullman, J. D. (1979). *Introduction to Automata Theory, Languages, and Computation.* Reading, MA: Addison-Wesley.

Howson, C. & Urbach, P. (1989). *Scientific Reasoning* La Salle, ILL: Open Court.

Jaynes, E. T. (1957). Information theory and statistical mechanics, I and II. *Physical Review,* **106:** 620-30; **107:** 171-190.

Jaynes, E. T. (1968). Prior probabilities. *IEEE Transactions on Systems Science and Cybernetics,* **4**, 227-241.

Jaynes, E. T. (1979). Where do we stand on maximum entropy. In R. Levine & M. Tribus (Eds.), *The Maximum Entropy Formalism.* Cambridge, MA: MIT Press.

Jeroslow, R. G. (1989). *Logic Based Decision Support.* Amsterdam: Elsevier.

Kahneman, D., Slovic, P., & Tversky, A. (1982). *Judgement under Uncertainty: Hueristics and Biases.* Cambridge, U.K.: Cambridge University Press.

Karimi, R., Rasmussen, N., & Wolf, L. (1980). Qualitative and Quantitative Reliability Analysis of Safety Systems. MIT Energy Laboratory Report No. MIT-EL 80-015.

Keeney, R. & Raiffa, H. (1976). *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* New York: Wiley.

Klir, G. & Folger, T. (1988). *Fuzzy Sets, Uncertainty, and Information.* Englewood Cliffs, NJ: Prentice-Hall.

Lad, F. (1993). *Operational Subjective Statistical Methods.* Unpublished manuscript.

Lad, F. & Dickey, J. (1990). The fundamental theorem of prevision. In *Economic Decision-Making: Games, Econometrics, and Optimisation.* J. Gabszewicz, J. Richard, & L. Wolsey (Eds.), New York: North-Holland.

Lad, F., Dickey, J. & Rahman, A. (1990). The fundamental theorem of prevision. *Statistica, 50*, 19-38.

Lad, F., Dickey, J. & Rahman, A. (1991). Numerical application of the fundamental theorem of prevision. *Journal of Statistical Computation and Simulation.* 19-38.

Luce, R. D. & Raiffa, H. (1957). *Games and Decisions: Introduction and Critical Survey.* New York: Wiley.

Luenberger, D.G. (1984). *Linear and Non-Linear Programming.* Reading, MA: Addison-Wesley.

Martz, H. F., & Waller, R. A. (1982). *Bayesian Reliability Analysis.* New York: Wiley.

Mendelson, E. (1987). *Introduction to Mathematical Logic* (3rd ed.). Monterey, CA: Wadsworth & Brooks/Cole.

Miller, R., McNeil, M., Challinor, S., Massarie, F. & Myers, J. (1986). The INTERNIST-1 / Quick Medical Reference Project: Status Report. *Western Journal of Medicine,* **145**, 816-822.

Myers, T. S. & Osherson, D. N. (1992). On the psychology of ampliative inference. *Psychological Science,* 3, 121-135.

Neal, R. M. (1992). Connectionist learning of belief networks. *Artificial Intelligence*, **56**, 71-113.

Neopolitan, R. (1990). *Probabilistic Reasoning in Expert Systems: Theory and Algorithms.* New York: Wiley Interscience.

Nilsson, N. J. (1986). Probabilistic logic. *Artificial Intelligence*, **28**, 71-87.

Nilsson, N. J. (1993). Probabilistic logic revisited. *Artificial Intelligence*, **59**, 39-42.

Osherson, D. N., Shafir, E., & Smith, E. E. (1993). Ampliative inference: On choosing a probability distribution. *Cognition,* **49,**, 189-210.

Osherson, D. N., Smith, E. E., Myers, T. S., Shafir, E., & Stob, M. (1993). Extrapolating human probability judgement. *Theory and Decision.*

Pages, A. & Gondran, M. (1986). *System Reliability: Evaluation and Prediction in Engineering.* London: North Oxford.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems.* San Mateo: Morgan-Kaufman.

Pearl, J. (1990a). Numerical uncertainty in expert systems. In G. Shafer & J. Pearl (Eds.) *Readings in Uncertain Reasoning.* San Mateo: Morgan-Kaufman, 339-344.

Pearl, J. (1990b). The bayesian approach. In G. Shafer & J. Pearl (Eds.) *Readings in Uncertain Reasoning.* San Mateo: Morgan-Kaufman, 339-344.

Pednault, E. P. D., Zucker, S. W., & Muresan, L. V. (1981). On the independence assumption underlying subjective bayesian updating. *Artificial Intelligence,* **16,** 213-222.

Quinlan, J. R. (1983). Inferno: A cautious approach to uncertain inference. *The Computer Journal,* **26**, 255-269.

Ramsey, F. P. (1964). Truth and probability. In *Studies in Subjective Probability.* H. E. Kyburg & H. E. Smokler (Eds.), New York: Wiley.

Resnik, M. D. (1987). *Choices: An Introduction to Decision Theory.* Minneapolis, MN: University of Minnesota Press.

Rosenkrantz, R. D. (1977). *Inference, Method and Decision: Towards a Bayesian Philosophy of Science.* Boston: Reidel.

Ross, S. (1985). *Introduction to Probability Models.* Boston: Academic Press.

Ross, S. (1988). *A First Course in Probability.* New York: Macmillan.

Savage, L. J. (1972). *The Foundations of Statistics.* New York: Dover.

Sedgewick, R. (1983). *Algorithms.* Reading, MA: Addison-Wesley.

Shafer, G. (1979). *Mathematical Theory of Evidence.* San Mateo, CA: Morgan-Kaufman.

Shafer, G. & Pearl, J. (1990). (ed.) *Readings in Uncertain Reasoning.* San Mateo, CA: Morgan-Kaufman.

Seidenfeld, T. (1986). Entropy and Uncertainty. *Philosophy of Science,* **53,** 467-491.

Shortlife, E. H., & Buchanan, B. G. (1975). A model of inexact reasoning in medicine. *Mathematical Biosciences.* **23,** 351-379.

Skyrms, B. (1986). *Choice & Chance: An Introduction to Inductive Logic.* Belmont, CA: Wadsworth.

Smith, C. A. B. (1961). Consistency in statistical inference and decision. *Journal of the Royal Statistical Society*, series B, **23**, 1-25.

Strang, G. (1988). *Linear Algebra and its Applications.* 3rd ed. San Diego, CA:Harcourt, Brace, Jovanovich.

Szolovits, P. & Pauker, S. G. (1978). Categorical and probabilistic reasoning in medical diagnosis. *Artificial Intelligence,* **11,** 115-144.

Tsao, H. S. J., & Fang, S. C., & Lee, D. N. (1992). On the optimal entropy analysis. *European journal of operational research,* **59,** 324-329.

Tversky, A. & Kahneman, D. (1974). *Judgement under uncertainty: Hueristics and biases. Science,* **185,** 1124-1131.

von Winterfeldt, D. & Edwards, W. (1986). *Decision Analysis and Behavioral Research.* Cambridge, U.K.: Cambridge University Press.

United States Nuclear Regulatory Commission. (1975). Reactor safety study – An assessment of accident risks in U. S. Comercial Nuclear Power Plants.(NUREG-75/014).

United States Nuclear Regulatory Commission. (1981). *Fault Tree Handbook.* (NUREG-0492).

Wolfram Research, Inc. (1991). *Mathematica: A System for Doing Mathematics by Computer.* Urbana, ILL: Wolfram Research, Inc.

Wall, L, & Schwartz, R. (1990). *Programming Perl.* Sebastopol, CA: O'Reilly & Associates.

Whittle, P. (1982). Non-linear programming. *Handbook of Applicable Mathematics, Volume IV: Analysis*, W. Ledermann & S Vajda (Eds.). New York: Wiley.

Williams, P. M. (1980). Bayesian conditionalisation and the principle of minimum information. *British Journal of the Philosophy of Science,* **31,** 131-144.

Winston, P. H. (1992). *Artificial Intelligence.* Reading, MA: Addison-Wesley.