Statistics for Engineering and Information Science

Finn V. Jensen

Bayesian Networks and Decision Graphs

Springer

Statistics for Engineering
and Information Science

Series Editors
M. Jordan, S.L. Lauritzen, J.F. Lawless, V. Nair

Springer Science+Business Media, LLC

Statistics for Engineering and Information Science

Akaike and Kitagawa: The Practice of Time Series Analysis.
Cowell, Dawid, Lauritzen, and Spiegelhalter: Probabilistic Networks and Expert Systems.
Doucet, de Freitas, and Gordon: Sequential Monte Carlo Methods in Practice.
Fine: Feedforward Neural Network Methodology.
Hawkins and Olwell: Cumulative Sum Charts and Charting for Quality Improvement.
Jensen: Bayesian Networks and Decision Graphs.
Marchette: Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint.
Vapnik: The Nature of Statistical Learning Theory, Second Edition.

Finn V. Jensen

Bayesian Networks and Decision Graphs

With 184 Illustrations

Springer

Finn V. Jensen
Aalborg University
Department of Computer Sciences
Fredrik Bajers Vej 7E
DK-9220 Aalborg Ø
Denmark
fvj@cs.auc.dk

Library of Congress Cataloging-in-Publication Data
Jensen, Finn V.
Bayesian networks and decision graphs / Finn V. Jensen.
p. cm. — (Statistics for engineering and information science)
Includes bibliographical references and index.

ISBN 978-1-4757-3504-8 ISBN 978-1-4757-3502-4 (eBook)
DOI 10.1007/978-1-4757-3502-4

1. Bayesian statistical decision theory—Data processing. 2. Machine learning. 3. Neural networks (Computer science). 4. Decision making. I. Title. II. Series.
QA279.5 .J47 2001
519.5'42—dc21 2001020441

Printed on acid-free paper.

9 8 7 6 5 4

www.springer-ny.com

Preface

Ever since the first machines were constructed, artists and scientists have shared a vision of a human-like machine: an autonomous self-moving machine that acts and reasons like a human being. Much effort has been put into this dream, but we are still very far from having androids with even the tiniest similarity to humans.

This does not mean that all of these efforts have been wasted. As a spinoff, we have seen a long series of inventions that can take over very specialized sections of human work. These inventions fall into two categories: machines that can make physical changes in the world and thereby substitute human labor, and machines that can perform activities usually thought of as requiring intellectual skills.

In contemporary science and engineering, we still have this split into two categories. The activity of the first category is mainly concentrated on the construction of robots. The aim is to construct autonomous machines performing "sophisticated" actions such as searching for a cup, finding a way from the office to a lavatory, driving a vehicle in a deserted landscape or walking on two legs. Construction of such robots requires computers to perform certain kinds of artificial intelligence. Basically, it is the kind of intelligence that humans share with most mammals. It involves skills such as visual recognition of items, sound recognition, learning to abstract crucial items from a scene or control of balance and position in 3-D space. Although they are very challenging research tasks, and they certainly require enormous computing power and very sophisticated algorithms, you would not say that these skills are intellectual, and the basis for the activity is the physical appearance of a device that moves. To say it another way: the success criterion is how the algorithms work when controlling a physical machine in real time.

The activity in the second category is basically concerned with reasoning and human activities that we presumably do not share with other animals. The activity is separated from matter. When performed, no changes in the physical world need to take place. The first real success was the automated calculator: a machine that can perform very large and complicated arithmetic calculations. Automated calculation skill is nowadays hardly considered artificial intelligence, and we are now acquainted with computers performing tasks that decades ago were considered highly intellectual (e.g. taking derivatives of functions or reduction of mathematical expressions). When an activity has been understood so well that it can be formalized, it will soon be performed by computers, and gradually we acknowledge that this activity does not really require intelligence.

A branch of research in the second category has to do with reasoning. The first successes were in logical reasoning. Propositional logic is fully formalized, and although some tasks are NP-complete and therefore in some situations intractable for a computer, we have for propositional logic completed the transition from "intellectual task" to "we have computers to do this for us."

Unfortunately, logical reasoning is very limited in scope. It deals with how to infer from propositions that you know are true. Very often you do not know a proposition for certain, but you still need to perform inference from your incomplete and uncertain knowledge. Actually, this is the most common situation for human reasoning. Reasoning under uncertainty is not yet so well understood that it can be formalized entirely for computers. There are several approaches to reasoning under uncertainty. The approach taken in this book is (subjective) probability theory. When the reasoning ends up in a conclusion on a decision, we use *utilities*, and we assume that the decision taken is the one that maximizes the expected utility. In other words, the approach prescribes a certain behavior. We may not always expect this behavior from human beings, and therefore the approach is also termed *normative*. There are alternative approaches to reasoning under uncertainty. Most prominent is *possibility theory*, which in certain contexts is called *fuzzy logic*. The interested reader may consult the wide literature on these approaches.

The aim of normative systems can in short be termed human *wisdom*: **to take decisions on the basis of accumulated and processed experience**. The tasks are of the following types:

- using observations to interpret a situation;
- focusing a search for more information;
- deciding for intervening actions;
- adapting to changing environments;
- learning from experience.

A damping factor for properly exploiting the advances in artificial intelligence has for a long time been the lack of successes in robotics. An autonomous agent that moves, observes, and changes the world must carry a hardly controllable body.

Therefore, the advances have mainly been exploited in *decision support systems*, computer systems which provide advice for humans on highly specialized tasks. With the Internet, the scope of artificial intelligence has widened considerably. The Internet is an ideal nonphysical world for intelligent agents, which are pure spirits without bodies. In the years to come, we will experience a flood of intelligent agents on the Internet, and companies as well as private persons will be able to launch their own agents to explore and collect information on the Internet. Also, we will experience the dark sides of human endeavor. Some agents will destroy, intrude, tell lies and so on, and we will have to defend ourselves against them. Agents will meet agents, and they will have to decide how to treat each other, they will have to learn from past experience, and they will have to adapt to changing environments.

During the 1990s, Bayesian networks and decision graphs attracted a great deal of attention as a framework for building normative systems, not only in research institutions but also in industry. Contrary to most other frameworks for handling uncertainty, a good deal of theoretical insight as well as practical experience is required in order to exploit the opportunities provided by Bayesian networks and decision graphs.

On the other hand, many scientists and engineers wish to exploit the possibilities of normative systems without being experts in the field. This book should meet that demand. It is intended for both classroom use and self-study, and it addresses persons who are interested in exploiting the approach for the construction of decision support systems or bodyless agents.

The theoretical exposition in the book is self-contained, and the mathematical prerequisite is some prior exposure to calculus and elementary graph theory (except for Section 3.4, which requires familiarity with gradients of functions over several variables). The book falls into two parts. In the first part, the emphasis is on gaining practical experience with the use of Baysian networks and decision graphs, and we have assumed that the reader has access to a computer system for handling Bayesian networks and influence diagrams (the exercises marked with an ^{E} require such a system). There are many systems, academic as well as commercial. The following systems can be downloaded for academic use free of charge: Bayesware (www.bayesware.com), BN Toolbox (www.cs.berkeley.edu /~murphyk/Bayes/bnsoft.html), BucketElim (www.ics.uci.edu/~irinar), Genie (www2.sis.pitt.edu/~genie), Hugin (www.hugin.com), Java Bayes (www.cs.cmu.edu/~javabayes/Home), Netica (www.norsys.com), and XBAIES (www.staff.city.ac.uk/~rgc/webpages/xbpage.html). The presentation in this part is based very much on examples, and for overview purposes there is a summary section at the end of each chapter.

The second part is devoted to presenting basic algorithms for normative systems, and the algorithms are exploited to introduce new types of features for decision support systems and bodyless agents. Alhough the exposition is self-contained, it is more mathematically demanding, and it requires that the reader be familiar with working with texts in the mathematical language.

The book is an introduction to Bayesian networks and decision graphs. Many results are not mentioned or just treated superficially. The following textbooks and monographs can be used for further study:

- Judea Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann Publishers, 1988.
- Russell Almond, *Graphical Belief Modelling*, Chapman & Hall, 1995.
- Steffen L. Lauritzen, *Graphical Models*, Oxford University Press, 1996.
- Enrique Castillo, José M. Gutiérrez, and Ali S. Hadi, *Expert Systems and Probabilistic Network Models*, Springer-Verlag, 1997.
- Robert G. Cowell, A. Philip Dawid, and Steffen L. Lauritzen, *Probabilistic Networks and Expert Systems*, Springer-Verlag, 1999.

The annual *Conference on Uncertainty in Artificial Intelligence* (www.auai.org) is the main forum for researchers working with Bayesian networks and decision graphs, so the best general references for further reading are the proceedings from these conferences.

Another relevant conference is the biannual *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty* (ECSQARU). The conference deals with various approaches to uncertainty calculus, and the proceedings are published in the Springer-Verlag series *Lecture Notes in Artificial Intelligence.*

I wish to express my gratitude to several people for ideas during the preparation of the book. First I thank the Ph.D. students at the Research Unit for Decision Support Systems, Olav Bangsø, Søren L. Dittmer, Anders L. Madsen, Thomas D. Nielsen, and Dennis Nilsson, and my colleagues at Aalborg University, Poul S. Eriksen, Uffe Kjærulff, Steffen L. Lauritzen, and Kristian G. Olesen. I also thank the many academic colleagues in the U.S. and Europe with whom I have had the pleasure of exchanging ideas, in particular Linda van der Gaag, Helge Langseth, Claus Skaanning, Marco Valtorta, Jiří Vomlel, Marta Vomlelová, and Yang Xiang. I also thank two years of undergraduate students who had to cope with unfinished drafts of notes for parts of the course on decision support systems. Finally, I am much indebted to Camilla Jørgensen for her very competent LaTeX-coding and for several corrections to my English.

This book is supported by a web site, www.cs.auc.dk/~fvj/bnid.html, which will support readers with solutions and models for selected exercises,

a list of errata, special exercises, and other links relevant to the issues in the book.

Aalborg, January 2001 Finn V. Jensen

Contents

Part I

A Practical Guide to Normative Systems

1
Causal and Bayesian Networks

1.1 Reasoning under Uncertainty

1.1.1 Car start problem

The following is an example of reasoning, which humans do daily.

"In the morning, my car will not start. I can hear the starter turn, but nothing happens. There may be several reasons for my problem. I can hear the starter roll, so there must be power in the battery. Therefore, the most probable causes are that the fuel has been stolen overnight or that the spark plugs are dirty. It may also be due to dirt in the carburetor, a leak in the ignition system, or something more serious. To find out, I first look at the fuel meter. It shows $\frac{1}{2}$ full, so I decide to clean the spark plugs."

To have a computer do the same kind of reasoning, we need answers to questions such as: "What made me conclude that among the probable causes "stolen fuel" and "dirty spark plugs" are the two most probable?" or"What made me decide to look at the fuel meter, and how can an observation concerning fuel make me conclude on the unrelated spark plugs?" To be more precise, we need ways of representing the problem and ways of performing inference in this representation such that a computer can simulate this kind of reasoning and perhaps do it better and faster than humans.

For propositional logic, Boolean logic is the representation framework, and various derived structures, such as truth tables and binary decision diagrams, are invented together with efficient algorithms for inference.

In logical reasoning, we use four kinds of logical connectives: conjunction, disjunction, implication, and negation. In other words, simple logical statements are of the kind "if it rains, then the lawn is wet," "both John and Mary have caught the flu," "either they stay at home or they go to the cinema," or "the lawn is not wet." From a set of logical statements, we can deduce new statements. From the two statements "if it rains, then the lawn is wet" and "the lawn is not wet," we can infer that it does not rain.

When we are dealing with uncertain events, it would be nice if we could use similar connectives with certainties rather than truth values attached, so we may extend the truth values of propositional logic to "certainties," which are numbers between 0 and 1. A certainty 0 means "certainly not true," and the higher the number the higher the certainty. Certainty 1 means "certainly true."

We could then work with statements such as "if I take a cup of coffee while on break, I will with certainty 0.5 stay awake during the next lecture"or "if I take a short walk during break, I will with certainty 0.8 stay awake during the next lecture." Now, suppose I take a walk as well as have a cup of coffee. How certain can I be to stay awake? To answer this, I need a rule for how to *combine* certainties. In other words, I need a function that takes the two certainties 0.5 and 0.8 and returns a number, which should be the certainty resulting from combining the certainty from the two statements.

The same is needed for *chaining*: "if a then b with certainty x," and "if b then c with certainty y." I know a, so what is the certainty of c?

It has turned out that any function for combination and chaining will in some situations lead to wrong conclusions.

Another problem, which is also a problem for logical reasoning, is abduction: I have the rule "a woman has long hair with certainty 0.7." I see a long haired-person. What can I infer about the person's sex?

1.1.2 Causal networks

A way of structuring a situation for reasoning under uncertainty is to construct a graph representing causal relations between events.

A reduced car start problem

To simplify the situation, assume that we have the events $\{yes, no\}$ for *Fuel?*, $\{yes, no\}$ for *Clean Spark Plugs?*, $\{full, \frac{1}{2}, empty\}$ for *Fuel Meter Standing*, and $\{yes, no\}$ for *Start?*. In other words, the events are clustered around variables, each with a set of outcomes, also called *states*. We know that the state of *Fuel?* and the state of *Clean Spark Plugs?* have a causal impact on the state of *Start Problem?*. Also, the state of *Fuel?* has an impact on the state of *Fuel Meter Standing*. This is represented by the graph in Figure 1.1.

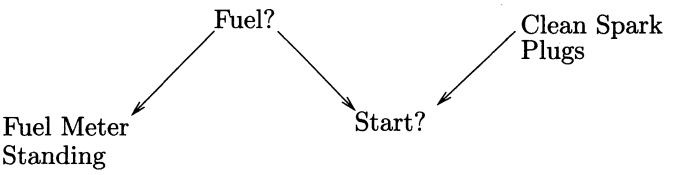

FIGURE 1.1. A causal network for the reduced car start problem.

If we add a direction from *yes* to *no* inside each variable (and from *full* to *empty*), we can also represent directions of the impact. For the present situation, we can say that all the impacts are positive (with the direction); that is, the more the certainty of the cause is moved in a positive direction, the more the certainty of the caused variable will also be moved in a positive direction. To indicate this, we can label the links with the sign "+" as is done in Figure 1.2.

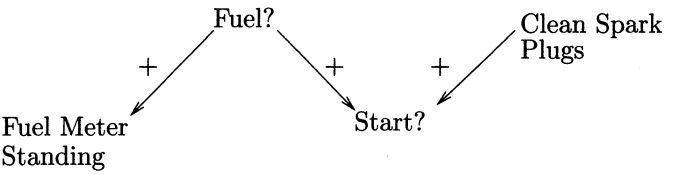

FIGURE 1.2. A causal network for the reduced car start problem with a sign indicating direction of impact.

We can use the graph in Figure 1.2 to perform some reasoning. Obviously, if I know that the spark plugs are not clean, then the certainty for no start will increase. However, my situation is the opposite. I realize that I have a start problem. As my certainty on *Start?* is moved in a positive direction, I find the possible causes for such a move more certain (*Clean Spark Plugs?* and *Fuel?*); that is, the sign "+" is valid for both directions. Now, because the certainty on *no* for *Fuel?* has increased, I will have a higher expectation that *Fuel Meter Standing* be in state *empty*.

The movement of the certainty for *Fuel Meter Standing* tells me that by reading the fuel meter I will get information related to the start problem. I read the fuel meter, it says $\frac{1}{2}$, and reasoning backward yields that the certainty on *Fuel?* is moved in a negative direction.

So far, the reasoning has been governed by simple rules that can easily be formalized. The conclusion is harder: "Lack of fuel does not seem to be the reason for my start problem, so most probably the spark plugs are not clean". Is there a formalized rule that makes this kind of reasoning on a causal network computerized? We will return to this problem in Section 1.2.

Note The reasoning has focused on changes of certainty. If, in certainty calculus, the actual certainty of a specific event must be calculated, then

knowledge of certainties prior to any information is also needed. In particular, prior certainties are required for the events that are not effects of causes in the network. If, for example, my car cannot start, the actual certainty that the fuel has been stolen depends on my neighborhood.

1.2 Causal Networks and d-Separation

A causal network consists of a set of *variables* and a set of *directed links* between variables. Mathematically, the structure is called a *directed graph.* When talking about the relations in a directed graph, we use the wording of family relations: if there is a link from A to B, we say that B is a *child* of A and A is a *parent* of B.

The variables represent events (propositions). A variable can have any number of states. A variable may, for example, be the color of a car (states *blue, green, red, brown*), the number of children in a family (states 0, 1, 2, 3 4, 5, 6, > 6), or a disease (states *bronchitis, tuberculosis, lung cancer*). Variables may have a countable or a continuous state set, but in this book we solely consider variables with a finite number of states.

In a causal network, a variable represents a set of possible states of affairs. A variable is in exactly one of its states; which one may be unknown to us.

As illustrated in Section 1.1.2, causal networks can be used to follow how a change of certainty in one variable may change the certainty for other variables. We present in this section a set of rules for that kind of reasoning. The rules are independent of the particular calculus for uncertainty.

Serial connections

Consider the situation in Figure 1.3. A has an influence on B, which in turn has an influence on C. Obviously, evidence on A will influence the certainty of B, which then influences the certainty of C. Similarly, evidence on C will influence the certainty on A through B. On the other hand, if the state of B is known, then the channel is blocked, and A and C become independent. We say that A and C are *d-separated given* B, and when the state of a variable is known, we say that it is *instantiated.*

We conclude that evidence may be transmitted through a serial connection unless the state of the variable in the connection is known.

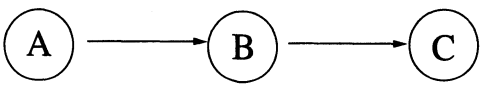

FIGURE 1.3. Serial connection. When B is instantiated, it blocks communication between A and C.

Diverging connections

The situation in Figure 1.4 is called a *diverging* connection. Influence can pass between all the children of A unless the state of A is known. We say that $B, C, \ldots, E$ are d-separated given A.

Evidence may be transmitted through a diverging connection unless it is instantiated.

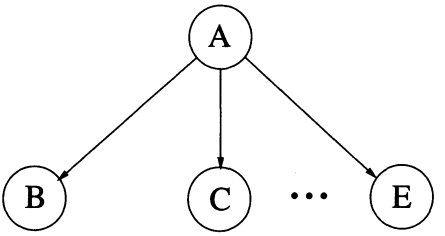

FIGURE 1.4. Diverging connection. If A is instantiated, it blocks for communication between its children.

Example Figure 1.5 shows the causal relations between *Sex* (male, female), *length of hair* (long, short), and *stature* (<168 cm, >168 cm).

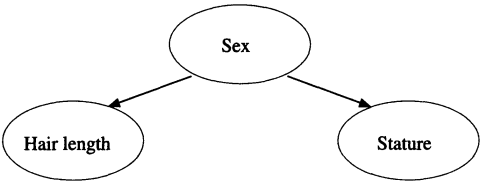

FIGURE 1.5. Sex has an impact on length of hair as well as stature.

If we do not know the sex of a person, seeing the length of his/her hair will tell us more about the sex, and this in turn will focus our belief on his/her stature. On the other hand, if we know that the person is a man, then length of hair gives us no extra clue on his stature.

Converging connections

A description of the situation in Figure 1.6 requires a little more care. If nothing is known about A except what may be inferred from knowledge of its parents $B, \ldots, E$, then the parents are independent: evidence on one of them has no influence on the certainty of the others. Knowledge of one possible cause of an event does not tell us anything about other possible causes. However, if anything is known about the consequences, then information on one possible cause may tell us something about the other causes. This is the *explaining away* effect illustrated in the car stop problem: a has occurred, and b as well as c may cause a. If we then get the

information that c has occurred, the certainty of b will decrease. Also, if we get the information that c has not occurred, then the certainty of b will increase. In Figure 1.8, examples are listed.

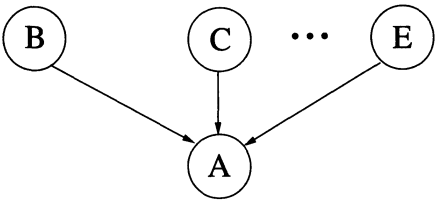

FIGURE 1.6. Converging connection. If A changes certainty, it opens for communication between its parents.

The conclusion is that evidence may only be transmitted through a converging connection if either the variable in the connection or one of its descendants has received evidence.

Remark Evidence on a variable is a statement of the certainties of its states. If the variable is instantiated, we call it *hard* evidence; otherwise it is called *soft*. Blocking in the case of serial and diverging connections requires hard evidence, whereas opening in the case of converging connections holds for all kinds of evidence.

Example Figure 1.7 shows the causal relations among *Salmonella* infection, *flu*, *nausea*, and *pallor*.

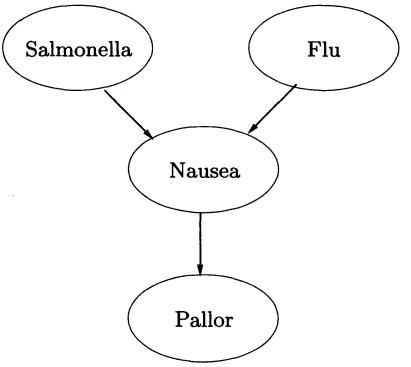

FIGURE 1.7. Salmonella and flu may cause nausea, which in turn causes pallor.

If we know nothing of nausea or pallor, then the information on whether the person has a *Salmonella* infection will not tell us anything about flu. However, if we have noticed that the person is pale, then the information that he/she does not have a *Salmonella* infection will make us more ready to believe that he/she has the flu.

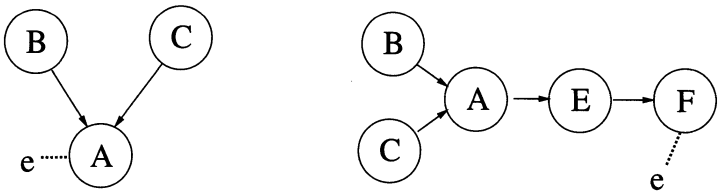

FIGURE 1.8. Examples where the parents of A are dependent. The dotted lines indicate insertion of evidence.

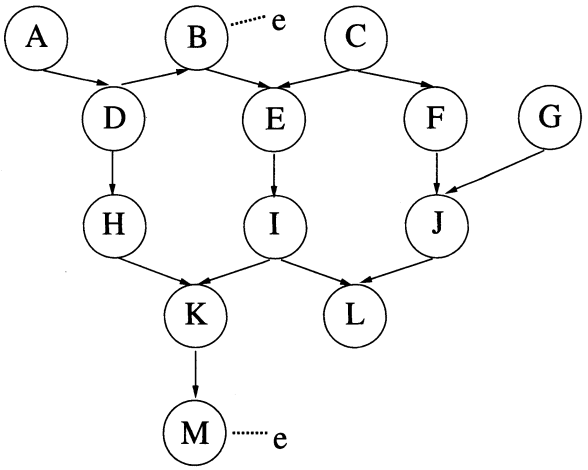

FIGURE 1.9. A causal network with B and M instantiated. A is d-separated from G only.

1.2.1 d-separation

The three preceeding cases cover all ways in which evidence may be transmitted through a variable, and following the rules it is possible to decide for any pair of variables in a causal network whether they are independent given the evidence entered into the network. The rules are formulated in the following.

Definition (d-separation): Two distinct variables A and B in a causal network are *d-separated* if, for all paths between A and B, there is an intermediate variable V (distinct from A and B) such that either

- the connection is serial or diverging and V is instantiated

  or

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

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

Figure 1.9 gives an example of a larger network. The evidence entered at B and M represents instantiation. If evidence is entered at A, it may be transmitted to D. The variable B is blocked, so the evidence cannot pass through B to E. However, it may be passed to H and K. Since the child M of K has received evidence, evidence from H may pass to I and further to E, C, F, J, and L, so the path $A-D-H-K-I-E-C-F-J-L$ is a d-connecting path.

Figure 1.10 gives two other examples.

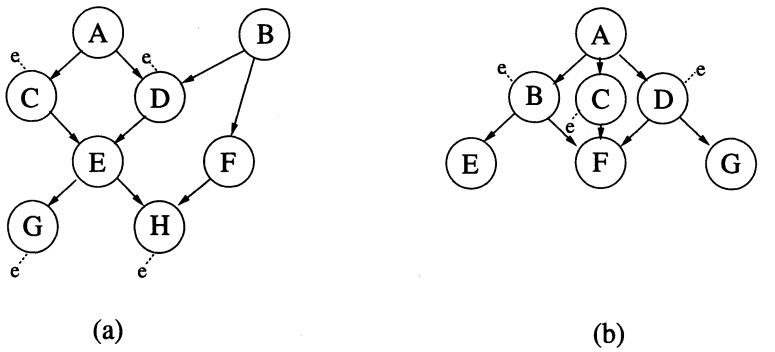

FIGURE 1.10. Causal networks with hard evidence entered (the variables are instantiated). (a) Although all neighbors of E are instantiated, it is d-connected to F, B, and A. (b) F is d-separated from the remaining uninstantiated variables.

Note that although A and B are d-connected, changes in the belief in A will not necessarily change the belief in B. To stress this, we will sometimes say that A and B are *structurally independent* if they are d-separated (see also Exercise 1.14).

You may wonder why we have introduced d-separation as a definition rather than as a theorem. A theorem should be as follows.

Claim If A and B are d-separated, then changes in the certainty of A have no impact on the certainty of B.

However, the claim cannot be established as a theorem without a more precise description of the concept of "certainty." You can take d-separation as a property of human reasoning and require that any certainty calculus comply with the claim.

Definition The *Markov blanket* of a variable A is the set consisting of the parents of A, the children of A, and the variables sharing a child with A.

Note If all variables in the Markov blanket for A are instantiated, then A is d-separated from the rest of the network (see Figure 1.11).

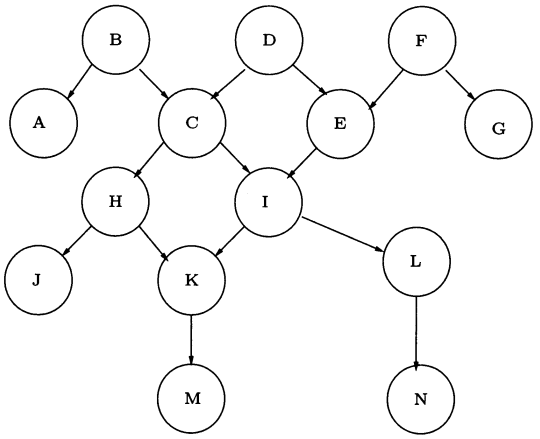

FIGURE 1.11. The Markov blanket for I is $\{C, E, H, K, L\}$. Note that if only I's neighbors are instantiated, then J is not d-separated from I.

1.3 Probability Calculus

So far, nothing has been said about the quantitative part of certainty assessment. Various certainty calculi exist, but in this book we only treat the so-called Bayesian calculus, which is *classical probability calculus*.

1.3.1 Basic axioms

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

(i) $P(a) = 1$ if and only if a is certain.

(ii) If a and b are mutually exclusive, then

$$P(a \vee b) = P(a) + P(b).$$

1.3.2 Conditional probabilities

The basic concept in the Bayesian treatment of certainties in causal networks is *conditional probability.* Whenever a statement of the probability $P(a)$ of an event a is given, then it is given conditioned by other known factors. A statement such as "the probability of the die turning up 6 is $\frac{1}{6}$" usually has the unsaid prerequisite that it is a fair die — or rather, as long as I know nothing of it, I assume it to be a fair die. This means that the statement should be "given that it is a fair die, the probability ...." In this way, any statement on probabilities is a statement conditioned on what else is known.

A conditional probability statement is of the following kind:

"Given the event b, the probability of the event a is x."

The notation for the preceding statement is $P(a \mid b) = x$.

It should be stressed that $P(a \mid b) = x$ does not mean that whenever b is true, then the probability for a is x. It means that if b is true, and *everything else known is irrelevant for* a, then the probability of a is x ("everything else" may be separated from a given b).

The fundamental rule

The *fundamental rule* for probability calculus is

$$P(a \mid b)P(b) = P(a, b), \tag{1.1}$$

where $P(a, b)$ is the probability of the joint event $a \wedge b$. Remembering that probabilities should always be conditioned by a context c, the formula should read

$$P(a \mid b, c)P(b \mid c) = P(a, b \mid c). \tag{1.2}$$

From (1.1) follows $P(a \mid b)P(b) = P(b \mid a)P(a)$, and this yields the well-known *Bayes' rule*

$$P(b \mid a) = \frac{P(a \mid b)P(b)}{P(a)}. \tag{1.3}$$

Bayes' rule conditioned on c reads

$$P(b \mid a, c) = \frac{P(a \mid b, c)P(b \mid c)}{P(a \mid c)}. \tag{1.4}$$

Formula (1.2) should be considered an axiom for probability calculus rather than a theorem. A justification for the formula can be found by counting frequencies. Suppose we have m cats (c) of which n are brown (b), and i

of the brown cats are Abyssinians (a). Then, the frequency of a's given b among the cats, $f(a \mid b, c)$, is $\frac{i}{n}$, the frequency of b's, $f(b \mid c)$, is $\frac{n}{m}$, and the frequency of brown Abyssinian cats, $f(a, b \mid c)$, is $\frac{i}{m}$. Hence,

$$f(a \mid b, c) f(b \mid c) = f(a, b \mid c).$$

Likelihood

Sometimes $P(a \mid b)$ is called the *likelihood of b given a*, and it is denoted $L(b \mid a)$.

The reason for this is the following. Assume $b_1, \ldots, b_n$ are possible scenarios with an effect on the event a, and we know a. Then, $P(a \mid b_i)$ is a measure of how likely it is that b_i is the cause. In particular, if all b_i's have the same prior probability, Bayes' rule yields

$$P(b_i \mid a) = \frac{P(a \mid b_i) P(b_i)}{P(a)} = kP(a \mid b_i),$$

where k is independent of i.

1.3.3 Subjective probabilities

The justification in the previous section for the fundamental rule was based on frequencies. This does not mean that we only consider probabilities based on frequencies. Probabilities may also be completely subjective estimates of the certainty of an event.

A subjective probability may, for example, be my personal assessment of the chances of selling more than 2,000 copies of this book in year 2001.

A way to assess this probability could be the following. I am given the choice between two gambles:

1. If more than 2,000 copies are sold in the year 2001, I will receive \$100.
2. I will by the end of year 2001 be allowed to draw a ball from an urn with n red balls and $100 - n$ white balls. If my ball is red, I will get \$100.

If all balls in the urn are red, I will prefer (2), and if all balls are white, I will prefer (1). There is a number n for which the two gambles are equally attractive, and, for this n, $\frac{n}{100}$ is my estimate of the probability of selling more than 2,000 copies of this book in year 2001. (I will not disclose the n to the reader.)

For subjective probabilities defined through such ball-drawing gambles, the fundamental rule can also be proved.

1.3.4 Probability calculus for variables

As stated in Section 1.2, the nodes in a causal network are *variables* with a *finite number of mutually exclusive states*.

If A is a variable with states $a_1, \ldots, a_n$, then $P(A)$ denotes a probability distribution over these states

$$P(A) = (x_1, \ldots, x_n); \qquad x_i \geq 0; \qquad \sum_{i=1}^{n} x_i = 1,$$

where x_i is the probability of A being in state a_i.

Notation The probability of A being in state a_i is denoted $P(A = a_i)$ and denoted $P(a_i)$ if the variable is obvious from the context.

If the variable B has states $b_1, \ldots, b_m$, then $P(A \mid B)$ denotes an $n \times m$ table containing numbers $P(a_i \mid b_j)$ (see Table 1.1).

	b_1	b_2	b_3
a_1	0.4	0.3	0.6
a_2	0.6	0.7	0.4

TABLE 1.1. An example of $P(A \mid B)$. Note that the columns sum up to 1.

$P(A, B)$, the joint probability for the variables A and B, is also a notation for an $n \cdot m$ table. It consists of a probability for each configuration (a_i, b_j) (see Table 1.2).

	b_1	b_2	b_3
a_1	0.16	0.12	0.12
a_2	0.24	0.28	0.08

TABLE 1.2. An example of $P(A, B)$. Note that the sum of all entries is 1.

When the fundamental rule (1.1) is used on variables A and B, the procedure is to apply the rule to the $n \cdot m$ configurations (a_i, b_j)

$$P(a_i \mid b_j)P(b_j) = P(a_i, b_j).$$

This means that in the table $P(A \mid B)$, for each j the column for b_j is multiplied by $P(b_j)$ to obtain the table $P(A, B)$. If $P(B) = (0.4, 0.4, 0.2)$, then Table 1.2 is the result of using the fundamental rule on Table 1.1. When applied to variables, we use the same notation for the fundamental rule:

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

From a table $P(A, B)$, the probability distribution $P(A)$ can be calculated. Let a_i be a state of A. There are exactly m different events for which A is in state a_i, namely the mutually exclusive events $(a_i, b_1), \ldots, (a_i, b_m)$. Therefore, by axiom (ii),

$$P(a_i) = \sum_{j=1}^{m} P(a_i, b_j).$$

This calculation is called *marginalization*, and we say that the variable B is marginalized out of $P(A, B)$ (resulting in $P(A)$). The notation is

$$P(A) = \sum_B P(A, B).$$

By marginalizing B out of Table 1.2, we get $P(A) = (0.4, 0.6)$.

The division in Bayes' rule (1.3) is treated in the same way as the multiplication in the fundamental rule (see Table 1.3).

	a_1	a_2
b_1	0.4	0.4
b_2	0.3	0.47
b_3	0.3	0.13

TABLE 1.3. $P(B \mid A)$ as a result of applying Bayes' rule to Table 1.1 and $P(B) = (0.4, 0.4, 0.2)$.

1.3.5 An algebra of potentials

For later use, we will list some properties of the algebra of multiplication and marginalization of probability tables. The tables need not be (conditional) probabilities, and they are generally called *potentials*.

A potential is a real-valued table over a *domain* of finite variables. $\mathrm{dom}(\phi)$ denotes the domain of the potential ϕ.

Two potentials can be *multiplied*, denoted by a (often hidden) dot. Multiplication has the following properties:

(i) $\mathrm{dom}(\phi_1\phi_2) = \mathrm{dom}(\phi_1) \cup \mathrm{dom}(\phi_2)$.

(ii) **The commutative law**: $\phi_1\phi_2 = \phi_2\phi_1$.

(iii) **The associative law**: $(\phi_1\phi_2)\phi_3 = \phi_1(\phi_2\phi_3)$.

(iv) **Existence of unit**: The number 1 is a potential over the empty domain, and $1 \cdot \phi = \phi$ for all potentials ϕ. The unit potential is denoted **1**.

A potential can be *marginalized*. $\sum_A \phi$ is a potential over $\mathrm{dom}(\phi) \setminus \{A\}$. Marginalization is *commutative*.

$$\sum_A \sum_B \phi = \sum_B \sum_A \phi.$$

For potentials of the form $P(A \mid V)$, where V is a set of variables, we have:

$B \setminus A$	a_1	a_2
b_1	x_1	x_2
b_2	x_3	x_4

$B \setminus C$	c_1	c_2
b_1	y_1	y_2
b_2	y_3	y_4

TABLE 1.4. $\phi_1(A, B)$ and $\phi_2(C, B)$.

$B \setminus A$	a_1	a_2
b_1	(x_1y_1, x_1y_2)	(x_2y_1, x_2y_2)
b_2	(x_3y_3, x_3y_4)	(x_4y_3, x_4y_4)

TABLE 1.5. $\phi_1(A, B) \cdot \phi_2(C, B)$. The two numbers in each entry correspond to the states c_1 and c_2.

(v) **The unit potential property**: $\sum_A P(A \mid V) = \mathbf{1}$.

For marginalization of a product, the following holds:

(vi) **The distributive law**: If $A \notin \mathrm{dom}(\phi_1)$, then $\sum_A \phi_1\phi_2 = \phi_1 \sum_A \phi_2$.

The distributive law is usually known as $ab + ac = a(b + c)$, and the preceding formula is actually the same law applied to tables. To verify it, consider the calculations in Tables 1.4-1.8.

We also use the term *projection* for marginalization. If A and B, for example, are marginalized out of $\phi(A, B, C)$, we may say that ϕ is *projected* down to C, and we use the notation $\phi^{\downarrow C}$. With this notation, the properties of marginalization look as follows (V and W denote sets of variables):

(vii) **The commutative law**: $(\phi^{\downarrow V})^{\downarrow W} = (\phi^{\downarrow W})^{\downarrow V}$.

(viii) **The distributive law**: If $\mathrm{dom}(\phi_1) \subseteq V$, then $(\phi_1\phi_2)^{\downarrow V} = \phi_1(\phi_2^{\downarrow V})$.

1.3.6 Calculation with joint probability tables

The reasoning under uncertainty described in Sections 1.1.1 and 1.1.2 can be performed if we have a joint probability distribution for the variables in question. For the problem in Figure 1.1, for example, we need a joint probability table for the variables *Fuel?*, *Clean Spark Plugs?*, *Fuel Meter Standing*, and *Start?*. We receive some evidence e (for example, *Start?=no* and *Fuel Meter Standing* $= \frac{1}{2}$), and we wish to calculate the new probability

$B \setminus A$	a_1	a_2
b_1	$x_1y_1 + x_1y_2$	$x_2y_1 + x_2y_2$
b_2	$x_3y_3 + x_3y_4$	$x_4y_3 + x_4y_4$

TABLE 1.6. $\sum_C \phi_1(A, B) \cdot \phi_2(C, B)$.

B	
b_1	$y_1 + y_2$
b_2	$y_3 + y_4$

TABLE 1.7. $\sum_C \phi_2(C, B)$.

$B \setminus A$	a_1	a_2
b_1	$x_1(y_1 + y_2)$	$x_2(y_1 + y_2)$
b_2	$x_3(y_3 + y_4)$	$x_4(y_3 + y_4)$

TABLE 1.8. $\phi_1(A, B) \sum_C \phi_2(C, B)$.

for the remaining variables (*Fuel?* and *Clean Spark Plugs?*). To illustrate the method, assume we have three variables, A, B, and C, with the joint probabilities as in Table 1.9. We receive evidence $A = a_2$ and $C = c_1$.

	b_1	b_2	b_3
a_1	(0, 0.05, 0.05)	(0.05, 0.05, 0)	(0.05, 0.05, 0.05)
a_2	(0.1, 0.1, 0)	(0.1, 0, 0.1)	(0.2, 0, 0.05)

TABLE 1.9. A joint probability table for the variables A, B, and C. The three numbers in each entry correspond to the states c_1, c_2, and c_3.

First, we focus on the part of the table corresponding to $A = a_2$ and $C = c_1$, and we have

$$P(a_2, B, c_1) = (0.1, 0.1, 0.2). \tag{1.5}$$

To calculate $P(B \mid a_2, c_1)$, we can use the fundamental rule

$$P(B \mid a_2, c_1) = \frac{P(a_2, B, c_1)}{P(a_2, c_1)}. \tag{1.6}$$

To be able to use (1.6), we need $P(a_2, c_1)$. This is achieved by marginalizing B out of (1.5), and we get $P(a_2, c_1) = 0.4$. Another way of doing the same is to say that we wish to transform $P(a_2, B, c_1)$ into a probability distribution. Because the numbers do not add up to one, we *normalize* the distribution by dividing each number with their sum. We get $P(B \mid a_2, c_1) = (0.25, 0.25, 0.5)$.

1.3.7 Conditional independence

The blocking of transmission of evidence as described in Section 1.2.1 is, in Bayesian calculus, reflected in the concept of *conditional independence.*

The variables A and C are *independent given the variable* B if

$$P(a_i \mid b_j) = P(a_i \mid b_j, c_k) \tag{1.7}$$

for all i, j, k.

This means that if the state of B is known, then no knowledge of C will alter the probability of A. (We will write $P(A \mid B) = P(A \mid B, C)$, although the two tables do not have the same dimensions.)

Remark If the condition B is empty, we simply say that A and C are independent.

Conditional independence appears in the cases of serial and diverging connections (see Figure 1.12).

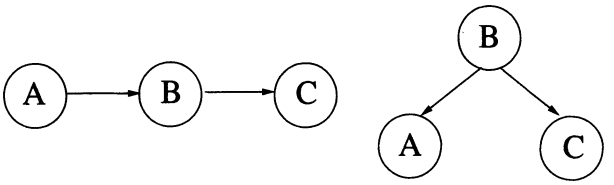

FIGURE 1.12. Examples where A and C are conditionally independent given B.

Definition (1.7) may look asymmetric; however, if (1.7) holds, then – by the conditioned Bayes' rule (1.4) – we get

$$P(C \mid B, A) = \frac{P(A \mid C, B) \cdot P(C \mid B)}{P(A \mid B)} = \frac{P(A \mid B) \cdot P(C \mid B)}{P(A \mid B)} = P(C \mid B).$$

The proof requires that $P(A \mid B) > 0$, that is, for states a, b with $P(A = a \mid B = b) = 0$, the calculation is not valid. However, for our considerations it does not matter; if B is in state b, then the evidence $A = a$ is impossible and will not appear, so why bother with the transmission of it?

1.4 Bayesian Networks

1.4.1 Definition of Bayesian networks

Causal relations also have a quantitative side, namely their *strength*. This is expressed by attaching numbers to the links.

Let A be a parent of B. Using probability calculus, it would be natural to let $P(B \mid A)$ be the strength of the link. However, if C is also a parent of B, then the two conditional probabilities $P(B \mid A)$ and $P(B \mid C)$ alone do not give any clue about how the impacts from A and C interact. They may cooperate or counteract in various ways, so we need a specification of $P(B \mid A, C)$.

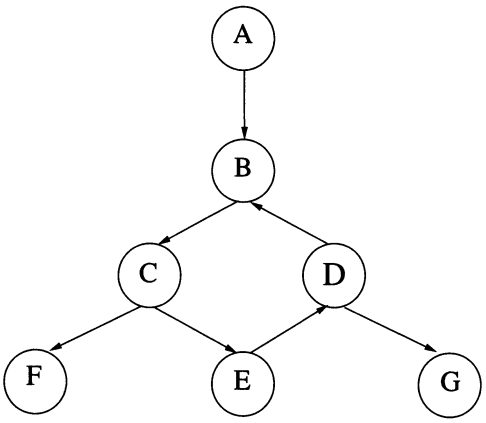

FIGURE 1.13. A directed graph with a feedback cycle. This is not allowed in Bayesian networks.

It may happen that the domain to be modeled contains feedback cycles (see Figure 1.13).

Feedback cycles are difficult to model quantitatively (this is an example of what differential equations are all about). For causal networks, no calculus has been developed that can cope with feedback cycles. Therefore, we require that the network does not contain cycles.

Definition A *Bayesian network* consists of the following:

- A set of *variables* and a set of *directed edges* between variables.
- Each variable has a finite set of mutually exclusive states.
- The variables together with the directed edges form a *directed acyclic graph* (DAG). (A directed graph is *acyclic* if there is no directed path $A_1 \to \cdots \to A_n$ s. t. $A_1 = A_n$.)
- To each variable A with parents $B_1, \ldots, B_n$, there is attached the potential table $P(A \mid B_1, \ldots, B_n)$.

Note that if A has no parents, then the table reduces to unconditional probabilities $P(A)$. For the DAG in Figure 1.14, the prior probabilities $P(A)$ and $P(B)$ must be specified. It has been claimed that prior probabilities are an unwanted introduction of bias to the model, and calculi have been invented in order to avoid it. However, as discussed in Section 1.1.2, prior probabilities are necessary not for mathematical reasons but because prior certainty assessments are an integral part of human reasoning about certainty (see also Exercise 1.9).

The definition of Bayesian networks does not refer to causality, and there is no requirement that the links represent causal impact. Instead, we require that *the d-separation properties implied by the structure hold.*

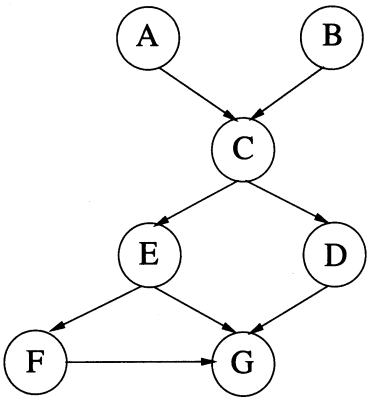

FIGURE 1.14. A directed acyclic graph (DAG). The probabilities to specify are $P(A)$, $P(B)$, $P(C \mid A,B)$, $P(E \mid C)$, $P(D \mid C)$, $P(F \mid E)$, and $P(G \mid D,E,F)$.

This also means that if A and B are d-separated given evidence e, then the probability calculus used for Bayesian networks must yield $P(A \mid e) = P(A \mid B,e)$ (see Section 1.4.4).

When building the structure of Bayesian network models, we need not insist on having the links go in a causal direction. On the other hand, we then need to check its d-separation properties to ensure that they correspond with our perception of the world.

1.4.2 A Bayesian network for car start

The Bayesian network for the reduced car start problem is the one in Figure 1.15.

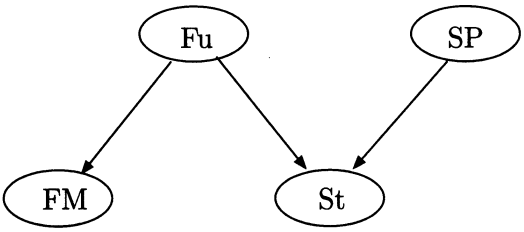

FIGURE 1.15. The causal network for the reduced car start problem. We have used the abbreviations *Fu (Fuel?), SP (Clean Spark Plugs?), St (Start?),* and *FM (Fuel Meter Standing).*

For the quantitative modeling, we need the probability assessments $P(Fu), P(SP), P(St \mid Fu,SP), P(FM \mid Fu)$. To avoid having to deal with numbers that are too small, let $P(Fu) = (0.98, 0.02)$ and $P(SP) =$

(0.96, 0.04). The remaining tables are given in Table 1.10. Note that the table for $P(FM \mid Fu)$ reflects the fact that the fuel meter may be malfunctioning, and the table for $P(St \mid Fu, SP)$ gives room for causes other than no fuel and dirty spark plugs.

	$Fu = yes$	$Fu = no$
$FM = full$	0.39	0.001
$FM = \frac{1}{2}$	0.60	0.001
$FM = empty$	0.01	0.998

$P(FM \mid Fu)$

	$Fu = yes$	$Fu = no$
$Sp = yes$	(0.99, 0.01)	(0,1)
$Sp = no$	(0.01, 0.99)	(0,1)

$P(St \mid Fu, Sp)$

TABLE 1.10. Conditional probabilities for the model in Figure 1.15. The numbers (x, y) in the lower table represent $(St = yes, St = no)$.

1.4.3 The chain rule for Bayesian networks

Let $U = \{A_1, \ldots, A_n\}$ be a universe of variables. If we have access to the joint probability table $P(U) = P(A_1, \ldots, A_n)$, then we can also calculate $P(A_i)$ as well as $P(A_i \mid e)$, where e is evidence (see Section 1.3.6). However, $P(U)$ grows exponentially with the number of variables, and U need not be very large before the table becomes intractably large. Therefore, we look for a more compact *representation* of $P(U)$, a way of storing information from which $P(U)$ can be calculated if needed.

A Bayesian network over U is such a representation. If the conditional independencies in the Bayesian network hold for U, then $P(U)$ can be calculated from the potentials specified in the network.

Theorem 1.1 (The chain rule for Bayesian networks) *Let* BN *be a Bayesian network over* $U = \{A_1, \ldots, A_n\}$. *Then, the joint probability distribution* $P(U)$ *is the product of all potentials specified in* BN

$$P(U) = \prod_i P(A_i \mid pa(A_i)), \tag{1.8}$$

where $pa(A_i)$ *is the parent set of* A_i.

Proof: (Induction in the number of variables in the universe U.)

If U consists of one variable, then the theorem is trivial.

Assume the chain rule to be true for all networks consisting of $n-1$ variables, and let U be the universe for a DAG with n variables. Since the network is acyclic, there is at least one variable A without children. Consider the DAG with A removed.

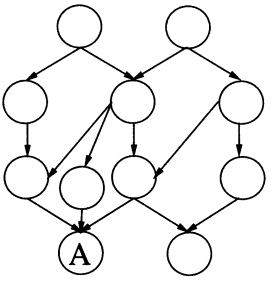

FIGURE 1.16. A DAG with n variables. If the variable A is removed, the induction hypothesis can be applied.

From the induction hypothesis, we have that $P(U \setminus \{A\})$ is the product of all specified probabilities — except $P(A \mid pa(A))$.

By the fundamental rule, we have

$$P(U) = P(A \mid U \setminus \{A\}) \cdot P(U \setminus \{A\}).$$

Since A is d-separated from $U \setminus (\{A\} \cup pa(A))$ given $pa(A)$ (see Figure 1.16), we get

$$P(U) = P(A \mid U \setminus \{A\}) P(U \setminus \{A\}) = P(A \mid pa(A)) \cdot P(U \setminus \{A\}).$$

The right-hand side in the preceding formula is the product of all specified probabilities. □

1.4.4 Bayesian networks admit d-separation

In the proof of the chain rule for Bayesian networks, we used d-separation properties induced by the causal network. As claimed in Section 1.2.1, d-separation is a property of human reasoning and the chain rule is a result of this claim. Because we have established that the joint probability of the universe is the product of the attached conditional probabilities (1.8), we can ask whether the d-separation properties are also buried in this formula. In other words, if the causal network yields that C is d-separated from A given the set V, can we then from (1.8) infer that $P(C \mid A, V) = P(C \mid V)$? We will not give a proof of this fact but only a couple of indications.

Serial connection

We will prove that if A is serially connected to C through B, then $P(C \mid B, A) = P(C \mid B)$.

If A, B, and C are the only variables, the chain rule for Bayesian networks yields

$$P(A,B,C) = P(A)P(B \mid A)P(C \mid B) = P(A,B)P(C \mid B).$$

Then,

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

Converging connection
Let A and B be parents of C. We wish to prove $P(A \mid B) = P(A)$, which is the same as $P(A,B) = P(A)P(B)$.

If A, B, and C are the only variables, the chain rule for Bayesian networks yields

$$P(A,B,C) = P(A)P(B)P(C \mid B,A).$$

Then, by the distributive law, we have

$$P(A,B) = \sum_C P(A)P(B)P(C \mid B,A) = P(A)P(B)\sum_C P(C \mid B,A).$$

Because the "columns" in a conditional probability table sum to 1, we have that $\sum_C P(C \mid B,A)$ is a table of 1's. Therefore, $P(A,B) = P(A)P(B)$.

1.4.5 Car start revisited

In this section, we apply the rules of probability calculus to the car start problem. This is done to illustrate that probability calculus can be used to perform the reasoning in the example – in particular, explaining away. In Chapter 5, we give a general algorithm for probability updating in Bayesian networks. We will use the Bayesian network from Section 1.4.2 to perform the reasoning in Section 1.1.1.

We will use the joint probability table for the reasoning. The joint probability table is calculated from the chain rule for Bayesian networks

$$P(Fu, FM, SP, St) = P(Fu)P(SP)P(FM \mid Fu)P(St \mid Fu, SP).$$

The result is given in Tables 1.11 and 1.12.

	$FM = full$	$FM = \frac{1}{2}$	$FM = empty$
$Sp = yes$	$(0.363, 0)$	$(0.559, 0)$	$(0.0093, 0)$
$Sp = no$	$(0.00015, 0)$	$(0.00024, 0)$	$(3.9 \cdot 10^{-6}, 0)$

TABLE 1.11. The joint probability table for $P(Fu, FM, SP, St = yes)$. The numbers (x, y) in the table represent $(Fu = yes, Fu = no)$.

	$FM = full$	$FM = \frac{1}{2}$	$FM = empty$
$Sp = yes$	$(0.00367, 1.9 \cdot 10^{-5})$	$(0.00564, 1.9 \cdot 10^{-5})$	$(9.4 \cdot 10^{-5}, 0.0192)$
$Sp = no$	$(0.01514, 8 \cdot 10^{-7})$	$(0.0233, 8 \cdot 10^{-7})$	$(0.000388, 0.000798)$

TABLE 1.12. The joint probability table for $P(Fu, FM, SP, St = no)$. The numbers (x, y) in the table represent $(Fu = yes, Fu = no)$.

The evidence $St = no$ tells us that we are in the context of Table 1.12. By marginalizing FM and Fu out of Table 1.12 (summing each row), we get $P(SP, St = no) = (0.02864, 0.03965)$.

We get the conditional probability $P(SP \mid St = no)$ by dividing with $P(St = no)$. This is easy. $P(St = no)$ is the sum of the two numbers in $P(SP, St = no)$, and we get $P(SP \mid St = no) = (0.42, 0.58)$. Another way of saying it is that the distribution we end up with will be a set of numbers that sum to 1. If they do not, normalize by dividing by the sum. In the same way, we get $P(Fu \mid St = no) = (0.71, 0.29)$.

Next, we get the information that $FM = \frac{1}{2}$, and the context for calculation is limited to the part with $FM = \frac{1}{2}$ and $St = no$. The numbers are given in Table 1.13.

	$Fu = yes$	$Fu = no$
$Sp = yes$	0.00564	$1.9 \cdot 10^{-5}$
$Sp = no$	0.0233	$8 \cdot 10^{-7}$

TABLE 1.13. $P(Fu, SP, St = no, FM = \frac{1}{2})$.

By marginalizing and normalizing, we get $P(Fu|St = no, FM = \frac{1}{2}) = (0.999, 0.001)$ and $P(SP|St = no, FM = \frac{1}{2}) = (0.196, 0.804)$, so the calculus did catch the explaining away effect.

1.4.6 Evidence

Bayesian networks are used for calculating new probabilities when you achieve particular information. The information so far has been of the type "$A = a$", where A is a variable and a is a state of A. Let A have n states with $P(A) = (x_1, \ldots, x_n)$, and assume that we get the information e that A can only be in state i or j. This statement expresses that all states except i and j are impossible, and we have the probability distribution $P(A, e) = (0, \ldots, 0, x_i, 0, \ldots, 0, x_j, 0, \ldots, 0)$. Note that $P(e)$, the prior probability of e, is the sum of $P(a, e)$. Note also that $P(A, e)$ is the result of multiplying $P(A)$ with $(0, \ldots, 0, 1, 0, \ldots, 0, 1, 0, \ldots, 0)$, where the 1's are at the ith and jth places.

Definition Let A be a variable with n states. A *finding* on A is an n-dimensional table of zeros and ones.

To distinguish between the statement e "A is in either state i or j" and the corresponding finding, we sometimes use the underline notation $\underline{e}$ for the finding. Semantically, a finding is a statement that certain states of A are impossible.

Now, assume that you have a joint probability table, $P(U)$, and let $\underline{e}$ be the preceding finding. The joint probability table $P(U, e)$ is the table resulting from $P(U)$ by giving all entries with A not in state i or j the value zero and leaving the other entries unchanged. This is the same as multiplying $P(U)$ with $\underline{e}$,

$$P(U, e) = P(U) \cdot \underline{e}.$$

Note that $P(e) = \sum_U P(U, e) = \sum_U (P(U) \cdot \underline{e})$. Using the chain rule for Bayesian networks, we have the following theorem.

Theorem 1.2 *Let BN be a Bayesian network over the universe U, and let $\underline{e}_1, \ldots, \underline{e}_n$ be findings; then*

$$P(U, e) = \prod_{A \in U} P(A \mid pa(A)) \cdot \prod_i \underline{e}_i,$$

and for $A \in U$ we have

$$P(A \mid e) = \frac{\sum_{U \setminus \{A\}} P(U, e)}{P(e)}.$$

Some types of evidence cannot be represented as findings. I may, for example, receive a statement from someone that the chance of A being in state a_1 is twice as high as for a_2. This type of evidence is called *likelihood evidence*. It is possible to treat this kind of evidence in Bayesian networks. The preceding statement is then represented by the distribution (0.67, 0.33), and Theorem 1.2 still holds. However, because it is unclear what it means that a likelihood statement is true, $P(e)$ cannot be interpreted as the probability of the evidence, and $P(U, e)$ has unclear semantics. We will not deal further with likelihood evidence.

1.4.7 Bucket elimination

As described in Section 1.3.6 and illustrated in Section 1.4.5, probability updating in Bayesian networks can be performed by using the chain rule to calculate $P(U)$, the joint probability table of the universe. However, U need not be large before $P(U)$ becomes intractably large. In this section, we illustrate how the calculations can be performed without having to deal with the full joint table. In Chapter 5, we give a detailed treatment of algorithms for updating.

Consider the Bayesian network in Figure 1.17, and assume that all variables have ten states. Assume that we have the evidence $e = \{D = d, F = f\}$, and we wish to calculate $P(A \mid e)$.

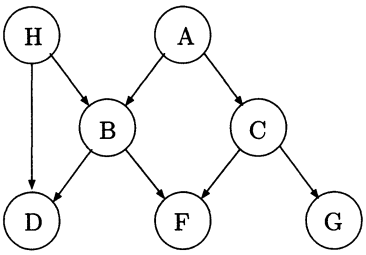

FIGURE 1.17. A Bayesian network. Evidence $e = \{D = d, F = f\}$.

We have

$$P(U, e) = P(A, B, C, d, f, G, H) = P(A)P(H)P(B \mid A, H)P(C \mid A)P(d \mid B, H)P(f \mid B, C)P(G \mid C),$$

where for example $P(d \mid B, H)$ denotes the table over B and H resulting from fixing the D-entry to the state d. We say that potential has been *instantiated to* $D = d$. Notice that we need not calculate the full table $P(U)$ with 10^7 entries. If we wait until evidence is entered, we will in this case only need to work with a table with 10^5 entries. Later, we show that we need not work with tables larger than $1,000$ entries.

To achieve $P(A, e)$, we marginalize the variables B, C, G, and H out of $P(A, B, C, d, f, G, H)$. The order in which we marginalize is unimportant (Section 1.3.5), so let us start with G; that is, we wish to calculate

$$\sum_G P(A, B, C, d, f, G, H) = \sum_G P(A)P(H)P(B \mid A, H)P(C \mid A)P(d \mid B, H)P(f \mid B, C)P(G \mid C).$$

In the left-hand product, only the last table contains G in its domain, and due to the distributive law (Section 1.3.5) we have

$$\sum_G P(A, B, C, d, f, G, H) = P(A)P(H)P(B \mid A, H)P(C \mid A)P(d \mid B, H)P(f \mid B, C)\sum_G P(G \mid C),$$

and we need only calculate $\sum_G P(G \mid C)$. Actually, for each state c of C, we have $\sum_G P(G \mid c) = 1$, and we need not perform any calculation at all. We have

$$P(A, B, C, d, f, H) = \sum_G P(A, B, C, d, f, G, H) = P(A)P(H)P(B \mid A, H)P(C \mid A)P(d \mid B, H)P(f \mid B, C).$$

Next, we marginalize H out. Using the distributive law again, we get

$$\sum_H P(A,B,C,d,f,H) =$$
$$P(A)P(C \mid A)P(f \mid B,C)\sum_H P(H)P(B \mid A,H)P(d \mid B,H).$$

We multiply the three tables $P(H), P(B \mid A,H)$, and $P(d \mid B,H)$, and we marginalize H out of the product. The result is a table $T(B,A)$, and we have

$$P(A,B,C,d,f) = P(A)P(C \mid A)P(f \mid B,C)T(B,A).$$

Finally, we calculate this product and marginalize B and C out of it.

Notice that we never work with a table of more than three variables compared to the five variables in $P(A,B,C,d,f,G,H)$.

The method just used can be described in the following way: we start with a set V of tables, and whenever we wish to marginalize a variable X, we take from V all tables with X in their domains. Calculate the product of them, marginalize X out of it, and place the resulting table in V. This is called *eliminating the variable* X, and the process of repeatedly eliminating a variable from an initial set of tables is called *bucket elimination*.

1.4.8 Graphical models — formal languages for model specification

From a mathematical point of view, the basic property of Bayesian networks is the chain rule: a Bayesian network is a compact representation of the joint probability table over its universe. In this respect, a Bayesian network is one type of compact representation among many others. However, there is more to it than this. From a knowledge engineering point of view, a Bayesian network is a type of *graphical model*. The structure of the network is formulated in a graphical communication language for which the language features have very simple semantics, namely causality. This does not mean that "causality" is an easy concept. It may be very difficult to experience causality, and philosophically the concept is not fully understood. However, most often humans can communicate sensibly about causal relations in a knowledge domain. Furthermore, the graphical specification also specifies the requirements for the quantitative part of the model (the conditional probabilities). In Chapter 2, we extend the modeling language, and in Chapter 4 we present other types of graphical models.

Graphical models are communication languages. They consist of a qualitative part, where features from graph theory are used, and a quantitative part consisting of *potentials*, which are real-valued functions over sets of

nodes from the graph. The graphical part specifies the kind of potentials and their domains.

First, graphical models can be used for interpersonal communication. The graphical specification is easy for humans to read, and it helps focus attention, for example in a group working jointly on building a model. For interpersonal communication, the semantics of the various graph-theoretic features must be rather well-defined.

The next step in the use of graphical models has to do with communication to a computer. You wish to communicate a graphical model to a computer, and the computer should be able to process the model and give answers to various queries. In order to achieve this, the specification language must be formally defined with a well-defined syntax and semantics.

The first concern when constructing a graphical model language for a type of modeling task is to ensure that it is sufficiently well-defined that it can be communicated to a computer. This covers the graphical part as well as the specification of potentials. The next concern is the scope of the language: what is the range of domains and tasks that you will be able to model with this language? The final concern is tractability: do you have algorithms such that in reasonable time the computer can process a model and query to provide answers?

The Bayesian network is a sufficiently well-defined language, and behind the graphical specification in the user interface, the computer systems have an alpha-numeric specification language, which for some systems is open to the user. Actually, the language for Bayesian networks is a context-free language with a single context-sensitive aspect (no directed cycles).

The scope of the Bayesian network language is hard to define, but the examples in the next chapter show that it has a very broad scope.

Tractability is not a yes or no issue. As described in Chapter 5, there are algorithms for probability updating in Bayesian networks, but basically probability updating is NP-hard. This means that some models have an updating time exponential in the number of nodes.

On the other hand, the running times of the algorithms can be easily calculated without actually running them. In Chapters 5 and 7, we treat complexity issues for the various graphical languages presented.

1.5 Summary

d-separation in causal networks

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

- the connection is serial or diverging, and V is instantiated

or

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

The fundamental rule for probability calculus

$$P(A \mid B,C)P(B \mid C) = P(A,B \mid C).$$

Bayes' rule

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

Marginalization

$$P(A) = \sum_B P(A,B) = P(A,b_1) + \ldots + P(A,b_n).$$

The distributive law

If $A \notin \mathrm{dom}(\phi_1)$, then $\sum_A \phi_1\phi_2 = \phi_1 \sum_A \phi_2$.

Conditional independence

A and C are independent given B if $P(A \mid B) = P(A \mid B,C)$.

Definition of Bayesian networks

A Bayesian network consists of the following:

- A set of *variables* and a set of *directed edges* between variables.
- Each variable has a finite set of mutually exclusive states.
- The variables together with the directed edges form a *directed acyclic graph* (DAG).
- To each variable A with parents $B_1,\ldots,B_n$, there is attached the potential table $P(A \mid B_1,\ldots,B_n)$.

Admittance of d-separation in Bayesian networks

If A and B are d-separated in a Bayesian network with evidence e entered, then $P(A \mid B,e) = P(A \mid e)$.

The chain rule for Bayesian networks

Let BN be a Bayesian network over $U = \{A_1,\ldots,A_n\}$. Then, the joint probability distribution $P(U)$ is the product of all conditional probabilities specified in BN

$$P(U) = \prod_i P(A_i \mid pa(A_i)),$$

where $pa(A_i)$ are the parents of A_i. Let $\underline{e}_1, \ldots \underline{e}_m$ be findings, and then

$$P(U, e) = \prod_i P(A_i \mid pa(A_i)) \prod_j \underline{e}_j$$

and

$$P(A \mid e) = \frac{\sum_{U \setminus \{A\}} P(U, e)}{P(e)}.$$

1.6 Bibliographical Notes

The connection between causation and conditional independence was studied by Spohn (1980). The concepts of causal network, d-connection, and the definition in Section 1.2.1 are due to Pearl (1986) and Verma (1987). A proof that Bayesian networks admit d-separation can be found in (Pearl 1988) or in (Lauritzen 1996). Bayesian networks have a long history in statistics, and in the first half of the 1980s they were introduced to the field of expert systems through work by Pearl (1982) and Spiegelhalter and Knill-Jones (1984). The first real-world applications of Bayesian Networks were Munin (Andreassen et al. 1989) and Pathfinder (Heckerman et al. 1992). The basis for the inference method presented in Section 1.4.7 originates from D'Ambrosio (1991) and was modified to the presented bucket elimination in (Dechter 1996).

1.7 Exercises

Exercise 1.1 To illustrate that simple rules cannot cope with uncertainty reasoning, consider the following two cases:

(*i*) I have an urn with a red ball and a white ball in it. If I add a red ball and shake it, what is the certainty of drawing a red ball in one draw? If I add a white ball instead, what is the certainty of drawing a red ball? If I combine the two actions, what is the certainty of drawing a red ball?

(*ii*) When shooting, I am more certain to hit the target if I close the left eye. I am also more certain to hit the target if I close the right eye. What is the combined certainty if I do both?

Exercise 1.2 Construct a causal network and follow the reasoning in the following story. Mr. Holmes is working in his office when he receives a phone call from his neighbor, who tells him that Holmes' burglar alarm has gone off. Convinced that a burglar has broken into his house, Holmes rushes to

his car and heads for home. On his way, he listens to the radio, and in the news it is reported that there has been a small earthquake in the area. Knowing that earthquakes have a tendency to turn on burglar alarms, he returns to work.

Exercise 1.3 In the graphs in Figure 1.18, determine which variables are d-separated from A.

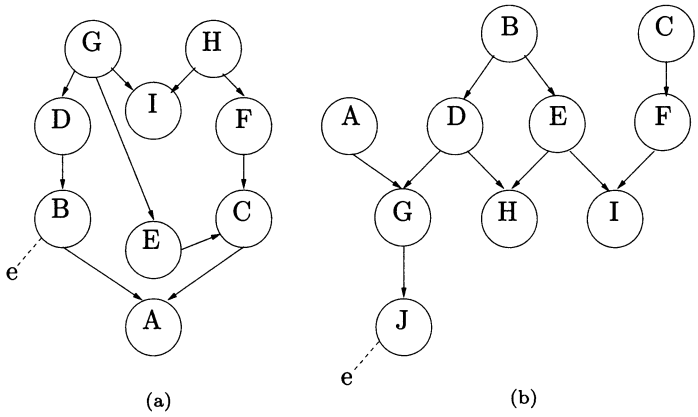

FIGURE 1.18. Figure for Exercise 1.3.

Exercise 1.4 Let A be a variable in a DAG. Assume that all variables in A's Markov blanket are instantiated. Show that A is d-separated from the remaining uninstantiated variables.

Exercise 1.5 Let D_1 and D_2 be DAGs over the same variables. D_1 is an *I-submap* of D_2 if all d-separation properties of D_1 also hold for D_2. If also D_2 is an I-submap of D_1, they are said to be *I-equivalent*.

Which of the four DAGs in Figure 1.19 are I-equivalent?

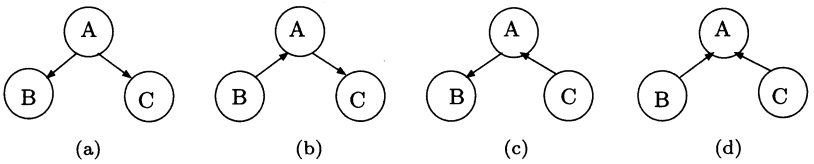

FIGURE 1.19. Figure for Exercise 1.5.

Exercise 1.6 Calculate $P(A)$, $P(B)$, $P(A \mid B)$, and $P(B \mid A)$ from the table $P(A, B)$ (Table 1.14).

Exercise 1.7 In Table 1.15, a joint probability table for the binary variables A, B, and C is given.

	b_1	b_2	b_3
a_1	0.05	0.10	0.05
a_2	0.15	0.00	0.25
a_3	0.10	0.20	0.10

TABLE 1.14. $P(A, B)$ for Exercise 1.6.

	b_1	b_2
a_1	(0.006, 0.054)	(0.048, 0.432)
a_2	(0.014, 0.126)	(0.032, 0.288)

TABLE 1.15. $P(A, B, C)$ for Exercise 1.7.

(*i*) Calculate $P(B, C)$ and $P(B)$.
(*ii*) Are A and C independent given B?

Exercise 1.8 Consider the DAG (a) in Exercise 1.5.

(*i*) Show that $P(B \mid A, C) = P(B \mid A)$.

We have $P(A) = (0.1, 0.9)$ and the conditional probability tables in Table 1.16.

(*ii*) Calculate $P(A, B, C)$.

Exercise 1.9 Table 1.17 describes a test T for an event A. The number 0.01 is the frequency of *false negatives*, and the number 0.001 is the frequency of *false positives*.

(*i*) The police can order a blood test on drivers under the suspicion of having consumed too much alcohol. The test has the preceding characteristics. Experience says that 20% of the drivers under suspicion do in fact drive with too much alcohol in their blood. A driver is taken with a positive blood test. What is the probability that the driver is guilty of driving under the influence of alcohol?

(*ii*) The police block a road, take blood samples of all drivers, and use the same test. It is estimated that one out of 1,000 drivers has too much alcohol in their blood. A driver has a positive test result. What is the

	a_1	a_2
b_1	0.2	0.3
b_2	0.8	0.7

$P(B \mid A)$

	a_1	a_2
c_1	0.5	0.6
c_2	0.5	0.4

$P(C \mid A)$

TABLE 1.16. Conditional probability tables for Exercise 1.8.

	$A = yes$	$A = no$
$T = yes$	0.99	0.001
$T = no$	0.01	0.999

TABLE 1.17. Table for Exercise 1.9. Conditional probabilities $P(T \mid A)$ characterizing test T for A.

probability that the driver is guilty of driving under the influence of alcohol?

Exercise 1.10 Let $P(c_i \mid b_j) \neq 0$ for all i, j. Prove that A and C are independent given B if, and only if, $P(A, C \mid B) = P(A \mid B) \cdot P(C \mid B)$.

Exercise 1.11 ^{E} Install an editor for Bayesian networks (a list of web sites for free downloading can be found in the Preface).

Exercise 1.12 ^{E} Construct a Bayesian network for Exercise 1.9.

Exercise 1.13 ^{E} Construct a Bayesian network to follow the reasoning from Exercise 1.2. Use your own estimates of probabilities for the network.

Exercise 1.14 ^{E} Consider the Bayesian network in Figure ?? with conditional probabilities given in Table 1.18. Use your system to investigate whether A and C are independent.

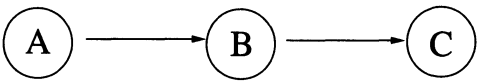

FIGURE 1.20. Figure for Exercise 1.14.

Exercise 1.15 ^{E} Use your system and Section 1.4.2 to perform the reasoning in Section 1.1.2.

	$A = yes$	$A = no$
b_1	0.6	0.2
b_2	0.1	0.5
b_3	0.2	0.1
b_4	0.1	0.2

$P(B \mid A)$

	b_1	b_2	b_3	b_4
$C = yes$	0.8	0.8	0.2	0.2
$C = no$	0.2	0.2	0.8	0.8

$P(C \mid B)$

TABLE 1.18. Tables for Exercise 1.14.

2
Building Models

Bayesian networks create a very efficient language for building models of domains with inherent uncertainty. However, as can be seen from the calculations in Section 1.4.5, it is a tedious job to perform evidence transmission even for very simple Bayesian networks. Fortunately, software tools that can do the calculation job for us are available. In the rest of this book, we assume that the reader has access to such a system (some URLs are given in the Preface).

Therefore, we can start by concentrating on how to use Bayesian networks in model building and defer a presentation of the methods for probability updating to Chapter 5.

In Section 2.1, we examine through examples the considerations when determining the structure of a Bayesian network model. Section 2.2 gives examples of estimation of the conditional probabilities. The examples cover theoretically well-founded probabilities as well as probabilities taken from databases and purely subjective estimates. Section 2.3 introduces various modeling tricks to use when the amount of numbers to acquire is overwhelming.

2.1 Catching the Structure

The first thing to have in mind when organizing a Bayesian network model is that its purpose is to give estimates of certainties for events that are not observable (or only observable at an unacceptable cost), and the primary

task in model building is to identify these events. We call them *hypothesis events.* The hypothesis events detected are then grouped into sets of mutually exclusive events to form *hypothesis variables.* The next thing to have in mind is that in order to come up with a certainty estimate, we should provide some information channels, and the task is to identify the types of achievable information that may reveal something about the hypothesis variables. These types of information are grouped into *information variables*, and a typical piece of information is a statement that a certain variable is in a particular state, but softer statements are also allowed.

Having identified the variables for the model, the next thing will be to establish the directed links for a causal network.

2.1.1 Milk test

> Milk from a cow may be infected. To detect whether the milk is infected, you have a test, which may give either a *positive* or a *negative* test result. The test is not perfect. It may give a positive result on clean milk as well as a negative result on infected milk.

We have two hypothesis events: *milk infected* and *milk not infected*, and because they are mutually exclusive, they are grouped into the variable *Infected?* with the states *yes* and *no*. The types of information are the test results, which can be either *positive* or *negative*. For this, we establish the variable *Test* with states *pos* and *neg*.

The causal direction between the two variables is from *Infected?* to *Test* (see Figure 2.1).

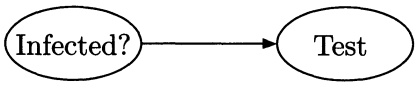

FIGURE 2.1. The Bayesian network for the milk test.

Certainly, no sensible person will claim that the positive test result may infect the milk. However, our reasoning is directed this way, and therefore you may in more complex situations be tempted to give the link between "disease" and "symptom" a wrong direction.

From one day to another, the state of the milk can change. Cows with infected milk will heal over time, and a clean cow has a risk of having infected milk the next day. Now, imagine that the farmer performs the test each day. After a week, he has not only the current test result but also the six previous test results. For each day, we have a model like the one in Figure 2.1. These seven models should be connected such that past knowledge can be used for the current conclusion. A natural way would be

to let the state of the milk yesterday have an impact on the state today. This yields the model in Figure 2.2.

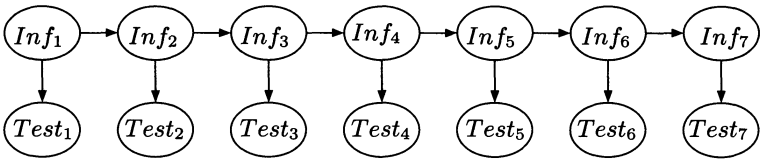

FIGURE 2.2. A seven-day model for milk test.

The model in Figure 2.2 contains a set of hidden assumptions, which can be read from the d-separation properties.

First, the model assumes the *Markov property*: if we know the present, then the past has no influence on the future. In the language of d-separation, the assumption is that, for example, Inf_{i-1} is d-separated from Inf_{i+1} given Inf_i. If we know that the milk on day four is infected, then this can be used to forecast the probability that the milk will be infected on day five. This forecast will not be improved by knowing that the milk was not infected on day three. For various diseases, such an assumption will not be valid. Some diseases have a natural span of time. For example, if I have the flu today but was healthy yesterday, then I will most probably have the flu the day after tomorrow. On the other hand, if I have had the flu for four days, then there is a good chance that I will be cured the day after tomorrow. If the Markov property of Figure 2.2 does not reflect reality, the model should be changed. For example, it may be argued that you also need to go an extra day back, and the model will be as in Figure 2.3.

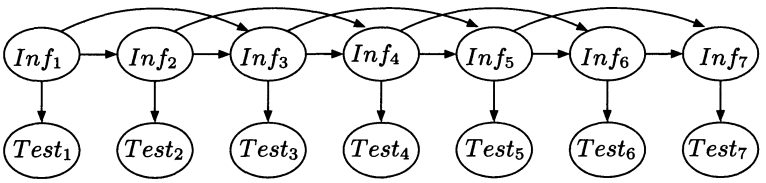

FIGURE 2.3. A seven-day model with a two-day memory of infection.

Notice that although in practice we will never know the state of the infection nodes, it makes a difference whether the memory links are included. In the reasoning, we cannot exploit knowledge of the exact state of the previous infection node, but we may use a probability distribution based on a test result.

The second hidden assumption has to do with the test. Any two test nodes are d-separated given any infection node on the path. This means that the fault probability of the test is independent of whether it was previously correct. In other words, the fact that the test was wrong yesterday has no influence on whether the test will be correct today. If this does not reflect the behavior of the test, you may, for example, include its performance yesterday in the model. This is done in Figure 2.4.

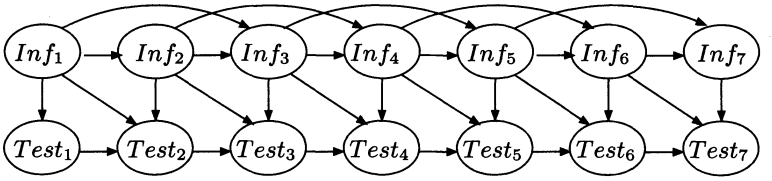

FIGURE 2.4. A seven-day model with two-day memory for infection and a one-day memory of correctness of test.

2.1.2 Cold or angina?

> I wake up in the morning with a sore throat. It may be the beginning of a cold or I may suffer from angina. If it is severe angina, I will not go to work. To gain more insight, I can take my temperature, and I can look down my throat for yellow spots.

Here we have five hypothesis events *Cold?* {*no, yes*} and *Angina?* {*no, mild, severe*}. The hypothesis events must be organized into a set of variables with mutually exclusive states. We may use the variables indicated previously, but we may also use only one variable *Sick?* with states {*no, cold, mild angina, severe angina*}. In the latter case, suffering from both cold and angina is excluded as a possibility. We choose to use the two variables *Cold?* and *Angina?*.

The information variables are *Sore Throat?* {*no, yes*}, *See Spots?* {*no, yes*}, and *Fever?* {*no, low, high*}. The variable *Fever?* causes a problem because it really is continuous. In Section 2.3.8, we give a method on how to transform a continuous variable to a finite one.

Now it is time to consider the causal structure between the variables. We need not worry about how information is transmitted through the network. The only thing to worry about is which variables have a direct causal impact on other variables.

In this example, we have that *Cold?* has a causal impact on *Sore Throat?* and *Fever?* while *Angina?* has an impact on all information variables. The model is given in Figure 2.5.

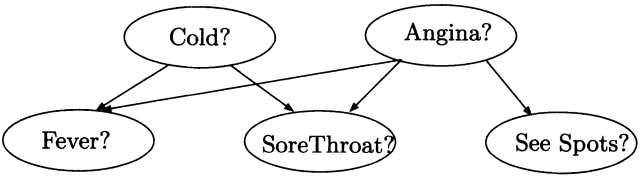

FIGURE 2.5. A model for *Cold or angina?*.

The next thing to check is whether the conditional independencies laid down in the model correspond with reality. The model in Figure 2.5 yields, for example, that if we know the state of *Angina?*, then seeing spots will not have an impact on the expectation either for *Fever?* or for *Sore Throat?*. If we do not agree, we may introduce a link from *See Spots?* to, for example, *Fever?*. For now, we will accept the conditional independencies given by the model.

2.1.3 Insemination

> Six weeks after insemination of a cow, there are two tests for the result: blood test (BT) and urine test (UT).

Following the method from Section 2.1.1, we construct a model as in Figure 2.6, where Pr represents a possible pregnancy.

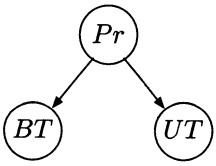

FIGURE 2.6. A model for pregnancy.

Next, we will analyze the conditional independencies stated by the model. We ask the expert whether it is correct that the outcomes of the two tests are independent given Pr. More specifically, assume we know that the cow is pregnant. From this, we infer some expectations for the test results. Now, if we get a negative test result from the blood test, will this change our expectation for the urine test? The experts say that it will, and we must conclude that the model is not a proper reflection of reality.

There are several ways to change the model. You might, for example, introduce a link between the two test nodes, but there is no natural direction. To find out what to do, you must study the process more carefully, and it turns out that what the two tests actually do is trace indications of hormonal changes in the cow. A more refined model will involve a variable

Ho reflecting whether hormonal changes have taken place in the cow, and the model will be as in Figure 2.7.

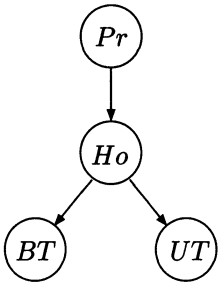

FIGURE 2.7. A more correct model for pregnancy. Both the blood test (BT) and the urine test (UT) measure the hormonal state (Ho).

For the model in Figure 2.7, it does not hold that BT and UT are independent given Pr. The model states that BT and UT are independent given Ho (which should be checked). If the model in Figure 2.6 is used for diagnosing a possible pregnancy, a negative outcome of both the blood test and the urine test will be counted as two independent pieces of evidence and therefore overestimate the probability for the insemination to have failed (see Exercise 2.6).

In the model in Figure 2.7, we have introduced the variable Ho, which is neither a hypothesis variable nor an information variable. Such variables are called *mediating variables*. Mediating variables are often introduced when two variables are not (conditionally) independent as opposed to the situation in the current model. Some standard situations are illustrated in Figure 2.11.

2.1.4 Simple Bayes models

The first Bayesian diagnostic systems were constructed through the following procedure

- Let the possible diseases be collected into one hypothesis variable H with prior probability $P(H)$.
- For all information variables I, acquire the conditional probability distribution $P(I \mid H)$ (the likelihood of H given I).
- For any set of observations $f_1, \ldots, f_n$ on the variables $I_1, \ldots, I_n$, calculate the product $L(H \mid f_1, \ldots, f_n) = P(f_1 \mid H) \cdot P(f_2 \mid H) \cdot \cdots \cdot P(f_n \mid H)$. This product is called the *likelihood* for H given $f_1, \ldots, f_n$. The posterior probability for H is calculated as $\mu P(H) \cdot L(H \mid f_1, \ldots, f_n)$, where μ is a normalization constant.

The preceding calculations reflect the simple model shown in Figure 2.8 (see Exercise 2.9).

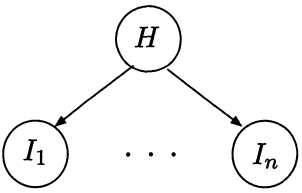

FIGURE 2.8. A simple Bayes model.

The model assumes that the information variables are independent given the hypothesis variable. As can be seen from the insemination example, the assumption need not hold, and if the model is used anyway, the conclusions may be misleading.

2.1.5 A simplified poker game

> In this poker game, each player receives three cards and is allowed two rounds of changing cards. In the first round, you may discard any number of cards from your hand and get replacements from the pack of cards. In the second round, you may discard at most two cards. After the two rounds of card changing, I am interested in an estimate of my opponent's hand.

The hypothesis events are the various types of hands in the game. They may be classified in the following way (in increasing rank): nothing special, 1 ace, 2 of the same value, 2 aces, flush (3 of a suit), straight (3 of consecutive value), 3 of the same value, straight flush. Ambiguities are resolved according to rank. This is, of course, a simplification, but it is often necessary to do so when modeling. The hypothesis events are collected into one hypothesis variable *OH* (opponent's hand) with the preceding classes as states.

The only information to acquire is the number of cards the player discards in the two rounds. By saying this, we are making an approximation again. The information on the cards you have seen is relevant for your opponent's hand. If, for example, you have seen three aces, then he cannot have two aces. Therefore, the information variables are *FC* (first change) with states *0, 1, 2, 3* and *SC* (second change) with states *0, 1, 2*.

A causal structure for the information variables and the hypothesis variable could be as in Figure 2.9. However, this structure will leave us with no clue as to how to specify the probabilities.

What we need are mediating variables describing the opponent's hands in the process: the initial hand *OH0* and the hand *OH1* after the first change of cards. The causal structure will then be as in Figure 2.10.

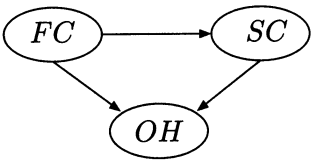

FIGURE 2.9. An oversimplified structure for the poker game. The variables are *FC* (first change), *SC* (second change), and *OH* (opponent's hand).

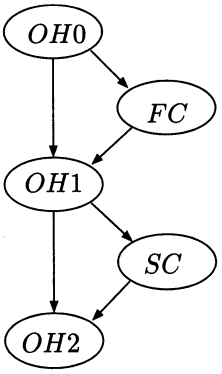

FIGURE 2.10. A structure for the poker game. The two mediating variables *OH0* and *OH1* are introduced. *OH2* is the variable for my opponent's final hand.

To determine the states of *OH0* and *OH1*, we must produce a classification that is relevant for the determination of the states of the children (*FC* and *OH1*, say). We may let *OH0* and *OH1* have the states *nothing special, 1 ace, 2 of consecutive value, 2 of a suit, 2 of the same value, 2 of a suit and 2 of consecutive value, 2 of a suit and 2 of the same value, 2 of consecutive value and 2 of the same value, flush, straight, 3 of the same value, straight flush.*

We defer further discussion of the classification to the section on specifying the probabilities (Section 2.2.3).

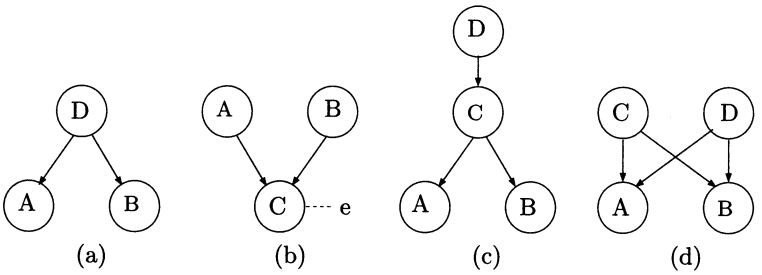

FIGURE 2.11. Examples where an intermediate variable C "solves" undirected dependencies. A and B in examples (a) and (b) are not independent, whereas A and B in examples (c) and (d) are not independent given D.

2.1.6 Causality

In the examples presented in the previous section, there was no problem in establishing the links and their directions. However, you cannot expect this part of the modeling always to go smoothly.

First, causal relations are not always obvious – recall the debate on whether smoking causes lung cancer or whether a person's sex has an impact on his/her ability in the technical sciences. Furthermore, causality is not a well understood concept. Is a causal relation a property of the real world or, rather, is it a concept in our minds helping us to organize our perception of the world? We will not, however, go into the scientific debate on causality and how to discover causal relations. We make one point only, that causality has to do with actions where the state of the world is changed. You may, for example, find yourself confronted with two correlated variables A and B, but you cannot determine a direction. If you observe the state of A, you will change your belief of B and vice versa. A good test then is to imagine that some outside agent *fixes* the state of A. If this does not make you change the belief of B, then A is not a cause of B.

On the other hand, if this imagined test indicates no causal arrow in any direction, then you should look for an event that has a causal impact

on both A and B. If C is such a candidate, then check whether A and B become independent given C (see Figure 2.11).

2.2 Determining the Conditional Probabilities

The basis for the conditional probabilities in a Bayesian network can have a different epistemological status, ranging from well-founded theory over frequencies in a database to subjective estimates. We will give examples of each type.

2.2.1 Milk test

For the milk test in Figure 2.1, we need $P(Infected?)$ and $P(Test \mid Infected?)$. The retailer of the test should provide $P(Test \mid Infected?)$. Any producer of such kinds of tests is supposed to have performed a series of tests yielding the relevant numbers. There are two relevant numbers: the frequency of *false positives*, $P(Test = pos \mid Infected? = no)$, and the frequency of *false negatives*, $P(Test = neg \mid Infected? = yes)$. Let both numbers be 0.01.

The numbers provided by the retailer are not sufficient for the user of the test. In the case of a positive test result, the milk may still be clean, and to come up with a probability we need the prior probabilities $P(Infected?)$.

An estimate of the prior probability would in this case be the daily frequency λ of infected milk for each cow at the particular farm. To estimate λ may be a bit tricky because the farmer may have no experience with actually testing the milk from each specific cow with a perfect test. Assume that this particular farm has 50 cows, and the milk from all cows is poured into a container and transported to the dairy, which tests the milk with a very precise test. The farmer's experience is that on average the dairy reports his milk to be infected once a month.

Now, we must make various assumptions. The first assumption could be that the daily λ is the same for all cows. The next assumption could be that outbreaks of infected milk for the cows in the farm are independent. This yields a coin-tossing model with $P(Infected? = yes) = \lambda$. The information we have is that if we toss fifty coins at the same time, the frequency of at least one of them coming up with $Infected? = yes$ is 1 out of 30, that is, in 29 days out of 30, none of the cows are infected. The probability that all 50 cows are clean is $(1 - \lambda)^{50}$. We have

$$(1 - \lambda)^{50} = \frac{29}{30},$$

which yields the estimate

$$\lambda = 1 - \left(\frac{29}{30}\right)^{0.02} \approx 0.0007.$$

This completes the model, and next you can use a computer system to edit it and to calculate posterior probabilities. The interesting question for this situation is: If we get a positive test result, what is the probability that the milk is infected? This is left as an exercise (see Exercise 2.3).

For the seven-day model in Figure 2.2, we also need $P(Inf_{i+1} \mid Inf_i)$. There are two numbers to estimate: the risk of becoming infected and the chance of being cured. These numbers must be based on experience. For the sake of the example, let the risk of becoming infected be 0.0002 and the chance of being cured 0.3. This gives the numbers in Table 2.1.

		Inf_i	
		yes	no
Inf_{i+1}	yes	0.7	0.0002
	no	0.3	0.9998

TABLE 2.1. $P(Inf_{i+1} \mid Inf_i)$.

For the seven-day model with a two-day memory of infection (Figure 2.3), we need $P(Inf_{i+1} \mid Inf_i, Inf_{i-1})$. If we assume that the risk of being infected is the same as before, that the infection always lasts at least two days, and that after this the chance of being cured is 0.4 each of the following days, then the numbers are as in Table 2.2 (see Exercise 2.8).

		Inf_{i-1}	
		yes	no
Inf_i	yes	0.6	1
	no	0.0002	0.0002

TABLE 2.2. $P(Inf_{i+1} = yes \mid Inf_i, Inf_{i-1})$.

For the seven-day model with two-day memory of infection as well as correctness of test (Figure 2.4), we furthermore need $P(Test_{i+1} \mid Inf_i, Inf_{i+1}, Test_i)$. If we assume that a correct test has a 99.9% chance of being correct next time, and an incorrect test has a 90% risk of also being incorrect next time, we can calculate all required numbers for the four-dimensional table. However, by introducing mediating variables, $Corct_i$, the specification of numbers could be easier, and the tables would be smaller. Figure 2.12 shows how the model could be simplified.

With the preceding assumptions, the required tables are as in Table 2.3.

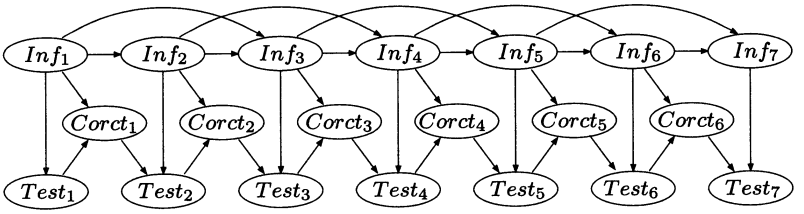

FIGURE 2.12. A seven-day model with a two-day memory for infection and a one-day memory of correctness of test.

		Inf_i	
		yes	no
$Test_i$	yes	1	0
	no	0	1

		$Corct_{i-1}$	
		yes	no
Inf_i	yes	0.999	0.1
	no	0.001	0.9

TABLE 2.3. $P(Corct_i = yes \mid Inf_i, Test_i)$ and $P(Test_i = pos \mid Inf_i, Corct_{i-1})$.

2.2.2 Stud farm

The stallion Brian has sired Dorothy with the mare Ann and sired Eric with the mare Cecily. Dorothy and Fred are the parents of Henry, and Eric has sired Irene with Gwenn. Ann is the mother of both Fred and Gwenn, but their fathers are in no way related. The colt John with the parents Henry and Irene has been born recently; unfortunately, it turns out that John suffers from a life-threatening hereditary disease carried by a recessive gene. The disease is so serious that John is displaced instantly, and as the stud farm wants the gene out of production, Henry and Irene are taken out of breeding. What are the probabilities for the remaining horses to be carriers of the unwanted gene?

The genealogical structure for the horses is given in Figure 2.13.

The only information variable is John. Before the information on John is acquired, he may have three genotypes: he may be sick (aa), a carrier (aA), or he may be pure (AA). The hypothesis events are the genotypes of all other horses in the stud farm.

The conditional probabilities for inheritance are both empirically and theoretically well-studied, and the probabilities are as shown in Table 2.4.

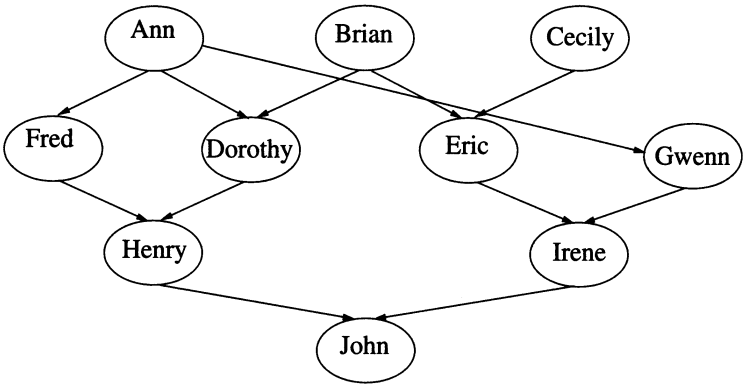

FIGURE 2.13. Genealogical structure for the horses in the stud farm.

	aa	aA	AA
aa	(1, 0, 0)	(0.5, 0.5, 0)	(0, 1, 0)
aA	(0.5, 0.5, 0)	(0.25, 0.5, 0.25)	(0, 0.5, 0.5)
AA	(0, 1, 0)	(0, 0.5, 0.5)	(0, 0, 1)

TABLE 2.4. $P(\text{Child} \mid \text{Father, Mother})$ for genetic inheritance. The numbers (α, β, γ) are the child's probabilities for $(aa,\ aA,\ AA)$.

The inheritance tables could be as in Table 2.4. However, for all horses except John, we have additional knowledge. Since they are in production, they cannot be of type *aa*. A way to incorporate this would be to build a Bayesian network where all inheritance is modeled in the same way and afterward enter the findings that all horses but John are not *aa*. It is also possible to calculate the conditional probabilities directly. If we first consider inheritance from parents that may only be of genotype aA or AA, we get Table 2.5.

	aA	AA
aA	(0.25, 0.5, 0.25)	(0, 0.5, 0.5)
AA	(0, 0.5, 0.5)	(0, 0, 1)

TABLE 2.5. P(Child | Father, Mother) when the parents are not sick.

The table for John is as in Table 2.5. For the other horses, we know that *aa* is impossible. This is taken care of by removing the state *aa* from the distribution and normalizing the remaining distribution. For example, $P(\text{Child} \mid aA, aA) = (0.25, 0.5, 0.25)$, but since *aa* is impossible, we get the distribution $(0, 0.5, 0.25)$, which is normalized to $(0, 0.67, 0.33)$. The final result is shown in Table 2.6.

	aA	AA
aA	(0.67, 0.33)	(0.5, 0.5)
AA	(0.5, 0.5)	(0, 1)

TABLE 2.6. P(Child | Father, Mother) with *aa* removed.

In order to deal with Fred and Gwenn, we introduce the two unknown fathers, I and K, as mediating variables and assume that they are not sick. For the horses at the top of the network, we specify prior probabilities. This will be an estimate of the frequency of the unwanted gene, and there is no theoretical way to derive it. Let us assume that the frequency is such that the prior belief of a horse being a carrier is 0.01.

In Figure 2.14, the final model with initial probabilities is shown, Figure 2.15 gives the posterior probabilities given that John is *aa*, and in Figure 2.16 you can see the posterior probabilities with the prior beliefs at the top changed to 0.0001. Note that the sensitivity to the prior beliefs is very small for the horses where the posterior probability for *carrier* is well beyond 0, for instance in the cases of Ann and Brian.

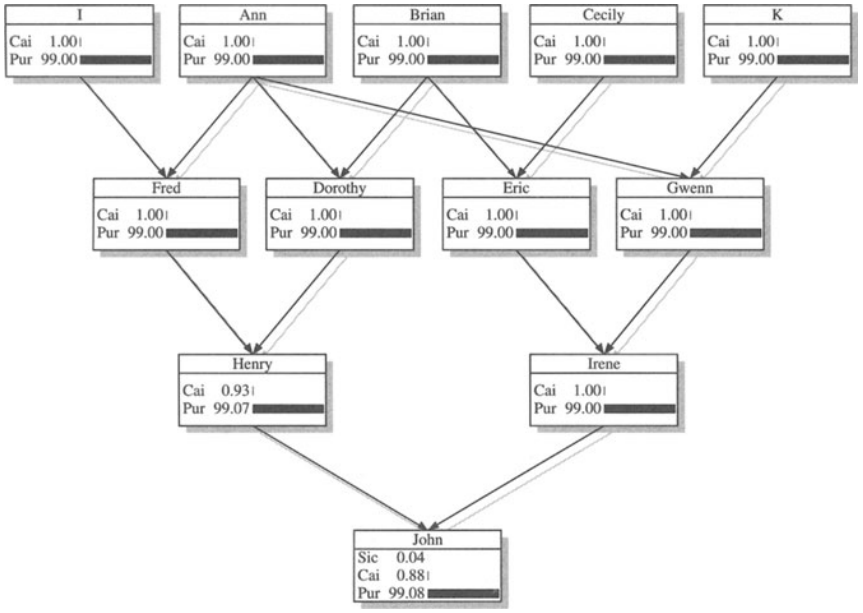

FIGURE 2.14. The stud farm model with initial probabilities.

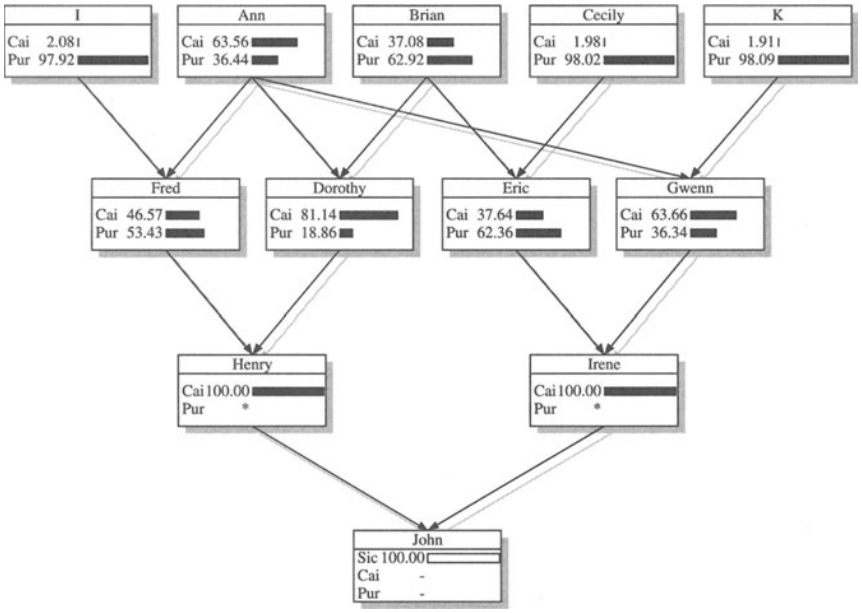

FIGURE 2.15. Stud farm probabilities given that John is sick.

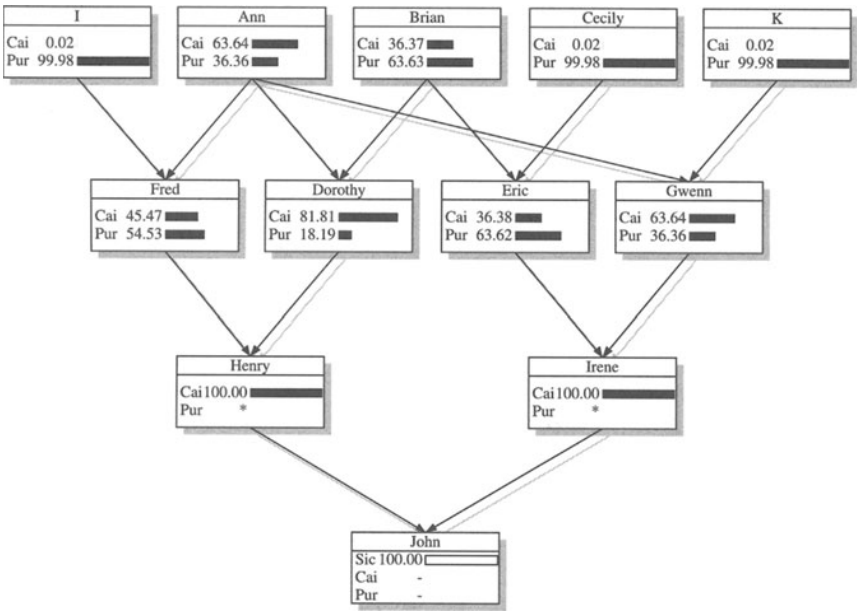

FIGURE 2.16. Stud farm probabilities with prior probabilities for top variables changed to (0.0001, 0.9999).

2.2.3 *Conditional probabilities for the poker game*

In the stud farm example, the conditional probabilities were mainly established through theoretical considerations. This should also be attempted for the model of the poker game developed in Section 2.1.5, but it cannot be carried through entirely.

Consider for example $P(FC \mid OH0)$. It is not possible to give probabilities that are valid for any opponent. It is heavily dependent on the opponent's insight, psychology, and game strategies. We will assume the following strategy:

If nothing special (*no*), then change 3.

If 1 ace (*1 a*), then keep the ace.

If 2 of consecutive value (*2 cons*), 2 of a suit (*2 s*), or 2 of the same value (*2 v*), then discard the third card.

If 2 of a suit and 2 of consecutive value, then keep 2 of a suit. (This strategy could be substituted by a random strategy for keeping either 2 of a suit or 2 of consecutive value.)

If 2 of a suit and 2 of the same value or 2 of consecutive value and 2 of the same value, then keep the 2 of the same value.

If flush (*fl*), straight (*st*), 3 of the same value (*3 v*), or straight flush (*sfl*), then keep it.

Based on the preceding strategy, a logical link between *FC* and *OH0* is established. Note that the strategy makes the states for combined hands redundant. They play no role, and therefore we remove them.

The strategy for $P(SC \mid OH1)$ is the same except that in the case of *no*, only 2 cards are discarded.

The remaining probabilities to specify are $P(OH0), P(OH1 \mid OH0, FC)$, and $P(OH2 \mid OH1, SC)$.

P(OH0)

The states are (*no, 1 a, 2 cons, 2 s, 2 v, fl, st, 3 v, sfl*).

Through various (approximated) combinatorial calculations, the prior probability distribution is found to be $P(OH0) = (0.1672, 0.0445, 0.0635, 0.4659, 0.1694, 0.0494, 0.0353, 0.0024, 0.0024)$

P(OH1 | OH0, FC)

Due to the logical links between *OH0* and *FC*, it is sufficient to consider only nine out of the possible 36 parent configurations, namely (*no, 3*), (*1 a, 2*), (*2 cons, 1*), (*2 s, 1*), (*2 v, 1*), (*fl, 0*), (*st, 0*), (*3 v, 0*), (*sfl, 0*). The last four are obvious. In Table 2.7, the results of approximate combinatorial calculations are given.

		(OH0, FC)				
		(no, 3)	*(1 a, 2)*	*(2 cons, 1)*	*(2 s, 1)*	*(2 v, 1)*
	no	0.1583	0	0	0	0
	1 a	0.0534	0.1814	0	0	0
	2 cons	0.0635	0.0681	0.3470	0	0
	2 s	0.4659	0.4796	0.3674	0.6224	0
OH1	*2 v*	0.1694	0.1738	0.1224	0.1224	0.9592
	fl	0.0494	0.0536	0	0.2143	0
	st	0.0353	0.0383	0.1632	0.0307	0
	3 v	0.0024	0.0026	0	0	0.0408
	sfl	0.0024	0.0026	0	0.0102	0

TABLE 2.7. $P(OH1 \mid OH0, FC)$ for the nonobvious parent configurations.

The probabilities for the remaining parent configurations may be whatever is convenient, so put, for example, $P(OH1 \mid 3\ v,\ 1) = (1, 0, \cdots, 0)$.

P(OH2 | OH1, SC)

First, a table $P(OH2' \mid OH1, SC)$ similar (but not identical in the numbers) to Table 2.7 can be calculated. However, the states of $OH2'$ are not

the ones we are interested in. We are interested in the *value* of the hand, and a state such as *2 cons* is of no value unless one of them is an ace. Therefore, the probabilities for the states of $OH2'$ are transformed to probabilities for *OH2*. For the transformation, the following rules are used:

$$1\ a = 1\ a + \frac{1}{6}(2\ cons + 2\ s),$$

$$no = no + \frac{5}{6}(2\ cons + 2\ s).$$

The probabilities of *2 a* are calculated specifically. The resulting probabilities are given in Table 2.8.

		(OH1, Sc)				
		(no, 2)	*(1 a, 2)*	*(2 cons, 1)*	*(2 s, 1)*	*(2 v, 1)*
	no	0.5613	0	0.5903	0.5121	0
	1 a	0.1570	0.2425	0.1181	0.1024	0
	2 v	0.1757	0.0667	0.1154	0.1154	0.8838
OH2	*2 a*	0.0055	0.1145	0.0096	0.0096	0.0736
	fl	0.0559	0.0559	0	0.2188	0
	st	0.0392	0.0392	0.1666	0.0313	0
	3 v	0.0027	0.0027	0	0	0.0426
	sfl	0.0027	0.0027	0	0.0104	0

TABLE 2.8. $P(OH2 \mid OH1, SC)$ for the nonobvious configurations.

Using a model such as the one in Figure 2.10 and with the conditional probability tables specified in this section, we have established a model for assisting a (novice) poker player. However, if my opponent knows that I use the system, he may choose to change his strategies. His goal is to win rather than to obtain good hands, and he therefore may choose a strategy that makes me overestimate his hand. For instance, it seems a good strategy to discard two cards instead of three in the case of *no*. I will be convinced that he has an ace, and his chances for a good hand are not substantially reduced. We will return to this point in Chapter 4 on decision making.

2.2.4 Transmission of symbol strings

A language L over 2 symbols $(\mathbf{a}, \mathbf{b})$ is transmitted through a channel. Each word is surrounded by the delimiter symbol c. In the transmission some characters may be corrupted by noise and be confused with others.

A five-letter word is transmitted. Give a model that can determine the probabilities for the transmitted symbols given the received symbols.

There are five hypothesis variables $T_1, \ldots, T_5$ with states a, b and five information variables $R_1, \ldots, R_5$ with states a, b, c. Mediating variables for the delimiters before and after the word may also be considered. There is a causal relation from T_i to R_i. Furthermore, there may also be a relation from T_i to $T_{i+1} (i = 1, \ldots, 4)$. You could also consider more involved relations from pairs of symbols to symbols, but for now we refrain from doing that. The structure is given in Figure 2.17.

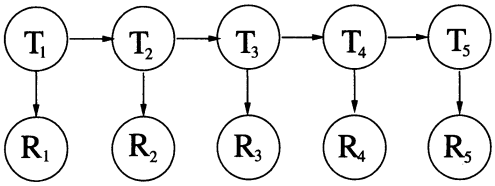

FIGURE 2.17. A model for symbol transmission. T_i are the symbols transmitted; R_i are the symbols received.

The conditional probabilities can be established through experience. The probabilities $P(R_i \mid T_i)$ will be based on statistics describing the frequencies of confusion. Let Table 2.9 be the result.

	$T = a$	$T = b$
$R = a$	0.80	0.15
$R = b$	0.10	0.80
$R = c$	0.10	0.05

TABLE 2.9. $P(R \mid T)$ under transmission.

You may obtain the probabilities $P(T_{i+1} \mid T_i)$ by investigating the five-letter words in L. What is the frequency of the first letter? What is the frequency of the second letter given that the first letter is **a**? You continue to do this for each letter. You can refine this frequency analysis by also taking the frequencies of the words into consideration. Let Table 2.10 be the result of a frequency analysis.

First 2 letters	Last 3 letters							
	aaa	**aab**	**aba**	**abb**	**baa**	**bab**	**bba**	**bbb**
aa	0.017	0.021	0.019	0.019	0.045	0.068	0.045	0.068
ab	0.033	0.040	0.037	0.038	0.011	0.016	0.010	0.015
ba	0.011	0.014	0.010	0.010	0.031	0.046	0.031	0.045
bb	0.050	0.060	0.056	0.057	0.016	0.023	0.015	0.023

TABLE 2.10. Frequencies of five-letter words in L. The word **abaab**, for example, has frequency 0.040.

You can calculate the required probabilities from Table 2.10 using the fundamental rule. The prior probabilities for T_1 are (0.5, 0.5), and $P(T_2, T_1)$ is achieved by adding the elements in each row. Table 2.11 gives two conditional probabilities.

	a	b
a	0.6	0.4
b	0.4	0.6

$P(T_2 \mid T_1)$

	a	b
a	0.24	0.74
b	0.76	0.26

$P(T_3 \mid T_2)$

TABLE 2.11. Two conditional probabilities for five-letter words in L.

An alternative model would be to have a hypothesis variable, *Word*, with 32 states and with Table 2.10 as prior probabilities (see Figure 2.18).

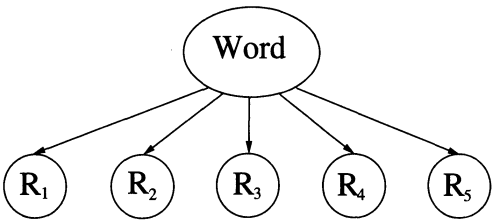

FIGURE 2.18. An alternative model for symbol transmission. *Word* is the set of possible transmitted words.

This is manageable because of the small amount of five-letter words over $\{\mathbf{a}, \mathbf{b}\}$; but if the alphabet had 24 symbols, and if six-letter words were considered, the number of states in *Word* would become intractably large. On the other hand, the model of Figure 2.17 may be too simple to catch the dependencies in Table 2.10, so the task really is to analyze the table in order to find the simplest structure describing it. There are methods for doing this, and we will return to this topic in Section 3.2.

2.2.5 Cold or angina?

The estimation of the conditional probabilities for the example introduced in Section 2.1.2 has a very subjective flavor based on my own experience with colds and anginas. I will estimate the following probabilities: $P(Cold?), P(Angina?), P(Sore\ Throat? \mid Cold?, Angina?), P(Fever? \mid Cold?, Angina?), P(See\ Spots? \mid Angina?)$.

Because in the morning I do not recall having been chilly yesterday, the prior probabilities $P(Cold?)$ and $P(Angina?)$ are my subjective recollections of how often I wake up in the morning with a cold or with angina.

Because cold is more frequent than angina, I put $P(Cold?) = (0.97, 0.03)$ and $P(Angina?) = (0.993, 0.005, 0.002)$, (the order of the states are taken from Section 2.1.2).

Without angina or with mild angina, I will not see spots. With severe angina, I would expect to see spots, but I may not. I put $P(See\ Spots? \mid Angina? = severe) = (0.1, 0.9)$.

P(Sore Throat? | Cold?, Angina?)

If I suffer from neither a cold nor angina, I have a background probability 0.05 of having a sore throat in the morning. A cold as well as angina may give me a sore throat. If I only have a cold, the probability of a sore throat is 0.4. If I have mild angina, the probability of a sore throat is 0.7, and in the case of severe angina, I will certainly have a sore throat. What if I have both a cold and mild angina? I do not have sufficient experience to come up with a reliable estimate. Instead, I can use the two conditional probabilities from before: out of 100 mornings, I will wake up five mornings with a "background produced" sore throat. Out of the remaining 95 mornings, the cold yields a sore throat in 40% of them, that is, 38 mornings. Out of the remaining 57 mornings, mild angina will cause a sore throat in 70% of them: 39.9 mornings. In total, if I have both mild angina and a cold, I will have a sore throat in 82.9 mornings out of 100. The number 82.9 indicates an unjustified precision, and we set the probability to 0.85. In Section 2.3.2 on "noisy or", we give a systematic treatment of this method of estimating probabilities. The full table for $P(Sore\ Throat? \mid Cold?, Angina?)$ is given in Table 2.12. It is left as an exercise to complete the model.

	Angina? = no	*Angina? = mild*	*Angina? = severe*
Cold? = no	0.05	0.7	1
Cold? = yes	0.4	0.85	1

TABLE 2.12. $P(Sore\ Throat? = yes \mid Cold?, Angina?)$.

2.2.6 Why causal networks?

As mentioned previously, the structure of a Bayesian network need not reflect cause–effect relations. The only requirement is that the d-separation properties of the network hold for the domain modeled. There is, however, good reason to strive for causal networks. The model in Figure 2.19 can be used to illustrate some of the points. We have a disease *Dis* and two tests, *Ts* and *Tt*.

When diagnosing, you usually reason opposite to the directions of the arrows in Figure 2.19, and trained physicians are usually inclined to provide conditional probabilities in the diagnostic direction. A model might look like the one in Figure 2.20 (a).

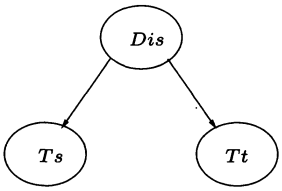

FIGURE 2.19. A model for a disease with two tests.

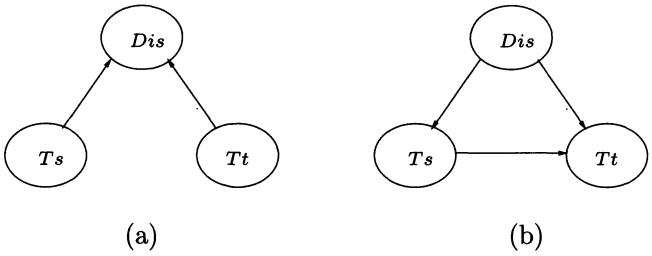

FIGURE 2.20. Diagnostic models for the situation in Figure 2.19: (a) with a wrong independence, (b) with no (conditional) independence.

The model in Figure 2.20 (a) is not correct. According to this model, Ts and Tt are independent, and there is no way to correct it by specifying the potentials in a sophisticated manner. To correct the model, you must add some extra structure. You may, for example, introduce a link from Ts to Tt, as is done in Figure 2.20 (b). Therefore, to get a correct model, it is not sufficient to acquire $P(Dis \mid Ts, Tt)$ together with the "priors" $P(Ts)$ and $P(Tt)$. This also illustrates another point, namely that a correct model of a causal domain is minimal with respect to links. In other words, if for some reason you wish to represent a causal relation with a link directed opposite to the causal direction, then the total number of links will not decrease.

The model in Figure 2.19 has another advantage over the models in Figure 2.20, namely that the conditional probabilities $P(Ts \mid Dis)$ and $P(Tt \mid Dis)$ are more stable than the conditional probabilities specified for the models in Figure 2.20. The conditional probabilities for Figure 2.19 reflect general properties of the relation between diseases and tests, and they are the ones that a manufacturer of tests can publish, whereas the conditional probabilities for Figure 2.20 are a mixture of disease–test relations and prior frequencies of the disease.

It may happen that it is not possible to acquire the conditional probabilities for a correct model, but instead other types of conditional probabilities are available. Assume, for example, that for the model in Figure 2.19 we can only acquire the potentials $P(Dis \mid Ts)$, $P(Dis \mid Tt)$, $P(Ts)$, and $P(Tt)$. Using Bayes' rule on $P(Dis \mid Ts)$ and $P(Ts)$, we get $P(Dis)$ and $P(Ts \mid Dis)$. The same can be done with $P(Dis \mid Tt)$ and $P(Tt)$. If the

two calculations of $P(Dis)$ give the same result, we have the required potentials. If, on the other hand, the two calculations disagree, there is no safe way to solve the conflict. It can happen in many different situations that you have a set of potentials, but the model requires another set and there is no safe way of inferring the needed potentials. It is a lively area of research to construct engineering methods for getting the best out of what you have.

In Chapter 4, we deal with *interventions*. They provide another good reason for constructing causal models. An intervention is an action that has an impact on the state of particular variables. The impact of an intervention will spread in the causal direction and not opposite to the causal direction. If the model does not reflect causal directions, it cannot be used to simulate the impact of interventions.

2.3 Modeling Methods

Much scepticism of Bayesian networks stems from the question of where do the numbers come from?. As shown in the previous section, they come from many different sources. If you are building a model over a domain where experts actually *do* take decisions based on estimates, why should you not be able to make your Bayesian network estimate at least as well as the experts? You can, for example, use the technique described in Section 1.3.3 to acquire the probabilities from the experts. The acquisition of numbers is, of course, not without problems, and in this section we give some methods that can help you in this job. Also, we provide some modeling tricks.

2.3.1 Undirected relations

It may happen that the model must contain dependence relations among variables A, B, C, say, but it is neither desirable nor possible to attach directions to them. (In that case, the model is called a *chain graph*. A chain graph is an acyclic graph with both directed and nondirected links, where *acyclic* means that all cycles consist of only nondirected links.) The relation may, for example, be a description of possible configurations. This difficulty may be overcome by using conditional dependence as described in Section 1.2.1 (converging influence).

Let $R(A, B, C)$ describe the relation in numbers from [0,1]. Add a new variable D with two states y and n and let A, B, C be parents of D (see Figure 2.21).

Let $P(D = y \mid A, B, C) = R(A, B, C); P(D = n \mid A, B, C) = 1 - R(A, B, C)$ and enter the evidence $D = y$.

Example I have washed two pairs of socks in the washing machine. The washing has been rather hard on them, so they are now difficult to

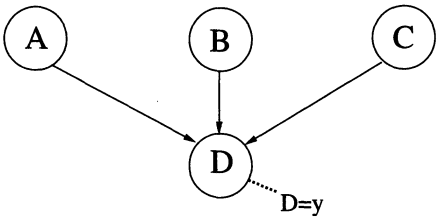

FIGURE 2.21. A way to introduce undirected relations among A, B, and C.

distinguish. However, it is important for me to pair them correctly. To classify the socks, I have pattern and color. A classification model may be like the one in Figure 2.22. The variables S_i have states t_1 and t_2 for the two types, the variables P_i have two pattern types, and the variables C_i have two color types. The constraint that there are exactly two socks of each type is described in Table 2.13.

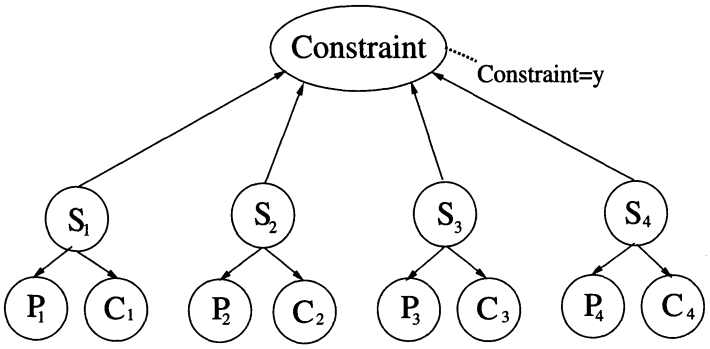

FIGURE 2.22. A model for classifying pairs of socks.

S_1	t_1	t_1	t_1	t_1	t_1	t_1	t_1	t_1	t_2	t_2	t_2	t_2	t_2	t_2	t_2	t_2
S_2	t_1	t_1	t_1	t_1	t_2	t_2	t_2	t_2	t_1	t_1	t_1	t_1	t_2	t_2	t_2	t_2
S_3	t_1	t_1	t_2	t_2	t_1	t_1	t_2	t_2	t_1	t_1	t_2	t_2	t_1	t_1	t_2	t_2
S_4	t_1	t_2	t_1	t_2	t_1	t_2	t_1	t_2	t_1	t_2	t_1	t_2	t_1	t_2	t_1	t_2
P	0	0	0	1	0	1	1	0	0	1	1	0	1	0	0	0

TABLE 2.13. The table for $P = P(\text{Constraint} = y \mid S_1, S_2, S_3, S_4)$; t_1 and t_2 are the two states of S_1, S_2, S_3, S_4.

The situation is more subtle if the relation $R(A, B, C)$ is of probabilistic nature. If A, B, and C have no parents, $R(A, B, C)$ can be a joint probability table. On the other hand, if A has a parent, then it is not obvious what $R(A, B, C)$ represents. We will not deal with this problem but refer the reader to the literature on chain graphs.

2.3.2 *Noisy or*

When a variable A has several parents, you must specify $P(A \mid c^*)$ for each configuration c^* of the parents. If you take the distributions from a database, the number of cases for each configuration may become too small. Also, the configurations may be too specific for any expert. You may also be in the situation that you have reasonable estimates of $P(A \mid B)$ and $P(A \mid C)$, but you require $P(A \mid B, C)$. Then, you should look for assumptions that reduce the number of distributions to specify.

Consider in Section 2.2.5 the conditional probability $P(\textit{Sore Throat?} \mid \textit{Cold?, Angina?})$. It was possible to get estimates of $P(\textit{Sore Throat?} \mid \textit{Cold?})$ and $P(\textit{Sore Throat?} \mid \textit{Angina?})$, but is there a general way to describe how they then combine into $P(\textit{Sore Throat?} \mid \textit{Cold?, Angina?})$? The following is a way of describing it.

There are three events causing me to have a sore throat in the morning:

- the "background event," which in 5% of the mornings yields a sore throat;
- *cold*, which causes a sore throat with probability 0.4;
- *angina*, which when *mild* causes a sore throat with probability 0.7.

The preceding uncertainty can be interpreted as follows. If any of the causes are present, then I have a sore throat unless something has prevented it. In other words, if I have mild angina, then I have a sore throat unless some other circumstances prevent it, and there is a 30% chance that it is prevented. In the same way, there is a 60% chance that some inhibitor prevents me from having a sore throat although I have a cold, and the background event is prevented with probability 0.95.

Now, if we assume that the preventing factors are independent, then the combined probabilities are easy to calculate as one minus the product of the appropriate probabilities for the inhibitors (note that the background event is always a fact). The probabilities are given in Table 2.14.

	Angina? = no	*Angina? = mild*	*Angina? = severe*
Cold? = no	0.05	$1 - 0.95 \cdot 0.3$	1
Cold? = yes	$1 - 0.95 \cdot 0.6$	$1 - 0.95 \cdot 0.3 \cdot 0.6$	1

TABLE 2.14. Calculation of $P(\textit{Sore Throat? = yes} \mid \textit{Cold?, Angina?})$. Note that some numbers are slightly different from the corresponding numbers in Table 2.12.

The preceding construction is an example of the simplifying assumption called a *noisy or*.

Let $A_1, \ldots, A_n$ be binary variables listing all the causes of the binary variable B. Each event $A_i = y$ causes $B = y$ unless an *inhibitor* prevents it, and the probability for that is q_i (see Figure 2.23).

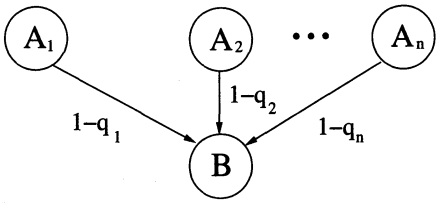

FIGURE 2.23. The general situation for noisy or. q_i is the probability that the impact of A_i is inhibited.

In other words, $P(B = n \mid A_i = y) = q_i$. We assume that *all inhibitors are independent.* Then, $P(B = n \mid A_1, A_2, \ldots, A_n) = \prod_{j \in Y} q_j$, where Y is the set of indices for variables in the state y. For example,

$$\begin{aligned} P(B = y \mid A_1 = y, A_2 = y, A_3 = \cdots = A_n = n) \\ &= 1 - P(B = n \mid A_1 = y, A_2 = y, A_3 = \cdots = A_n = n) \\ &= 1 - q_1 \cdot q_2. \end{aligned}$$

By assuming "noisy or," the number of probabilities to estimate grows linearly with the number of parents.

Note 1 We require $P(B = y \mid A_1 = \cdots = A_n = n)$ to be 0. This may seem to restrict the applicability of the approach. However, as in the preceding example, if $P(B = y) > 0$ when none of the causal events in the model are on, then introduce a background event that is always on.

Note 2 The complementary construction to noisy or is called *noisy and.* A set of causes shall all be on in order to have an effect. However, the causes have random inhibitors, which are mutually independent.

Note 3 Noisy or can be modeled directly without performing the calculations (see Figure 2.24). This highlights the assumptions behind the noisy or gate. If a cause is on, then its effect may be prevented by an inhibitor, and the probabilities for the inhibitors to be present are independent.

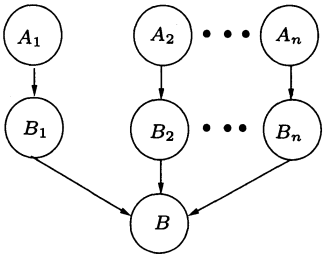

FIGURE 2.24. Direct modeling of a noisy or gate. $P(B_i \mid A_i)$ is the original $P(B \mid A_i)$, and $P(B \mid B_1, \ldots, B_n)$ is logical or.

2.3.3 Divorcing

Let $A_1, \ldots, A_n$ be a list of variables all of which are causes of B. If you wish to specify $P(B \mid A_1, \ldots, A_n)$, you might have a very large knowledge acquisition task ahead of you. Either you need to ask the experts on the distribution of B given very specific parent configurations or, if the table must be extracted from databases, you need a very large set of cases. The following example illustrates the problem.

Granting a loan

A bank will decide on a mortgage loan for a customer who wishes to purchase a house. The customer is asked to fill in a form giving information on various financial and personal matters together with various key information on the house. The answers are used to estimate the probability that the bank will get its money back.

The information can be the following: type of job, yearly income, other financial commitments, number and types of cars in the family, number of previous addresses during the last five years, number of children in the family, number of divorces, size and age of the house, price of the house, and type of environment.

In principle, each slot in the form represents a variable with a causal impact on the variable *Money back?*. If we assume each parent variable to have five states, we have already listed a parent space with $5^{11} \approx 5{,}000{,}000$ configurations. For each configuration, we request a distribution for A. No person can estimate that number of distributions, nor can he or she estimate a distribution for a divorced businesswoman with a yearly income of \$50,000, having loans of \$70,000 already, one car, three previous addresses, two children, wanting to purchase a twenty-year-old house of 150 m^2 at the price of \$200,000 in a farming area. Also, if the distributions are to be taken from a database, the bank will need at least 50,000,000 cases that may not be more than 10 years old.

To handle this kind of task, we *divorce* the parents as was done in Section 2.2.1. The set of parents $A_1, \ldots, A_i$ for B is divorced from the parents $A_{i+1}, \ldots, A_n$ by introducing a mediating variable C, making C a child of $A_1, \ldots, A_i$ and a parent of B (see Figure 2.25).

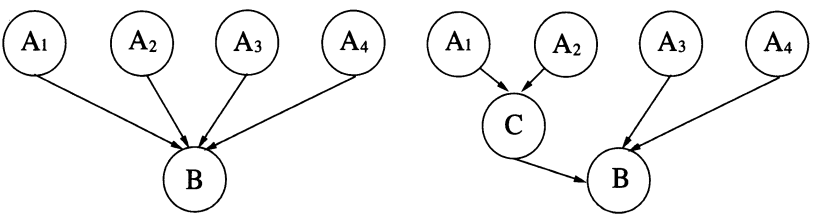

FIGURE 2.25. A_1 and A_2 are divorced from A_3 and A_4 by introducing the variable C.

The assumption behind divorcing is the following (with reference to Figure 2.25).

The set of configurations (A_1, A_2) can be partitioned into the sets $c_1 \ldots c_m$ such that whenever two configurations (a_1, a_2) and (a_1', a_2') are elements in the same c_i, then $P(B \mid a_1', a_2', A_3, A_4) = P(B \mid a_1, a_2, A_3, A_4)$. The divorcing variable then has $c_1 \ldots c_m$ as states.

In the example of granting a loan, it will be impossible to perform an analysis as before, and you will group the variables based on another type of insight into the domain. For example, the variables in the house can be grouped and given a common child variable describing how safe the mortgage will be, the financial variables may be grouped for a variable describing the applicant's financial abilities, and the remaining variables describe the applicant's stability.

In connection with the example of granting a loan, it should be noted that if we only want to perform a classification, then we need not build a Bayesian network. Other techniques such as statistical classifiers or neural networks may be more adequate. However, if we also wish to calculate decision recommendations, we will need the posterior probabilities provided by a Bayesian network. We will deal further with this in Chapter 4.

2.3.4 Noisy functional dependence

There are ways of directing the divorcing. "Noisy or" and "noisy and" are examples of a general method called *noisy functional dependence.*

Headache

Headache (Ha) may be caused by fever (Fe), hangover (Ho), fibrositis (Fb), brain tumor (Bt), and other causes (Ot), and you may choose to soothe it with aspirin ($As.$)(We ignore the effect aspirin has on fever.) Let Ha have the states *no, mild, moderate, severe.* The various causes support each other in the effect. If, for example, $Ho = y$ or $Fb = y$ are present, then they may yield a *mild* Ha, but if both are present, then the Ha would be *moderate.* Furthermore, if also $As = y$, then Ha may drop to *no* or *mild.* Although the various parents of Ha combine in a rather involved manner, we still have the feeling that the impacts of the causes are independent. This kind of independence can be described as follows: if the headache is at level l, and we add an extra cause for headache, then the result is a headache at level q independent of how the initial state has been caused.

Assume that we can estimate conditional probabilities of type $P(Ha \mid C)$, and we want to combine the effects of the various causes. For this, we can imagine that we attach a number to the states of Ha: *no* 0, *mild* 1, *moderate* 2, *severe* 4, and the "adding up" of the effects consists in adding the numbers. A model could be similar to the one in Figure 2.26.

The hidden assumption behind this method of adding up is that the effect from any cause is independent of the current state of headache, and

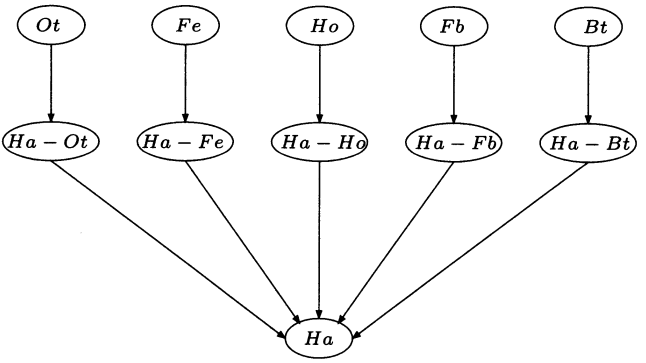

FIGURE 2.26. A model for causes of headache. The bottom node adds up the effects.

it is faithfully reflected in the numbers attached to the headache states. To make it explicit in the model, we can give each headache node a child with numbers as states, these nodes are given a common child that adds the numbers, and a new node translates the numbers to Ha states (see Figure 2.27).

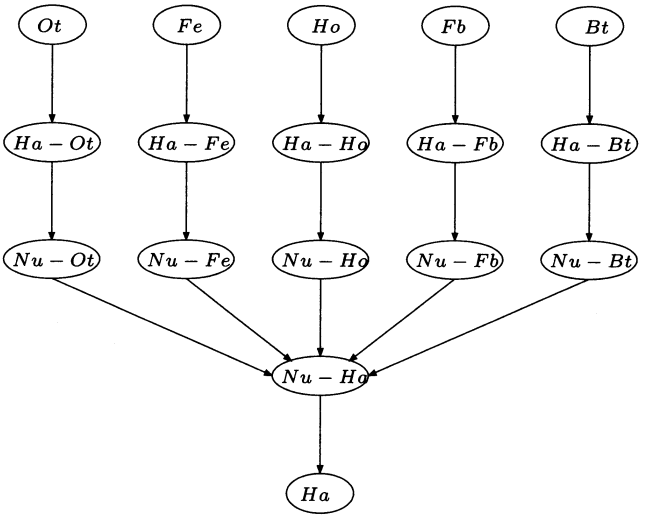

FIGURE 2.27. A model that adds the headache states by transforming to numbers, adding and transforming back to headache states again.

Now, for $P(Nu{-}Ha \mid Nu{-}Ot, Nu{-}Fe, Nu{-}Ho, Nu{-}Fb, Nu{-}Bt)$ we can perform divorcing, we can add one number at a time (see Figures 2.28 and 2.29), or we can represent the function in any other kind of compact way.

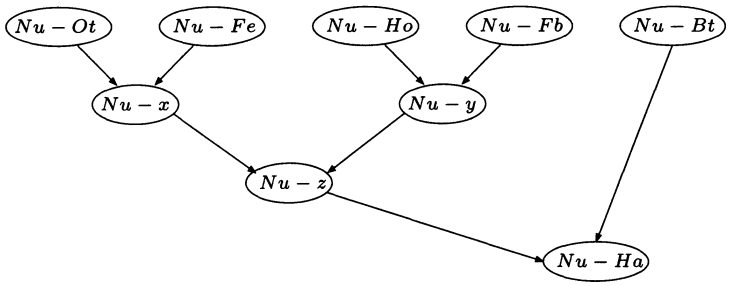

FIGURE 2.28. The adder represented through divorcing.

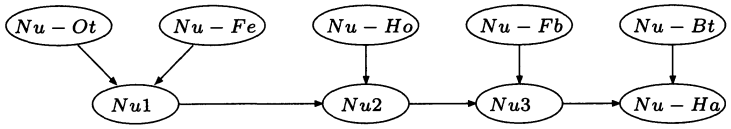

FIGURE 2.29. The adder represented through adding one number at a time.

The effect of aspirin can be included in two different ways. Either it subtracts a number from the sum or it has a direct effect on the headache state.

2.3.5 *Time-stamped models*

Working with domains that evolve over time, you can introduce a discrete time stamp and have a model for each unit of time. We call such a local model a *time slice.*

Consider, for example, the model for infected milk in Figure 2.30.

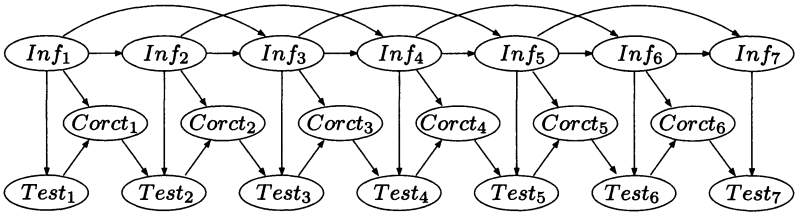

FIGURE 2.30. A seven-day model with two-day memory for infection as well as correctness of test.

For each time slice i, you have three variables $Inf_i, Test_i$, and $Corct_i$. The three variables are connected in a time slice, as depicted in Figure 2.31.

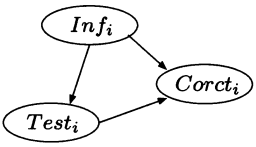

FIGURE 2.31. A time slice for infected milk.

The time slices are connected through *temporal links* to constitute a full model. If the structure of the time slices is identical, and if the temporal links are the same, we say that the model is a *repetitive temporal model.* If the conditional probabilities also are identical, we call the model *strictly repetitive.*

Special kinds of time-stamped models are the *hidden Markov models.* They are stricly repetitive models with an extra assumption (the Markov property): the past has no impact on the future given the present.

The model in Figure 2.30 is not a hidden Markov model because influence from Inf_{i-1} may flow to Inf_{i+1} no matter our knowledge of time slice i. The model can be transformed to a hidden Markov model by introducing a copy Inf'_i of Inf_{i-1} in the ith time slice (see Figure 2.32).

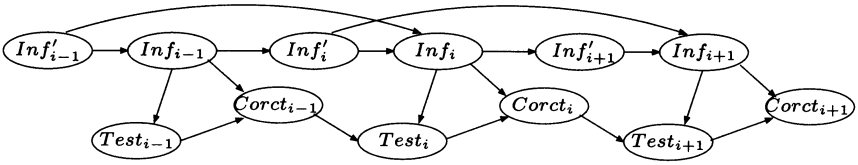

FIGURE 2.32. The model of Figure 2.30 transformed to a hidden Markov model.

The reason for the term *hidden* Markov model is that under the surface (the test results) there is a hidden activity which cannot be observed (the infections).

From a modeling point of view, it is quite straightforward to work with time-stamped models. However, they will often yield calculation problems (see Exercise 2.20 and Chapter 5).

The time used in temporal models need not be real time. The model for transmission of symbols in Section 2.2.4 can be considered a temporal repetitive model. It is not a hidden Markov model because the conditional probabilities are not identical.

A *Kalman filter* is a hidden Markov model where exactly one variable has relatives outside the time slice. The model in Figure 2.2 is a Kalman filter. A *Markov chain* is a Kalman filter consisting of exactly one variable in each time slice. Note that a hidden Markov model can be transformed to a Markov chain by taking the cross product of all variables in each time slice.

When modeling domains that are evolving over time, there is a distinction between *finite horizon* and *infinite horizon* domains. The infected milk problem is an infinite horizon domain, and a typical finite horizon domain is a corn field from sowing to harvest.

Specifying a repetitive temporal model can be eased by introducing a couple of new features to the specification language. Apart from the structure of a time slice, you must specify the number of time slices and the temporal links. The number of slices can be written in a special box, and you can introduce a special kind of arrow to specify temporal links. A number attached to a temporal link can specify the number of time steps to jump (if no number is specified, the link goes from slice i to slice $i+1$). In Figure 2.33, we have used an extended specification language for the model in Figure 2.30.

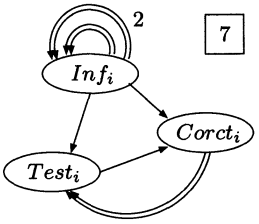

FIGURE 2.33. A compact specification of the model in Figure 2.30 (an extension of Figure 2.31). The $\rightrightarrows$ indicates a temporal link. The number "2" attached to one of them specifies that it jumps two time steps (no number attached means a jump from slice i to slice $i+1$).

These features are sufficient for specifying strictly repetitive models. If the conditional probabilities vary over the time slices, you will need further language features.

2.3.6 Expert disagreements

It may happen that we are in a situation where the experts disagree on the conditional probabilities for a model. Consider the model in Figure 2.34, and assume that we have three experts who agree on $P(B)$ and $P(C \mid A)$, but they disagree on $P(A)$ and $P(D \mid B, C)$. For the three experts, we have $P(A = y) = (0.1, 0.3, 0.4)$, and the tables for $P(D \mid B, C)$ can be seen in Table 2.15.

		B	
		y	n
C	y	(0.4, 0.4, 0.6)	(0.7, 0.9, 0.7)
	n	(0.6, 0.4, 0.5)	(0.9, 0.7, 0.9)

TABLE 2.15. $P(D = y \mid B, C)$ for the three different experts s_1, s_2, s_3.

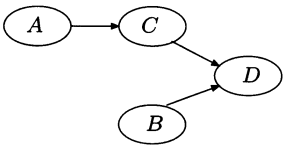

FIGURE 2.34. A model with expert disagreements. All variables are binary.

If you have equal confidence in the three experts, you can take the mean of the three numbers. If your confidence in the experts varies, you may incorporate this and calculate a weighted average. For example, you may give the first two experts a confidence weight 1 and the third expert a confidence weight 2. Because the total confidence weight is 4, you get a confidence distribution (0.25, 0.25, 0.5), and for A you have $P(A = y) = 0.25 \cdot 0.1 + 0.25 \cdot 0.3 + 0.5 \cdot 0.4 = 0.3$. $P(D \mid B, C)$ is shown in Table 2.16.

		B	
		y	n
C	y	0.5	0.75
	n	0.5	0.85

TABLE 2.16. $P(D = y \mid B, C)$ weighted with confidence distribution (0.25, 0.25, 0.5).

The confidence distribution can be incorporated into the model by introducing a variable S with states s_1, s_2, and s_3. S has a link to the nodes, the tables of which the three experts disagree upon (see Figure 2.35).

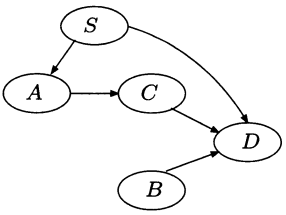

FIGURE 2.35. The model from Figure 2.34 with the experts represented explicitly by the node S.

S is given the confidence distribution (0.25, 0.25, 0.5) as before, and the child variables have a conditional probability table for each expert. The table $P(D = y \mid B, C, S)$ is as in Table 2.15.

By modeling the different expert opinions explicitly, you have prepared the model for *adaptation*. Whenever you have a case with evidence e entered into the model, you will get $P(S \mid e)$, which is an indication from the case telling which expert to believe, that is, you get a new confidence distribution

		S_i		
		s_1	s_2	s_3
S_{i+1}	s_1	1	0	0
	s_2	0	1	0
	s_3	0	0	1

TABLE 2.17. $P(S_{i+1} \mid S_i)$ for the model in Figure 2.36.

that can be used for the next case. This adaptation process can be modeled as a Kalman filter model (see Figure 2.36), where time is counted in cases.

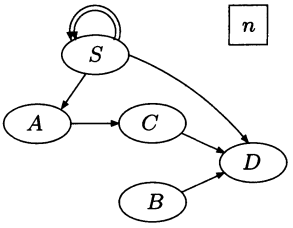

FIGURE 2.36. Adaptation of expert confidence modeled as a Kalman filter model. $P(S_{i+1} \mid S_i)$ is the identity table given in Table 2.17.

Note that with this model new cases can be used to revise conclusions on previous cases.

2.3.7 Interventions

You may wish to incorporate actions that change the state of some variables. You may, for example, wish to model the result of cleaning the spark plugs in the car start problem. If you use the model in Figure 1.15 directly and enter your cleaning of the spark plug by entering $SP = yes$, you get incorrect results. The problem is that you may not have a start problem anymore, and the state of *St* may be changed due to your action. The problem is called *persistence*. You may extend the model in Figure 1.15 with a variable *Clean?*, but then you must also introduce new nodes for the variables that may change state. Because you have a causal model, the nonpersistent nodes are the descendants of the nodes affected by the intervention (see Figure 2.37). *Clean?* has a special status in the model. It is not meaningful to give it prior probabilities, and the descendants of the nodes have no meaning before a decision on *Clean?* has been taken. Therefore, it is customary to give this kind of node a rectangular shape.

The conditional probabilities for new nodes are natural. If *Clean?* = *no*, then $SP-C$ is in the same state as SP, and if *Clean?* = *yes* and $SP = yes$, then the probability that $SP-C = no$ is the probability that you can clean

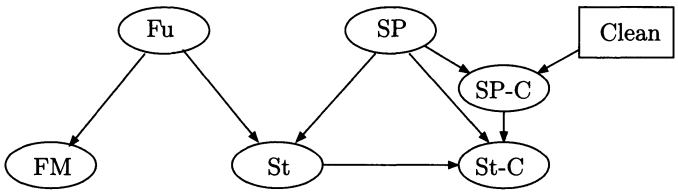

FIGURE 2.37. A network modeling the effect of cleaning the spark plugs.

the spark plugs properly. For $St - C$, you still have a start problem unless it was due to dirty spark plugs and they have been properly cleaned.

2.3.8 Continuous variables

It is possible to build Bayesian network models including continuous variables. However, due to a lack of technical development, there are several constraints. First, Bayesian networks can only handle *conditional Gaussian* distributions (given a parent configuration, the child distribution is a linear combination of normal distributions). A second constraint is structural. A continuous variable is not allowed to have a discrete child. This means, for example, that you cannot model whether a person is dead or alive as a consequence of the person's age. The reason for this constraint is mathematical/technical.

We will not deal with direct representation of continuous variables. Instead of a direct representation, you may transform a continuous variable into a finite variable by cutting the real line in intervals.

Consider the *Cold or Angina?* example from Section 2.1.2. The variable *Fever?* is continuous. For each combination of states of *Cold?* and *Angina?*, we must specify a distribution for *Fever?*. It would be natural to specify mean μ and variance σ^2 for a normal distribution, and this is also what you will do if you have a Bayesian network system which is able to handle conditional Gaussian distributions. On the other hand, if the model is extended with a child, *Headache?*, or *Fever?*, then you would violate the structural constraint, and you will have to transform *Fever?* into a finite variable.

Assume that you have the specification in Table 2.18. We must specify intervals for a finite set of states. For the three states *no, low,* and *high,* it would be natural to use knowledge of fever. In other situations, you would try to determine intervals such that for each parent configuration most of the probability mass is concentrated in few intervals. This may not be possible, and it will often be a delicate matter to establish a good set of intervals. In the current situation, we define low fever to be in the interval $(37.5°, 38.5°)$. Consequently, *no* is $(-\infty, 37.5°)$ and *high* is $(38.5°, \infty)$. Next, you use Table 2.18 to calculate the probability mass for each interval. The result is given in Table 2.19.

		Cold?	
		no	yes
Angina?	*no*	(37°, 0.25)	(37.5°, 0.75)
	mild	(38°, 0.5)	(38.5°, 1)
	severe	(39°, 0.75)	(39.5°, 1.25)

TABLE 2.18. Means and variances for the *Fever?* variable.

		Cold?	
		no	yes
Angina?	*no*	(0.834, 0.165, 0.01)	(0.5, 0.376, 0.124)
	mild	(0.24, 0.52, 0.24)	(0.159, 0.341, 0.5)
	severe	(0.042, 0.24, 0.718)	(0.037, 0.149, 0.814)

TABLE 2.19. The result of sampling Table 2.18 to the intervals for *no, low,* and *high.*

2.4 Special Features

A Bayesian network model is primarily used for belief updating. However, you may request other kinds of information from a model. This section outlines some types of requests. Chapter 6 gives a more detailed presentation. To illustrate the features in this section, we use the sore throat example from Section 2.1.2 (see Figure 2.38). However, we change the potentials slightly: when I suffer from mild angina, I will see yellow spots with probability 0.01, and it also happens with probability 0.001 that I have severe angina without a sore throat, provided that I do not have a cold. The rest of the potentials can be found in Sections 2.2.5 and 2.3.8.

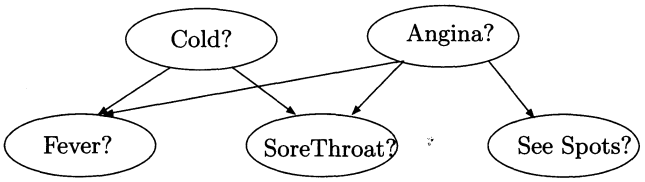

FIGURE 2.38. The sore throat model.

We use the evidence $e = \{Fever? = no,\ Sore\ Throat? = no,\ See\ Spots? = yes\}$ (do not ask why I looked down my throat.)

2.4.1 Joint probability tables

Because it is not unusual to suffer from both cold and angina, it may be of interest to use the model in Figure 2.38 to calculate the joint probability

table $P(Angina?, Cold? \mid e)$. This can be done by use of the fundamental rule:

$$P(Angina?, Cold? \mid e) = P(Angina? \mid Cold?, e)P(Cold? \mid e).$$

Read $P(Cold? \mid e)$ from the system; then first enter and propagate $Cold? = yes$ and then $Cold? = no$ to get $P(Angina? \mid Cold?, e)$.

This method is conceptually easy, but if you request the joint table for many variables, it is computationally very time-consuming. Other methods are presented in Chapter 6.

2.4.2 Most probable explanation

Instead of requesting the full joint probability table, I may request the most probable configuration of *Cold?* and *Angina?*. This can be achieved much faster than by calculating $P(Cold?, Angina? \mid e)$.

In general, you have a set of instantiated variables and you request the most probable configuration of the remaining variables. This is also called the *most probable explanation*, MPE. MPE can be calculated similarly to probability updating (see Section 1.4.7 and Chapter 5). The only difference is that instead of marginalizing by summing out you take the maximum. The distributive law for max reads $\max(ab, ac) = a\max(b, c)$. In the general form, it says

$$\text{If A} \notin \text{dom}(\phi_1), \text{ then } \max_A \phi_1\phi_2 = \phi_1 \max_A \phi_2.$$

Most Bayesian network systems have a special feature for calculating MPE.

2.4.3 Data conflict

Although the evidence e yields posterior probabilities for *Cold?* as well as for *Angina?*, it is more likely that I have misinterpreted what I saw in the throat. In other words, in the light of neither fever nor sore throat, it is very likely that the evidence *See Spots?* $= yes$ is faulty. It would be nice if the system by itself could raise a flag indicating that the evidence does not seem coherent.

To investigate coherence of the evidence, a *conflict measure* is defined. The idea behind the measure is that correct findings from a coherent case covered by the model support each other, and therefore we will expect them to be positively correlated. Let $e = \{e_1, \ldots e_m\}$ be a set of findings. The conflict measure on e is defined as

$$\text{conf}(e) = \log_2 \frac{P(e_1) \cdot \ldots \cdot P(e_m)}{P(e)}.$$

The conflict measure is easy to calculate because $P(e)$ is communicated by the system (see Exercise 2.1.3) and $P(e_i)$ can be read from the model in its initial state. If conf(e) is positive, the findings are not positively correlated, and we can take this as an indication that the evidence is conflicting. To be quite accurate, a high conflict measure is an indication that there is discrepancy between model and evidence. This may be due to flawed findings, it may be because we are faced with a very rare case, or the situation may not be covered by the model. This is discussed in more detail in Section 6.5.

2.4.4 Sensitivity analysis

Sensitivity analysis refers to analysing how sensitive the conclusions (the probabilities of the hypothesis variables) are to minor changes. The changes may be variations of the parameters of the model or may be changes of the evidence (*SE analysis*). In general, sensitivity analysis is rather technical and in this section we only give some hints. It is treated in more detail in Chapter 6.

Consider the angina example. The conclusion is $P(Angina?|e) = (0, 0.98, 0.02)$. SE analysis consists of answering questions such as "what are the crucial findings?," "what if one of the findings was changed or removed?," or "what set of findings would be sufficient for the conclusion?." If we consider the conclusion to be that I suffer from mild angina, we see that the finding *See Spots*? = *yes* is not sufficient in itself because it indicates severe angina, nor is any of the other findings. Instead, *See Spots*? = *yes* together with *Sore Throat* = *no* is sufficient, and with these two findings fixed the conclusion is insensitive to any finding on *Fever*?.

Now, consider the parameters $t = P(Sore\ Throat? = no \mid Angina? = severe, Cold? = no)$ and $s = P(See\ Spots? = y \mid Angina? = mild)$. The initial values of t and s are 0.001 and 0.01, respectively. What we might look for is a functional expression for $P(Angina? = mild \mid e)(t)$ and $P(Angina? = mild \mid e)(s)$. This is called one-way sensitivity analysis. We might also request two-way sensitivity analysis by establishing $P(Angina? = mild \mid e)(t, s)$.

It follows from a general theorem that $P(e)(t)$ as well as $P(Angina? = mild, e)(t)$ are linear expressions (see Section 6.7), and hence $P(Angina? = mild \mid e)(t)$ is a fraction of two linear expressions. From the initial propagation, we can acquire $P(e)(0.001)$ and $P(Angina? = mild \mid e)(0.001)$. By changing t to 0.002 and propagating, we get $P(e)(0.002)$ and $P(Angina? = mild \mid e)(0.002)$. These four values are sufficient for determining the four constants in the functional expression for $P(Angina? = mild \mid e)(t)$.

2.5 Summary

Types of variables when building a Bayesian network model

Hypothesis variables. Variables with a state that is asked for. They are, however, either impossible or too costly to observe.

Information variables. Variables that can be observed.

Mediating variables. Variables introduced for a special purpose. It may be to reflect properly the independence properties in the domain, to facilitate the acquisition of conditional probabilities, to reduce the number of distributions to acquire for the network, or other purposes.

Warning: it is tempting to introduce mediating variables in order to have a more refined model of the domain; however, if they do not serve any other purpose you should get rid of them. They jeopardize performance.

Acquiring conditional probabilities

Theoretically well-founded probabilities as well as frequencies and purely subjective estimates can be used for the same network.

If the number of distributions is too large for a reasonable estimation, a simplifying assumption can reduce it.

Noisy or. Let B have the parents $A_1, \ldots, A_n$ (all variables binary). Suppose that $A_i = y$ causes $B = y$ unless it is inhibited by an inhibitor Q_i which is active with probability q_i. Assume that the inhibitors are independent. Then,

$$P(B = n \mid a_1, \ldots, a_n) = \prod_{j \in Y} q_j,$$

where Y is the set of indices for the states y.

Divorcing. Let B have the parents $A_1, \ldots, A_n$. Assume that the set of configurations of $(A_1, \ldots, A_i)$ can be partitioned into the sets $c_1, \ldots, c_m$ such that whenever two configurations a^* and a_1^* of $(A_1, \ldots, A_i)$ are elements in the same c_j, then

$$P(B \mid a^*, A_{i+1}, \ldots, A_n) = P(B \mid a_1^*, A_{i+1}, \ldots, A_n).$$

Then, $A_1, \ldots, A_i$ can be divorced from $A_{i+1}, \ldots, A_n$ by introducing a mediating variable C with states $c_1, \ldots, c_m$, making C a child of $A_1, \ldots, A_i$ and a parent of B.

Other tricks

Undirected relations — in particular, logical constraints — can be modeled by introducing a dummy child of the constrained variables and letting its potential reflect the relation.

For specification language for *repeating structures*, see Figure 2.39.

Expert disagreements on potentials can be represented in a model by introducing a node representing the experts.

Continuous variables can be transformed into finite variables.

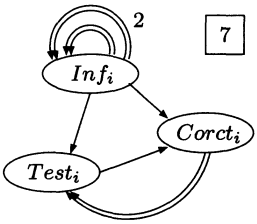

FIGURE 2.39. A compact specification of a repeating structure with 7 slices. The $\rightrightarrows$ indicates a temporal link. The number "2" attached to one of them specifies that it jumps two time steps (no number attached means a jump from slice i to slice $i+1$).

2.6 Bibliographical Notes

Simple Bayes was used by de Dombal et al. (1972). Noisy or was first described by Pearl (1986); divorcing was used in MUNIN (Andreassen, Jensen, Andersen, Falck, Kjærulff, Woldbye, Sørensen, Rosenfalck, and Jensen 1989). Exercise 2.21 is based on Cooper (1990). Chain graphs are treated in depth by Lauritzen (1996). Time-stamped models were introduced to Bayesian networks by Kjærulff (1995). The compact representation of repetitive structures was suggested by Bangsø and Wuillemin (2000). Andreassen (1992) discusses various ways of transforming conditional Gaussian variables into finite variables. A method not described in this chapter is *similarity networks* (Heckerman 1990). The method helps in elicitating the conditional probabilities. Other elicitation methods can be found in (Druzdzel and van der Gaag 1995). References for the special features in Section 2.4 are given in Section 6.8.

2.7 Exercises

Exercise 2.1 Assume that three mornings in a row I wonder whether my sore throat is due to cold or angina. Construct a model.

Exercise 2.2 Construct a model extending the model in Section 2.1.3 with a scanning test.

Exercise 2.3 ^{E} Construct a model for a single milk test (Section 2.2.1). What is the probability of infected milk given a positive test result?

Exercise 2.4 ^{E} Minced meat purchased in the supermarket may be infected. On average, it happens once out of 600 times. A test with results *positive* and *negative* can be used. If the meat is *clean*, the test result will be *negative* in 499 out of 500 cases, and if the meat is *infected*, the test result will be *positive* in 499 out of 500 cases.

	$Pr = y$	$Pr = n$
$Ho = y$	0.9	0.01
$Ho = n$	0.1	0.99

	$Ho = y$	$Ho = n$
$BT = y$	0.7	0.1
$BT = n$	0.3	0.9

	$Ho = y$	$Ho = n$
$UT = y$	0.8	0.1
$UT = n$	0.2	0.9

TABLE 2.20. Tables for Exercise 2.6.

Construct a Bayesian network and use a software system to calculate the probability of $infected$ for meat with a positive test result.

Exercise 2.5 ^{E} Complete the Bayesian network for Cold or angina? and perform a self-diagnosis.

Exercise 2.6 ^{E} Consider the insemination example from Section 2.1.3. Let the probabilities be as in Table 2.20 ($Ho = y$ means that hormonal changes have taken place). $P(Pr) = (0.87, 0.13)$.

(*i*) What is $P(Pr \mid BT = n, UT = n)$?

(*ii*) Construct a simple Bayes model. Determine the conditional probabilities for model by using the model above. What is $P(Pr \mid BT = n, UT = n)$ in this model?

Exercise 2.7 ^{E} Use the model from Exercise 2.6 to calculate $P(UT = y, BT = y)$. Enter the two pieces of evidence into the model and prompt your system to update probabilities. As a side effect, the system computes $P(e)$, the probability of the evidence entered. Find out how your system provides it.

Exercise 2.8 ^{E}

(*i*) Implement the seven-day model in Figure 2.12. Are the initial probabilities stable over time?

(*ii*) Consider the conditional probability $P(Inf_2 \mid Inf_1)$ with $P(Inf1) = (0.0007, 0.9993)$ and assume that the risk of becoming infected is 0.0002. We require stable initial probabilities: $P(Inf_2) = P(Inf_1) = (0.0007, 0.9993)$. Show that the chance of being cured must be 2/7.

(*iii*) Consider the conditional probabilities $P(Inf_{i+2} \mid Inf_i, Inf_{i+1})$, and assume that the risk of being infected is the same as before. We require stable initial probabilities. Show that the chance of being cured for a more than one day infection must be 0.4.

Exercise 2.9 Show that the procedure described in Section 2.1.4 is equivalent to updating in the model in Figure 2.8.

Exercise 2.10 ^{E} Consider the stud farm example in Section 2.2.2 and let the prior probability for aA be 0.005.

(*i*) Enter the model into your Bayesian network system.

(*ii*) Add to the model the frequency 0.001 for mutation of the gene from A to a.

(*iii*) Construct a model for the situation in part (*i*), but for a recessive gene borne by the female sex chromosome. (Note that horses with the disease are taken out of production.)

Exercise 2.11 ^{E} Consider the transmission example from Section 2.2.4.

(*i*) From Table 2.10, calculate the remaining conditional probabilities for the model in Figure 2.17.

(*ii*) Implement the model.

(*iii*) The sequence *baaca* is received. What is the most probable symbol transmitted according to the model in Figure 2.17? What is the most probable word?

(*iv*) What is the most probable word according to the model in Figure 2.18?

Exercise 2.12 ^{E} Consider the simplified poker game in Sections 2.1.5 and 2.2.3.

(*i*) Implement the system.

(*ii*) Extend the system with a facility giving the chances that your hand is better than your opponent's hand.

Exercise 2.13 You are confronted with three doors, A, B, and C. Behind exactly one of the doors there is \$10,000. When you have pointed at a door, an official opens another one with nothing behind it. After he has done so, you are allowed to alter your choice. Should you do that?

Exercise 2.14 Extend the model in Figure 2.22 to incorporate constraints on color and pattern for the same sock.

Exercise 2.15 The *drive* in golf is the first shot when playing a hole. If you drive with a *spoon* (a particular type of golf club), there is a 2% risk of a miss (a bad drive), and $\frac{1}{4}$ of the good drives have a length of 180 m, $\frac{1}{2}$ are 200 m, and $\frac{1}{4}$ have a length of 220 m. You may also use a *driver* (another type of golf club). This will on average increase the length by 10%, but you will also have 3 times as high a risk of a miss. Both wind and the slope of the hole may affect the result of the drive. Wind doubles the risk of a miss, and the length is affected by 10% (longer if the wind is from behind and shorter otherwise). A downhill slope yields 10% longer drives, and an uphill slope decreases the length of the drive by 10%.

Estimate the probabilities for miss and length given the various factors.

Exercise 2.16 The *putt* is (hopefully) the last shot on a golf hole. My ball is lying 1 m away from the hole, and under normal circumstances I will miss 1 putt out of 10. However, when it rains, I miss 1 out of 7, if it is

$As \setminus Ha_1$	*no*	*mild*	*moderate*	*severe*
y	(1, 0, 0, 0)	(0.7, 0.3, 0, 0)	(0.1, 0.7, 0.2, 0)	(0, 0.1, 0.7, 0.2)
n	(1, 0, 0, 0)	(0, 1, 0, 0)	(0, 0, 1, 0)	(0, 0, 0, 1)

TABLE 2.21. $P(Ha \mid Ha_1, As)$ for Exercise 2.19.

windy, I miss 1 out of 4, if the green is curved, I miss 1 out of 3, and if I am putting for a birdie (one under par), I miss 1 out of 2.

Estimate the probabilities for success and failure given the various factors.

Exercise 2.17 Show that the model in Figure 2.24 corresponds to the one in Figure 2.23.

Exercise 2.18 ^{E} Show that noisy or may be modeled as described in Figures 2.28 and 2.29. Apply this model to the putting problem of Exercise 2.16, and compare the amount of numbers to specify.

Exercise 2.19

(*i*) Complete the model in Section 2.3.4.

$$P(Ha) = P(Ha \mid Ot = y) = (0.93, 0.04, 0.02, 0.01)$$

$$P(Ha \mid Fe = y) = P(Ha \mid Ho = y) = P(Ha \mid Fb = y) = (0.1, 0.8, 0.1, 0)$$

$$P(Ha \mid Bt = y) = (0.3, 0.2, 0.2, 0.3).$$

(*ii*) Include aspirin on the basis of Table 2.21.

Exercise 2.20 ^{E} Consider the model in Figure 2.40. All variables have ten states.

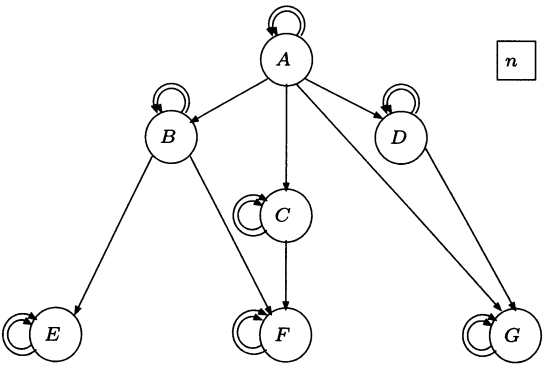

FIGURE 2.40. A time-stamped model for Exercise 2.20.

(*i*) Implement one time slice (with any set of potentials).

(*ii*) Implement three time slices.

(*iii*) How many time slices can you implement before your system reports that it requires extra memory?

Exercise 2.21 ^{E} The following relations hold for the Boolean variables A, B, C, D, E, and F:

$(A \vee \neg B \vee C) \wedge (B \vee C \vee \neg D) \wedge (\neg C \vee E \vee \neg F) \wedge (\neg A \vee D \vee F) \wedge$
$(A \vee B \vee \neg C) \wedge (\neg B \vee \neg C \vee D) \wedge (C \vee \neg E \vee \neg F) \wedge (A \vee \neg D \vee F)$.

(*i*) Is there a truth value assignment to the variables making the expression true? (Hint: represent the expression as a Bayesian network.)

(*ii*) We receive the evidence that A is false and B is true. Is there a truth value assignment to the other variables making the expression true?

(*iii*) The *satisfiability problem* for propositional calculus is: given a Boolean expression $\mathbb{E}$ (over n Boolean variables), is there a truth value assignment to the variables that makes $\mathbb{E}$ true?

Show that a method for calculation of probabilities in Bayesian networks yields a method for solving the satisfiability problem for propositional calculus. (Hint: assume that $\mathbb{E}$ is in conjunctive normal form.)

(*iv*) Show that probability calculation in Bayesian networks is NP-hard.

Exercise 2.22 ^{E}

(*i*) Take your model from Exercise 2.5 and enter the evidence $e = \{Fever? = no,\ Sore\ Throat? = no,\ See\ Spots? = yes\}$. How does your system react?

Change the potentials such that $P(Sore\ Throat? = no \mid Angina? = severe,\ Cold? = no) = 0.001$, and $P(See\ Spots? \mid Angina? = mild) = 0.01$.

(*ii*) Calculate $P(Cold?, Angina? \mid e)$.

(*iii*) Calculate MPE(e).

(*iv*) Calculate conf(e).

(*v*) Determine $P(Angina? = mild \mid e)(s)$, where $s = P(See\ Spots? = yes \mid Angina? = mild)$.

3

Learning, Adaptation, and Tuning

In Chapter 2, we mainly considered modeling situations where a domain expert establishes the structure of the network. With respect to estimating the potentials for the model (the conditional probabilities), it can be based on combinations of theoretical considerations, a database of cases, and pure subjective estimates. You may be so fortunate that you have a large database of cases or you expect to collect cases in the future. In that case, you would like to exploit the information for model building or for future change.

In this chapter, we introduce some basic techniques for these tasks. More sophisticated techniques have been invented, but the approaches to the problems addressed are similar.

In Section 3.2, we first consider *batch learning*: using a database of cases to establish structure as well as potentials for a Bayesian network. Section 3.3 deals with the situation where a Bayesian network has been constructed and repeatedly is used to process cases. *Adaptation* is the process of modifying the network to better reflect the experience represented by the accumulated cases. The last task in this chapter is *tuning*, where you have constructed a model and now have some explicit request to it in the form of prescribed posterior probabilities. How can you modify the parameters in the model to meet the request? The approach taken to tuning is similar to neural network training, namely gradient descent on the parameters. A parameter is an entrance in a conditional probability table. Sections 3.2–3.4 can be read independently and, for Section 3.4, readers not familiar with partial derivatives and gradients are advised to consult a textbook on calculus.

Common to all three tasks is that they are very active areas of research and standards have not been established. As a consequence, features for learning, adaptation, and tuning are not yet a natural part of Bayesian network software. With respect to computer-aided exercises, readers are referred to the provider of their Bayesian network software. On the book's web site (www.cs.auc.dk/~fvj/bnid.html) you will find links to software and exercises.

3.1 Distance Measures

In various situations, you will need to compare distributions. You have a distribution $\mathbf{x} = (x_1, \ldots, x_n)$, which is considered a "true" or "target" distribution, and there are other distributions $\mathbf{y} = (y_1, \ldots, y_n)$ and $\mathbf{z} = (z_1, \ldots, z_n)$, which are "approximations." You wish to choose between $\mathbf{y}$ and $\mathbf{z}$. To help your decision, you look for a measure of distance between distributions.

For example, let $\mathbf{x} = (0.3, 0.3, 0.2, 0.2)$, $\mathbf{y} = (0.2, 0.4, 0.3, 0.1)$, and $\mathbf{z} = (0.5, 0.1, 0.2, 0.2)$. Which one of $\mathbf{y}$ and $\mathbf{z}$ is closest to $\mathbf{x}$?

The problem of distance can be approached in the following way. Assume I perform a large set of forecasts on the basis of the distributions. For each forecast, I get a penalty if I am wrong. If the true forecasting distribution is $\mathbf{x}$ but I use the distribution $\mathbf{y}$ for forecasting, I will get some average penalty $ES(\mathbf{x}, \mathbf{y})$, and $ES(\mathbf{x}, \mathbf{y}) - ES(\mathbf{x}, \mathbf{x})$ is the distance from $\mathbf{y}$ to $\mathbf{x}$.

Let us as an example consider a symmetric die. For the outcomes, the true distribution is $(1/6, 1/6, 1/6, 1/6, 1/6, 1/6)$, and assume that I forecast the distribution $(0.2, 0.2, 0.1, 0.1, 0.2, 0.2)$. Consider the situation where the rolling die shows "2."

There are two widely used ways of giving penalties: the *quadratic (or Brier) scoring rule* and the *logarithmic scoring rule.*

The quadratic scoring rule translates the actual outcome to a $0-1$ distribution and takes the sum of the squares between the forecasted distribution and outcome. If we let a_i denote the ith outcome, we define the penalty for outcome a_i as

$$S_Q(a_i, \mathbf{y}) = (1 - y_i)^2 + \sum_{j \neq i} y_j^2 = 1 - 2y_i + \sum_j y_j^2.$$

For the dice example, the outcome "2" is translated to $(0, 1, 0, 0, 0, 0)$ and the penalty is $0.2^2 + 0.8^2 + 0.1^2 + 0.1^2 + 0.2^2 + 0.2^2 = 0.78$.

If the true frequencies of the outcomes are $\mathbf{x}$, the average penalty is

$$ES_Q(\mathbf{x}, \mathbf{y}) = \sum_i x_i S_Q(a_i, \mathbf{y}),$$

and we define

$$\text{dist}_Q(\mathbf{x}, \mathbf{y}) = ES_Q(\mathbf{x}, \mathbf{y}) - ES_Q(\mathbf{x}, \mathbf{x}).$$

Through pencil pushing (see Exercise 3.1), we get

$$\text{dist}_Q(\mathbf{x}, \mathbf{y}) = \sum_i (x_i - y_i)^2.$$

For the preceding example, we get the distances $\text{dist}_Q(\mathbf{x}, \mathbf{y}) = 0.04$ and $\text{dist}_Q(\mathbf{x}, \mathbf{z}) = 0.08$, and we conclude that $\mathbf{y}$ is closer to $\mathbf{x}$ than $\mathbf{z}$ is. We call dist_Q the *Euclidean distance* (although we do not take the square root.)

The logarithmic scoring rule is defined as

$$S_L(a_i, \mathbf{y}) = \log_2 \frac{1}{y_i} = -\log_2 y_i.$$

The idea behind this scoring rule is a consideration of how much information you get from observing an event with probability p. If $p = 1$, the observation tells you nothing at all, and the smaller the probability, the higher the surprise. The concept of *entropy* originates from the same considerations. Entropy is the information expected to be achieved from an experiment with distribution $\mathbf{x}$: $H(\mathbf{x}) = -\sum_i x_i \log_2 x_i$.

The expected score from forecasting $\mathbf{x}$ when $\mathbf{y}$ is the true distribution is $-\sum_i y_i \log_2 x_i$, and we get for the distance from $\mathbf{x}$ to $\mathbf{y}$

$$\text{dist}_K(\mathbf{x}, \mathbf{y}) = \sum_i y_i(\log_2 y_i - \log_2 x_i).$$

dist_K is called the *Kullback–Leibler* divergence. Note that $\text{dist}_K(\mathbf{x}, \mathbf{y})$ is not symmetric in $\mathbf{x}$ and $\mathbf{y}$.

The quadratic scoring rule and the logarithmic scoring rule have the nice property that they are *strictly proper*: the only distribution yielding a minimal score is the true one (see Exercise 3.2). Therefore, the derived distance measures have the property that $\text{dist}(\mathbf{x}, \mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$.

3.2 Batch Learning

Batch learning consists in learning Bayesian network models from large databases. It is a very lively research area, and several methods have been developed. However, although the statistical theory is well understood, no method has so far become "the standard." This is due to practical problems. Although you know what you should do, you do not have the time or space to do it. Section 3.2.1 gives a simple example to present some of the reasoning behind batch learning. However, the method presented can rarely be used in practice, and in Section 3.2.2 we give some hints on the practical problems. Readers looking for detailed knowledge of batch learning methods are referred to the literature listed in the Preface.

3.2.1 Example: strings of symbols

In Section 2.2.4, a model was used for the relations between the letters in the words transmitted (see Figure 3.1), and the table of frequencies (Table 2.10) was used to determine the probabilities for the model. We call this model M_{Simp}. However, there may be other models, and we need some means of evaluating these possible models. There are two matters to consider:

- How well can the original table be reconstructed from the model?
- How much space does the model require?

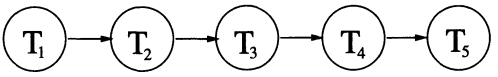

FIGURE 3.1. The Bayesian network, M_{Simp}, used in Section 2.2.4 for modeling the relation between the letters in the words transmitted.

The chain rule (Theorem 1.1) applied to M_{Simp} yields the joint probability table

$$P^*(T_1, T_2, T_3, T_4, T_5) = P(T_1)P(T_2 \mid T_1)P(T_3 \mid T_2)P(T_4 \mid T_3)P(T_5 \mid T_4).$$

The result is shown in Table 3.1.

First 2 letters	Last 3 letters							
	aaa	**aab**	**aba**	**abb**	**baa**	**bab**	**bba**	**bbb**
aa	0.016	0.023	0.018	0.021	0.044	0.067	0.050	0.061
ab	0.030	0.044	0.033	0.041	0.011	0.015	0.012	0.014
ba	0.010	0.016	0.012	0.014	0.029	0.045	0.033	0.041
bb	0.044	0.067	0.059	0.061	0.016	0.023	0.017	0.021

TABLE 3.1. $P^*(T_1, T_2, T_3, T_4, T_5)$, the joint probability table determined by M_{Simp}.

A direct comparison of Table 2.10 and Table 3.1 shows some differences, and we may or may not say that M_{Simp} is a sufficiently accurate representation of Table 2.10.

We have the true distribution in Table 2.10, and the one used for forecasting is Table 3.1. We get

$$\text{dist}_Q(P, P^*) = 0.000337.$$

It is more or less up to you to decide how large a distance to accept and to fix a threshold. For our tables, we could say that the difference between the probabilities should not be larger than 0.004, so a reasonable threshold would be $32(0.004)^2 = 0.000512$, yielding M_{Simp} acceptable.

3.2.2 Search for possible structures

We look for Bayesian networks that represent a joint probability table within an acceptable distance from $P(Word)$, but this is not sufficient. It may happen that several models are acceptable, and bearing in mind that we look for simple models, we also must take the size of the model into account.

Let M be a Bayesian network with variables U. For each variable A with parents $pa(A)$, we define $Sp(A)$ to be the number of entries in $P(A \mid pa(A))$, and the size is

$$Size(M) = \sum_{A \in U} Sp(A).$$

For example, $Size(M_{\text{Simp}}) = 2 + 4 + 4 + 4 + 4 = 18$.

To take care of the tradeoff between size and distance, we define an *acceptance measure*

$$\text{Acc}(P, M^*) = Size(M^*) + k\text{dist}(P, P^*),$$

where P^* is the joint probability table for U determined by M^*, and k is a positive real number.

In the problem with transmission of symbol strings, we use $\text{dist}_Q(P, P^*)$ and choose $k = 10{,}000$, and we will work with a distance threshold of 0.0005. The task is to determine an acceptable Bayesian network that minimizes Acc.

In principle, we investigate all possible DAGs over the variables T_1, T_2, T_3, T_4, T_5. However, there are too many of them; we therefore add some structure constraints to the models considered. In our case, it is natural to say that a directed link from T_i to T_j is only allowed if $i < j$.

We start with the largest model M_{Max} meeting the structure constraint. It is shown in Figure 3.2.

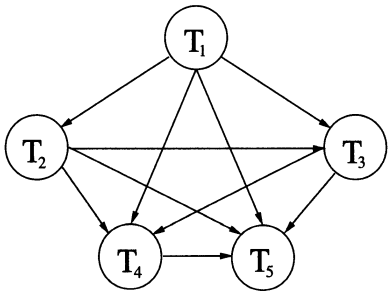

FIGURE 3.2. The model M_{Max}, the largest model meeting the constraint that a link from T_i to T_j is only allowed if $i < j$.

Let $P(\mathit{Word} \mid M_{\text{Max}})$ denote the distribution determined by M_{Max}. From the following calculation, we see that $P(\mathit{Word} \mid M_{\text{Max}}) = P(\mathit{Word})$:

$$\begin{aligned} P(\mathit{Word}) &= P(T_1, T_2, T_3, T_4, T_5) \\ &= P(T_5 \mid T_1, T_2, T_3, T_4) P(T_1, T_2, T_3, T_4) \\ &= P(T_5 \mid T_1, T_2, T_3, T_4) P(T_4 \mid T_1, T_2, T_3) P(T_1, T_2, T_3) \\ &= P(T_5 \mid T_1, T_2, T_3, T_4) P(T_4 \mid T_1, T_2, T_3) P(T_3 \mid T_1, T_2) \\ &\quad P(T_2 \mid T_1) P(T_1) \\ &= P(\mathit{Word} \mid M_{\text{Max}}). \end{aligned}$$

Therefore, M_{Max} is below the threshold, and $\text{Acc}(P, M_{\text{Max}})$ is calculated to be 62.

There are 2^{10} DAGs to investigate. This is a heavy task, but if we follow a procedure where we – starting with M_{Max} – successively delete links, we need not remove any further links when a model is rejected. The result of this search is that the model M_{Min} in Figure 3.3 is the best one.

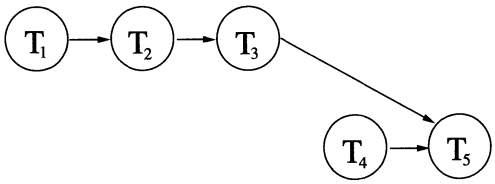

FIGURE 3.3. M_{Min}. The best model for $P(\mathit{Word})$.

The tables for M_{Min} are given in Table 3.2, and the joint probability table determined by M_{Min} is shown in Table 3.3.

	T_1	
T_2	a	b
a	0.6	0.4
b	0.4	0.6

$P(T_2 \mid T_1)$

	T_2	
T_3	a	b
a	0.25	0.75
b	0.75	0.25

$P(T_3 \mid T_2)$

	T_3	
T_4	a	b
a	(0.45, 0.55)	(0.4, 0.6)
b	(0.5, 0.5)	(0.4, 0.6)

$P(T_5 \mid T_3, T_4)$

TABLE 3.2. Conditional probabilities for M_{Min}; $P(T_1) = P(T_4) = (0.5, 0.5)$.

We have $\text{Acc}(P, M_{\text{Min}}) = 20.14$ and $\text{Acc}(P, M_{\text{Simp}}) = 21.37$.

3.2.3 Practical issues

The method described in Section 3.2.1 was illustrated through an example with a very small set of variables. In such a case, there is rarely reason to look for a model; the joint probabilities in Table 2.10 can be used.

First 2 letters	Last 3 letters							
	aaa	**aab**	**aba**	**abb**	**baa**	**bab**	**bba**	**bbb**
aa	0.017	0.021	0.019	0.019	0.045	0.068	0.045	0.068
ab	0.034	0.040	0.037	0.038	0.010	0.015	0.010	0.015
ba	0.011	0.014	0.010	0.010	0.031	0.045	0.030	0.045
bb	0.051	0.062	0.057	0.057	0.015	0.023	0.015	0.023

TABLE 3.3. $P(Word \mid M_{\text{Min}})$.

If the number of variables is so large that the joint probability table is intractably large, it is also impossible to start with a model such as M_{Max} (the conditional probability table for the last variable is as large as the joint probability table for the universe).

The batch learning situation will often be that you have a very large set of cases C in a data warehouse. The task of extracting patterns from the cases in a data warehouse is called *data mining*. A goal for data mining could be to establish a Bayesian network model, and the search starts from a tractable model, which then may be changed incrementally by adding, removing, or redirecting a link. However, the search space of models is intractably large, and you must constrain it. With ten variables, there are more than $4 \cdot 10^{18}$ different DAGs.

There are various standard ways of constraining the models to consider. First, causality is exploited. If possible, the nodes are clustered in a causal hierarchy. You may, for example, consider a medical domain where you have disease nodes **D**, symptom nodes **S**, risk factor nodes **R**, and treatment nodes **T**. Then, you need not consider links from a node in **S** to a node in **D**. The full hierarchy is shown in Figure 3.4.

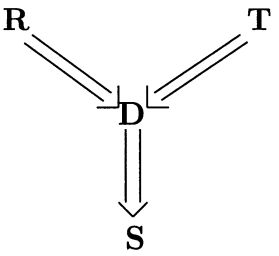

FIGURE 3.4. A causal hierarchy for clusters of nodes. Directed links are allowed only inside a cluster or downward in the hierarchy.

If **R** and **T** have two nodes, and **D** and **S** have three nodes, then the hierarchy allows approximately $2 \cdot 10^{12}$ different DAGs. This is a considerable reduction compared to $4 \cdot 10^{18}$, but still it is extremely many.

Another way of reducing the model space is to use expert statements. All positive as well as negative statements on the presence of links reduce

the number by a factor between 2 and 3. If, for example, the expert states that the nodes in **D** are independent given **R** and **T** (three links missing), then the model space is reduced by a factor of 25.

Another complexity issue has to do with the calculation of the distance. For the data warehouse situation, it is difficult to work with the Euclidean distance. Instead, scoring can be used: each case c is entered into the model M, and M is given a score $S^M(c)$. The logarithmic scoring rule is relatively easy to use:

$$S^M(c) = -\log_2 P^M(c).$$

Enter the case into the model, propagate it, and take the normalization constant. The sum of the scores, $S^M(C)$, is used for comparing models; the lower the score, the better. If we take the distribution of the cases as the "true" distribution P^C, we have (see Exercise 3.5)

$$S^M(C) - S^C(C) = n \cdot \mathrm{dist}_K(P^M, P^C),$$

where n is the number of cases in C. In other words, the difference in score between two models is proportional to the difference in Kulbach–Leibler divergence with P^C.

$S^M(C)$ has another characterization. If we assume all cases are independent, we have

$$P^M(C) = \Pi_{c \in C} P(c)$$

and

$$S^M(C) = -\log_2 P^M(C).$$

$\log_2 P^M(C)$ is also called the *log-likelihood of M given C* (notice that $P^M(C) = P(C \mid M)$).

So far, we have assumed the distribution from the database to be the true one, and the task was to find a compact representation approximating it. Usually, it is too bold an assumption to consider the database as a true distribution. It is more correct to consider a database as a *sample* from an unknown true distribution. Table 2.10 may, for example, be based on 1,000 words. This means that although M_{Min} is closer to P, it may still be that P is sampled from M_{Simp}.

Let C be a database of cases, and let $\mathbb{M}$ be a set of models. What you would try to do is maximize $P(M \mid C)$ for $M \in \mathbb{M}$. $P(M \mid C)$ is a rather awkward probability to calculate. However, Bayes' rule (1.3) can help us:

$$P(M \mid C) = \frac{P(C \mid M)P(M)}{P(C)} = \frac{P^M(C)P(M)}{P(C)}.$$

Since $P(C)$ is independent of M, it plays no role when determining the maximum. If there is no prior knowledge of the probabilities of the models,

we will assume them to be equally likely. Hence, the (log-)likelihood is a measure of goodness. Either you can look for a model of maximal likelihood or you can balance the likelihood with size.

There are many other practical problems. Very often the database is not a list of cases for which the states of all variables are known. You may have many missing values. It may also happen that you have several databases over different overlapping sets of variables and with different numbers of cases. Also, if the model M is large and the number of cases in C is extremely large, the calculation of $P^M(C)$ can be very time-consuming. Therefore, you look for incremental methods for fast calculation of $P^N(C)$ on the basis of $P^M(C)$ if N has almost the same structure as M.

3.3 Adaptation

When constructing a Bayesian network, you will almost always be uncertain of the correctness of the conditional probabilities chosen. Usually you would allow each probability to range within an interval, and a number in this interval is chosen. This type of uncertainty is called *second-order uncertainty*.

Second-order uncertainty raises two questions:

- Does the second-order uncertainty have an impact on the conclusions from the model?
- Are there systematic ways of reducing the second-order uncertainty?

The first question was discussed in Section 2.4.4 and is addressed in Section 6.7. In this section, we address the second question. We will look at a situation where certain parameters are open for modification.

When a system is at work, you repeatedly get new cases, and you would like to learn from these cases. As described in Section 2.3.6, the situation may be that you are pretty certain of the structure of the network. However, the conditional probabilities are dependent on a context that varies from place to place, and you want to build a system that automatically adapts to the particular context in which it is placed.

In Figure 3.5, the variable A is directly influenced by B and C, and the strength is modeled by $P(A \mid B, C)$. The uncertainty in $P(A \mid B, C)$ may be modeled explicitly by introducing an extra parent, T, for A (Figure 3.5 (b)). The variable T can be considered as a type variable. To reflect the frequencies of the context types, a prior distribution $P(T)$ is given.

When a case is entered into the network, the propagation will yield a new distribution $P^*(T)$, and we may say that the change of the distribution for T reflects what has been learned from the case. $P^*(T)$ can now be used as a new prior distribution. All variables whose table is dependent on the context will be children of T.

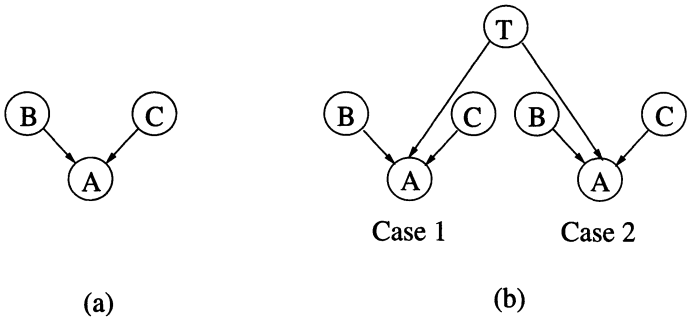

FIGURE 3.5. Adaptation through a type variable T. The distribution of T is updated by *Case 1* and used in the next case.

3.3.1 Fractional updating

If the uncertainty of the conditional probabilities cannot be modeled explicitly through a type variable, statistical methods can be used. The statistical task is to modify the estimates of the parameters gradually with the cases entered. This is intractable unless some assumptions on the dependencies between the parameters are added. For the preceding situation, the dependence is modeled through the T variable. If this is not possible, two simplifying assumptions are often used.

Global independence says that the second-order uncertainty for the various variables is independent. This means that we can modify the tables for the variables independently.

Local independence says that the uncertainties of the distributions for different parent configurations are independent. To be more precise, let (b_i, c_j) and (b_i', c_j') be different configurations; then the (second-order) uncertainty on $P(A \mid b_i, c_j)$ is independent of the (second-order) uncertainty on $P(A \mid b_i', c_j')$, and the two distributions can be modified independently.

Consider $P(A \mid B, C)$, and let all variables be ternary. Under the assumptions of global and local independence, we may now think of $P(A \mid b_i, c_j) = (x_1, x_2, x_3)$ as a distribution established through a number of past cases where (B, C) was in state (b_i, c_j). We can then express our certainty of the distribution by a fictitious sample size s. The larger the sample size, the smaller the second-order uncertainty, so we work with a *sample size* s, a set of counts (n_1, n_2, n_3) such that $s = n_1 + n_2 + n_3$ and

$$P(A \mid b_i, c_j) = \left(\frac{n_1}{s}, \frac{n_2}{s}, \frac{n_3}{s}\right).$$

Let us first consider a couple of simple cases before we take the general case.

1. We get a new case e with $B = b_i$ and $C = c_j$ and with $A = a_1$. Then, $n_1 := n_1 + 1$ and $s := s + 1$, and the probabilities are updated as

follows:

$$x_1 := \frac{(n_1 + 1)}{(s + 1)}; x_2 := \frac{n_2}{(s + 1)}; x_3 := \frac{n_3}{(s + 1)}.$$

2. We get a new case e with $B = b_i$ and $C = c_j$, but for A we only have a distribution $P(A \mid e) = P(A \mid b_i, c_j, e) = (y_1, y_2, y_3)$. Then, we cannot work with integer counts, and we update $n_k := n_k + y_k$ and $s := s + 1$. Accordingly, we get

$$x_k := \frac{(n_k + y_k)}{(s + 1)}.$$

In general, we get a case with $P(b_i, c_j \mid e) = z$. Then, $s := s + z$. To update the counts, we use the distribution $P(A \mid b_i, c_j, e) = (y_1, y_2, y_3)$. Because the sample size is only increased with z, we take $n_k := n_k + zy_k$, and we get

$$x_k := \frac{(n_k + zy_k)}{(s + z)}.$$

This scheme is known as *fractional updating*. Unfortunately, the scheme has a serious drawback, namely that it tends to overestimate the count of s, thereby overestimating our certainty of the distribution. Assume for example that $e = \{B = b_i, C = c_j\}$. Then, the case tells us nothing about $P(A \mid b_i, c_j)$, but nevertheless fractional updating will add a count of 1 to s and take it as a confirmation of the present distribution.

A statistically correct updating is intractable. However, an approximation of it can be performed. It does not have the same drawback as mentioned for fractional updating. Essentially, the distribution is updated as in fractional updating; however, the sample size is modified in a different way. We will not go into that.

3.3.2 Fading

It is often a problem for fractional updating that the initial counts are kept when the system is trying to adapt to the environment. Particularly when the conditional probabilities in the environment change over time, the accumulated counts will prevent the system from following the changes. Also, because fractional updating has a tendency to overestimate counts, vacuous counts will build up to make parameters too resistant to change. Therefore, to keep the flexibility of parameters, it may be good to prevent the sample size from growing unbounded.

An idea for solving this problem is the following. For example, let the ternary variable X have sample size s and counts (n_1, n_2, n_3), and assume that we get a count of 1 for x_1. Now, instead of counting n_1 up by one, we first multiply the counts by a fading factor, $q \in (0, 1)$. Hence, we get

$$s := sq + 1; n_1 := n_1 q + 1; n_2 := n_2 q; n_3 := n_3 q.$$

If we assume that all counts will be of value 1, the influence from the past will fade away exponentially. In the limit, we get a sample size s^*, where

$$s^* = \frac{1}{(1-q)}.$$

s^* is called the *effective sample size*, and it represents a steady-state situation. If $s = s^*$ and we get a new count, we have

$$s := s^* q + 1 = \frac{q}{(1-q)} + 1 = \frac{1}{(1-q)} = s^*.$$

Instead of declaring a fading factor, you may declare an effective sample size s^*, and the fading factor is then

$$q^* = \frac{(s^* - 1)}{s^*}.$$

This idea can be used for each modifiable distribution $P(X \mid \pi)$. The effective sample size need not be the same for all distributions. The effective sample size to declare is dependent on how resistant to change you wish the distribution to be. The higher the resistance, the higher the effective sample size.

Fading can be implemented such that the effective sample size is preserved. In other words, if the sample size for a distribution is equal to the declared effective sample size s^*, then it will not be changed when adapting to a new case.

Let $P(A \mid \pi)$ be declared modifiable with effective sample size s^*, and assume we have $P(\pi \mid e) = y$ for a case. Then, fractional updating yields a new count of y. If the sample size will be preserved in the steady-state situation, we have for the fading factor q

$$s^* q + y = s^*.$$

Hence,

$$q = \frac{(s^* - y)}{s^*}.$$

Note that if $P(\pi \mid e) = 1$, then $q = q^*$, and if $P(\pi \mid e) = 0$, then $q = 1$.

3.3.3 Specification of initial sample size

Frequently, the uncertainty of a parameter is expressed as an interval $[x, y]$. To exploit the technique for adaptation, the second-order uncertainty expressed by this interval will be translated to an initial sample size and a

set of counts. The specification of the interval $[x, y]$ for $t = P(A = a)$ can be interpreted as "I expect the value of t to be somewhere in the middle of the interval, and I am 90% sure that the value is in the interval." In other words, you have a distribution of t with mean close to $\frac{1}{2}(x + y)$ and with 90% of the density mass inside $[x, y]$.

As an example, take the interval $[0.3, 0.4]$ for the state a of the binary variable A. We interpret the interval as before, and assume that the distribution is the result of s samples out of which n were in state a. The distribution for t is the so-called beta distribution with mean $\mu = \frac{n}{s}$ and with variance $\sigma^2 = \frac{\mu(1-\mu)}{(s+1)}$. It holds that 90% of the probability mass lies in the interval $[\mu - \sigma, \mu + \sigma]$, so we seek values for s and n such that $\mu \approx 0.35$ and $\sigma \approx 0.05$, and we get $n = 31.5$ and $s = 90$.

3.3.4 Example: strings of symbols

Consider the transmission of symbols example from Section 2.2.4 with the model from Figure 2.17. Assume that every tenth word is sent through an error-correcting code such that you know for certain the word transmitted. You wish to adapt the parameters of the model to the words actually transmitted and received.

First, you can use the coded words to adapt the error rates: $P(R_i \mid T_i)$. Choose the effective sample size 100 for all parameters. This gives the fading factor 0.99. Also, let the initial sample be 100. The counts are given in Table 3.4.

	$T = a$	$T = b$
$R = a$	80	15
$R = b$	10	80
$R = c$	10	5

TABLE 3.4. Initial counts for $P(R \mid T)$.

Whenever a coded word is received, you have five cases (excluding the redundancy bits in the code). Assume that **baaba** was sent but **baaca** received. This means that the distribution $P(R \mid a)$ is modified three times and the distribution $P(R \mid b)$ is changed twice. For $P(R \mid a)$ we get the counts $((80{\cdot}0.99{+}1){\cdot}0.99{+}1){\cdot}0.99{+}1, 10{\cdot}0.99^3, 10{\cdot}0.99^3) = (80.6, 9.7, 9.7)$, and for $P(R \mid b)$ we get the counts $(15{\cdot}0.99^2, (80{\cdot}0.99{+}1){\cdot}0.99, 5{\cdot}0.99^2{+}1) = (14.7, 79.4, 5.9)$.

The noncoded words cannot be used for adaptation of $P(R \mid T)$, but they can be used for modifying $P(T_1)$ as well as $P(T_{i+1} \mid T_i)$. Assume that we receive the word $e =$ **baaca**. Let us concentrate on modifying $P(T_2 \mid T_1)$. Let the initial sample size be 50 for $T_1 = a$ and 150 for $T = b$. From Table 2.11, we infer the count table as given in Table 3.5.

	$T_1 = a$	$T_1 = b$
$T_2 = a$	30	60
$T_2 = b$	20	90

TABLE 3.5. Initial counts for $P(T_2 \mid T_1)$.

The model from Exercise 2.11 yields $P(T_1 \mid e) = (0.13, 0.87)$, $P(T_2 \mid T_1 = a, e) = (0.81, 0.19)$, and $P(T_2 \mid T_1 = b, e) = (0.66, 0.34)$. The fading factors are $(100 - 0.13)/100 = 0.9987$ and $(100 - 0.87)/100 = 0.9913$. We get for $P(T_2 \mid a)$ the counts $(30 \cdot 0.9987 + 0.13 \cdot 0.81, 20 \cdot 0.9987 + 0.13 \cdot 0.19) = (30.07, 20.00)$ and for $P(T_2 \mid b)$ we get $(60 \cdot 0.9913 + 0.87 \cdot 0.66, 90 \cdot 0.9913 + 0.87 \cdot 0.34) = (60.05, 89.5)$.

Note that the sample size increases for the part with initial sample size smaller than the effective sample size and decreases for the part with initial sample size larger than the effective sample size.

3.3.5 Adaptation to structure

As for the parameters in a model, it may happen that the structure of the model does not fit the cases you meet. If you use incremental adaptation of parameters, you will often experience that the changes of parameter values to a large degree will compensate for a slightly incorrect structure. Anyway, the structural inaccuracy may be so substantial that parameter adjustments cannot compensate. Unfortunately, no handy method for incremental adaptation of structure has been constructed. The reason is that structural changes are performed in jumps, and the justification for a jump is based on accumulated experience rather than a single case.

Basically, there are two ways out: you can accumulate the cases and run a batch learning program now and then, or you can work concurrently with several models.

If you accumulate cases and regularly perform batch learning, you may easily run into space problems, and you will look for a compact way of representing past cases. Bayesian networks are very well suited for this. You store a Bayesian network of acceptable size representing the past cases (see Section 3.2) together with a number indicating the number of cases. When batch learning is performed, you sample the appropriate number of cases from the network (see Section 5.7). However, if you represent the past with a Bayesian network, you introduce bias, and when you sample you reproduce the bias. You also have the problem that when you sample, you reproduce the space problem you tried to avoid.

The second way is similar to the "expert disagreement approach." Assume that you have three alternative models M_1, M_2, M_3 with initial normalized weights w_1, w_2, w_3. A case with evidence e is entered into all mod-

els, and propagation yields $P_i(A \mid e)$ as well as $P_i(e)$, where P_i represents the distribution for M_i and A is any variable. Then,

$$P(A \mid e) = w_1 P_1(A \mid e) + w_2 P_2(A \mid e) + w_3 P_3(A \mid e).$$

We interpret w_i as the probability of the model given the past. To calculate the new weights, we have

$$w_i := P(M_i \mid e) = \frac{P(e \mid M_i)P(M_i)}{P(e)} = \frac{w_i P_i(e)}{\sum_j w_j P_j(e)}.$$

Usually, you will have a set of *questionable* links: there may or may not be a link from A to B. For each questionable link, the difference is the domain of a single potential. If you have n questionable links, there are up to 2^n different models to investigate. However, because the models are very similar, there will be much saving in the calculation task.

For example, you may start off with the largest M and choose to work with models that only deviate from M on a single questionable link. It is then very easy to calculate $P_i(e)$ for all models M_i, and if the new model becomes the most likely, it is also easy to shift to the models deviating from the new model with only one questionable link (although there is a problem with the initial weights for the new models in focus). We will not go into detail about this.

The alternative models approach should be extended by also letting the models adapt the parameters. In this case, each model is represented as a set of conditional probabilities with attached counts and virtual sample sizes. Whenever a new case arrives, fractional updating, for example, can be performed for each model before the new weights are calculated. As for the calculation of weights, if the alternative models are based on questionable links, much calculation is saved.

3.4 Tuning

We have a Bayesian network BN. For this network, we have some evidence e, and for a particular variable A we have $\mathbf{x} = P(A \mid e) = (x_1, \ldots, x_n)$. We may have a prior request $\mathbf{y} = (y_1, \ldots, y_n)$ for $P(A \mid e)$, so we want to tune the network such that $P(A \mid e) = \mathbf{y}$. Assume that the structure of BN is fixed, but for the conditional probabilities we have some freedom described by a set of modifiable parameters $\mathbf{t} = (t_1, \ldots, t_m)$ with an initial set of values $\mathbf{t}_0$. We want to set the parameters so that $P(A \mid e)$ is sufficiently close to $\mathbf{y}$.

If $(t_1, \ldots, t_n)$ are parameters in the Bayesian network BN over the universe U, then $P(U)$ is a function of $(t_1, \ldots, t_n)$, as are also $P(A \mid e)$ and $P(e)$. We must make clear what is meant by "the probabilities are functions

of the parameters." Let A be a binary variable, and let π be a configuration of A's parents $pa(A)$. Then, $t = P(A = a \mid \pi)$ is a parameter, but consequently we have $P(A = \neg a \mid \pi) = 1 - t$, and it covaries with t. If A has more than two states, we assume *proportional scaling*: the remaining probabilities are scaled with the same factor. If A has n states, and a_1 is a parameterized state, we assume that $P(A \mid \pi) = (t, (1-t)x_2, \ldots, (1-t)x_n)$, where $\sum x_i = 1$.

It is possible to deal with several parameters in the same distribution. If, for example, the first two states are parameterized, we would require $P(A \mid \pi) = (t, s, (1 - t - s)x_3, \ldots, (1 - t - s)x_n)$. Then, s does not scale when t is changed. In the following, we assume proportional scaling, and we also assume that there is at most one parameter per distribution.

The task is to set the parameters such that the distance $\text{dist}(\mathbf{x}, \mathbf{y})$ is as small as possible. If the parameters cannot be set in such a way that the distance is close to zero, it is an indication of an incorrect structure.

If it is possible to determine $\text{dist}(\mathbf{x}, \mathbf{y})$ as a function of $\mathbf{t}$, you might be so fortunate that the problem can be solved directly. However, usually the problem cannot be solved directly even when the function is known, and a *gradient descent* method can be used

1. Calculate **grad** $\text{dist}(\mathbf{x}, \mathbf{y})$ with respect to the parameters $\mathbf{t}$.
2. Give $\mathbf{t}_0$ a displacement $\triangle\mathbf{t}$ in the direction opposite to the direction of **grad** $\text{dist}(\mathbf{x}, \mathbf{y})(\mathbf{t}_0)$; that is, choose a step size $\alpha > 0$ and let $\triangle\mathbf{t} = -\alpha$**grad** $\text{dist}(\mathbf{x}, \mathbf{y})(\mathbf{t}_0)$.
3. Iterate this procedure until the gradient is close to $\mathbf{0}$.

From the definitions of the two distance measures, we see

$$\frac{\partial}{\partial t}\text{dist}_Q(\mathbf{x}, \mathbf{y}) = \sum_i 2(x_i - y_i)\frac{\partial x_i}{\partial t}$$

and

$$\frac{\partial}{\partial t}\text{dist}_K(\mathbf{x}, \mathbf{y}) = -\sum_i \frac{y_i}{x_i}\frac{\partial x_i}{\partial t}.$$

The y_i's are known, and the x_i's are available through updating in BN, so what we need are **grad** $x_i(\mathbf{t})$ for all i.

If the variable A is binary, we have $\mathbf{x} = (x, 1 - x)$, $\mathbf{y} = (y, 1 - y)$, and

$$\text{dist}_Q(\mathbf{x}, \mathbf{y}) = 2(x - y)^2,$$

$$\text{dist}_K(\mathbf{x}, \mathbf{y}) = y(\log_2 y - \log_2 x) + (1 - y)(\log_2(1 - y) - \log_2(1 - x)),$$

and

$$\textbf{grad}\ \text{dist}_Q(\mathbf{x}, \mathbf{y}) = 4(x - y)\textbf{grad}x, \tag{3.1}$$

$$\mathbf{grad}\ \mathrm{dist}_K(\mathbf{x},\mathbf{y}) = \left(\frac{(x-y)}{x(1-x)}\right)\mathbf{grad}x.$$

From these formulas, we see that the gradient is $\mathbf{0}$ if and only if either x is independent of all the parameters or $x = y$.

3.4.1 Example

Let BN be the Bayesian network in Figure 3.6 with initial probabilities from Table 3.6. Let C be the observation variable and A the variable of interest. Assume also that the parameters are $t = P(\neg a)$ and $s = P(\neg c \mid \neg b)$. Initially, we have $\mathbf{t}_0 = (0.5, 0.4)$.

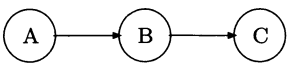

FIGURE 3.6. A small Bayesian network for illustration.

$B \setminus A$	a	$\neg a$
b	1	0.3
$\neg b$	0	0.7

$C \setminus B$	b	$\neg b$
c	1	0.6
$\neg c$	0	0.4

TABLE 3.6. Parameters for the network in Figure 3.6. $P(A) = (0.5, 0.5)$.

Assume that we require $P(A \mid c) = (0.4, 0.6) = (y, 1-y)$. Through updating, we get $x = P(a \mid c) = 0.58$. We calculate $P(a \mid c)$ as a function of $\mathbf{t}$:

$$P(A,c) = \sum_B P(A)P(B \mid A)P(B \mid c) = (1-t, t-0.7ts),$$

$$P(a \mid c) = \frac{P(a,c)}{\sum_A P(A,c)}.$$

We get

$$P(a \mid c) = x(t,s) = \frac{(1-t)}{(1-0.7ts)}.$$

The request is

$$\frac{1-t}{1-0.7ts} = 0.4,$$

which yields

$$s = \frac{t-0.6}{0.28t} = \frac{25}{7} - \frac{15}{7t}.$$

The set of parameters t meeting the request is shown in Figure 3.7.

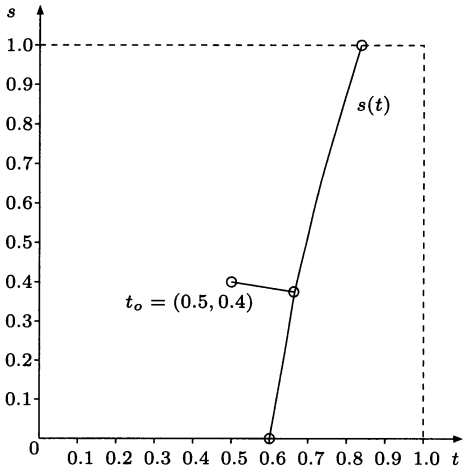

FIGURE 3.7. The graph of $s(t)$ consists of the parameter pairs (t, s) meeting the request $P(a \mid c) = 0.4$.

Out of the infinite number of parameter pairs $(t, s(t))$, we choose one. If we do not wish to choose any of the extremes $(0.6, 0)$ or $(\frac{5}{6}, 1)$, it would be natural to choose the point closest to $\mathbf{t}_0 = (0.5, 0.4)$. This point is characterized by the property that the normal contains $\mathbf{t}_0$ (see Figure 3.7). Through standard calculations, we get the following equation in t:

$$t^4 - \frac{1}{2}t^3 + \frac{666}{98}t - \frac{225}{49} = 0.$$

A root is $t = 0.668$, and we get $s = 0.364$. For this very simple example, it was possible to calculate the closest parameter setting meeting the request. The situation need not be much more complex before a direct calculation becomes intractable.

The gradient descent method will in this example go as follows

$$\mathbf{grad}\ x(t) = \frac{1}{(1 - 0.7ts)^2}(0.7s - 1, (1 - t)0.7t),$$

$$\mathbf{grad}\ x(t_0) = (-0.97, 0.24).$$

Formula (3.1) yields

$$\begin{aligned} \mathbf{grad}\ \mathrm{dist}_Q(\mathbf{x}, \mathbf{y}) &= 4(0.58 - 0.4)(-0.97, 0.24) \\ &= (-0.70, 0.18). \end{aligned}$$

Using a step size of 0.2, we get

$$\triangle \mathbf{t} = (0.14, -0.036)$$

and

$$\mathbf{t}_1 = (0.640, 0.364); P^1(a \mid c) = 0.43.$$

The process is repeated

$$\mathbf{grad}\ x(t_1) = (-1.06, 0.23),$$
$$\mathbf{grad}\ \mathrm{dist}_Q(x, y) = (-0.13, 0.03),$$
$$\mathbf{t}_2 = (0.686, 0.358); P^2(a \mid c) = 0.380.$$

Repeating once more yields

$$\mathbf{t}_3 = (0.672, 0.361); P^3(a \mid c) = 0.395.$$

3.4.2 Determining $P(A \mid e)$ as a function of t

The gradient descent method seems to require that we can calculate $\mathbf{x}$ as a function of the parameters $\mathbf{t}$. It was possible for the preceding small example, but the method used will in general be intractable.

However, the partial derivatives can be calculated without knowing $\mathbf{x}$ as a function of the parameters. The following theorem, which is proved in Chapter 6, can be used.

Theorem 3.1 *Let BN be a Bayesian network over the universe U. Let t be a parameter and let e be evidence entered in BN. Then, assuming proportional scaling, we have*

$$P(e)(t) = \alpha t + \beta,$$

where α and β are real numbers.

The theorem shows that the probability of evidence is a linear function in each parameter. This yields, for example, that $P(e)$ as a function of two parameters has the form $\alpha t + \beta ts + \gamma s + \delta$.

Because "$A = a$" can be treated as evidence, we also have that $P(A = a, e)$ is a linear function in t, and we have the following corollary.

Corollary 3.1 Let BN be a Bayesian network over the universe U with parameter t. Let $A \in U$, let a be a state of A, and let e be evidence. Then, assuming proportional scaling, we have

$$P(A = a \mid e)(t) = \frac{\alpha t + \beta}{\gamma t + \delta},$$

where α, β, γ, and δ are real numbers.

The constants in Theorem 3.1 and Corollary 3.1 can be determined by giving the parameter different values and propagating (as mentioned in Section 1.4.4, a side effect of propagation is a calculation of $P(e)$). For example, to determine $P(e)$, use two different parameter values t' and t'' and determine $x_1 = P(e)(t')$ and $x_1 = P(e)(t'')$. Using Theorem 3.1, you get two linear equations for determining α and β. In the next section, a more efficient method is given.

3.4.3 Explicit modeling of parameters

If your system does not support parameter variation, you can model parameters explicitly in a Bayesian network. This explicit modeling can be an aid for calculating the coefficients in Section 3.4.2 as well as the gradients for tuning.

Construction Let $t = P(A = a \mid \pi)$ be a parameter with initial value t_o and π a configuration of $pa(A)$. For the sake of notation, let $P(A \mid \pi)(t) = (t, (1-t)x_2, \ldots, (1-t)x_n)$. Give A a new parent T with states 0 and 1 and with prior distribution $(1 - t_o, t_o)$. Set $P^*(A \mid pa(A), T) = P(A \mid pa(A))$ except for the configuration π, where we set $P^*(A \mid \pi, T = 0) = P(A \mid \pi)(0)$ and $P^*(A \mid \pi, T = 1) = P(A \mid \pi)(1)$. The construction is illustrated in Figure 3.8.

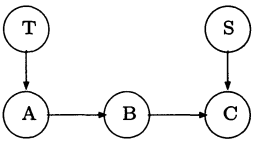

FIGURE 3.8. Explicit modeling of the parameters t and s for the example in Figure 3.6.

$C \setminus B$	b	$\neg b$
c	(1, 1)	(1, 0)
$\neg c$	(0, 0)	(0, 1)

$P(C \mid B, S)$

TABLE 3.7. A table for the network in Figure 3.8. $P(S) = (0.6, 0.4)$.

Through this construction, the original conditional probability distribution $P(A \mid pa(A))$ is decomposed into $P_0(A \mid pa(A)) = P^*(A \mid pa(A), T = 0)$ and $P_1(A \mid pa(A)) = P^*(A \mid pa(A), T = 1)$, and we have

$$\begin{aligned} P(A \mid pa(A)) &= P^*(A \mid pa(A), T = 1)t_0 + P^*(A \mid pa(A), T = 0)(1 - t_0) \\ &= \sum_T P^*(A \mid pa(A), T)P^*(T). \end{aligned}$$

Also setting the state of T to 0 corresponds to setting the parameter $t = 0$ in BN, and setting $T = 1$ corresponds to setting $t = 1$ in BN.

Proposition 3.1 *Let BN be a Bayesian network over the universe U and with probability distribution P. Let t be a parameter as previously de-*

scribed with initial value t_0. Let BN^* be the result of applying the previous construction to BN, and let P^* denote the probability distribution of BN^*. Let $e_1, \ldots, e_n$ be evidence on U.

Then,

$$P^*(U \mid e) = P(U \mid e)(t_0),$$

and hence for all $X \in U$ we have

$$P^*(X \mid e) = P(X \mid e)(t_0).$$

Proof: By the construction, we have

$$\sum_T P^*(A \mid pa(A), T)P^*(T) = P(A \mid pa(A))(t_0).$$

Using the chain rule, we get

$$\begin{aligned}
P^*(U,e) &= \sum_T P^*(U,T,e) \\
&= \sum_T P^*(T)P^*(A \mid pa(A),T) \prod_{X \in U \setminus \{A\}} P(X \mid pa(x)) \prod_i \underline{e}_i \\
&= P(A \mid pa(A))(t_0) \prod_{X \in U \setminus \{A\}} P(X \mid pa(X)) \prod_i \underline{e}_i \\
&= \prod_{X \in U} P(X \mid pa(x)) \prod_i \underline{e}_i \\
&= P(U,e)(t_0).
\end{aligned}$$

□

Proposition 3.1 yields that we can model the parameters explicitly, and by changing the prior probabilities of the parameter nodes it is easy to experiment with various parameter values. The explicit modeling can be further exploited. When evidence e has been entered and propagated, we get a distribution $P(T \mid e)$. Actually, the systems for updating calculate $P(T, e)$, and from these two numbers we get $P^*(T \mid e)$ as well as $P^*(e) = P(e)$.

Proposition 3.2 *Let BN be a Bayesian network with evidence e, and let BN^* be as before with the parameter $\mathbf{t_0}$ modeled explicitly. Put $P(e) = P^*(e) = y$ and $P^*(T = 1 \mid e) = x$. Then,*

$$P(e) = \alpha t + \beta$$

with

$$\alpha = y\frac{(x - t_0)}{t_0(1 - t_0)}, \qquad (t_0 \neq 0, 1),$$

$$\beta = y\frac{1 - x}{1 - t_0},$$

so if there are several parameters in BN^*, then the coefficients with respect to each parameter can be calculated through the same propagation.

Proof: We have

$$y = \alpha t_0 + \beta,$$

$$x = P^*(T = 1 \mid e) = \frac{P^*(e \mid T = 1)P(T = 1)}{P^*(e)} = \frac{P(e)(1)t_0}{y} = \frac{(\alpha + \beta)t_0}{y}. \tag{3.2}$$

Hence,

$$xy = (\alpha + \beta)t_o. \tag{3.3}$$

Solving (3.2) and (3.3) with respect to α and β yields the result.

□

Corollary 3.2 Assumptions as in Proposition 3.2.

1. The gradient of $P(e)$ can be calculated in the explicit model BN^*.
2. Let b be a state of some variable B. Then, the gradient of $P(B = b \mid e)$ can be calculated in the explicit model BN^*.

Hence, we can in the explicit model perform gradient descent tuning.

Proof:

1. The gradient of $P(e)$ is the vector of α's calculated as shown in Proposition 3.2.
2. Enter $B = b$ as evidence together with e and propagate. This yields for each parameter t the coefficients in

   $$P(B = b, e)(t) = \alpha t + \beta.$$

   The rest is standard calculus because $P(B = b \mid e)$ is a fraction of two linear functions.

□

$B \setminus A$	a	$\neg a$
b	$1 - ts$	$1 - s$
$\neg b$	$1 - t$	0

TABLE 3.8. $P(C = c \mid A, B)$ as a noisy or with the parameters t and s.

3.4.4 The example revisited

By modeling the parameters $\mathbf{t} = (t, s)$ explicitly, we get for $\mathbf{t}_0 = (0.5, 0.4)$

$$P(c) = 0.86, P(T = 1 \mid c) = 0.4186, P(S = 1 \mid c) = 0.3023.$$

The formulas in Proposition 3.2 yield

$$P(c) = -0.28t + 1 = -0.35s + 1.$$

We enter $A = a$ and get

$$P(a, c) = 0.5, P(T = 1 \mid a, c) = 0, P(S = 1 \mid a, c) = 0.4,$$

and Proposition 3.2 yields for $\mathbf{t}_0 = (0.5, 0.4)$

$$P(a, c) = -t + 1 = 0s + 0.5.$$

Then,

$$\frac{\partial}{\partial t} P(a \mid c) = \frac{\partial}{\partial t} \frac{1 - t}{1 - 0.28t} = \frac{-0.72}{(1 - 0.28t)^2},$$

$$\frac{\partial}{\partial s} P(a \mid c) = \frac{\partial}{\partial s} \frac{0.5}{1 - 0.35s} = \frac{0.175}{(1 - 0.35s)^2}.$$

We get

$$\mathbf{grad}\ P(a \mid c)(\mathbf{t}_0) = (-0.97, 0.24),$$

and we proceed as in Section 3.4.1.

3.4.5 Dependent parameters and resistance

Compared to neural networks, Bayesian networks have a very large set of simple parameters, and if they are all open for modification, the updating task will be very heavy and convergence may be slow. Therefore, it is a good idea to reduce the number of parameters. This can be done, for example, through *divorcing*.

A standard technique such as *noisy or* will also reduce the number of parameters. However, in "noisy or" the parameters are not simple. For example, take the conditional table in Table 3.8. The three simple parameters $(x_1, x_2, x_3) = (P(c \mid a, b), P(c \mid a, \neg b), P(c \mid \neg a, b))$ in the table are

functions of (t, s), and instead of the partial derivatives with respect to (x_1, x_2, x_3), we need the partial derivative of a composed function with respect to (t, s). From elementary calculus, we have

$$\frac{\partial f}{\partial t} = \sum_i \frac{\partial f}{\partial x_i} \cdot \frac{\partial x_i}{\partial t} = -s\frac{\partial f}{\partial x_1} - \frac{\partial f}{\partial x_2},$$

and the requested gradient is easy to calculate from the gradient with respect to the simple parameters.

This technique can also be used in situations where simple parameters covary. In temporal networks, for example, some simple parameters are identical, or it may happen that an effect is damped by a constant factor from time slice to time slice.

In the tuning procedure, we have only the choice of declaring a variable modifiable or not. There may be reasons why we would prefer to change some parameters rather than others. Some parameters may, for example, be based on a large amount of known cases, and others may be rather ignorant guesses.

Therefore, we may attach a positive real number to each modifiable parameter expressing its resistance to change. Assume in our example that you are much more certain of t than you are of s. You may quantify this to a pair of *resistance measures* (5, 1). Instead of a displacement in the opposite direction of the gradient, you divide each component with the appropriate resistance, and we get the direction (0.14, −0.18). Because each component is multiplied by a positive number, this direction is also a direction of decreasing distance. By repeating the procedure, you will reach another point of the curve in Figure 3.7.

There are ways of quantifying the resistance. If, for example, you have virtual sample sizes attached to the parameters, they can serve as resistance measures.

If you have intervals for the parameters, you would like to constrain the tuning procedure so it does not cross the interval bounds. To keep the parameter t within the interval $(a; b)$, you can attach a resistance that is a function of t, such as $\frac{1}{(b-t)(t-a)}$.

3.5 Summary

Distance measures

$$\text{dist}_Q(\mathbf{x}, \mathbf{y}) = \sum_i (x_i - y_i)^2,$$

$$\text{dist}_K(\mathbf{x}, \mathbf{y}) = \sum_i y_i(\log_2 y_i - \log_2 x_i).$$

Both distance measures are based on strictly proper scoring rules.

Batch learning

Size measure:

$$Size(M) = \sum_{A \in U} Sp(A),$$

where $Sp(A)$ is the number of entries in A's potential.

Acceptance measure:

$$\text{Acc}(P, M^*) = Size(M^*) + k\text{dist}(P, P^*).$$

General search method: Choose a threshold t for $\text{dist}(P, P^*)$ and a k for $\text{Acc}(P, M^*)$. Among the models with distance to P less than t, choose one of minimal $\text{Acc}(P, M^*)$.

Heuristic for searching through the models: Partition the variables into sets $Z_1, \ldots, Z_m$ and establish a partial order $\leq$ for the sets such that links between nodes must follow the partial order. Whenever a model M^* with a distance beyond the threshold t is reached, we need not consider submodels of M^*.

Adaptation

Adaptation through type variables: The second-order uncertainty can be characterized as uncertainty about which table out of $\mathbf{t}_1, \ldots, \mathbf{t}_m$ is the correct one for $P(A \mid pa(A))$.

Add a type variable T with states $t_1, \ldots t_m$ and with A as child. The prior probability $P(t_1, \ldots, t_m)$ reflects your belief in the various tables. Put $P(A \mid pa(A), t_i) = \mathbf{t}_i$.

Whenever a case e has been processed, the probability $P(t_1, \ldots t_m \mid e)$ is used as the new prior for the next case.

Fractional updating: Assume that the second-order uncertainty obeys both the global and local independence requirements. For each parent configuration π, choose a fictitious sample size n expressing the present certainty of $P(A \mid \pi)$. This yields a fictitious sample size $n_a = nP(a \mid \pi)$ for the configuration (a, π).

When a case has been processed, it yields $P(a, \pi \mid e)$. Add $P(a, \pi \mid e)$ to n_a. Thereby, the sample is increased by $P(\pi \mid e)$.

Warning: fractional updating reduces the second-order uncertainty too quickly.

Fading: Instead of counting up with n_a, first multiply the counts for π by a fading factor. A fading factor q can be established from an effective sample size s^*

$$q = \frac{(s^* - P(\pi \mid e))}{s^*}.$$

The alternative model approach: If there is explicit uncertainty in the model – that is, if there are alternative models $M_1, \ldots, M_m$ – they can be weighted initially and run in parallel. After each case, the weights are modified.

Tuning

The set of parameters $\mathbf{t}$ open for modification; $\mathbf{x}(\mathbf{t})$ the current distribution in the model; $\mathbf{y}$ the target distribution.

1. Calculate **grad** $\text{dist}(\mathbf{x}, \mathbf{y})$ with respect to the parameters $\mathbf{t}$.
2. Give $\mathbf{t}_0$ a displacement $\triangle\mathbf{t}$ in the direction opposite to the direction of **grad** $\text{dist}(\mathbf{x}, \mathbf{y})(\mathbf{t}_0)$; that is, choose a step size $\alpha > 0$ and let $\triangle\mathbf{t} = -\alpha$**grad** $\text{dist}(\mathbf{x}, \mathbf{y})(\mathbf{t}_0)$.
3. Iterate this procedure until the gradient is close to $\mathbf{0}$.

We have

$$\frac{\partial}{\partial t}\text{dist}_Q(\mathbf{x}, \mathbf{y}) = \sum_i 2(x_i - y_i)\frac{\partial x_i}{\partial t},$$

$$\frac{\partial}{\partial t}\text{dist}_K(\mathbf{x}, \mathbf{y}) = -\sum_i \frac{y_i}{x_i}\frac{\partial x_i}{\partial t}.$$

Because $P(e)(t) = \alpha t + \beta$, we know that $x_i(t)$ is the ratio of two linear functions, and the partial derivatives can be calculated for all parameters through two propagations (Chapter 5).

Explicit modeling of parameters: Let $t = P(A = a \mid \pi)$ be a parameter with initial value t_o and π a configuration of $pa(A)$. For the sake of notation, let $P(A \mid \pi)(t) = (t, (1-t)x_2, \ldots, (1-t)x_n)$. Give A a new parent T with states 0 and 1 and with prior distribution $(1 - t_o, t_o)$. Set $P^*(A \mid pa(A), T) = P(A \mid pa(A))$ except for the configuration π, where we set $P^*(A \mid \pi, T = 0) = P(A \mid \pi)(0)$ and $P^*(A \mid \pi, T = 1) = P(A \mid \pi)(1)$.

The numbers required for the calculation of gradients are then easily accessed by using a normal *BN* system.

Resistance: The gradient method for tuning can be extended with measures of resistance for the various parameters.

3.6 Bibliographical Notes

Learning has a long history in statistics, and learning Bayesian networks from data is a very lively research area. (Mitchell 1997) is an AI-oriented introduction to *machine learning*. Edwards and Havranek (1985) introduce a model selection procedure well suited for Bayesian networks. The relation to Bayesian networks was established through work in the early 1990s ((Fung and Crawford 1990), (Spiegelhalter and Lauritzen 1990), (Dawid and Lauritzen 1993), (Cooper and Herskovits 1992)). (Cowell et al. 1999) is a good reference for further reading, but (Buntine 1994), (Buntine 1996), and (Jordan 1998) are also recommended. An improved version of fractional

updating for Bayesian networks was presented by Spiegelhalter and Lauritzen (1990), and fading is introduced in (Olesen et al. 1992). The tuning method presented was introduced by Jensen (1999), based on work by Russell et al. (1995) and Castillo et al. (1996).

3.7 Exercises

Exercise 3.1 Show that $ES_Q(\mathbf{x},\mathbf{y}) - ES_Q(\mathbf{x},\mathbf{x}) = \sum_i (x_i - y_i)^2$.

Exercise 3.2 (A proof that the quadratic scoring rule is strictly proper)
(*i*) Show that $ES_Q(\mathbf{x},\mathbf{y})$ is minimal if $\mathbf{x} = \mathbf{y}$. (Hint: use the fact that the function $f(x) = x^2 - 2ax$ is minimal for $x = a$.)
(*ii*) Show that if $ES_Q(\mathbf{x},\mathbf{y}) = ES_Q(\mathbf{x},\mathbf{x})$, then $\mathbf{x} = \mathbf{y}$. (Hint: use Exercise 3.1.)

Exercise 3.3 ^{E} You have four binary variables A, B, C, and D. Table 3.9 gives the distribution of 2,000 cases.

		CD			
		yy	*nn*	*ny*	*nn*
AB	*yy*	568	63	27	242
	yn	64	8	3	28
	ny	27	2	7	63
	nn	244	26	62	566

TABLE 3.9. Table for Exercise 3.3.

You have the structural constraint $(A, B) \Rightarrow (C, D)$; that is, links may not go from C or D to A or B. Determine the structures M for which $Size(M) \leq 14$.

Exercise 3.4 Assume that you have a database C with n cases of configurations over the variables V. Let $sp(V)$ denote the set of configurations over V and let $\#(\underline{v})$ denote the number of cases of the configuration v. Define $P^C(\underline{v}) = \#(\underline{v})/n$. Let P^M be a probability distribution over $sp(V)$. Assume that $P^C(\underline{v}) = 0$ if and only if $P^M(\underline{v}) = 0$ and discount these configurations.

(*i*) Show that
$$\begin{aligned} S^M(C) &= -\Sigma_{c\in C} \log_2 P^M(c) \\ &= -\Sigma_{\underline{v}\in sp(V)} \#(\underline{v}) \log_2 P^M(\underline{v}) \\ &= -\Sigma_{\underline{v}\in sp(V)} n P^C(\underline{v}) \log_2 P^M(\underline{v}). \end{aligned}$$

(*ii*) Show that $S^M(C) - S^C(C) = n \cdot \mathrm{dist}_K(P^M, P^C)$.

$B \setminus A$	y	n
y	0.9	0.4
n	0.1	0.6

$P_1(B \mid A)$

$B \setminus A$	y	n
y	0.6	0.4
n	0.4	0.6

$P_2(B \mid A)$

TABLE 3.10. Table for Exercise 3.5.

Exercise 3.5 ^{E} You have the model $A \rightarrow B$ and $P(A) = (0.7, 0.3)$. Two experts give the tables in Table 3.10, and you have no reason to believe more in one expert than in the other.

(*i*) Construct an adaptation model with the experts modeled explicitly.

(*ii*) What happens when you adapt to the following sequence of (A, B) states: $< (n, y)(n, y)(y, n) >$?

(*iii*) Process a sequence of cases with $A = y$ where the state of B is $(n, y, n, y, y, n, n, y, y, y)$. What are your beliefs in the experts now, and what is $P(B \mid A)$?

Exercise 3.6 You have the same model as in Exercise 3.5, but $P(B \mid A)$ is the one in Table 3.11.

$B \setminus A$	y	n
y	0.75	0.4
n	0.25	0.6

TABLE 3.11. Table for Exercise 3.6.

For $P(B \mid A = y)$, you have an initial sample size of 12.

(*i*) Perform fractional updating from the sequence in Exercise 3.5 (*iii*).

(*ii*) Perform fractional updating on the same sequence but with fading factor 0.9.

Exercise 3.7 The network from Example 3.4.1 in its initial state has sample sizes $s_t = 25$, $s_s = 10$, and $s_u = 25$ for the three parameters. It now receives 20 cases with $C = c$ out of which 10 have $A = a$ (the rest have $A = \neg a$). For the cases with $A = a$, all cases have $B = b$, and in the rest, 4 had $B = \neg b$.

(*i*) Adapt the network without fading.

(*ii*) Adapt the network with effective sample sizes 25, 10, and 25 for t, s, and u, respectively.

(*iii*) Adapt the network to the same cases but without the information on B.

Exercise 3.8 Perform the calculations of Example 3.4.1 by use of a direct representation of the parameters t, s.

Exercise 3.9 Assume that in Example 3.4.1 we require $P(A \mid c) = (0.5, 0.5)$, and assume that $t = 0.6$ is fixed.

(*i*) Use the technique from Example 3.4.1 to tune the parameters s and u.

(*ii*) Use the technique from Section 3.4.3 to tune the parameters s and u.

Exercise 3.10 Let D be a child of C, and let C have parents A and B, all variables being binary. $P(A)$ and $P(B)$ have even distributions; $P(D \mid c) = (0.1, 0.9)$, $P(D \mid \neg c) = (0.6, 0.4)$, and $P(c \mid A, B)$ is as specified in Table 3.8. Tune the parameters t, s to the prescribed behavior $P(a \mid d) = 0.8$.

Exercise 3.11 Consult the web site `www.cs.auc.dk/~fvj/Learning.html` for exercises in structural learning.

4
Decision Graphs

A Bayesian network serves as a model for a part of the world, and the relations in the model reflect causal impact between events. The reason for building these computer models is to use them when taking decisions. In other words, the probabilities provided by the network are used to support some kind of decision making. In principle, there are two kinds of decisions, namely *test decisions* and *action decisions*.

A test decision is a decision to look for more evidence to be entered into the model, and an action decision is a decision to change the state of the world. In real life, this distinction is not very sharp; tests may have side effects, and by performing a treatment against a disease, evidence on the diagnosis may be acquired. In order to be precise, we should say that decisions have two *aspects*, namely a test aspect and an action aspect. The two aspects are handled differently in connection with Bayesian networks, and accordingly we treat them separately.

Actions should also be divided into two types, namely *intervening actions*, which force a change of state for some variables in the model, and *non-intervening actions*, where the impact is not a part of the model.

Although both observations and intervening actions change the probability distributions in the model, they are fundamentally different. To highlight this, consider the example in Figure 4.1.

If T yields a temperature of 37°C (99°F), both the probability for *Flu* and for *Sleepy* are decreased. On the other hand, if the fever is set to 37°C by taking aspirin, then it does not cure the flu (although some people seem to believe so), but still the probability for *Sleepy* is reduced due to the

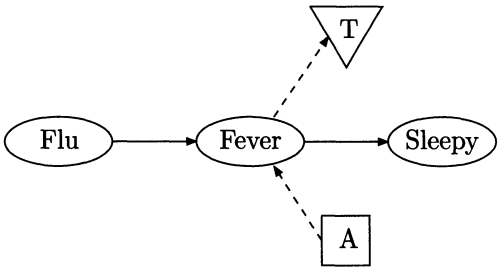

FIGURE 4.1. A simple flu decision model with an action (aspirin) and a test (temperature) attached. The action has no impact on $P(Flu)$.

reduced fever, that is, *the impact of intervening actions can only follow the direction of the causal links.*

The example stresses the important point already made in Section 2.2.6 concerning the use of Bayesian networks. Using Bayes' theorem, it is easy to establish the model in Figure 4.2, which reflects diagnostic reasoning from symptoms to disease.

FIGURE 4.2. A Bayesian network equivalent to the one in Figure 4.1.

From the point of view of entering evidence and propagating probabilities, the two Bayesian networks in Figures 4.1 and 4.2 are equivalent, so why bother emphasizing that the links in the network should be causal links? The difference becomes apparent when taking an aspirin. In Figure 4.2, taking an aspirin will cure the flu but will have no effect on sleepiness.

4.1 One Action

4.1.1 Fold or call?

Consider the poker example in Section 2.1.5 as extended in Exercise 2.11 with variables MH (my hand) and BH (best hand) (see Figure 4.3).

The reason why I am interested in knowing which hand is best is that I shall take a decision on an action. For this game, the rules are that we both placed \$1 on the table to get the initial hand, and after the rounds of card changing, my opponent places \$1 extra (in this game she is forced to place \$1 no matter her hand). Now, I may either *fold* or *call*. If I fold, my opponent takes the pot, and if I call, I place \$1 on the table, and we compare the hands. The player with the best hand takes the pot (in case of a draw we share).

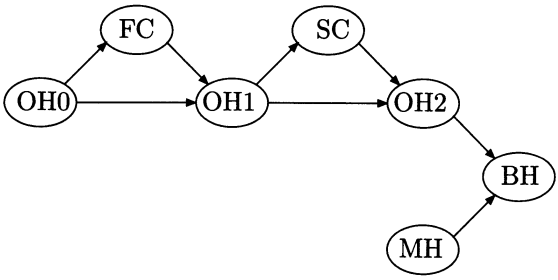

FIGURE 4.3. The poker model extended with variables for my hand and best hand.

My decision problem when deciding to fold or to call can be represented graphically by extending the Bayesian network with a couple of extra nodes. The decision options are represented by a rectangular node D with states *fold* and *call*. Another type of node, U, represents the possible outcomes in \$. U is called a *utility node*, and the outcomes are called *utilities*. The variables determining this outcome are *BestHand* and D, and this is indicated graphically through a directed link from *BestHand* and D to the diamond-shaped node U.

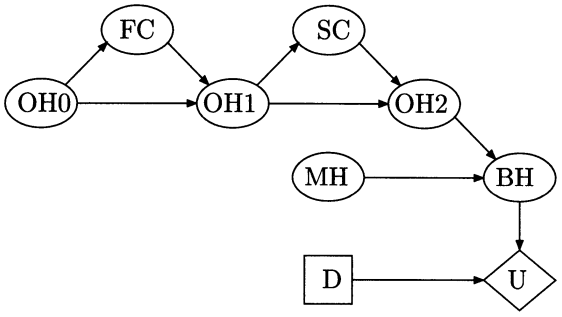

FIGURE 4.4. Graphical representation of my decision problem of whether to fold or call. The variable D is a decision variable. The variable U represents the outcome in \$, and the links into U indicate functional dependencies.

When I have extended the Bayesian network to the preceding model, I will use the model to give advice on the decision D. I have observed my opponent's change of cards (for example, two cards and one card), and I know my own hand (for example, a flush). The probability for *BestHand* is calculated, and it is used to calculate $EU(call)$, *the expected utility* of calling: the sum of the various wins and losses weighted by their probability. The formula for *call* is

$$EU(call) = \sum_{BH} U(BH,\ call)P(BH|evidence).$$

The expected utility of folding is 0, so I call if $EU(call) > 0$.

4.1.2 Mildew

Two months before the harvest of a wheat field, the farmer observes the state Q of the crop, and he observes whether it has been attacked by mildew, M. If there is an attack, he will decide on a treatment with fungicides.

There are five variables:

- Q with states fair (f), not too bad (n), average (a), and good (g);
- M with states *no*, little (l), moderate (m) and severe (s);
- H (state of the crop at time of harvest) with the states from Q plus rotten (r), bad (b), and poor (p) (farmers in all countries tend to describe their harvests in pessimistic terms);
- OQ (observation of Q) with the same states as Q;
- OM (observation of M) with the same states as M.

Furthermore, there is an action node A with actions *no*, light (l), moderate (m), and heavy (h) and a variable M^* describing the mildew attack after the action. H has a table U attached to it. $U(H)$ is a utility function giving the outcome of the harvest for each state of the crop. The cost of the actions is modeled as a table C attached to $A(C(A) \leq 0)$. The total utility is $U + C$. Figure 4.5 gives a model.

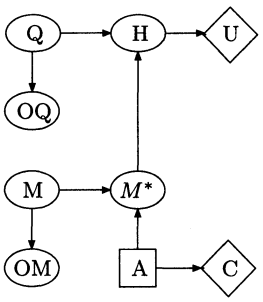

FIGURE 4.5. A decision model for Mildew.

With evidence e (statements on OQ and OM), he wishes to determine the optimal action, that is, he needs to calculate the expected utility of the various options. For each state a of A, calculate $P(H \mid A = a, e)$, and

$$EU(A \mid e) = C(A) + \sum_H U(H)P(H \mid A, e).$$

4.1.3 One action in general

The general situation with one decision variable is as described in Figure 4.6. There is a Bayesian network structure with chance nodes and directed links. The network is extended with a single decision node D. D may have an impact on the structure. In other words, there may be a link from D to some chance nodes. Furthermore, there is a set of utility functions, $U_1, \ldots, U_n$, over domains $X_1, \ldots, X_n$. Graphically, a utility function is represented as a diamond-shaped node with incoming links from the nodes in its domain.

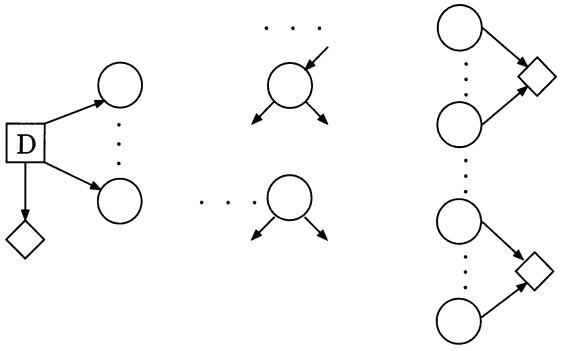

FIGURE 4.6. A graphical representation of a one-action decision scenario.

The task is to determine the action that yields the highest expected utility, that is, with evidence e achieved, we calculate

$$EU(D \mid e) = \sum_{X_1} U_1(X_1)P(X_1 \mid D, e) + \ldots + \sum_{X_n} U_n(X_n)P(X_n \mid D, e),$$

and a state d maximizing $EU(D \mid e)$ is chosen as an optimal action.

Note In the treatment of Bayesian networks, we have so far only considered situations where we request posterior probabilities for single variables. This may not be sufficient because X_i may contain more than one variable. In Chapter 6, we will address the task of calculating joint distributions for sets of variables.

4.2 Utilities

We treat decision problems in the framework of *utility theory*. Decisions are made because they may be of use in some way. Therefore, the various decisions should be evaluated on the basis of the usefulness of their consequences. We assume that "usefulness" is measured on a numerical scale called a *utility scale*, and if several kinds of utilities are involved in the same decision problem, then the scales have a common unit. This assumption may seem dubious; it is treated in the extensive literature on utility theory (e.g., (Lindley 1971) and (Winterfeldt and Edwards 1986)).

4.2.1 Management of effort

> In your computer science studies, you attend two courses, *Graph Algorithms* and *Decision Support Systems*. In the middle of the term, you realize that you cannot keep pace. You can either reduce your effort in both courses slightly or you can decide to attend one of the courses superficially only. What is the best decision?

You have three possible actions:

Gd Keep pace in Graph Algorithms and follow Decision Support Systems superficially.

SB Slow down in both courses.

Dg Keep pace in Decision Support Systems and follow Graph Algorithms superficially.

The results of the actions are your final marks for the courses. The marks are integers between 0 and 5, where 0 and 1 are failing marks. You have certain expectations for the marks given your effort in the rest of the term. They are shown in Table 4.1.

A way of solving your decision problem would be to say that the numeric value of the mark is a utility, and you want to maximize the sum of the expected marks. The calculations would then be

$$EU(Gd) = \sum_{m \in GA} P(m \mid kp)m + \sum_{m \in DSS} P(m \mid fs)m = 3.5 + 2.3 = 5.8,$$

$$EU(SB) = \sum_{m \in GA} P(m \mid sd)m + \sum_{m \in DSS} P(m \mid sd)m = 2.9 + 3.2 = 6.1,$$

$$EU(Dg) = \sum_{m \in GA} P(m \mid fs)m + \sum_{m \in DSS} P(m \mid kp)m = 2.3 + 3.9 = 6.2.$$

	kp	*sd*	*fs*
0	0	0	0.1
1	0.1	0.2	0.1
2	0.1	0.1	0.4
3	0.2	0.4	0.2
4	0.4	0.2	0.2
5	0.2	0.1	0

$P(GA \mid effort)$

	kp	*sd*	*fs*
0	0	0	0.1
1	0	0.1	0.2
2	0.1	0.2	0.2
3	0.2	0.2	0.3
4	0.4	0.4	0.2
5	0.3	0.1	0

$P(DSS \mid effort)$

TABLE 4.1. The conditional probabilities of the final marks in Graph Algorithms (*GA*) and Decision Support Systems (*DSS*) given the efforts *keep pace* (*kp*), *slow down* (*sd*), and *follow superficially* (*fs*).

From this, you would conclude that you should follow Graph Algorithms superficially but keep pace in Decision Support Systems.

However, do the marks really reflect your utilities? If, for example, you had the same number of marks but the numeric values were $0, 5, 6, 8, 9, 10$, you would have come to another conclusion. The problem is that you cannot expect that a difference of 1 in mark number always represents the same difference in utility. Actually, in this case your subjective utility is not increasing in the numeric value of the mark: the rule at your university that if you fail, you are given another chance, but if you pass, you are not allowed to try again to get a better mark. Therefore, you find that the worst mark to get is a 2 rather than a 0!

To overcome this problem, the mark scale is mapped into a utility scale going from 0 to 1. The best possible mark (5) is given the utility 1, and the worst possible mark (2) gets the utility 0.

The intermediate marks are given utilities by imagining that you have a choice between two games:

> *Game 1:* You get for certain the mark x;
>
> *Game 2:* You get mark 5 with probability p, and you get 2 with probability $1 - p$.

Which game would you prefer?

If $p = 0$, you would prefer Game 1, and for $p = 1$ Game 2 would be best. For some p between 0 and 1, you would be indifferent, and *this p is the utility for the mark x.*

In Table 4.2, we have performed the utility assessment for you. The utilities assessed are only for one course. We will now assume that the utility of marks for several courses is the sum of the individual utilities. Note that this is not evident (it might, for example, be that you prefer two 2's to failing both courses, which would delay your studies considerably).

In Figure 4.7, the decision model is illustrated.

Mark	0	1	2	3	4	5
Utility	0.05	0.1	0	0.6	0.8	1

TABLE 4.2. Utilities for the various marks (the same for both courses).

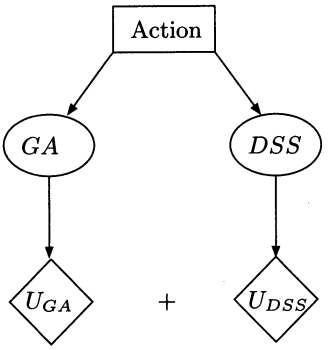

FIGURE 4.7. A decision model for effort.

To find the optimal decision, the calculations are

$$EU(action) = \sum_{m \in GA} P(m \mid action) U_{GA}(m) + \sum_{m \in DSS} P(m \mid action) U_{DSS}(m).$$

We get $EU(Gd) = 1.015, EU(SB) = 1.07, EU(Dg) = 1.045$, and the optimal action is SB.

4.3 Value of Information

As mentioned previously, there is a difference between action decisions and test decisions. In this section, we deal with test decisions.

In the context of Bayesian networks, a *test* is a procedure that discloses the state of one or several variables in the network. Often there is a particular node in the network representing the test outcome (see, for example, Section 2.1.1).

A typical decision situation is that you may choose among some actions, but before deciding on the action you also have the option to perform some tests. The question is which test to perform, if any.

4.3.1 Test for infected milk?

Consider the infected milk scenario from Figure 2.1 and Section 2.2.1 (to keep things simple, we assume infection and test to be independent between the days). The farmer has 50 cows, and the milk from each cow is poured

into a container and transported to the dairy. The dairy checks the milk carefully, and if it is infected it is thrown away. After having milked a cow, the farmer may test the milk before pouring it into the container. The value of the milk is \$2 per cow, and the price of the test is 6 cents. The question is whether he shall perform the test. The test situation can be illustrated graphically as in Figure 4.8.

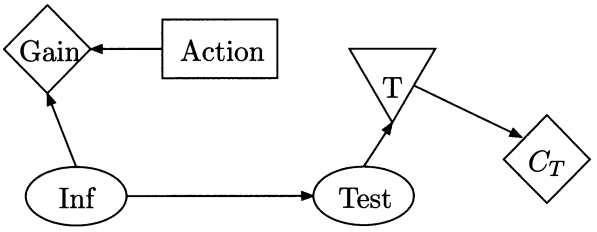

FIGURE 4.8. The test scenario for infected milk. The triangular node represents a test option.

To establish the utilities, let us assume that the farmer has clean milk from the 49 other cows. If the farmer pours the milk into the container, he will gain \$100 if it is not infected (provided the rest of the milk in the container is not infected), and he will gain nothing if it is infected. If he throws the milk away, he will gain \$98 no matter the state of the milk.

If the farmer does not perform a test, the probability of the milk being infected is 0.0007. The expected utility of pouring the milk into the container is

$$EU(pour) = 0.9993 \cdot 100 + 0.0007 \cdot 0 = 99.93.$$

Because the expected utility of pouring the milk into the container is larger than 98, he will do this.

The reason for performing the test is that some outcome will make the farmer change the decision. To put it in another way, if the decision is the same no matter the outcome of the test, then it is not worth the bother to perform it. Only a positive test result may change the current decision. An easy calculation (use your system) yields $P(clean \mid pos) = 0.935$. The expected utility of pouring given a positive test result is

$$EU(pour \mid pos) = 0.935 \cdot 100 = 93.5,$$

so if the test is positive, the farmer changes his decision. The next concern is whether the test is worth its price. There are two possibilities: the test is negative and the milk is poured, or the test is positive and the milk is thrown away. The probability of the first possibility is calculated as 0.9893, and the second possibility has the probability 0.0107. Hence, the expected benefit of performing the test is

$$EU(Test) = 0.9893 \cdot 100 + 0.0107 \cdot 98 = 99.98.$$

The farmer has an increase in expected utility from 99.93 to 99.98 at the price of \$0.06, and it is not worth it to perform the test.

4.3.2 Myopic hypothesis driven data request

In the preceding example, we attached a value to the various information scenarios, namely the expected utility of the optimal action. The driving force for evaluating an information scenario was how the distribution of the variable *Infected?* was affected. We call this kind of data request *hypothesis driven*: the distribution of a hypothesis variable H is the target of the analysis. To formulate it in more general terms, there is a *value function* V attached to the distribution $P(H)$. Usually, the value function is a maximal utility for an action variable A:

$$V(P(H)) = \max_{a \in A} \sum_{h \in H} U(a,h)P(h).$$

If test T with cost C_T yields the outcome t, then the value of the new information scenario is

$$V(P(H \mid t)) = \max_{a \in A} \sum_{h \in H} U(a,h)P(h \mid t).$$

Since the outcome of T is not known, we can only calculate the *expected value*

$$EV(T) = \sum_{t \in T} V(P(H \mid t)) \cdot P(t).$$

The *expected benefit* of performing test T is

$$EB(T) = EV(T) - V(P(H)).$$

The *expected profit* is

$$EP(T) = EB(T) - C_T.$$

The hard part in the calculation of the expected profit is $P(H \mid T)$. This will usually require one propagation per state of T.

If there are several possible tests to perform, we are faced with a new problem. We may calculate the expected profit of each test, but we cannot be sure that the best choice is the one with the highest expected (positive) profit. A proper analysis of the data request situation should consist of an analysis of all possible sequences of tests (including the empty sequence). To avoid such an intractable analysis, the so-called *myopic* approximation is often used: If you are allowed to perform at most one test, which one

will you choose? The answer is the one with the highest expected profit if it is positive.

The myopic approach does not guarantee an optimal sequence. Sometimes a single test does not yield anything by itself, whereas its outcome may be crucial for selecting a second very informative test. We give an example in the next section.

Now, assume you have the tests $T_1, \ldots, T_m$, and let H be the hypothesis variable. To calculate the expected profit for all tests, you need $P(H \mid T_i)$ for each T_i. This can be achieved by propagating each possible outcome of each possible test. It can also be achieved in a simpler way. By propagating the states of H rather than the states of the tests, we get $P(T_i \mid H)$ for all T_i. Bayes' rule yields

$$P(H \mid T_i) = P(T_i \mid H)\frac{P(H)}{P(T_i)}.$$

Because $P(T_i)$ and $P(H)$ are available initially, we do not need more propagations than there are states in H.

4.3.3 Nonutility value functions

If there is no proper model for actions and utilities, the reason for acquiring more information is to decrease the uncertainty of the hypothesis. This means that you will give high values to probabilities close to zero and one, while probabilities in the middle area should have low values. A classical function with this property is *entropy* (see Section 3.1).

The formula for entropy of a distribution over H is

$$ENT(P(H)) = -\sum_{h \in H} P(h)\log_2(P(h)),$$

where $p \log_2 p = 0$ if p vanishes.

Entropy is a measure of how much the probability mass is scattered around on the states. If the distribution is even, and the number of states in H is n, then the entropy is $\log_2 n$. In the limit, if the probability mass is concentrated in one state, the entropy is 0. Because we want the value function to increase with preference, we let an entropy-based value function be

$$\begin{aligned} V(P(H)) &= -ENT(P(H)) \\ &= \sum_{h \in H} P(h)\log_2(P(h)). \end{aligned}$$

Variance

If the states of H are numeric, another classical measure can be used, namely the *variance*. Again, since small variances are preferred, the value

function becomes

$$V(P(H)) = -\sum_{h \in H} (h - \mu)^2 P(h),$$

where $\mu = \sum_{h \in H} hP(h)$.

It is up to the modeler to specify the value function. If decisions with known utilities are attached to the hypothesis variable, then the utility value function should be preferred. If this is not the case, the user will mainly be interested in the precision of a diagnosis.

In the case of a Boolean hypothesis with states 0 and 1, the entropy function is $\log p^p(1-p)^{1-p}$, and the variance function is $-p(1-p)$. These two functions reflect that the value of p increases as it approaches its bounds 0 and 1. The entropy function is rather drastic in the way that the slope is infinite for 0 and 1. Therefore, small changes of p close to 0 and 1 will be highly valued. On the other hand, the variance is of polynomial degree 2, and the slope close to the bounds is 1 and -1, giving changes almost even value no matter how close they are to the bound.

Other value functions

In principle, any value function may be used. However, a particular class of functions called *convex functions* are best suited.

Definition A function $f : R^n \rightarrow R$ is convex if, for any two points P_1, P_2 on the graph of f, the line segment P_1P_2 lies above the graph (see Figure 4.9). Mathematically, the property is expressed as follows

$$\forall t \in [0,1], \forall \underline{x}, \underline{y} \in R^n : tf(\underline{x}) + (1-t)f(\underline{y}) \geq f(t\underline{x} + (1-t)\underline{y}).$$

The reason why a convex function is well suited is due to the following theorem, which we will not prove.

Theorem 4.1 *If the value function is a convex function, then the expected benefit of performing a test is never negative.*

Utility based functions are convex and so are entropy and variance.

4.3.4 Nonmyopic data request

It may happen that no test has a positive expected profit, while the expected benefit for a pair of tests is greater than their costs. In that case, the myopic approach is misleading. Consider the following simple example.

Assume that the farmer from Section 4.3.1 has another test T_2 at hand. T_2, which costs 20 cents, is more precise because the probability of false negatives as well as false positives is 0.001. It does not pay to spend 20 cents to improve the expected gain, which at most can increase from 99.93 to 100. Hence, a myopic analysis will tell the farmer not to perform any

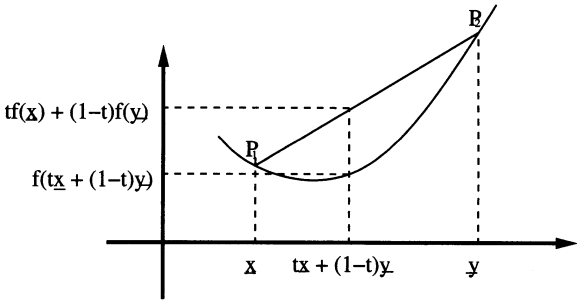

FIGURE 4.9. A convex function. The line segment between two points of the graph lies above the graph.

test. On the other hand, there may be a better strategy, namely to perform T_1, and if it is positive then double check with T_2 before the milk is thrown away. This strategy is illustrated in Figure 4.10.

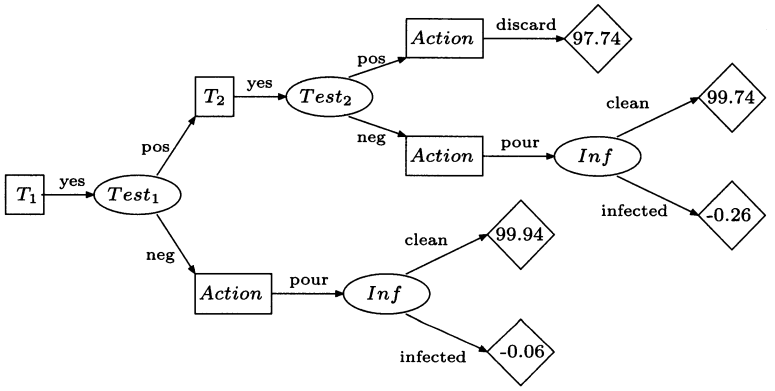

FIGURE 4.10. An alternative strategy to not testing. A path from the root to a leaf represents a sequence of decisions and observations. The numbers in the utility boxes are the gains for the paths.

To calculate the expected gain from the strategy in Figure 4.10, we need the probabilities for the five paths ending in a utility. For the upper path, for example, we need the probability that both $Test_1$ and $Test_2$ are positive. We can get them by using the Bayesian network in Figure 4.11. We get

$$\begin{aligned} EU(Strategy) &= 0.0007 \cdot 97.74 + 0.00998 \cdot 99.74 + 0.98872 \cdot 99.94 \\ &- 0.7 \cdot 10^{-6} \cdot 0.26 - 0.7 \cdot 10^{-5} \cdot 0.06 = 99.94, \end{aligned}$$

and we see that this strategy is slightly better than not to test at all.

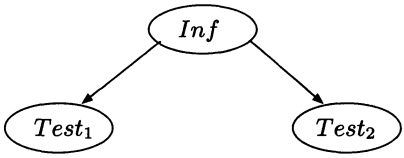

FIGURE 4.11. A Bayesian network for calculating the probabilities for the strategy in Figure 4.10.

4.4 Decision Trees

A classical way of representing decision scenarios with several decisions is *decision trees.*

The nonleaf nodes in a decision tree are decision nodes (rectangular boxes) or chance nodes (circles or ellipses), and the leaves are utility nodes (diamond shaped). The links in the tree have labels. A link from a decision node is labeled with the action chosen, and a link from a chance node is labeled by a state. Figures 4.12 and 4.13 show the graphical part of a decision tree for the milk example with two tests.

A decision tree is read from the root and downward. When you pass a decision node, the label tells you what the decision is, and when you pass a chance node, the label tells you the state of the node. If a decision node follows a chance node, then the node has been observed. We assume *no-forgetting*: when a decision is to be taken, the decision maker knows all the labels on the path from the root down to the current position in the decision tree. We adopt the shorthand *past* for the set of labels from the root to a position in the tree.

The quantitative part of a decision tree consists of probabilities and utilities. Each link from a chance variable has a probability attached to it, and each leaf node has a utility attached to it. Let A be a chance node at a particular position with past o, and let l be an outgoing link labeled with a. We attach $P(A = a \mid o)$ to the link. Let U be a leaf. We attach the utility of its past to U. See Figure 4.14 for an example.

We require decision trees to be complete: from a chance node there must be a link for each possible state, and from a decision node there must be a link for each possible decision option.

It may be impractical to have the probabilities indicated as labels on links. Instead, you may use a Bayesian network as a reference. You can, for example, complement the graphical part in Figures 4.12 and 4.13 with the Bayesian network in Figure 4.11.

4.4.1 A start problem

We now give another example of a decision scenario involving a sequence of decisions. It involves a test and three intervening actions.

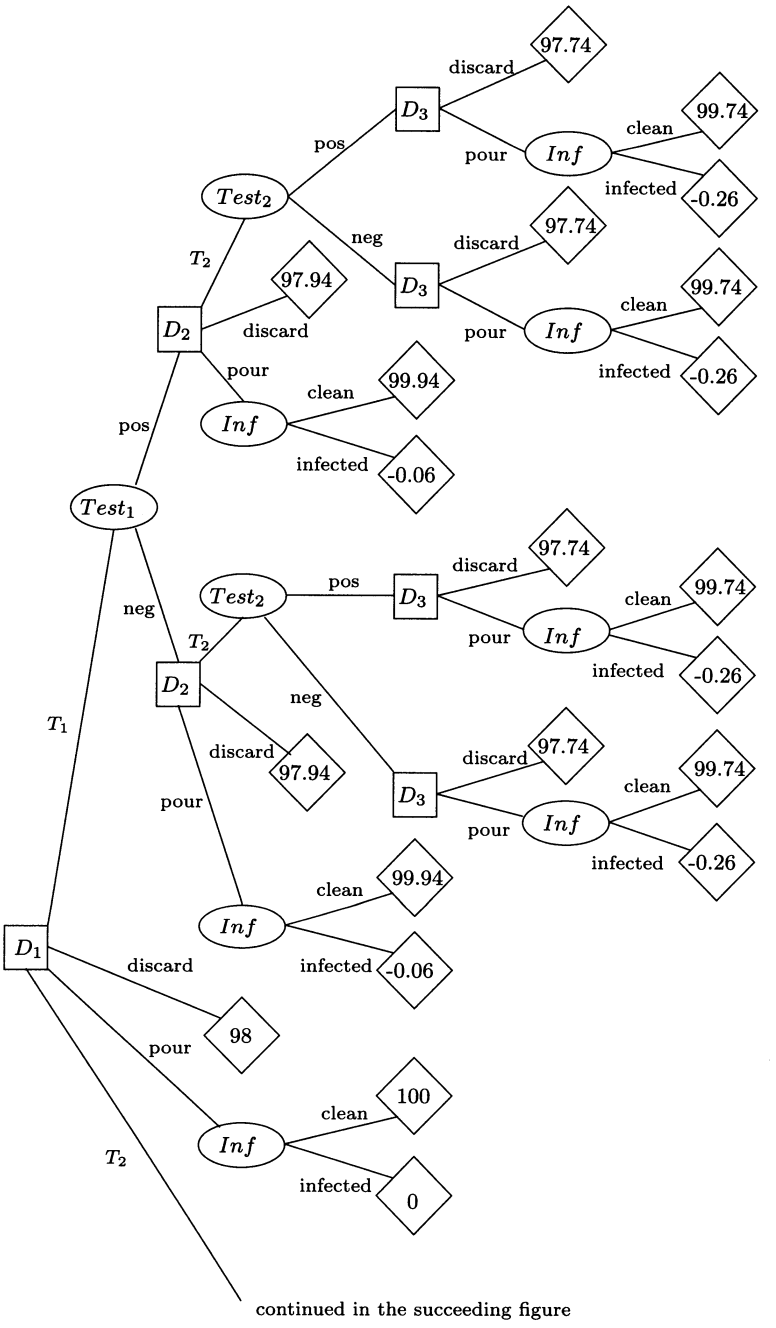

FIGURE 4.12. The graphical part of a decision tree for the milk problem from Section 4.3.4. The tree reflects that no test is performed when the milk has been poured or discarded. Note that nodes in a decision tree may share names.

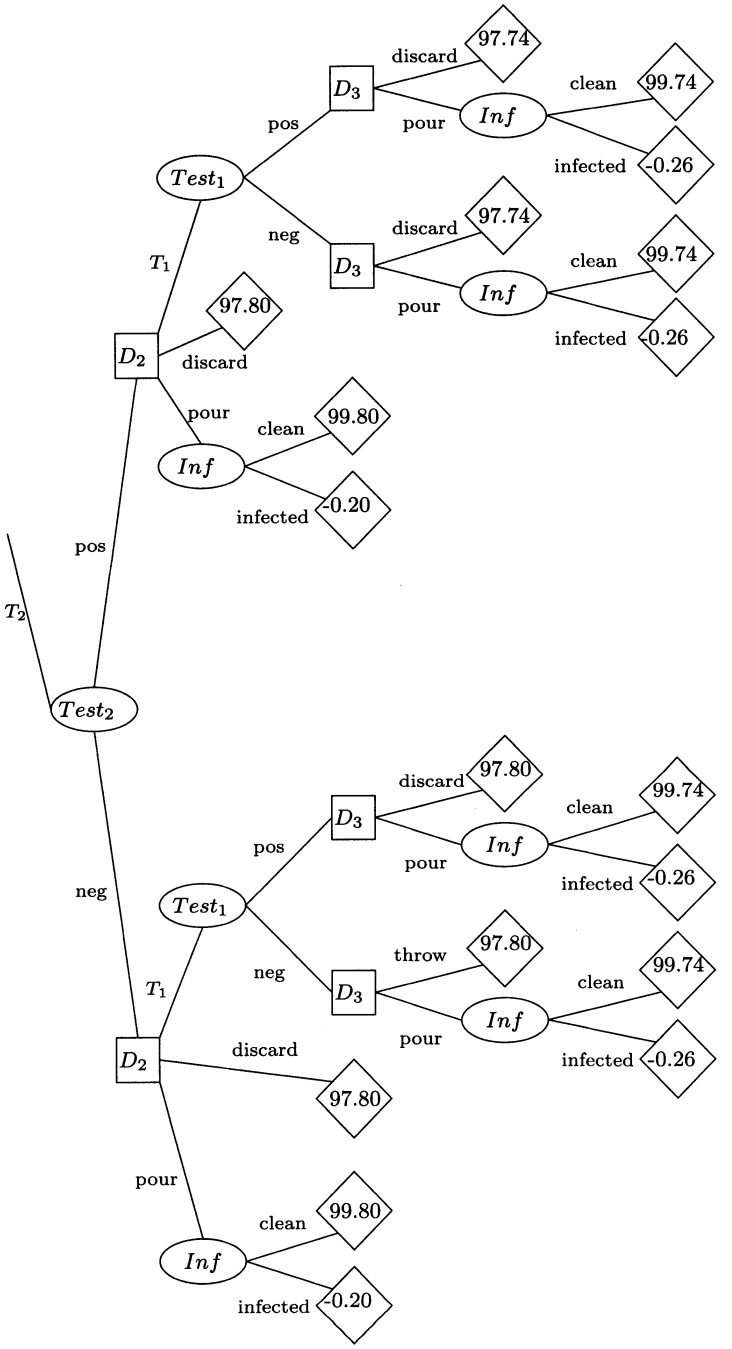

FIGURE 4.13. Continuation of diagram in Figure 4.12.

> In the morning, my car will not start. There are three possible faults: *spark plugs* with probability 0.3, *ignition system* with probability 0.2, and *other* with probability 0.5. I can perform two repair actions myself: SP, which at the cost of 4 minutes always fixes spark plugs; and IS, which takes 2 minutes and fixes the ignition system with probability 0.5. I can perform a test T, namely to check the charge on the spark plugs when starting. It takes half a minute, and it says *ok* if and only if the ignition system is okay. Finally, I can call road service RS, which at the cost of 15 minutes fixes everything. The car was okay yesterday evening, so I assume that there is at most one fault.

To work with utilities rather than costs, let us say that I have 30 minutes to fix the car and arrive at work, and I want to find a test–repair sequence that expectedly gives me as much time as possible for getting to work. Therefore, the utility of a test–repair sequence is the remaining time for getting to work.

A decision tree for this car start problem is shown in Figure 4.14. The probabilities for the decision tree are calculated from the model in Figure 4.15, where the technique from Section 2.3.7 is used.

4.4.2 Solving decision trees

A decision tree is solved by "rolling back": start with nodes that have only leaves as children. If the node is a chance node A, the expected utility for A is calculated. Each child of A has a utility u attached and the link has a probability p. We calculate the product up from each child, and their sum is attached to A. If the node is a decision node D, each child of D has a (expected) utility attached. Choose the child(ren) with maximal expected utility, highlight the link(s), and attach the value to D.

This is done recursively until the root is reached. The resulting value for the root is the expected utility if you adhere to the strategy of always maximizing the utility, and the paths from root to leaves following highlighted links when possible represent a strategy for the decision scenario.

A start problem (continued)

Figure 4.16 illustrates the calculations for solving the troubleshooting problem.

As can be seen from Figure 4.16, the expected utility is 16.96, and the strategy is first to perform the test T, and if it says *ok* then follow with SP and possibly RS. If T says $\neg ok$, then follow with IS and possibly RS. A strategy close to the optimal one is to start performing SP and if unsuccessful to follow with T.

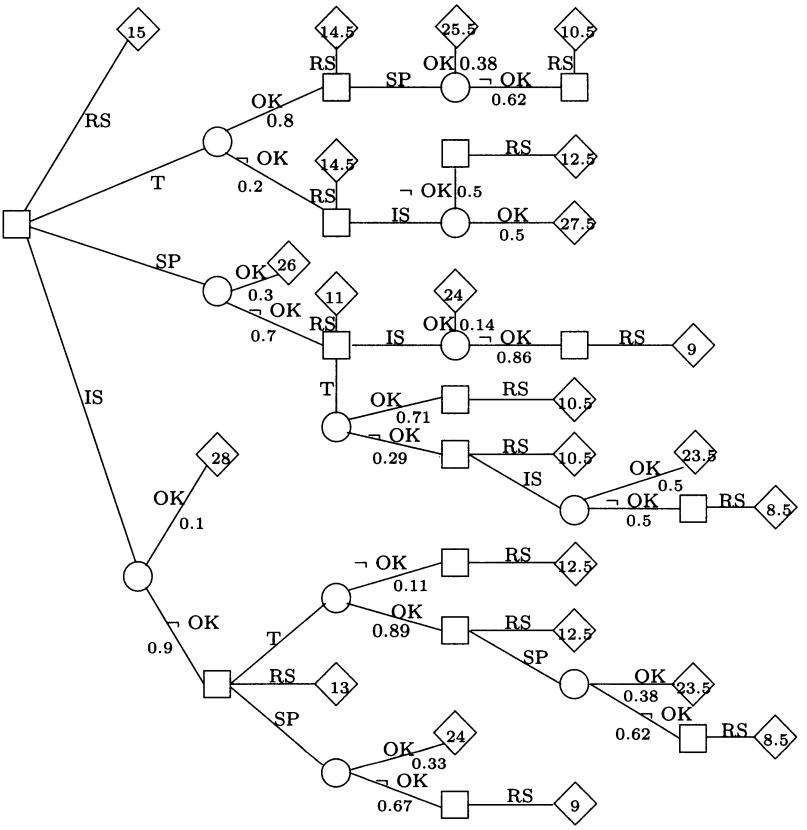

FIGURE 4.14. A decision tree for the current car start problem.

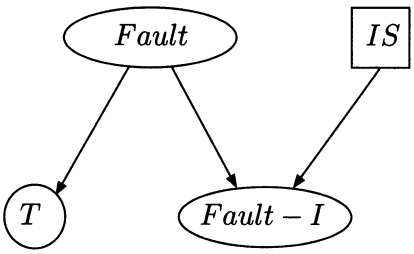

FIGURE 4.15. A model for calculating the probabilities for a decision tree for this car start problem. Due to the assumption of at most one fault, the faults are collected in the node *Fault* with states *is, sp*, and *other*.

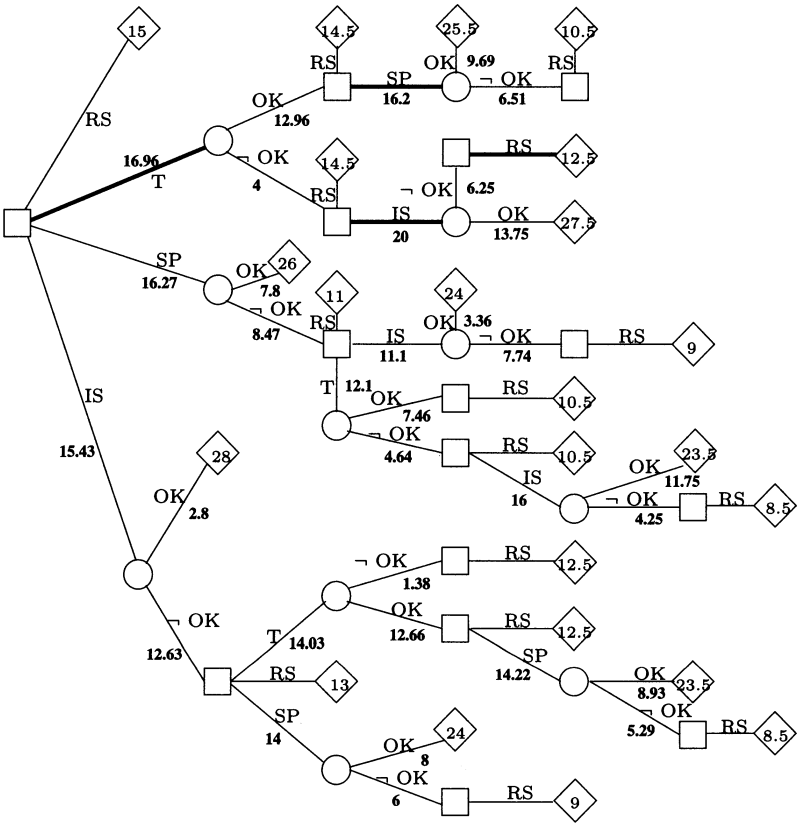

FIGURE 4.16. Results when solving the decision tree from Figure 4.14 (in boldface).

4.4.3 Coalesced decision trees

The main drawback of decision trees is that they grow exponentially with the number of decision and chance variables, and – as illustrated in the two examples – even very small decision scenarios require a very large decision tree. There are methods for reducing the complexity. They exploit symmetries in the decision scenario.

When a decision tree contains identical subtrees, they can be collapsed. In the milk problem, the situation will be the same when both tests are negative no matter the order in which the tests are performed. The succeeding part of the decision tree must be the same, and we can have the two outgoing links meet in a common decision node. Figure 4.17 shows the structure of a coalesced decision tree.

The procedure for solving a coalesced decision tree is the same as the procedure for normal decision trees.

4.5 Decision-Theoretic Troubleshooting

A fault causing a (man-made) device to malfunction is identified and eliminated through a sequence of troubleshooting steps. Some steps are *repair steps*, which may or may not fix the problem, some steps are *observation steps*, which cannot fix the problem but may give indications of the causes of the problem, and some steps have repair aspects as well as observation aspects. All steps have a cost in terms of money, time, or other factors or combinations thereof. The task is to find the cheapest strategy for sequencing the troubleshooting steps. In this section, we deal with pure repair steps and pure observation steps only, and we will call them *actions* and *questions*, respectively.

A troubleshooting problem can be represented and solved through a decision tree. However, as decision trees have a risk of becoming intractably large, we look for ways of pruning the decision tree. Also, a troubleshooting strategy may by itself be intractably large, and we look for ways of stepwise expanding the strategy through local calculations based on the actual past. The car start problems of Sections 1.1.1 and 4.4.1 are examples of troubleshooting tasks.

4.5.1 Action sequences

In this section, we consider a set of steps consisting of actions only. An action, A_i, has two possible outcomes, namely "$A_i = yes$" (the problem was fixed) and "$A_i = no$" (the action failed to fix the problem). Each action, A_i, has a cost $C_{A_i}(\varepsilon)$, which may depend on evidence ε. We sometimes use $C_i(\varepsilon)$ (or C_i) as shorthand for $C_{A_i}(\varepsilon)$. Because there are no questions, a *troubleshooting strategy* is a sequence of actions $s = \langle A_1, \ldots, A_n \rangle$ prescrib-

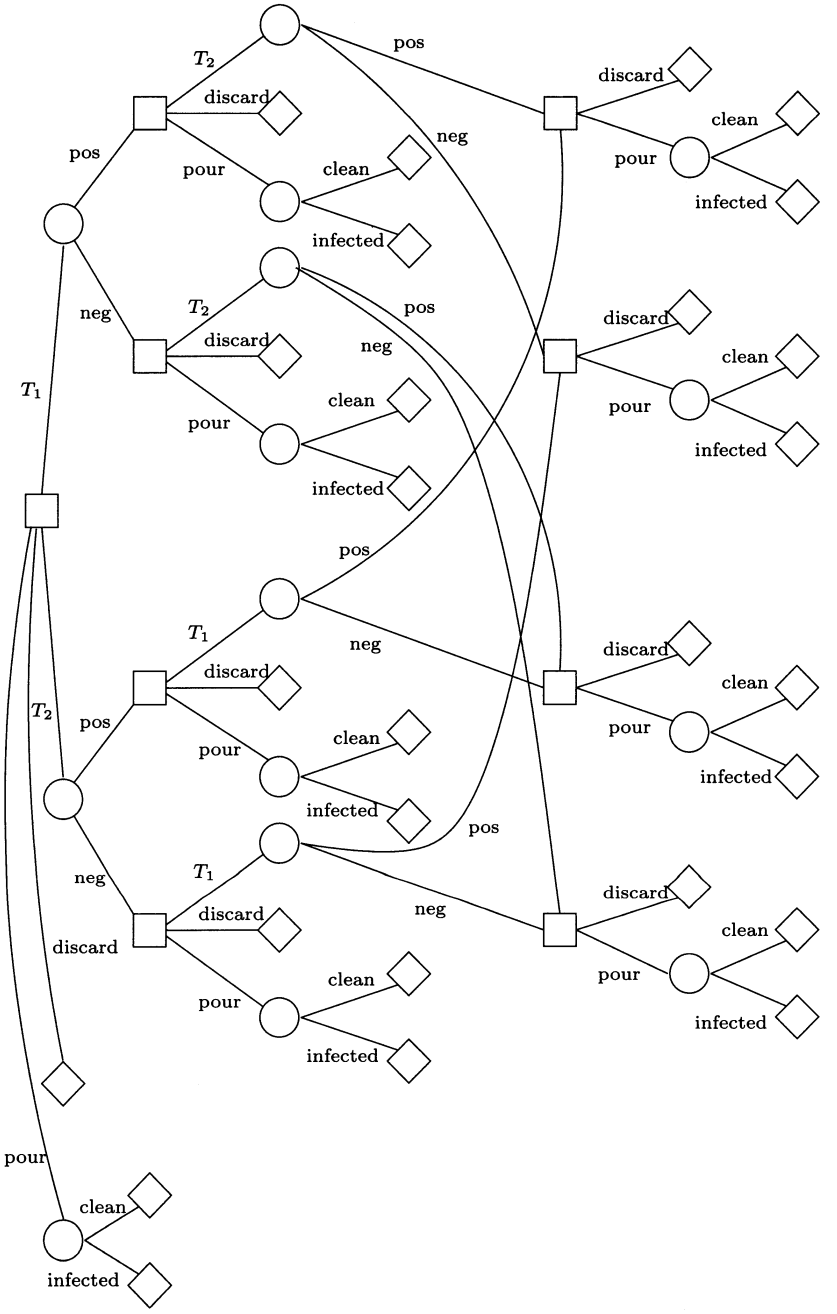

FIGURE 4.17. The structure of a coalesced decision tree for the milk problem.

ing the process of repeatedly performing the next action until an action fixes the problem or the last action has been performed.

When solving a troubleshooting problem, we have some initial evidence ε and in the course of executing actions in the troubleshooting sequence $s = \langle A_1, \ldots, A_n \rangle$ we collect further evidence, namely that the previous actions have failed. We let ε^i denote the evidence that the first i actions have failed, and we refer to a set of failed actions as *simple evidence*. In the following, we will not mention the initial evidence explicitly.

Definition The *expected cost of repair* (ECR) of a troubleshooting sequence $s = \langle A_1, \ldots, A_n \rangle$ with costs C_i is the mean of the costs until an action succeeds or all actions have been performed:

$$ECR(s) \equiv \sum_i ECR_i(s),$$

where

$$ECR_i(s) = C_i(\varepsilon^{i-1})P(\varepsilon^{i-1}).$$

Note that the term "expected cost of repair" may be misleading because we allow a situation where all actions have been performed without having fixed the problem. If this happens, it will happen with the same probability no matter the sequence, and therefore we need not estimate a cost for it. We may also extend the set of actions with a *call service* action, CS. We will return to this in Section 4.5.3.

Now, consider two neighboring actions A_i and A_{i+1} in s, and let s' be obtained from s by swapping the two actions. The contribution to $ECR(s)$ from the two actions is

$$C_i(\varepsilon^{i-1})P(\varepsilon^{i-1}) + C_{i+1}(\varepsilon^i)P(A_i = no, \varepsilon^{i-1}), \tag{4.1}$$

and the contribution to $ECR(s')$ from the two actions is

$$C_{i+1}(\varepsilon^{i-1})P(\varepsilon^{i-1}) + C_i(\varepsilon^{i-1}, A_{i+1} = no)P(A_{i+1} = no, \varepsilon^{i-1}). \tag{4.2}$$

The difference between (4.2) and (4.1) equals $ECR(s') - ECR(s)$, so we get

$$\begin{aligned} ECR(s') - ECR(s) = P(\varepsilon^{i-1}) \cdot \big(&C_{i+1}(\varepsilon^{i-1}) - C_i(\varepsilon^{i-1}) + \\ &C_i(\varepsilon^{i-1}, A_{i+1} = no)\ P(A_{i+1} = no \,|\, \varepsilon^{i-1}) - C_{i+1}(\varepsilon^i)\ P(A_i = no \,|\, \varepsilon^{i-1})\big). \end{aligned}$$

If s is an optimal troubleshooting sequence, we must have $ECR(s) \leq ECR(s')$, and therefore

$$\begin{aligned} &C_i(\varepsilon^{i-1}) + C_{i+1}(\varepsilon^i)P(A_i = no \,|\, \varepsilon^{i-1}) \leq \\ &\quad C_{i+1}(\varepsilon^{i-1}) + C_i(\varepsilon^{i-1}, A_{i+1} = no)P(A_{i+1} = no \,|\, \varepsilon^{i-1}). \end{aligned} \tag{4.3}$$

If it holds that the costs are independent of the actions taken, (4.3) can be rewritten as

$$\frac{P(A_i = yes \,|\, \varepsilon^{i-1})}{C_i} \geq \frac{P(A_{i+1} = yes \,|\, \varepsilon^{i-1})}{C_{i+1}}. \tag{4.4}$$

Definition Let A be a repair action and ε be the evidence compiled so far. The efficiency of A is defined as

$$ef(A|\varepsilon) = \frac{P(A = yes|\varepsilon)}{C_A(\varepsilon)}.$$

Proposition 4.1 *Let s be an optimal sequence of actions for which the costs are independent of the actions taken. Then, it must hold that* $ef(A_i|\varepsilon^{i-1}) \geq \mathrm{ef}(A_{i+1}|\varepsilon^{i-1})$.

In general, (4.3) can be used for pruning the decision tree, but Proposition 4.1 makes it even simpler. Assume that action B_i has been chosen at a branch where the options were $B_1, \ldots, B_m$ with current efficiencies $ef(B_1|\varepsilon), \ldots, ef(B_m|\varepsilon)$. Now, if B_i fails, only B_j's for which $ef(B_i|\varepsilon) \geq ef(B_j|\varepsilon)$ may be chosen, but after failure of B_j any action may be chosen.

Example 4.1 On a cold and wet morning, my car will not start. Moisture may have affected the ignition system or the carburetor, the spark plugs may be dirty, there may be a lack of fuel, or there may be a fault that I cannot fix myself. In that case, I call road service. I want to leave for work as soon as possible.

Table 4.3 gives the initial probabilities for the various causes. Because my car started yesterday evening, I assume that exactly one of the causes is present. I have one repair action for each possible cause, and except for *RS* the actions may not be perfect. The measure of precision is the probability of success given that the cause is present. Table 4.3 gives the precision and time requirement of the various actions.

	SP	*IS*	*Carb*	*Fl*	*RS*	*Other*
Cost	4 min.	2 min.	3 min.	1 min.	30 min.	n.a.
Prob.	0.3	0.1	0.1	0.1	n.a.	0.4
Prec.	0.8	0.7	0.6	0.95	1	n.a.

TABLE 4.3. Initial probabilities of the causes, precision, and cost in terms of minutes for the various repair actions.

To calculate the probabilities for the decision tree, I use the Bayesian network in Figure 4.18.

For the decision tree, I know that, whenever an action is performed, if it succeeds, the process is stopped, and if it is not stopped I will choose another action until I choose *RS*, which stops the process. Therefore, I need not represent the observation nodes for y/n of the result of the actions, and also all leaves in the decision tree must be *RS*.

I start with a root node where I will decide for the first decision. If I choose an action $A \neq RS$, I will investigate whether there is an action to perform if A fails. If all actions B have a higher efficiency than A,

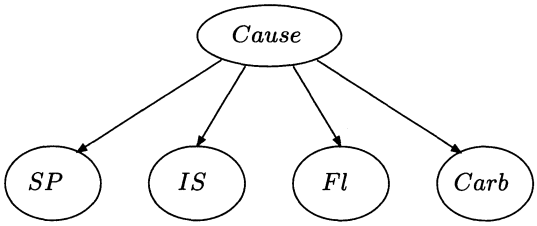

FIGURE 4.18. A Bayesian network for the current car problem. The *Cause* node has states $is, carb, sp, fl$, and *other*. The leaf nodes represent whether the action, when performed, will fix the problem.

then Proposition 4.1 yields that any action B after A will be suboptimal compared to swapping A and B.

Using the model in Figure 4.18, the efficiencies are calculated as $ef(SP) = 0.060, ef(IS) = 0.035, ef(Carb) = 0.020, ef(Fu) = 0.095, ef(RS) = 0.033$; that is, the action *Carb* should not be performed. Nothing yet prevents me from performing the other actions, and I expand the decision tree with a link for each of them.

Next, assume I have performed the action SP. Then, the next action can be any of the actions that initially has a lower efficiency than SP : $IS, Carb, RS$. We call them *triggered*. Among the triggered actions, I investigate whether Proposition 4.1 should prevent me from performing them. The efficiencies after an unsuccessful attempt with SP are $ef(IS \mid SP = n) = 0.046, ef(Carb \mid SP = n) = 0.026, ef(RS \mid SP = n) = 0.033$. Because $ef(Fu \mid SP = n) = 0.125$, we see that again a performance of *Carb* will have only suboptimal expansions. The tree so far is depicted in Figure 4.19.

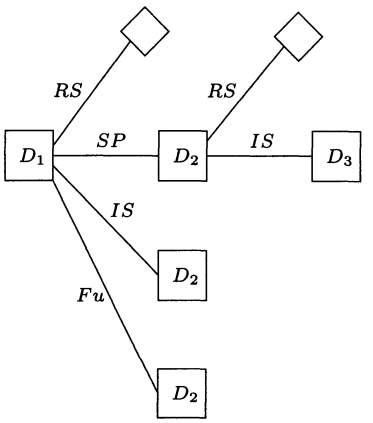

FIGURE 4.19. The decision tree expanded up to SP.

After an unsuccessful performance of IS, we have only $Carb$ and RS triggered. The efficiencies are calculated as $ef(Carb \mid SP = n, IS = n) = 0.029, ef(RS \mid SP = n, IS = n) = 0.033$, and because $ef(Fu \mid SP = n, IS = n) = 0.138$, the only expansion is RS.

Turning to the situation with IS as the first action and later the situation with Fu first, I do the same kind of analysis. The result is shown in Figure 4.20.

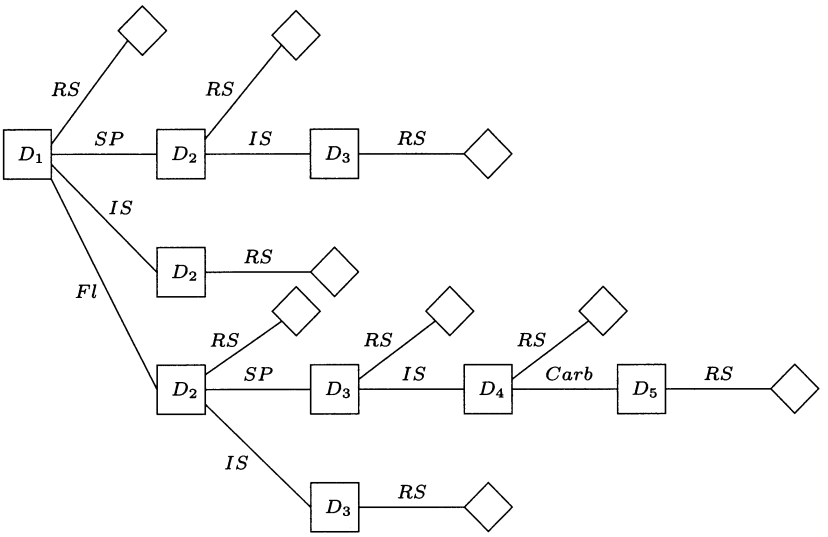

FIGURE 4.20. The pruned decision tree for the car problem using Proposition 4.1.

The tree in Figure 4.20 has eight non-RS links, compared to 32 in a coalesced decision tree for the same problem.

4.5.2 The greedy approach

It would be much easier to solve the troubleshooting problem if we could base the sequencing on a greedy approach: always choose an action with the highest efficiency. However, Proposition 4.1 does not guarantee that this approach will yield an optimal troubleshooting sequence.

In Figure 4.21, there are four possible causes, C_1, C_2, C_3, and C_4, for a device malfunctioning, and we assume that exactly one of the causes is present, and that the prior probabilities are 0.2, 0.25, 0.40, and 0.15, respectively. Assume that all actions have cost 1. Then, action A_2 has the highest efficiency, and if A_2 fails, then A_1 has higher efficiency than A_3. The sequence $\langle A_2, A_1, A_3 \rangle$ has $ECR = 1.50$. However, the sequence $\langle A_3, A_1 \rangle$ has $ECR = 1.45$.

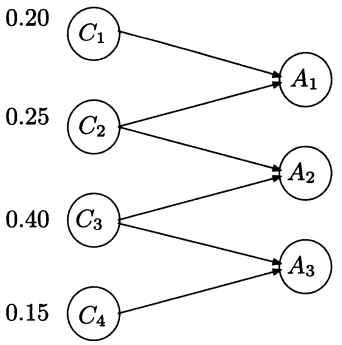

FIGURE 4.21. An example of dependent actions. The C's are causes for the device failing. The A variables represent actions. An action will repair a parent if faulty. A single fault is assumed.

To analyze why the decreasing efficiency approach does not guarantee an optimal sequence, let $\langle A_1, \ldots, A_n \rangle$ be a sequence ordered by decreasing efficiency. If the sequence is not optimal, there must be two actions A_i and A_j, $i < j$, which, in the optimal sequence, are taken in different order. At the time where A_i is chosen, we have

$$\frac{P(A_i = yes \,|\, \varepsilon)}{C_i} \geq \frac{P(A_j = yes \,|\, \varepsilon)}{C_j}.$$

In the optimal sequence, where A_j is chosen before A_i, we have

$$\frac{P(A_i = yes \,|\, \varepsilon')}{C_i} < \frac{P(A_j = yes \,|\, \varepsilon')}{C_j},$$

where ε and ε' are simple evidence (not involving A_i and A_j). We can infer that an action sequence $\langle A_1, \ldots, A_n \rangle$ is optimal if for all $i < j$ it holds that

$$ef(A_j \,|\, \varepsilon) \leq ef(A_i \,|\, \varepsilon),$$

where ε is simple evidence (not involving A_i and A_j).

Proposition 4.2 *Consider the following assumptions.*

- *The device has n different faults $F_1, \ldots, F_n$ and n different repair actions $A_1, \ldots, A_n$.*
- *Exactly one of the faults is present.*
- *Each action has a specific probability of repair, $p_i = P(A_i = \text{yes} \,|\, F_i)$, and $P(A_i = \text{yes} \,|\, F_j) = 0$ for $i \neq j$.*
- *The cost C_i of a repair action does not depend on the performance of previous actions.*

If these assumptions hold, then $\mathrm{ef}(A_j) \leq \mathrm{ef}(A_i)$ *implies that* $\mathrm{ef}(A_j \,|\, \varepsilon) \leq \mathrm{ef}(A_i \,|\, \varepsilon)$, *where* ε *is simple evidence (not involving* A_i *and* A_j*).*

Note that we do not assume the repair actions to be perfect. They may fail to fix a fault that they are supposed to fix.

Proof: Let A_m be an action that has failed. We calculate $P(A_i = yes \,|\, A_m = no)$ (for notational convenience, we omit mention of the current evidence). Due to the single-fault assumption, we have $P(A_m = \mathrm{no} \,|\, A_i = \mathrm{yes}) = 1$. Using Bayes' rule, we get

$$P(A_i = yes \,|\, A_m = no) = \frac{P(A_m = no \,|\, A_i = yes) P(A_i = yes)}{P(A_m = no)} = \frac{P(A_i = yes)}{P(A_m = no)}.$$

In other words, $P(A_m = no)$ is a normalizing constant for the remaining actions, and the relative order of efficiencies is preserved. □

The following theorem concludes the considerations.

Theorem 4.2 *Let* $s = \langle A_1, \ldots, A_n \rangle$ *be an action sequence for a troubleshooting problem fulfilling the conditions in Proposition 4.2. Assume that* s *is ordered according to decreasing initial efficiencies. Then,* s *is an optimal action sequence and*

$$ECR(s) = \sum_{i=1}^{n} C_i \left(1 - \sum_{j=1}^{i-1} p_j \right). \tag{4.5}$$

Proof: From the proof of Proposition 4.2, we have that the relative order of the efficiencies of the actions is preserved. For any action sequence s' that is not ordered according to $ef(A_i)$, there will be a j so that $ef(A_j) < ef(A_{j+1})$ and therefore $ef(A_j \,|\, \varepsilon^j) < ef(A_{j+1} \,|\, \varepsilon^j)$. Hence, s' can be improved by swapping A_j and A_{j+1}. From the definition, we have

$$ECR(s) = \sum_{i=1}^{n} C_i P(\varepsilon^i).$$

Due to the singlefault assumption, we have $P(\varepsilon^i) = 1 - \sum_{j=1}^{i-1} p_j$. □

4.5.3 Call service

The action call service (CS) will always solve the problem. The cost of CS is not the unknown price of fixing the device but the possible overhead of having outsiders fixing a problem you could have fixed yourself. The

efficiency of CS is $1/C_{CS}$ no matter what set of actions has been performed so far.

Let $s = \langle A_1, \ldots, A_n \rangle$ be an optimal action sequence resulting from a situation meeting the assumptions in Proposition 4.2. It may be that the sequence should be broken before A_n and service is called. According to Proposition 4.1, CS shall only be performed after an action of higher efficiency. It is a good idea to perform the CS action as soon as it has maximal efficiency. However, this is not guaranteed to be optimal. The question of finding an optimal action sequence including CS is of higher combinatorial complexity. Instead of looking for a sequencing of actions each of which must eventually be performed if the other actions fail, we now look for a subset of actions and a sequencing of them. We will not go further into this problem.

4.5.4 Questions

The outcome of a question may shed light on any of the possible faults, or it may be focused on a particular fault.

The troubleshooting task is to interleave actions and questions such that the expected cost is minimal. To do so, we must analyze the value of answers to questions.

Imagine that we are in the middle of a troubleshooting sequence; we have so far gained the evidence ε, and now we have the option to ask the question Q with cost C_Q. For simplicity, we assume that Q has only two outcomes, "yes" and "no." Assume that no matter what the outcome of Q, we are able to calculate the minimal expected cost of repair for the remaining sequence. Therefore, let ECR be the minimal expected cost if Q is not performed, and let $ECR_{Q=yes}$ and $ECR_{Q=no}$ denote the same for the outcomes "yes" and "no", respectively.

Then, the value of observing Q is

$$V(Q) = ECR - \Big(P(Q = yes\,|\,\varepsilon)ECR_{Q=yes} + P(Q = no\,|\,\varepsilon)ECR_{Q=no}\Big), \tag{4.6}$$

and Q is performed if and only if $V(Q) > C_Q$.

In order to determine whether to ask a question prior to an action, we must analyze all possible succeeding sequences, and if there are several actions and questions, it is in general intractable. In the future, we will also have question options to interleave.

A workable approximation is the *myopic* strategy, where it is assumed at any stage of troubleshooting that we allow questions to be asked, but in the future we allow only repair actions. In that case, the task reduces to calculating expected costs given the various outcomes of the possible questions, and the approaches from the previous section can be used.

4.5.5 The myopic repair-observation strategy

The following strategy is a workable approximation to the general troubleshooting task.

$\varepsilon :=$ "the device is not working properly";
while the device is not working properly **do**
calculate EGC: the expected cost of the greedy observationless repair sequence;
for all O **do**
for all states s of O **do**
calculate $P(O \text{ gives outcome } s \mid \varepsilon)$;
for all a **do**
calculate $p_a^s = P(a \text{ solves the problem} \mid O \text{ gives outcome } s, \varepsilon)$
od;
calculate EGC^s, the expected cost of the greedy observationless repair
sequence given O gives outcome s
od;
calculate $EGC_o = c_o + \sum_s P(O \text{ gives outcome } s \mid \varepsilon) EGC^s$
od;
choose the observation or action with lowest expected greedy cost; update ε according to the choice and result.
elihw.

4.6 Influence Diagrams

Some decision scenarios consist of the same sequence of decision-observation options. This is, for example, the case for the decision tree in Figure 4.22 with the model in Figure 4.23 for calculating the probabilities. This kind of decision scenario is called *symmetric*. A symmetric decision scenario can be represented as a chain of variables (see Figure 4.24).

4.6.1 Extended poker model

During a game of poker, I take a series of decisions. Before deciding on whether to call or fold, I also take decisions on the number of cards to change. When I decide on my first change of cards (MFC), I only know my initial hand ($MH0$), and when I decide for the second change of cards (MSC), I know my two hands, my opponent's first change of cards (OFC), and my own first change of cards. Accordingly, I then decide on whether to fold or call. Figure 4.25 gives a chain of variables, and Figure 4.26 is a Bayesian network model from which to take the probabilities when solving it. The probabilities to take are of the type $P(BH \mid OFC, USC, MH)$.

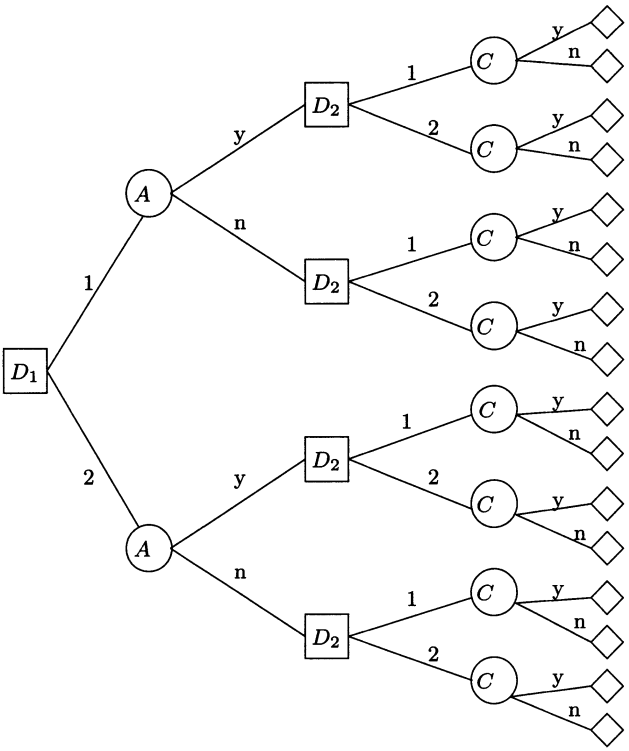

FIGURE 4.22. A decision tree.

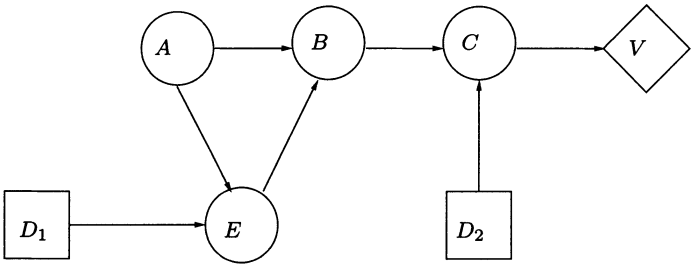

FIGURE 4.23. The model for calculating probabilities for the decision tree in Figure 4.22.

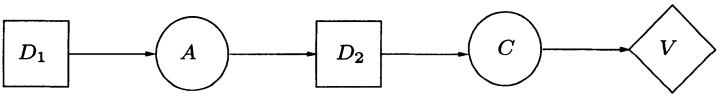

FIGURE 4.24. A chain representing the decision tree in Figure 4.22.

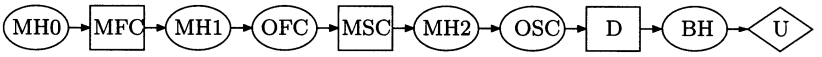

FIGURE 4.25. A chain of variables representing my poker game. Note: the order of $MH1$ and OFC as well as the order of $MH2$ and OSC may be interchanged.

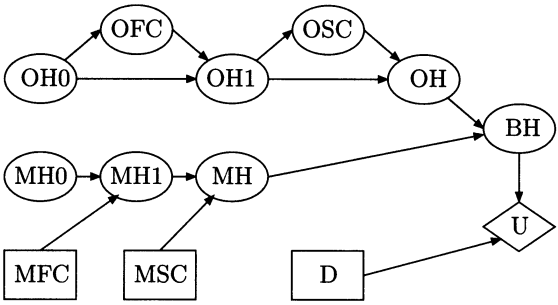

FIGURE 4.26. A Bayesian network model for the calculation of the probabilities required when solving the chain in Figure 4.25.

Two things are important for the chain of decisions and observations, namely the order of the decisions and the set of observations between two decisions.

The chain can be represented in the underlying Bayesian network (extended with decision nodes and utility nodes). To represent the order of decisions, you extend the graph with *precedence links*: directed links between decision nodes. To represent observations, you extend the graph with *information links*: a link from chance node A to decision node D if A is observed before D is decided upon but after the decision prior to D. The resulting graph is called an *influence diagram* (see Figure 4.27).

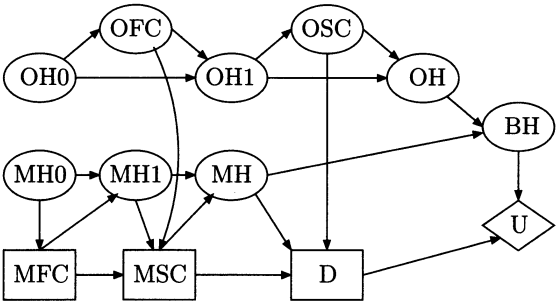

FIGURE 4.27. An influence diagram for my poker game: Figure 4.26 extended with precedence links and information links.

Some precedence links may be redundant. In Figure 4.27, for example, MFC has an impact on $MH1$, which is observed before MSC. There-

fore, MFC must precede MSC, and the link from MFC to MSC can be removed (see Figure 4.28).

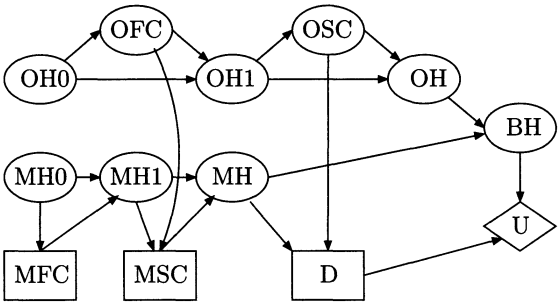

FIGURE 4.28. A pruned influence diagram for my poker game.

In Figure 4.28, we can read that, at the time of deciding D, I will know the states of the parents MH and OSC, and assuming that I do not forget my past, I will also know the states of $MSC, MH1, OFC$, and so on.

4.6.2 Definition of influence diagrams

Although influence diagrams were originally invented as a compact representation of decision trees for symmetric decision scenarios, they are now seen more as a decision tool extending Bayesian networks, and in the following we introduce influence diagrams this way.

Syntax

An *influence diagram* consists of a directed acyclic graph over chance nodes, decision nodes and utility nodes with the following structural properties:

- there is a directed path comprising all decision nodes;
- the utility nodes have no children.

For the quantitative specification, we require that:

- the decision nodes and the chance nodes have a finite set of mutually exclusive states;
- the utility nodes have no states;
- to each chance node A is attached a conditional probability table $P(A|pa(A))$;
- to each utility node U is attached a real-valued function over $pa(U)$.

Figure 4.29 gives an example of the structural part of an influence diagram.

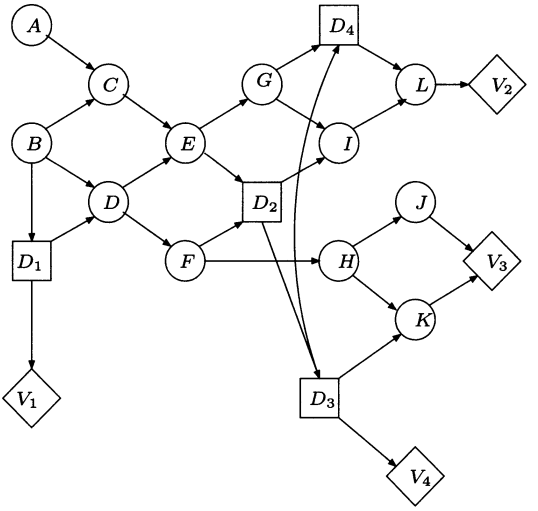

FIGURE 4.29. An example of the structure of an influence diagram. We have $I_0 = (B), I_1 = (E, F), I_2$ is empty, $I_3 = (G), I_4 = (A, C, D, H, I, J, K, L)$.

Semantics

Notice that links into a decision node yield no quantitative requirements. They are called *information links*, and they indicate that the state of the parent is known prior to the decision.

The structural requirement that there must be a path comprising all decision nodes ensures that the influence diagram defines a temporal sequence of decisions. This yields a partitioning of the chance variables according to the time of observation. The set I_0 is the set of variables observed before any decision is taken. The set I_1 is the set of variables observed after the first decision is taken but before the second decision, and the set I_i is the set of chance variables observed after the ith decision but before the $i+1$th decision. If there are n decisions, I_n is the set of variables that are not observed.

There is a hidden assumption behind the semantics of influence diagrams, namely *no-forgetting*: the decision maker remembers the past observations and decisions.

In some decision scenarios, two decisions may be independent in the sense that they can be taken in either order without changing the expected utilities. In Figure 4.29, the two decisions D_2 and D_3 are independent. Therefore, the link from D_2 to D_3 puts an unnecessary restriction on the decision maker. It could be removed and the representation would be meaningful, although the first structural requirement is violated. Unfortunately, it is not easy to characterize situations where decisions are independent,

and therefore we will keep the first structural requirement, which ensures a well-specified decision scenario.

If there is more than one utility node, then the entire utility is the sum of the individual utilities.

4.6.3 Solutions to influence diagrams

As for decision trees, we wish to get a strategy out of an influence diagram. Particularly, the basic request to a system processing influence diagrams is advice on the first decision. The way to calculate a strategy from an influence diagram is similar to the method of establishing a strategy from a decision tree. You start with the last decision and calculate the expected utility for the various options given the past. From this, you pick a decision of maximal expected utility, that is, you have a function which, given any configuration of the past, provides a decision as well as the expected utility of this decision. A strategy is a set of such policy functions.

In principle, the calculation of a strategy can be performed by folding the influence diagram out to a decision tree and using the technique for decision trees. However, it can be done much more efficiently by exploiting the structure of the influence diagrams. For the chain in Figure 4.24, we can use the model in Figure 4.23 to calculate $P(C \mid D_1, A, D_2)$. It is used to calculate an optimal policy $\sigma_2(D_1, A)$ for D_2, and when a policy for D_2 is determined, it can in combination with the model be used to calculate an optimal policy for D_2. In Chapter 7, we give details on how to exploit the structure. Still, there is a much higher risk of running into complexity problems than in the case of Bayesian networks. The risk comes from the size of the domains of the policy functions.

Fishing in the North Sea

Every year, the European Union undertakes very delicate political and biological negotiations to determine a volume of fishing for most kinds of fish in the North Sea. Oversimplified, you can say that each year we have a test for the volume of fish, and based on this test the volume of allowable catch is decided. This decision has an impact on the volume for next year (note that the decision on volume does not mean that only this volume is actually fished – quotas have a status similar to speed limits). Figures 4.30 and 4.31 give an influence diagram for a five-year strategy. Each variable has ten states.

The model in Figures 4.30 and 4.31 is an example of a partially observable Markov decision process (POMDP). This model has a complexity problem. For the fifth decision, all the past is relevant. Because there are nine ten-state variables in the past, the domain of the strategy function for FV_5 has 10^9 elements.

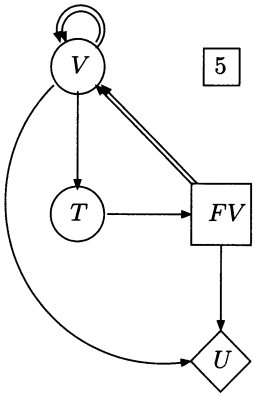

FIGURE 4.30. An influence diagram for a five-year strategy for fishing volumes of herring in the North Sea.

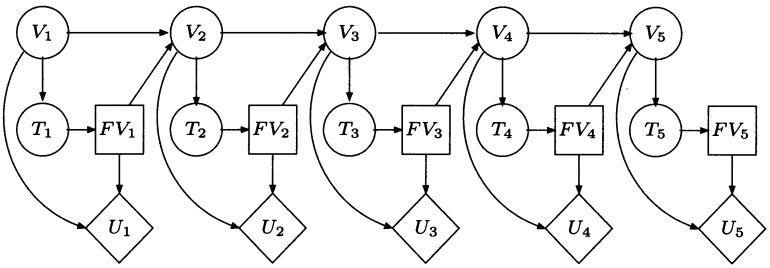

FIGURE 4.31. The out-folded version of the influence diagram in Figure 4.30.

This does not mean that whenever the past is intractably large, the computer must give up. Fortunately, it often happens that not all information from the past is relevant.

Sometimes even fairly small influence diagrams represent an intractable task, and then you must use various approximated methods. One method is *information blocking*. The principle in information blocking is to introduce variables which, when observed, d-separate most of the past from the present decision.

Fishing again

The problem with the model in Figures 4.30 and 4.31 is that all information from the past has an impact on how we will estimate the current volume of fish. We can make an approximation by allowing us to use only this year's test and fishing volume to estimate next year's volume of fish. In the model, we delete the arrow $V \rightrightarrows V$ and instead introduce an arrow $T \rightrightarrows V$ (see Figures 4.32 and 4.33).

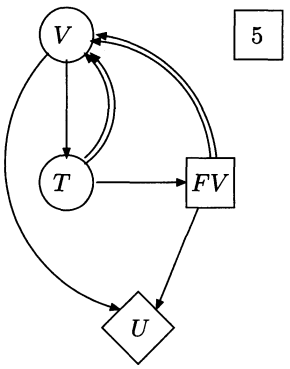

FIGURE 4.32. The influence diagram from Figures 4.30 and 4.31 approximated through information blocking.

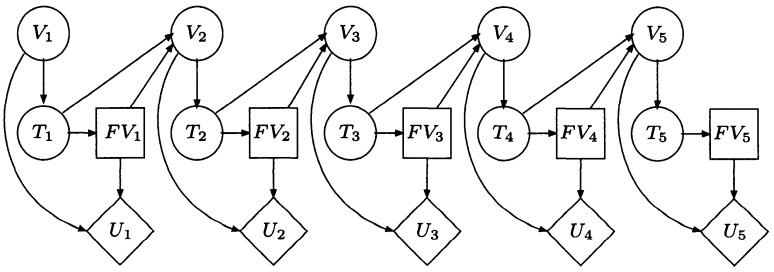

FIGURE 4.33. The out-folded version of the influence diagram in Figure 4.32.

To establish the potential $P(V_{i+1} \mid T_i, FV_i)$, use the model in Figures 4.30 and 4.31. We have

$$P(V_2, T_1 \mid FV_1) = \sum_{V_1} P(V_1)P(T_1 \mid V_1)P(V_2 \mid V_1, FV_1),$$

$$P(T_1 \mid FV_1) = \sum_{V_2} P(V_2, T_1 \mid FV_1),$$

$$P(V_2 \mid T_1, FV_1) = \frac{P(V_2, T_1 \mid FV_1)}{P(T_1 \mid FV_1)}.$$

This potential is used for all time slices.

The trick just shown is an example of a general information-blocking technique where you abstract the past into a *history variable* and only allow temporal links from observed variables and from the history variable (see Figure 4.34 for another example).

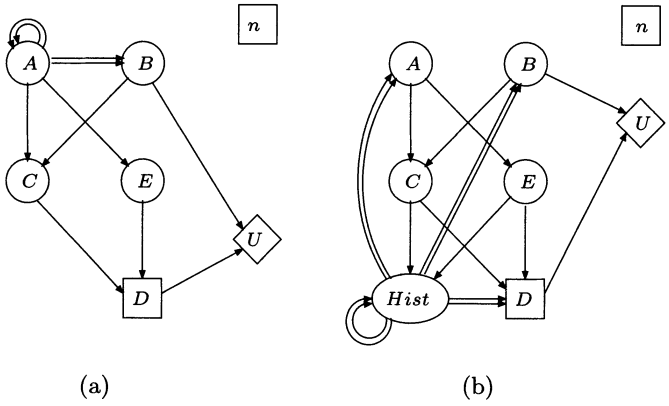

FIGURE 4.34. Information blocking through a history variable. The self-reference link for the history variable *Hist* makes it possible to abstract history more than one time slice back.

4.6.4 Test decisions in influence diagrams

Influence diagrams do not contain a special representation of test decisions. However, there is a general way of representing test options as decision variables. Assume that in the flu–fever example in Figure 4.1 I am in the situation that I can take my temperature before I decide whether to take an aspirin. The impact of taking an aspirin is modeled by introducing a new chance node, *A-Fever*, representing the fever after the aspirin decision A. The test decision is a decision on whether to observe the state of the chance node *Fever* (we assume the thermometer is accurate). This is represented as a triangular node, T_F, and a directed link from *Fever* to T_F

(see Figure 4.35). Furthermore, we add a link from T_F to A. Note that the information link does not go from *Fever* to A because the state of *Fever* will not be known if I decide not to observe it.

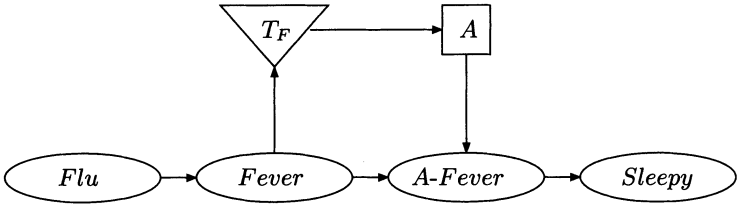

FIGURE 4.35. Representation of a test decision followed by an action decision.

The test node in Figure 4.35 can be represented as a decision node in the following way: represent the triangular node as a rectangular node T_F with states *yes* and *no*, add a chance variable *Fever'* with states *unobserved* and the states from *Fever*, let *Fever* and T_F be parents of *Fever'*, and add a link from *Fever'* to A indicating that *Fever'* is observed when the decision A is to be taken (see Figure 4.36).

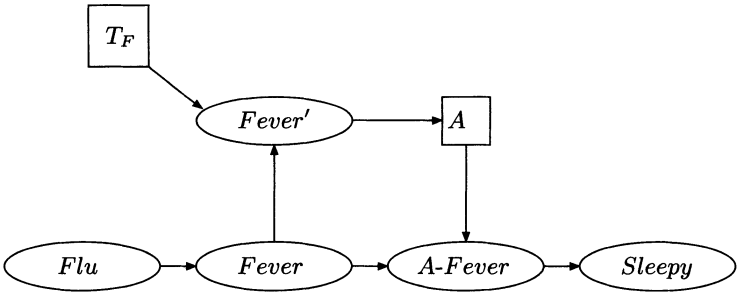

FIGURE 4.36. The network from Figure 4.35 with the test node transformed to a decision node.

The table for *Fever'* given T_F and *Fever* in Figure 4.36 is so that the state is *unobserved* if T_F is *no*, and if T_F is *yes*, then *Fever'* is in the same state as *Fever*.

This construction is general, and it is illustrated in Figure 4.37. In this way, methods developed for computing decision strategies can also be used for decision scenarios containing test decisions.

The construction can be made a bit simpler by extending the node A with the extra state *unobserved* and thereby avoiding the extra node A'. However, usually it is preferable not to change the nodes of the initial model.

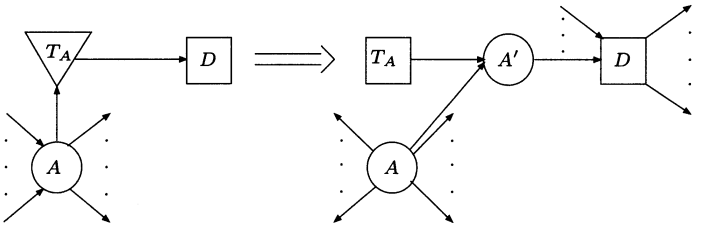

FIGURE 4.37. The general transformation of a test node to a decision node.

4.7 Summary

One action

Decision D, utility functions $U_1, \ldots, U_n$ over domains $X_1, \ldots, X_n$, evidence e. The expected utility is

$$EU(D \mid e) = \sum_{X_1} U_1(X_1)P(X_1 \mid D, e) + \ldots + \sum_{X_n} U_n(X_n)P(X_n \mid D, e),$$

and a state d maximizing $EU(D \mid e)$ is chosen as an optimal action.

Value of information

Value function (one utility function U, one action A):

$$V(P(H)) = \max_{a \in A} \sum_{h \in H} U(a, h)P(h).$$

Expected value of performing test T:

$$EV(T) = \sum_{t \in T} P(t) \max_{a \in A} \sum_{h \in H} U(a, h)P(h \mid t).$$

$EV(T)$ can be calculated for all tests T by entering the states of h as evidence and using Bayes' rule. Expected profit:

$$EP(T) = EV(T) - V(P(H)) - C_T.$$

Myopic approach: Choose repeatedly a test with the highest positive expected profit, if any.

Nonutility value functions:

- Entropy: $V(P(H)) = \sum_{h \in H} P(h) \log_2(P(h))$;
- Variance: $V(P(H) = -\sum_{h \in H}(h - \mu)^2 P(h)$, where $\mu = \sum_{h \in H} hP(h)$.

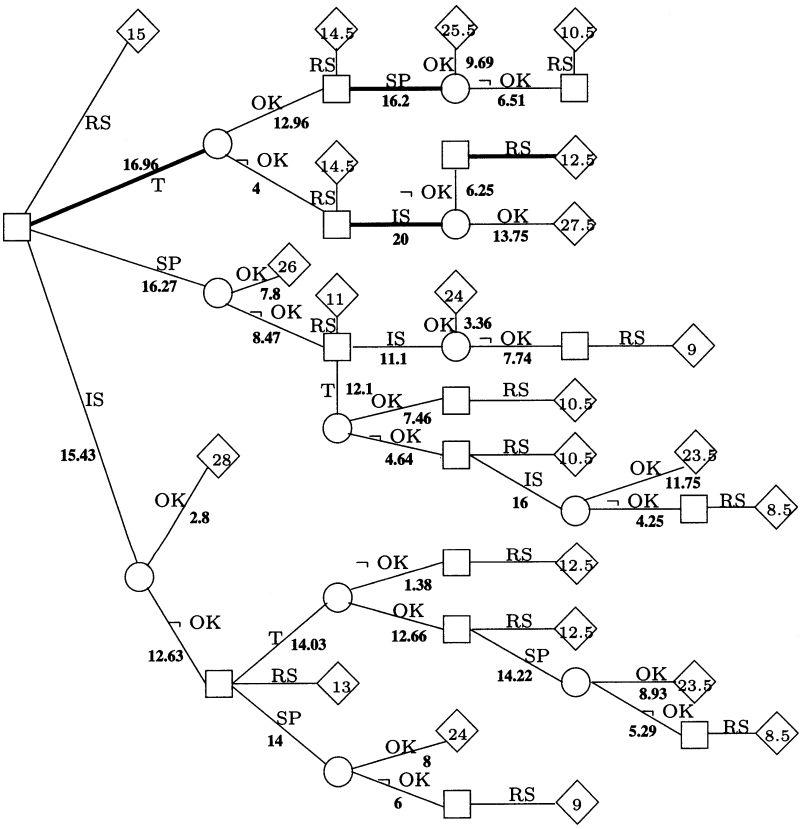

FIGURE 4.38. An example of a decision tree. The probabilities may be taken from a Bayesian network. The fat links indicate an optimal strategy.

Decision trees

An example is shown in Figure 4.38.

An optimal strategy is determined by "rolling back". Start with nodes that have only leaves as children. If the node is a chance node A, the expected utility for A is calculated. Each child of A has a utility u attached, and the link has a probability p. We calculate the product up from each child, and their sum is attached to A. If the node is a decision node D, each child of D has a (expected) utility attached. Choose the child(ren) with maximal expected utility, highlight the link(s), and attach the value to D.

Troubleshooting

The expected cost of repair of a troubleshooting sequence $s = \langle A_1, \ldots, A_n \rangle$ of repair actions is

$$ECR(s) = \sum_i C_i(\varepsilon^{i-1})P(\varepsilon^{i-1}),$$

where ε^j denotes the statement that the first j actions have failed.

For an optimal repair sequence, it holds that

$$\begin{aligned} C_i(\varepsilon^{i-1}) \quad &+ \quad C_{i+1}(\varepsilon^i)P(A_i = n \mid \varepsilon^{i-1}) \\ &\leq \quad C_{i+1}(\varepsilon^{i-1}) + C_i(\varepsilon^{i-1}, A_{i+1} = n)P(A_{i+1} = n \mid \varepsilon^{i-1}). \end{aligned}$$

The efficiency of a repair action is

$$ef(A \mid \varepsilon) = \frac{P(A = y \mid \varepsilon)}{C_A(\varepsilon)}.$$

If costs are independent of evidence, then for an optimal repair sequence it must hold that

$$ef(A_i \mid \varepsilon^{i-1}) \geq ef(A_{i+1} \mid \varepsilon^{i-1}),$$

and if for all $i < j$ it holds that

$$ef(A_j \mid \varepsilon) \leq ef(A_i \mid \varepsilon)$$

for all simple evidence ε (not involving A_i and A_j) of the type "actions $A, \ldots, B$ have failed," then the repair sequence $\langle A_1, \ldots, A_n \rangle$ is optimal (does not necessarily hold when call service is an option).

Questions: The value of getting an answer of Q is

$$V(Q) = ECR - \sum_{s \in Q} P(Q = s \mid \varepsilon)ECR_s,$$

where ECR_s is the expected cost of repair for an optimal sequence given evidence ε and "$Q = s$," and ECR is the expected cost of repair for an optimal sequence not starting with Q. Because neither ECR nor ECR_s are tractable, a myopic approach is often used.

Influence diagrams

An *influence diagram* consists of a directed acyclic graph over chance nodes, decision nodes, and utility nodes with the following structural properties:

- there is a directed path comprising all decision nodes;
- the utility nodes have no children.

For the quantitative specification, we require that:

- the decision nodes and the chance nodes have a finite set of mutually exclusive states;
- the utility nodes have no states;
- to each chance node A is attached a conditional probability table $P(A|pa(A))$;
- to each utility node V is attached a real-valued function over $pa(V)$.

Figure 4.39 gives an example of the structural part of an influence diagram. Methods for determining optimal strategies from influence diagrams are

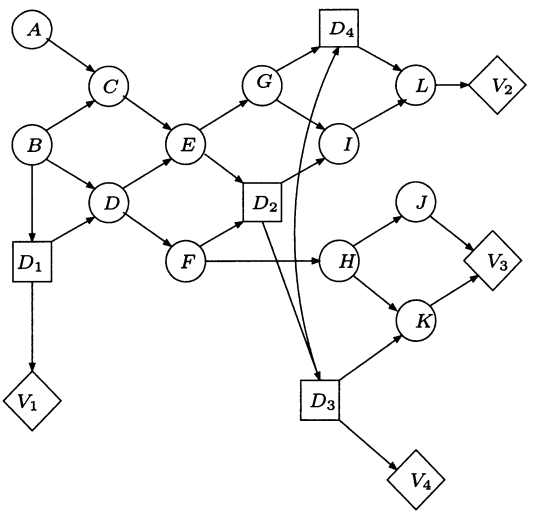

FIGURE 4.39. An example of the structure of an influence diagram. We have $I_0 = (B), I_1 = (E, F), I_2$ is empty, $I_3 = (G), I_4 = (A, C, D, H, I, J, K, L)$.

given in Chapter 7.

4.8 Bibliographical Notes

Decision theory has a long history but achieved a breakthrough in the work of von Neumann and Morgenstein (1953). Decision trees were introduced by Raiffa and Schlaifer (1961). Value of information is formally treated in (Howard 1966) and (Lindley 1971), where utilities are guiding the test selection. The myopic approximation was introduced by Gorry and Barnett (1968). In (Ben-Bassat 1978), entropy and variance are used. Troubleshooting based on decision theory was introduced by Kalagnanam and Henrion (1990), and it was further analyzed by Heckerman et al. (1995). Section 4.5 is an extension of this work. Proofs that various versions of troubleshooting are NP-complete can be found in (Sochorová and Vomlel 2000). Influence diagrams were proposed by Howard and Matheson (1984). The oil wildcatter's problem is due to Raiffa (1968). The used car buyer's problem is due to Howard (1984). Markov Decision Processes are treated in (Howard 1960) and (Puterman 1994).

4.9 Exercises

Exercise 4.1 Consider the management of effort example in Section 4.2.1.

(*i*) Let the marks be $0, 5, 6, 8, 9, 10$. What is the optimal decision if the numerical values are used as utilities?

(*ii*) Consider the approach where the marks are given subjective utilities. Show that action Gd can only be optimal if the mark 0 is given higher utility than mark 3.

Exercise 4.2 ^{E} Extend the model from Exercise 2.12 to a model for folding or calling.

Exercise 4.3 ^{E} Extend Exercise 2.15 with the following:

> In golf, the task is to use as few strokes as possible at each hole. I am driving at a hole 260 m long. If the drive is 265 m, I will on average use 1.8 strokes to finish the hole. If the drive is 240 m, on average 2 extra strokes are needed; 220 m requires 2.5 extra strokes; 200 m requires 2.7; 180 m 2.9 extra strokes; 160 m 3.1; 145 m 3.3; a drive of 290 m will carry the ball out in hazards requiring 3.5 extra strokes; if the drive is a miss, the ball will drop into a lake, and it will require 4.5 extra strokes to finish the hole.

Construct a system that helps me decide whether to use the spoon or the driver in the drive.

Exercise 4.4 ^{E} Consider the stud farm example from Section 2.2.2. Extend the model to be an aid for deciding for each horse whether it should be taken out of breeding. The tables below give the utilities.

	Carrier	*Pure*
Out	−10	−10
In	−40	100

Stallions

	Carrier	*Pure*
Out	−3	−3
In	−10	40

Mares

TABLE 4.4. Tables for Exercise 4.4.

Exercise 4.5 ^{E} Consider the insemination model from Exercise 2.6. Assume that you have the options to repeat the insemination or to wait for another six-week period. The cost of repeating the insemination is 65 regardless of the pregnancy state of the cow. If the cow is pregnant and you wait, it will cost you nothing, but if the cow is not pregnant and you wait, it will cost you further 30 (that makes a total of 95 for waiting plus the eventual repeated insemination). The cost of BT is 1 and the cost of UT is 2. Perform a myopic and a nonmyopic value of information analysis.

Exercise 4.6 Let the hypothesis variable H have n states. Introduce an action variable A with the same states as H; let the utility table be as follows:

$$U(h,a) = \begin{cases} 1 & \text{if } h \text{ and } a \text{ are the same} \\ 0 & \text{otherwise} \end{cases}$$

Show that a value function based on U corresponds to selecting a hypothesis state of highest probability.

Exercise 4.7 Solve the decision tree in Figure 4.40.

Exercise 4.8 ^{E} (The oil wildcatter's problem)

> An oil wildcatter must decide whether to drill or not to drill. The cost of drilling is \$70,000. If he decides to drill, the hole may be soaking (with a return of \$270,000), wet (with a return of \$120,000), or dry (with a return of \$0). The prior probabilities for soaking, wet, and dry are (0.2, 0.3, 0.5). At the cost of \$10,000, the oil wildcatter could decide to take seismic soundings of the geological structure at the site. The specifics of the test are given in Table 4.5.

(*i*) Solve the problem with a decision tree.
(*ii*) Solve the problem as a value of information problem.
(*iii*) Solve the problem with an influence diagram.

Exercise 4.9 (The used car buyer's problem)

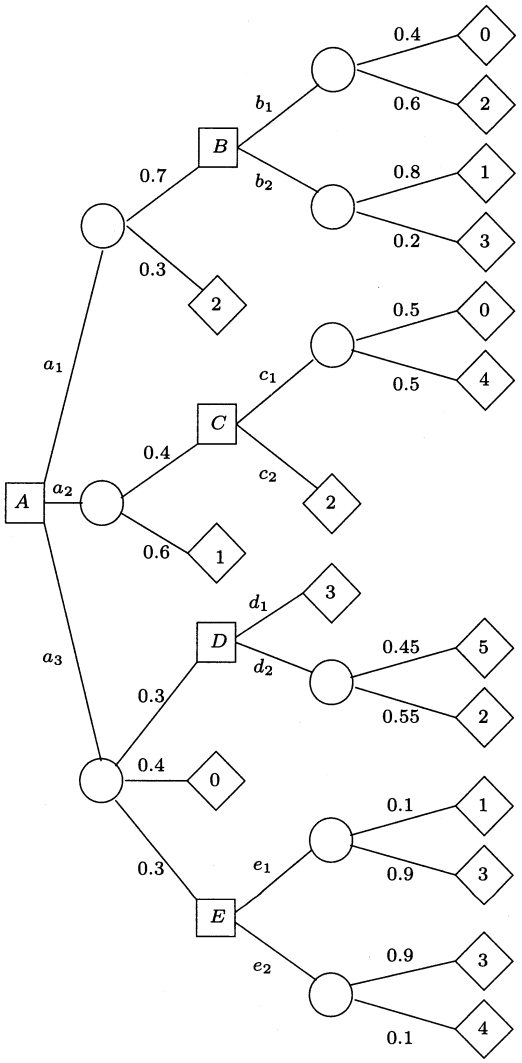

FIGURE 4.40. Figure for Exercise 4.7.

$T \setminus S$	dr	wt	so
n	0.6	0.3	0.1
o	0.3	0.4	0.4
c	0.1	0.3	0.5

$P(Test \mid Structure)$

TABLE 4.5. Table for Exercise 4.8. n, o, and c are the outcomes of the test.

Joe is considering buying a used car from a dealer for \$1,000. The market price of similar cars with no defects is \$1,100. Joe is uncertain whether the particular car he is considering is a "peach" or a "lemon." Of the ten major subsystems in the car, a peach has a serious defect in only one subsystem, whereas a lemon has a serious defect in six subsystems. The probability that the used car under consideration is a lemon is 0.2. The cost of repairing one defect is \$40, and the cost of repairing six defects is \$200.

For an additional \$60, Joe can buy the car from the dealer with an "antilemon guarantee." The antilemon guarantee will normally pay for 50% of the repair cost, but if the car is a lemon, then the guarantee will pay 100% of the repair cost.

Before buying the car, Joe has the option of having the car examined by a mechanic for an hour. In this period, the mechanic offers three alternatives t_1, t_2, t_3 as follows:

t_1: test the steering subsystem alone at a cost of \$9,

t_2: test the fuel and electrical subsystems for a total cost of \$13,

t_3: do a two-test sequence in which Joe can authorize a second test after the result of the first test is known. In this alternative, the mechanic will first test the transmission subsystem at a cost of \$10 and report the results to Joe. If Joe approves, the mechanic will then proceed to test the differential subsystem at an additional cost of \$4.

All tests are guaranteed to find a defect in the subsystem if a defect exists. We assume that Joe's utility for profit is linear in \$.

(*i*) Solve the problem with a decision tree.
(*ii*) Consider how to represent the problem as an influence diagram (you may add dummy states and variables as you wish).

Exercise 4.10 Use the greedy approach on Example 4.4.1.

Exercise 4.11 Solve the decision tree in Figures 4.12 and 4.13 (the probabilities can be taken from the model in Figure 4.11).

Exercise 4.12 Use the algorithm in Section 4.5.5 to solve the start problem in Section 4.4.

Exercise 4.13 Complete the reduced decision tree from Figure 4.17 and solve it.

Exercise 4.14 ^{E} Solve Exercise 2.13 as a decision problem.

Exercise 4.15 ^{E} Solve the example in Section 4.3.1 as an influence diagram.

Exercise 4.16 ^{E} Extend the poker model from Exercise 4.2 to the influence diagram in Figure 4.27.

Exercise 4.17 ^{E} Solve the milk problem in Section 4.3.4 with an influence diagram.

Exercise 4.18 ^{E} Represent the car start problem in Section 4.4.1 as an influence diagram. (What are the decision options at each step?)

Exercise 4.19 ^{E} Consider Exercise 4.17. The farmer may use 2, 3, 4, 5, 6, 10, 12, 15, 20, or 30 intermediate containers for the milk (pouring milk from 30, 20, 15, 12, 10, 6, 5, 4, 3 or 2 cows into the same container before pouring it into the main container). He will test the milk in the intermediate containers rather than for each individual cow. The cost of each container is 1 cent per day. What is his optimal strategy for number of containers and combination of tests?

Exercise 4.20 Unfold the influence diagrams in Figure 4.34 (with $n = 3$).

Part II

Algorithms for Normative Systems

5
Belief Updating in Bayesian Networks

In this chapter, we present an algorithm for probability updating. An efficient updating algorithm is fundamental to the applicability of Bayesian networks. As shown in Chapter 1, access to $P(U, e)$ is sufficient for the calculations. However, because the joint probability table increases exponentially with the number of variables, we look for more efficient methods. Unfortunately, no method guarantees a tractable calculation task. However, the method presented here represents a substantial improvement, and it is among the most efficient methods known.

5.1 Introductory Examples

To repeat the fundamentals from Chapter 1 and for pinpointing the issues in belief updating for Bayesian networks, we consider in this section a simple example. Consider the Bayesian network in Figure 5.1 over the universe U. The potentials specified for BN are $\phi_1 = P(A_1), \phi_2 = P(A_2 \mid A_1), \phi_3 = P(A_3 \mid A_1), \phi_4 = P(A_4 \mid A_2), \phi_5 = P(A_5 \mid A_2, A_3), \phi_6 = P(A_6 \mid A_3)$.

5.1.1 A single marginal

Let us first assume that we wish to calculate $P(A_4)$. From the chain rule, we have

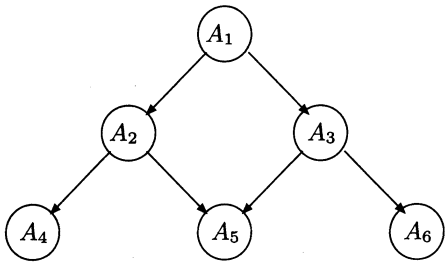

FIGURE 5.1. The simple Bayesian network, BN.

$$P(U) = \phi_1\phi_2\phi_3\phi_4\phi_5\phi_6 \text{ and } P(A_4) = \sum_{A_1,A_2,A_3,A_5,A_6} \phi_1\phi_2\phi_3\phi_4\phi_5\phi_6.$$

To avoid calculating $P(U)$, we use the distributive law (Section 1.3.6):

$$\begin{aligned} P(A_4) \;=\; & \sum_{A1} \phi_1(A_1) \sum_{A_2} \phi_2(A_2, A_1)\phi_4(A_4, A_2) \sum_{A_3} \phi_3(A_3, A_1) \\ & \sum_{A_5} \phi_5(A_5, A_2, A_3) \sum_{A_6} \phi_6(A_6, A_3). \end{aligned}$$

First, calculate $\phi'_6(A_3) = \sum_{A_6} \phi_6(A_6, A_3)$, then multiply $\phi'_6(A_3)$ on $\phi_5(A_5, A_2, A_3)$ and calculate $\phi'_5(A_2, A_3) = \sum_{A_5} \phi_5\ (A_5, A_2, A_3)\ \phi'_6(A_3)$; $\phi'_5(A_2, A_3)$ is multiplied on $\phi_3(A_3, A_1)$, and so forth. Notice that in the calculation of $\phi'_5(A_2, A_3)$ you can apply the distributive law again; that is, you need not multiply with $\phi'_6(A_3)$ before you marginalize A_3 out. The calculation is sketched graphically in Figure 5.2.

The reason for using the distributive law is to reduce the size of the tables to handle. The full joint, $P(U)$, requires a space incorporating all six variables. For the process illustrated in Figure 5.2, the largest potential to handle contains three variables. In Figure 5.3, the structure is repeated, but in each bucket we have indicated the variables to handle, and the variables in a mailbox indicate the domain of the potential communicated.

In the preceding calculations, we performed the marginalizations in a particular order, namely A_6, A_5, A_3, A_2, A_1, and this is reflected in the structure of Figure 5.2. Because marginalization is commutative (Section 1.3.6), it can be done in any order. It is standard to use the term *elimination order* rather than marginalization order. If we use the reversed elimination order, we get the structure in Figure 5.4.

Figure 5.5 illustrates the domains to handle for the last elimination order. As can be seen, the domains for the first order are smaller than the domains for the last order.

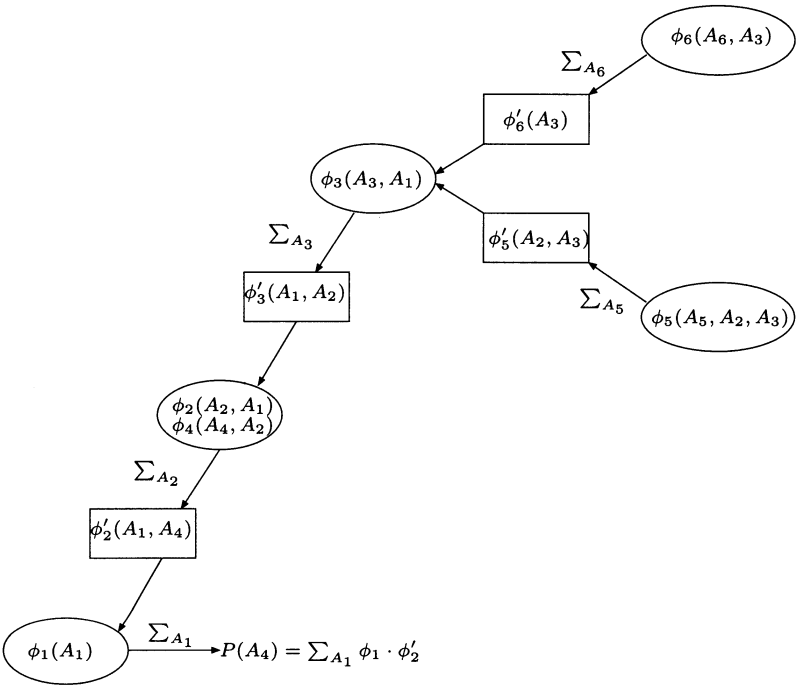

FIGURE 5.2. An illustration of the process of marginalizing down to A_4. The elliptic nodes are buckets containing potentials. In a bucket, the potentials are multiplied by the incoming potentials, a variable is marginalized out, and the result is placed in a mailbox (a rectangular node) for a neighbor bucket.

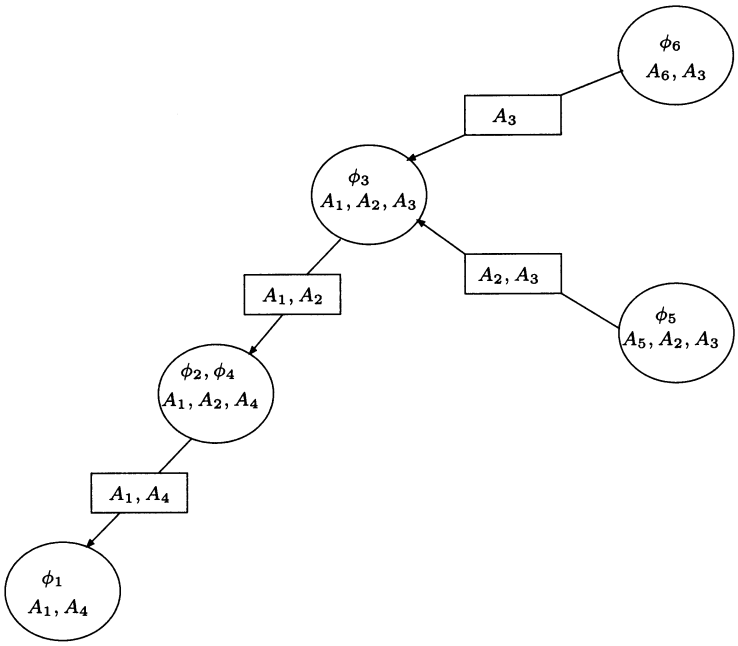

FIGURE 5.3. A structure indicating the domains of the various potentials to handle.

Because the size of the domains to handle is a good measure of complexity, we will address the task of finding an elimination order yielding the smallest domains to handle.

5.1.2 *Different evidence scenarios*

In the preceding calculations, we assumed that no evidence was entered into the network. By analyzing the process illustrated in Figure 5.2, we realize some simplifications. Because $\phi_5 = P(A_5 \mid A_2, A_3)$ and $\phi_6 = P(A_6 \mid A_3)$, we have that $\phi'_5 = \sum_{A_5} P(A_5 \mid A_2, A_3) = \mathbf{1}$ and $\phi'_6 = \sum_{A_6} P(A_6 \mid A_3) = \mathbf{1}$, where $\mathbf{1}$ is the unit potential (Section 1.3.5). Also,

$$\phi'_3 = \sum_{A_3} \phi_3 \phi'_5 \phi'_6 = \sum_{A_3} \phi_3 \mathbf{1} \cdot \mathbf{1} = \sum_{A_3} \phi_3 = \sum_{A_3} P(A_3 \mid A_1) = \mathbf{1}.$$

ϕ'_3 is void, and the entire process is reduced to calculating $\sum_{A_1} P(A_1) \sum_{A_2} P(A_2 \mid A_1) P(A_4 \mid A_2)$.

The nodes A_3, A_5, and A_6 are examples of so-called *barren nodes*. Nodes or any of their successors that have not received evidence are called barren. The conditional probability potential attached to a barren node has an impact only on descendant nodes.

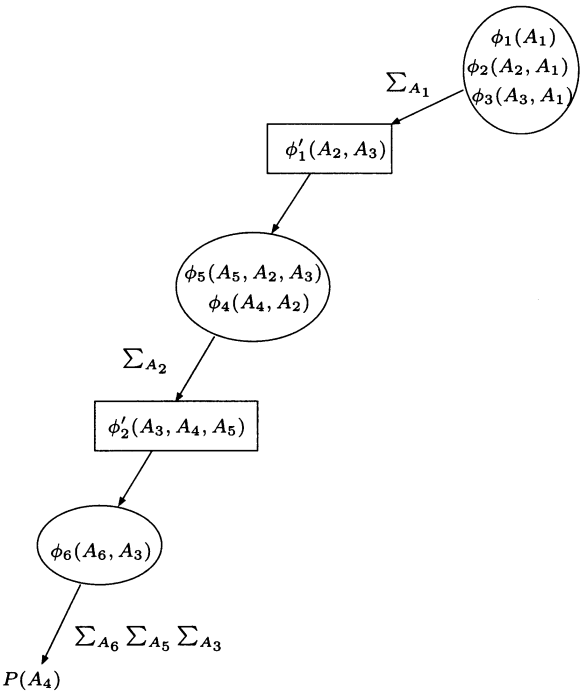

FIGURE 5.4. The structure resulting from eliminating in an order that is the reverse of the one from Figure 5.2.

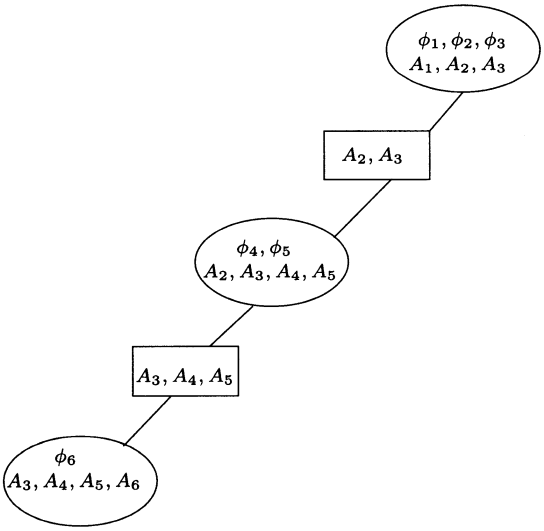

FIGURE 5.5. The domains for the elimination order A_1, A_2, A_3, A_5, A_6.

If we have the evidence $A_5 = a^5$ and $A_6 = a^6$, the evidence is represented as two 0–1 potentials, $\underline{e}_5$ and $\underline{e}_6$ (Section 1.4.6). The formula is

$$P(A_4, e) = \sum_{A_1, A_2, A_3, A_5, A_6} \phi_1 \phi_2 \phi_3 \phi_4 \phi_5 \phi_6 \underline{e}_5 \underline{e}_6,$$

and we have (Section 1.3.7)

$$P(A_4 \mid e) = \frac{P(A_4, e)}{\sum_{A_4} P(A_4, e)}.$$

To calculate $P(A_4, e)$, the effect on the frame in Figure 5.2 is that the two evidence potentials are added in the buckets with ϕ_5 and ϕ_6 attached to them (see Figure 5.6).

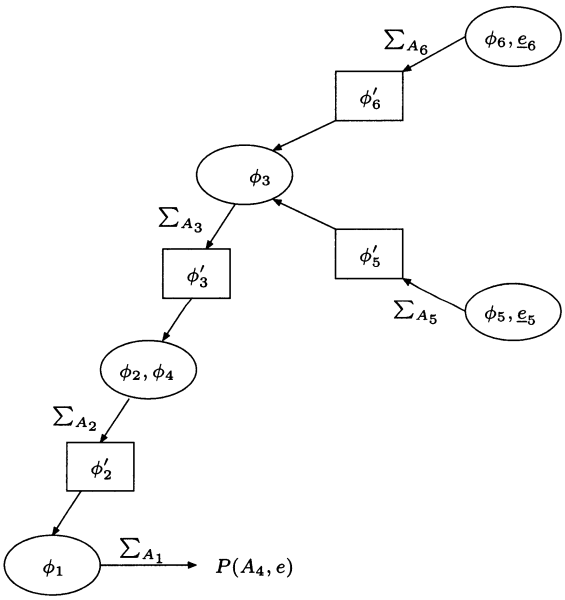

FIGURE 5.6. The frame from Figure 5.2 incorporating the evidence $e_5 : A_5 = a^5$ and $e_6 : A_6 = a^6$.

The effect of e is that the variables A_5 and A_6 are instantiated in the potentials ϕ_5 and ϕ_6, and the marginalizations of A_5 and A_6 are redundant; that is, $\phi'_5 = P(A_6 = a^6 \mid A_2, A_3)$ and $\phi'_6 = P(A_6 = a^6 \mid A_3)$.

The process in Section 5.1.1 is sufficiently general to encompass all types of evidence scenarios. The task is to supplement this general process with methods taking advantage of simplifications due to the particular evidence scenario, such as identification of barren nodes.

5.1.3 *All marginals*

Assume that we wish to compute $P(A_i, e)$ for all i. Without taking advantage of the special evidence scenario, we can for each node use the method from Section 5.1.1. Assume that we calculate $P(A_2, e)$ through the elimination order A_6, A_5, A_3, A_1, A_2. Then, the frame of potentials looks as in Figure 5.7.

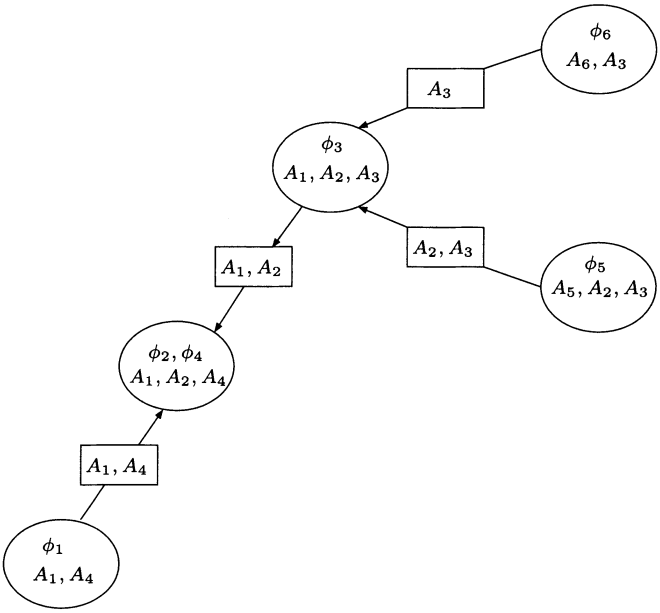

FIGURE 5.7. A frame for computing $P(A_2, e)$ through the elimination order A_6, A_5, A_3, A_1, A_2.

As can be seen, the frame in Figure 5.7 is very similar to the frame in Figure 5.4. Only one arrow is reversed, and many calculations from the calculation of $P(A_4, e)$ can be reused. In this chapter, we present a systematic way of exploiting reuse when calculating all marginals. The resulting method has a complexity equivalent to two single-variable marginalizations.

5.2 Graph-Theoretic Representation

As illustrated in Section 5.1, belief updating for Bayesian networks consists basically of calculating sums of products. In this section, we deal systematically with this task without explicit reference to Bayesian networks. The methods presented are general and can be applied to a large variety of tasks.

5.2.1 Task and notation

We will work with a set of real-valued potentials $\Phi = \phi_1, \ldots, \phi_m$ over finite variables from the universe $U = \{A_1, \ldots, A_n\}$.

Let ψ be any set of potentials. The product of all potentials in ψ is denoted $\prod \psi$. We will also use the notation $\prod_{i=1}^{k} \psi_i$ for the product $\psi_1, \ldots, \psi_k$, and if the borders are apparent from the context, we write $\prod \psi_i$.

$\sum_X \phi(X, Y, \ldots, Z)$ is the sum $\phi(x_1, Y, \ldots, Z) + \ldots + \phi(x_k, Y, \ldots, Z)$, and it is a potential over $(Y, \ldots, Z)$. We say that X has been *marginalized* out of $\phi(X, Y, \ldots, Z)$. If V is a set of variables, then $\sum_V$ is a notation for marginalizing out all variables in V. Because marginalization is commutative (Section 1.3.6), this notation is unambiguous.

Instead of sum notation, we may also use projection notation. We let $\phi^{\downarrow X}(X, Y, \ldots, Z)$ denote the potential resulting from marginalizing out $(Y, \ldots, Z)$. If W is a set of variables, then $\phi^{\downarrow W}$ denotes the result of marginalizing out all variables except the members of W.

Task Compute $(\prod \Phi)^{\downarrow A_i}$ for all A_i.

Definition 5.1 Let Φ be a set of potentials, and let X be a variable. X is *eliminated from* Φ through the following procedure:

1. Remove all potentials in Φ with X in their domains. Call the set Φ_X.
2. Calculate $\phi^{-X} = \sum_X \prod \Phi_X$.
3. Add ϕ^{-X} to Φ. Call the result Φ^{-X}.

Note that elimination of the variable X corresponds to using the distributive law on the product. Instead of calculating the product, we keep the factors in a bucket and do not multiply before we are forced to do so.

Proposition 5.1 *The task* $(\prod \Phi)^{\downarrow X}$ *is solved by repeatedly eliminating the variables except for* X.

It remains to establish an elimination order.

5.2.2 Domain graphs

To get an overview of the consequences of various elimination orders, the task is represented graphically.

Definition Let $\Phi = \{\phi_1, \ldots, \phi_m\}$ be potentials over $U = \{A_1, \ldots, A_n\}$ with $\mathrm{dom}\phi_i = D_i$. The *domain graph* for Φ is the undirected graph with the variables of U as nodes and with a link between pairs of variables being members of the same D_i.

For the sake of exposition, we assume throughout the chapter that the graphs considered are connected.

Example In Section 5.1.1, we dealt with a Bayesian network over the potentials $\Phi = \{\phi_1(A_1), \phi_2(A_2, A_1), \phi_4(A_4, A_2), \phi_3(A_3, A_1), \phi_5(A_5, A_2, A_3), \phi_6(A_6, A_3)\}$. The domain graph for Φ is given in Figure 5.8.

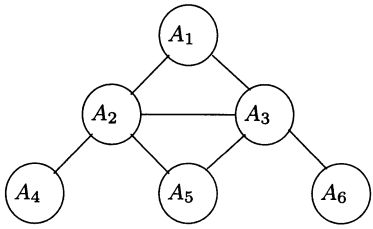

FIGURE 5.8. The domain graph for $\Phi = \{\phi_1(A_1), \phi_2(A_2, A_1), \phi_3(A_3, A_1), \phi_4(A_4, A_2), \phi_5(A_5, A_2, A_3), \phi_6(A_6, A_3)\}$.

Compared to the initial Bayesian network in Figure 5.1, we see that the link (A_2, A_3) is new. It is often called a *moral link* because it connects two nodes with a common child, and the domain graph for a Bayesian network is called the *moral graph*.

When we eliminate a variable X, we work with the product of all potentials with X in the domain. The domain of this product consists of X and its neighbors in the domain graph, and when X is eliminated, the resulting potential has all X's neighbors in its domain. This means that in the domain graph for Φ^{-X} all neighbors of X are pairwise linked. In Figure 5.9, we show the domain graph for the example in Figure 5.8 with A_3 eliminated.

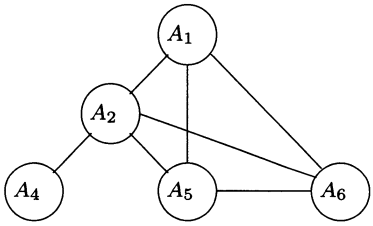

FIGURE 5.9. The domain graph for Φ^{-A_3} from Figure 5.8.

Note that the graph in Figure 5.9 has several new links. These new links are called *fill-ins*. The introduction of fill-ins highlights the fact that when eliminating A_3 you work with a potential over a domain that was not present originally. In order to avoid working with new domains, you try to avoid fill-ins. To say it another way, an elimination sequence that does not introduce fill-ins requires less space than an elimination sequence that introduces fill-ins.

In Section 5.1, we considered calculation of $P(A_4)$. In the graph-theoretic framework, it corresponds to constructing an elimination sequence ending with A_4. For the domain graph in Figure 5.8, it is possible to eliminate down to A_4 without introducing fill-ins: $A_6, A_5, A_3, A_1, A_2, A_4$. Such a sequence is called a *perfect elimination sequence*. There are several per-

fect elimination sequences ending with A_4, and an optimal elimination sequence will be found among them. In Figure 5.8, we see that the sequence $A_5, A_6, A_3, A_1, A_2, A_4$ as well as $A_1, A_5, A_6, A_3, A_2, A_4$ and A_6, A_1, A_3, A_5, A_2, A_4 are perfect elimination sequences.

Proposition 5.2 *Let $X_1, \ldots, X_k$ be a perfect elimination sequence, and let X_j be a node with a complete neighbor set.*[1] *Then, the sequence X_j, $X_1 \ldots, X_{j-1}, X_{j+1}, \ldots, X_k$ is also a perfect elimination sequence.*

Proof: If you start by eliminating X_j, you do not introduce fill-ins. Consider variable X_i. When you eliminate X_i, you look at the noneliminated neighbors, and if a pair of them is not linked, you introduce a fill-in. Eliminating X_j before X_i does not give X_i new neighbors, and it will not enforce new fill-ins when X_i is eliminated. □

The complexity of using a particular elimination sequence is characterized by the set of domains for the potentials used. For the elimination order $A_6, A_5, A_3, A_1, A_2, A_4$, the set of domains is $\{\{A_6, A_3\}, \{A_2, A_3, A_5\}$, $\{A_1, A_2, A_3\}, \{A_1, A_2\}, \{A_2, A_4\}\}$. If a domain is a subset of another domain, then it does not require extra space and we need not consider it. For example, the set $\{A_1, A_2\}$ is removed from the preceding domain set.

Definition The *domain set* of an elimination sequence is the set of domains of potentials produced during the elimination where potentials that are subsets of other potentials are removed.

Unfortunately, it does not hold that if you eliminate without introducing fill-ins, then the domain set consists only of domains from the initial set of potentials. For the preceding perfect elimination sequence, we have that when A_3 is eliminated you work with a potential with domain $\{A_1, A_2, A_3\}$, which is not one of the initial domains. However, there is no way to avoid it. No matter which of the three variables you eliminate first, you will produce a potential with all three variables in the domain. In general, it holds that if the set V of variables is a complete set in the domain graph, then any elimination sequence will contain a potential with a domain including V.

Proposition 5.3 *All perfect elimination sequences produce the same domain set, namely the set of cliques*[2] *of the domain graph.*

Proof: Let V be a clique. Let X be the first variable from V to be eliminated. When X is eliminated, we produce a domain D consisting of X and all its neighbors. Because all elements of V are neighbors of X, D must contain V. Let Y be a member of D. After elimination of X, there is a link between Y and all members of V. The elimination does not produce fill-ins, so the links must have been present initially, and because V is a maximal perfect

[1] A set of nodes is *complete* if all nodes are pairwise linked.

[2] A complete set is a *clique* if it is not a subset of another complete set (a maximal complete set).

set, Y must be a member of V. Hence, the cliques must be members of the domain set.

Assume that W is a member of the domain set. Because the elimination does not produce fill-ins, W must be a perfect set in the domain graph. If W is not maximal, it is a subset of a clique V, and V is a member of the domain set, so W cannot be a member. □

From Proposition 5.3, we can conclude that any perfect elimination sequence ending with the variable A is optimal with respect to calculating $P(A)$. The full task is to compute the marginals down to each variable, so the task can be solved by establishing an optimal elimination sequence for each variable.

5.3 Triangulated Graphs and Join Trees

Before continuing with the belief updating task, we deal in detail with some purely graph-theoretic concepts. They will be used on the belief updating task in the next section.

Definition An undirected graph with a perfect elimination sequence is called a *triangulated graph*. Note that the term "triangulated" may be misleading. The graph (b) in Figure 5.10 is not triangulated.

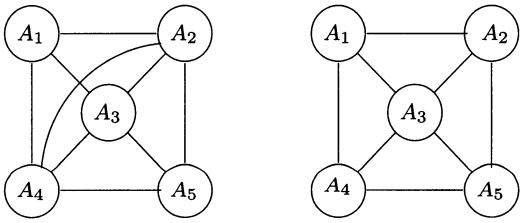

FIGURE 5.10. (a) A triangulated graph; (b) a nontriangulated graph.

Notation Let X be a node in an undirected graph. The set of neighbors of X we denote N_X, and the set of neighbors plus X we denote F_X, the family of X. If the nodes of the graph are enumerated, we use the index to write N_i rather than N_{X_i}. Nodes with a complete neighbor set are called *simplicial*. A neighbor to a node X is said to be *adjacent* to X. Note that X is simplicial if and only if F_X is a clique.

Proposition 5.4 *Let G be a triangulated graph, and let X be a simplicial node. Let G' be the graph resulting from eliminating X from G (see Figure 5.11). Then, G' is a triangulated graph.*

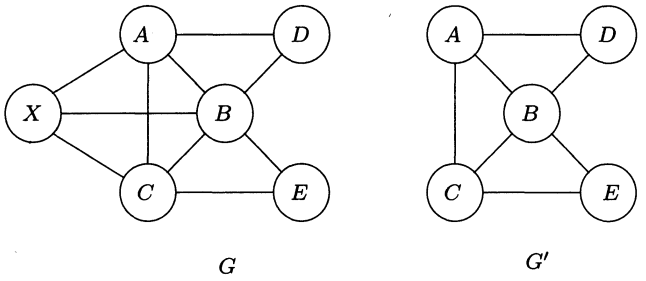

FIGURE 5.11. If F_X is a complete set, you eliminate X from G by simply removing X together with its links.

Proof: Proposition 5.2. □

Note that a triangulated graph always has at least one simplicial node, namely the first one in the elimination sequence. Actually, there are at least two.

Theorem 5.1 *A triangulated graph with at least two nodes has at least two simplicial nodes.*

Proof: We prove by induction a slightly stronger statement: let G be a noncomplete triangulated graph with at least three nodes. Then, it has at least two nonadjacent simplicial nodes.

Certainly, any noncomplete triangulated graph with three nodes has two nonadjacent simplicial nodes (see Figure 5.12).

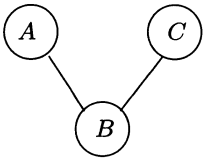

FIGURE 5.12. A connected noncomplete triangulated graph with three nodes.

Assume the statement to be true for all graphs with less than n nodes, and let G be a noncomplete triangulated graph with n nodes. The first node, X, in the elimination sequence is simplicial, and we must find another one not adjacent to X. Let G' be the graph resulting from removing X from G.

G' is triangulated, and any simplicial node in G' is either simplicial in G or a member of N_X.

Because G is not complete, it must contain nodes that are not members of N_X. If G' is complete, any of these nodes can do. If G' is not complete, we know from the induction hypothesis that it has at least two nonadjacent simplicial nodes. If both were neighbors of X, they would be adjacent. □

Corollary 5.1 In a triangulated graph, each variable A has a perfect elimination sequence ending with A.

Proof: Let A be any node in the triangulated graph G. Eliminate a simplicial node $X (X \neq A)$. Proposition 5.4 yields that the resulting graph is triangulated, and you can repeatedly apply Theorem 5.1 until only A is left. □

From Corollary 5.1, we see that if you have established one perfect elimination sequence, then you can easily establish a perfect elimination sequence down to any variable. In other words, you can for each variable A establish an optimal sequence of marginalizations for calculating $P(A)$. We give the details in Section 5.4.

Unfortunately, it does not hold that all domain graphs are triangulated. The following theorem gives an easy way of checking whether a graph is triangulated, and if it is, it also gives a simple way of establishing an elimination sequence.

Theorem 5.2 *An undirected graph is triangulated if, and only if, all nodes can be eliminated by successively eliminating a simplicial node X.*

Proof: If all nodes can be eliminated by successively eliminating simplicial nodes, then we produce a perfect elimination sequence, and the graph is triangulated.

Now, assume that the undirected graph is triangulated. Let us eliminate any simplicial node. Proposition 5.4 yields that the resulting graph is triangulated, and we can continue the procedure. □

To check whether a graph is triangulated, you repeatedly eliminate simplicial nodes. At some stage, you run into a situation where you cannot eliminate more nodes. If the node set is empty, then the graph is triangulated; if not, then the graph is not triangulated.

In general, it is NP-hard to determine the set of cliques in a graph. For triangulated graphs, Proposition 5.3 and Theorem 5.2 yield an easy procedure.

Construction To determine the set of cliques in a triangulated graph, you can do as follows

1. Eliminate a simplicial node X. F_X is a clique candidate.
2. If F_X does not include all remaining nodes, go to 1.
3. Prune the set of clique candidates by removing sets that are subsets of other clique candidates.

The resulting set is the set of cliques.

5.3.1 *Join trees*

Definition Let G be the set of cliques from an undirected graph, and let the cliques of G be organized in a tree T. T is a *join tree* if for any pair of nodes V, W all nodes on the path between V and W contain the intersection $V \cap W$.

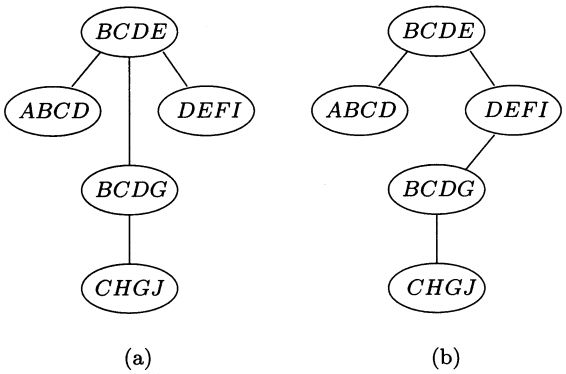

FIGURE 5.13. (a) A join tree; (b) not a join tree.

Theorem 5.3 *If the cliques of an undirected graph G can be organized into a join tree, then G is triangulated.*

Proof: Let the cliques be organized in a join tree, and let V be a leaf clique with unique neighbor clique W. Any member of V that is a member of another clique must – due to the join tree condition – also be a member of W. Therefore, V must contain at least one variable X not contained in any other clique (otherwise V would be a subset of W). Then, F_X must be complete, and X can be eliminated without creating fill-ins. We can repeat eliminating variables that are only members of V, and when all these have been eliminated, we have a graph G' with the same cliques as G except for V. Then, the join tree for G with the node V removed is a join tree for G', and we can continue with eliminating a variable from a leaf in G'. □

Theorem 5.4 *If the undirected graph G is triangulated, then the cliques of G can be organized into a join tree.*

The proof is a construction of a join tree from a triangulated graph. To illustrate the construction, we use the graph in Figure 5.14.

Construction Establish an elimination sequence in the following way. Start with a simplicial node X. Then, F_X is a clique. Continue eliminating nodes from F_X that have only neighbors in F_X. Give F_X an index i according to the number of nodes eliminated, and denote the set of the remaining nodes S_i. This set is called a *separator*. Choose a new clique in the graph G' with the eliminated nodes removed, and repeat the process

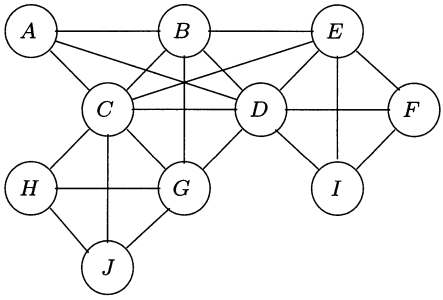

FIGURE 5.14. A triangulated graph.

with the index starting at i. Continue to do so until all cliques have been eliminated. Figure 5.15 shows the result of this process on the graph in Figure 5.14.

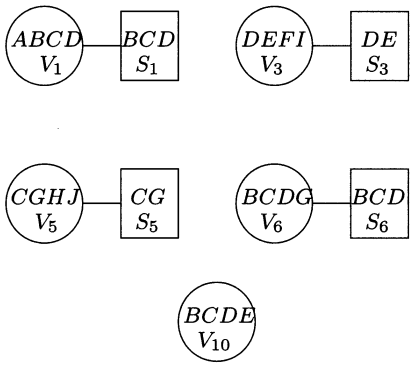

FIGURE 5.15. The cliques, separators, and indices resulting from the graph in Figure 5.14. The elimination sequence used is $A, F, I, H, J, G, B, C, D, E$.

When the parts have been established as indicated in Figure 5.15, each separator S_i is connected to a clique $V_j (j > i)$ such that $S_i \subset V_j$ (see Figure 5.16). This is always possible because S_i is a complete set, and when the first node from S_i is eliminated, it must be when dealing with a clique of higher index than i, and it must contain all of S_i. For convenience, we talk of the direction from V_i over S_i to V_j as upward, and we call V_j a parent of V_i.

We must prove that the structure constructed is a tree and has the join tree property. Each clique has at most one parent, so there cannot be multiple paths, and the structure is a tree.

To prove the join tree condition, consider the cliques V_i and $V_j (i < j)$, and let X be a member of both. There is a unique path between V_i and V_j, and we will prove that X is a member of all cliques on that path. Because X is not eliminated when dealing with V_i, it must be a member of S_i and,

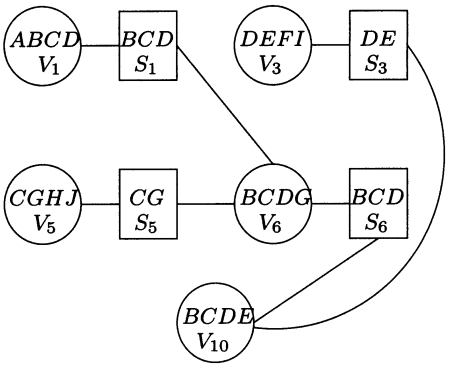

FIGURE 5.16. A join tree (expanded with separators) resulting from the construction applied on the graph in Figure 5.14.

from the construction, X must be a member of V_i's parent V_k. If $k = j$, we are finished; otherwise we continue the argument for the smallest of the two.

Remark The separators are called so because any separator S divides the graph into two parts, and all paths connecting the two parts must pass through S. If the graph is a moral graph for a Bayesian network, the two parts are d-separated given S.

A join tree provides the framework for constructing perfect elimination sequences. Notice namely that the simplicial nodes are those with all noneliminated neighbors in one clique, and two nodes are neighbors if they are members of the same clique. Hence, all perfect elimination sequences can be constructed from a join tree by repeatedly eliminating simplicial nodes.

5.4 Propagation in Junction Trees

Definition Let Φ be a set of potentials with a triangulated domain graph, G. A *junction tree* for Φ is a join tree for G with the following further structure: each potential ϕ in Φ is attached to a clique containing $\mathrm{dom}(\phi)$; each link has the appropriate separator attached; each separator contains two mailboxes, one for each direction.

If Φ is a set of conditional probabilities for a Bayesian network BN together with evidence potentials for the evidence e, we say that the junction tree represents BN with evidence e.

Notation The propagation algorithm presented here deals with sets of potentials. A set of potentials is a representation of the product of the member potentials. Let Φ be a set of potentials whose domains are subsets of V, and let W be a subset of V. Then, $\Phi^{\downarrow W}$ is a set of potentials resulting from successively eliminating the variables in $V \setminus W$ as described in Defini-

tion 5.1. Because the elimination order is arbitrary, this notation seems to introduce some ambiguity with respect to the functions in the resulting set. Because we treat the sets as representations of products, and the product is independent of the elimination order, we will not deal with this apparent ambiguity.

Example 5.2 Consider the set $\psi = \{\phi_1(A), \phi_2(A,B), \phi_3(A,C), \phi_4(C,D), \phi_5(C)\}$, and let $W = \{B,C\}$. Then, $\psi^{\downarrow W} = \{\sum_A \phi_1(A)\phi_2(A,B)\phi_3(A,C), \sum_D \phi_4(C,D), \phi_5(C)\}$.

Before giving a general description of the propagation algorithm, we will go through an example.

Example 5.3 Consider the Bayesian network in Section 5.1 with potentials $\phi_1 = P(A_1), \phi_2 = P(A_2 \mid A_1), \phi_3 = P(A_3 \mid A_1), \phi_4 = P(A_4 \mid A_2), \phi_5 = P(A_5 \mid A_2, A_3), \phi_6 = P(A_6 \mid A_3)$ and with the domain graph in Figure 5.8. We know that the elimination sequence $A_6, A_5, A_3, A_1, A_2, A_4$ is perfect. The domain graph has a join tree over the cliques $V_1 = \{A_3, A_6\}$, $V_2 = \{A_2, A_3, A_5\}$, $V_4 = \{A_1, A_2, A_3\}$, $V_6 = \{A_2, A_4\}$ and the separators $S_1 = \{A_3\}$, $S_2 = \{A_2, A_3\}$, $S_4 = \{A_2\}$. The junction tree is shown in Figure 5.17.

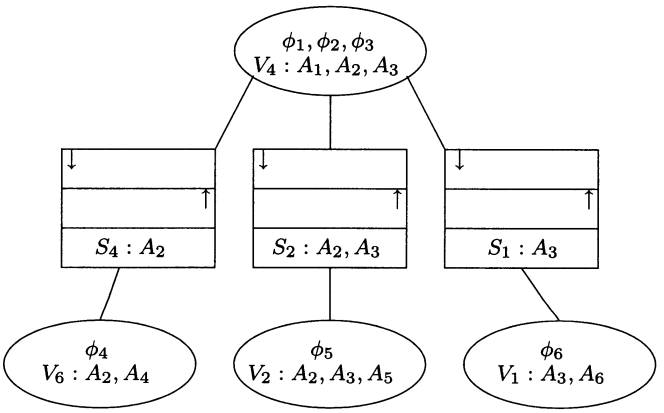

FIGURE 5.17. A junction tree for the Bayesian network in Figure 5.8.

To calculate $P(A_4)$, we find a clique containing $A_4 (V_6)$. It is made a temporary root, and we send messages in the direction of V_6 starting from the leaf cliques. The message $\psi_1 = \phi_6^{\downarrow A_3} = \phi_6^{\downarrow S_1}$ is placed in the appropriate S_1 mailbox, and the message $\psi_2 = \phi_5^{\downarrow \{A_2, A_3\}} = \phi_5^{\downarrow S_2}$ is placed in the appropriate S_2 mailbox. Next, V_4 assembles the incoming messages and the potentials held to the set $\Phi_4 = \{\psi_1, \psi_2, \phi_1, \phi_2, \phi_3\}$. The variables A_1 and A_3 are eliminated from Φ_4, and the result, $\psi^4 = (\phi_1\phi_2(\phi_3\psi_2\psi_1)^{\downarrow\{A_1,A_2\}})^{\downarrow A_2} = \sum_{A_1} \phi_1\phi_2 \sum_{A_3} \phi_3\psi_2\psi_1$, is placed in the appropriate mailbox (see Figure 5.18). Now, V_6 can collect its message, multiply it by ϕ_4, and marginalize A_2 out to get $P(A_4)$.

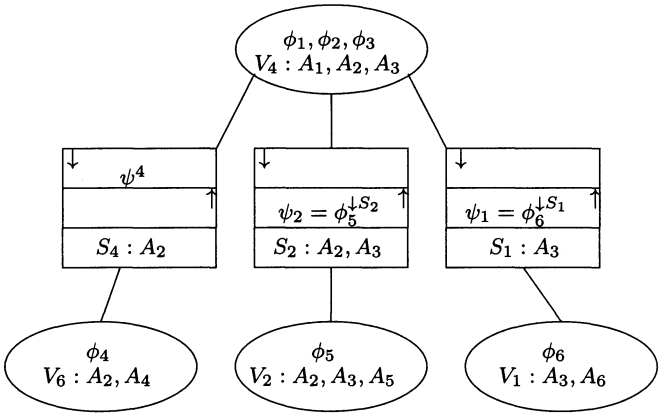

FIGURE 5.18. The cliques V_1 and V_2 have sent messages to their separators, and V_4 has sent the message $\sum_{A_1} \phi_1\phi_2 \sum_{A_3} \phi_3\psi_2\psi_1$ to S_4.

The process just described is called *to collect evidence* to V_6. To calculate the marginal for another variable X, we can collect to a clique containing X. If, for example, we wish to calculate $P(A_6)$, we can collect to V_1. We can also prepare the junction tree for the calculation of all marginals: send messages in the direction away from V_6. This process is called *to distribute evidence*. First, V_6 sends the message $\psi_4 = \phi_4^{\downarrow A_2}$ to S_4, and V_4 sends a message to S_2 as well as S_1 (see Figure 5.19). When the message for S_2 is calculated, the set $\{\psi_4, \phi_1, \phi_2, \phi_3, \psi_1\}$ is assembled, and A_1 is marginalized out. Here, we only multiply the potentials that have A_1 in the domain, and the message becomes a set of potentials: $\{\psi_4, \sum_{A_1} \phi_1\phi_2\phi_3, \psi_1\}$.

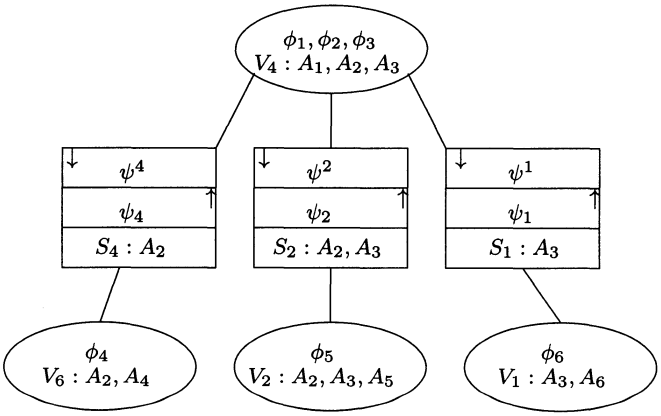

FIGURE 5.19. The junction tree after a full propagation. $\psi^2 = \{\psi_4, \sum_{A_1} \phi_1\phi_2\phi_3, \psi_1\}, \psi^1 = \sum_{A_2} \psi_2\psi_4 \sum_{A_1} \phi_1\phi_2\phi_3$.

When collect as well as distribute evidence has been performed, we have performed a full propagation, and to calculate a marginal $P(X)$ we find a clique V containing X. Take, for example, A_3. The clique V_1 contains A_3. The incoming message to V_1 is the message for collecting evidence to V_1, and therefore it corresponds to a perfect elimination sequence ending with the nodes A_6 and A_3. This means that the product $\phi_6\psi^1$ is the projection of the entire product down to $\{A_3, A_6\}$, and we can easily calculate $P(A_3)$ as well as $P(A_6)$.

There is a slightly easier way of calculating A_3. Consider the separator S_3. It consists of A_3 alone. For the product of the two messages of S_1, we have

$$\begin{aligned}\psi^1\psi_1 &= (\sum_{A_2}\psi_2\psi_4\sum_{A_1}\phi_1\phi_2\phi_3)(\sum_{A_6}\phi_6)\\ &= \sum_{A_5}\phi_5\sum_{A_4}\phi_4\sum_{A_1}\phi_1\phi_2\phi_3\sum_{A_6}\phi_6\\ &= (\phi_5\phi_4\phi_1\phi_2\phi_3\phi_6)^{\downarrow A_3} = P(A_3).\end{aligned}$$

Next, assume that you have the evidence $e = e_5 : "A_5 = a^5, "e_6 : A_6 = a^6$. The evidence e is represented as two 0–1 potentials $\underline{e}_5$ and $\underline{e}_6$. To calculate the probabilities $P(X, e)$, you place the two evidence potentials in appropriate cliques (V_2 and V_1) and perform a full propagation.

5.4.1 Lazy propagation in junction trees

Each clique V holds a set of potentials denoted Φ_V. Each separator has two mailboxes, one for each direction of the link. The messages stored in the mailboxes are sets of potentials. The messages are denoted ψ_S or ψ^S, depending on the direction.

The basic operation in the lazy propagation procedure is *message passing*.

Definition Let V be a clique with set of potentials Φ_V, and let S be a neighboring separator. Let $S_1, \ldots, S_k$ be the other neighbor separators of V. Assume that each S_i has received a message Ψ_i for V.

Then, V can *pass the message* $(\Phi_V \cup \Psi_1 \cup \ldots \cup \Psi_k)^{\downarrow S}$ to S, and we say that the direction $V - S$ is *triggered*.

The propagation method consists of repeatedly passing messages along triggered directions.

Proposition 5.5 *If you repeatedly pass messages along triggered directions in a junction tree, then you need not stop before a message has been passed in both directions over each link. In that situation, we say that the junction tree is* full.

Proof: See Exercise 5.19. □

As shown in Example 5.3, you can start off by directing all messages toward a chosen temporary root R. In other words, the junction tree is given a direction from R and outward, and the messages are passed in the opposite direction from leaves and inward (see Figure 5.20). This procedure is called CollectEvidence(R).

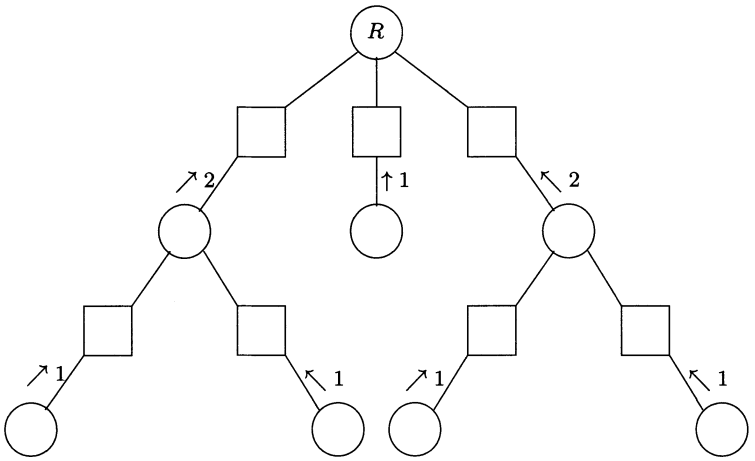

FIGURE 5.20. The message passing in CollectEvidence(R).

Notice that the message passing in CollectEvidence(R) corresponds to a perfect elimination sequence ending with the nodes of R.

To fill the junction tree after a CollectEvidence(R), you need only to place messages in the opposite directions. First, R passes a message to its neighbors, they in turn pass messages further outward, and so forth out to the leaves (see Figure 5.21). This procedure is called DistributeEvidence(R). Note that messages are only passed along triggered directions if DistributeEvidence(R) is performed after CollectEvidence(R) has been performed.

Theorem 5.5 *Let the junction tree T represent the Bayesian network BN over the universe U and with evidence e. Assume that T is full.*

1. *Let V be a clique with set of potentials Φ_V, and let $S_1, \ldots, S_k$ be V's neighbor separators and with V-directed messages $\Psi_1, \ldots, \Psi_k$. Then, $P(V, e) = \prod \Phi_V \prod \Psi_1 \cdot \ldots \cdot \prod \Psi_k$.*
2. *Let S be a separator with the sets Ψ_S and Ψ^S in the mailboxes. Then, $P(S, e) = \prod \Psi_S \prod \Psi^S$.*

Proof:

1. Consider the messages passed in the direction of V. They correspond to a CollectEvidence(V), and the message passing corresponds to

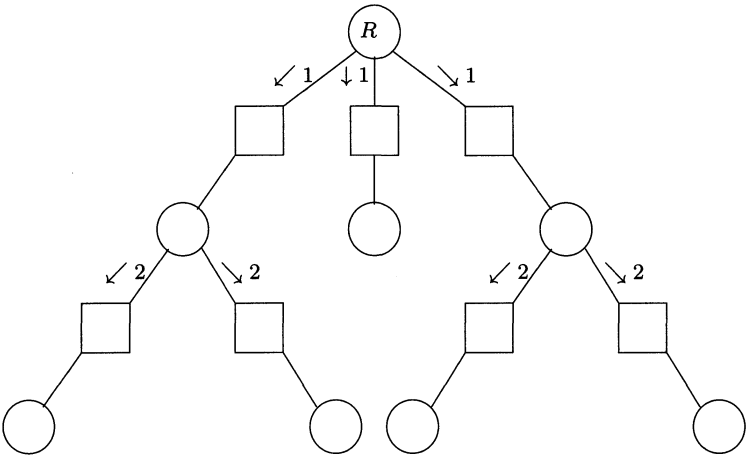

FIGURE 5.21. The messages passing in DISTRIBUTEEVIDENCE(R).

a perfect elimination sequence ending with the nodes of V. Therefore,

$$P(V,e) = P(U,e)^{\downarrow V} = \prod \Phi_V \prod \Psi_1 \cdot \ldots \cdot \prod \Psi_k.$$

2. Consider S_k as before. Because

$$\prod \Psi^k = \left(\prod \Phi_V \prod \Psi_1 \cdot \ldots \cdot \prod \Psi_{k-1}\right)^{\downarrow S_k},$$

we have

$$\begin{aligned} P(S_k,e) &= P(V,e)^{\downarrow S_k} = \left(\prod \Phi_V \prod \Psi_1 \cdot \ldots \prod \Psi_k\right)^{\downarrow S_k} \\ &= \left(\prod \Phi_V \prod \Psi_1 \cdot \ldots \cdot \prod \Psi_{k-1}\right)^{\downarrow S_k} \prod \Psi_k \\ &= \prod \Psi^k \prod \Psi_k. \end{aligned}$$

□

5.5 Exploiting the Information Scenario

As mentioned in the beginning of this chapter, the actual information scenarios can provide simplifications of the calculations. This is one of the reasons why we let lazy propagation work with sets of potentials rather than multiplied potentials.

5.5.1 *Barren nodes*

A node is *barren* if it has not received evidence and if its children are barren. Barren nodes do not contribute to the probabilities of nonbarren nodes, and therefore we need not take their potentials into account when calculating marginals of nonbarren nodes. This is illustrated in Figure 5.22.

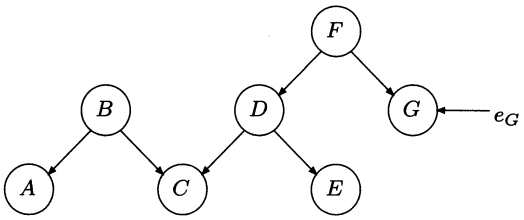

FIGURE 5.22. The nodes A, B, C, D, and E are barren.

In the calculation of $P(F \mid e)$, the part of the network with barren nodes can be discarded. Figure 5.23 shows a junction tree for the network.

To calculate $P(F \mid e)$, you can collect to the clique (F, G). We see that all marginalizations to perform are of the form $\sum_X P(X \mid pa(X))$, and from the unit potential property (Section 1.3.6) they are all **1**.

Now, assume that there is also evidence e_A for the variable A. Because A is d-separated from F, e_A does not affect $P(F \mid e)$. In the junction tree propagation, the message $\psi(B)$ from the clique (A, B) is no longer **1**. When the clique (B, C, D) produces a message for (D, F), the calculation is $\{P(C \mid B, D), P(a \mid B)\}^{\downarrow D}$. If we start marginalizing C out, we apply the unit potential property, and marginalizing B will result in a constant.

The handling of barren nodes can be taken care of using the following rule.

Barren node rule Let ψ be a set of potentials, and assume that we calculate $\psi^{\downarrow V}$. If $A \notin V$, and the only potential in ψ with A in the domain is of the form $P(A \mid W)$, then A is marginalized by discarding $P(A \mid W)$.

5.5.2 *d-separation*

When evidence is of the form that it instantiates a variable (hard evidence), then the domains to handle will be reduced with this variable. There are other simplifications due to instantiation: new pairs of variables may become d-separated, reducing the domains of the messages to communicate. We illustrate this with the example in Figure 5.24.

We will be interested in $P(E \mid e)$, and therefore we only consider collecting evidence to the clique (D, E). A junction tree for the network is shown in Figure 5.25.

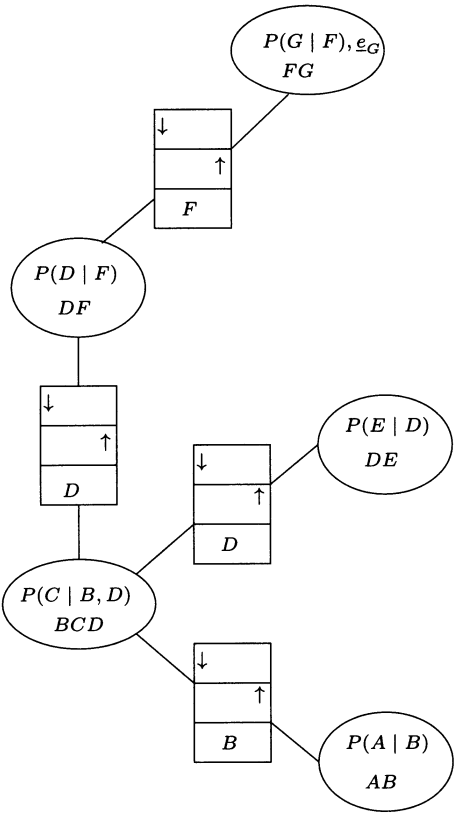

FIGURE 5.23. A junction tree for the network in Figure 5.22.

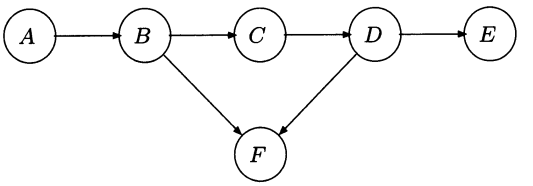

FIGURE 5.24. A Bayesian network.

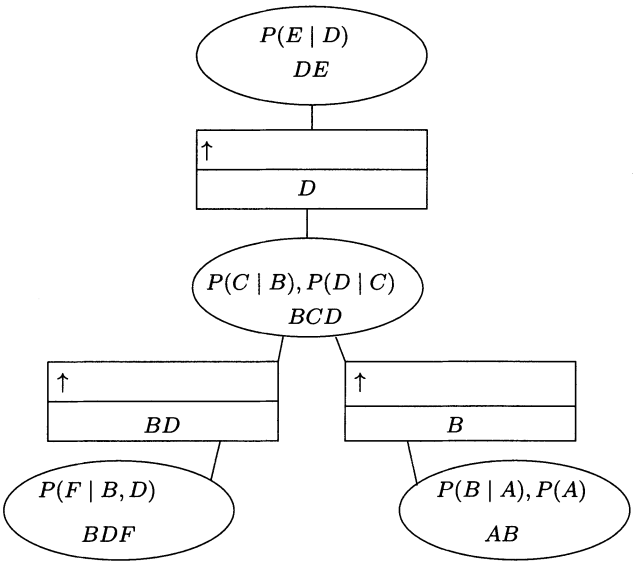

FIGURE 5.25. A junction tree for the Bayesian network in Figure 5.24. Only the upward mailboxes are indicated.

First, let A be instantiated to a. The messages are given in Figure 5.26, and we see that the evidence has an impact on $P(E, e)$ through the message $\psi_1(D)$.

Next, let C be instantiated to c. Then, A and E are d-separated. Figure 5.27 shows how this is reflected in the messages.

Finally, let F be instantiated to f. Then, A and E are not d-separated anymore. This is shown in Figure 5.28.

Note In the examples, we have entered evidence on a variable X by instantiating the potentials including X. In general, evidence can be entered by adding the corresponding evidence potential to a clique containing X, and the instantiation is effected when X must be marginalized. This means that the evidence potential is passed to separators containing X.

5.6 Nontriangulated Domain Graphs

So far, we have only considered propagation methods for potentials with a triangulated graph. For these methods, we know that the junction tree is a propagation framework having the smallest possible domains with which to work.

If the domain graph is not triangulated, we embed it in a triangulated graph and use its junction tree. In fact, we did so in Section 5.5.2 when we handled evidence.

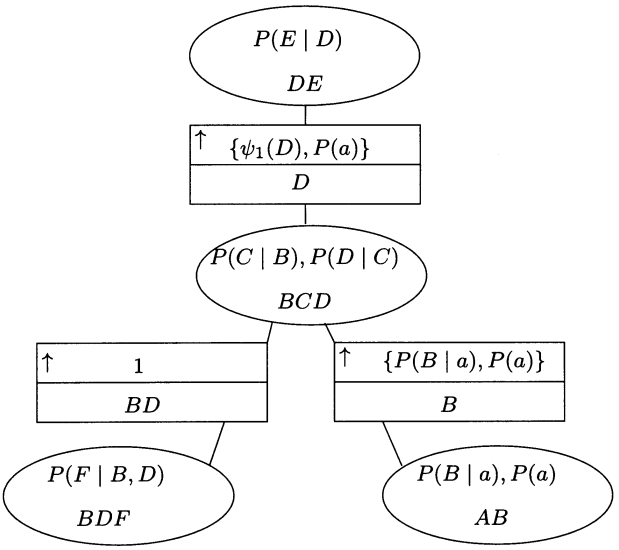

FIGURE 5.26. The messages when collecting to (D, E) for A instantiated. $\psi_1 = \sum_C P(D \mid C) \sum_B P(C \mid B) P(B \mid a) = \sum_C P(D \mid C) P(C \mid a) = P(D \mid a)$.

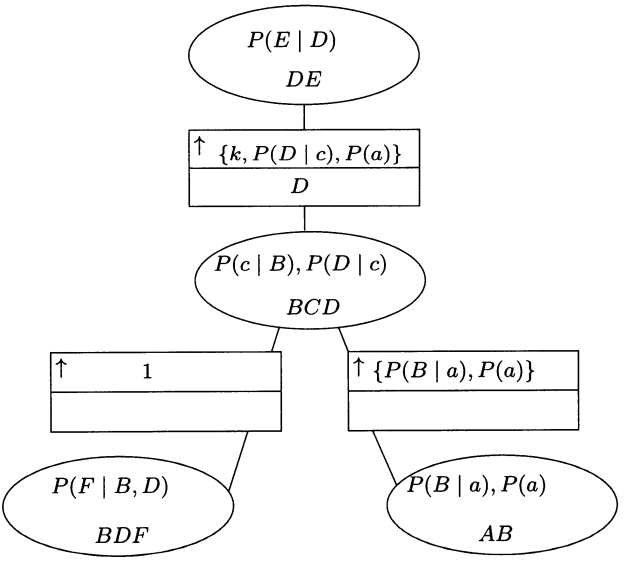

FIGURE 5.27. The messages when collecting to (D, E) for A and C instantiated. $k = \sum_B P(c \mid B) P(B \mid a) = P(c \mid a)$.

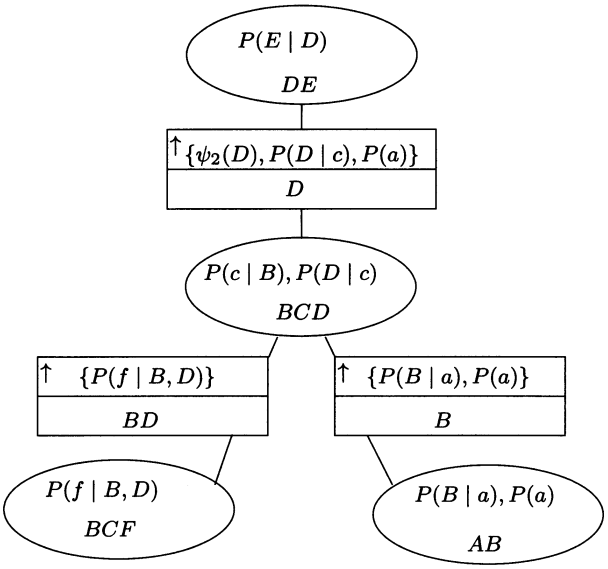

FIGURE 5.28. The messages when collecting to (D, E) for A, C, and F instantiated. $\psi_2(D) = \sum_B P(f \mid B, D)(c \mid B)P(B \mid a)$.

Example Consider the Bayesian network in Figure 5.29. After having eliminated the variables A, C, H, I, and J, we cannot eliminate any node without adding fill-ins, and the graph is not triangulated.

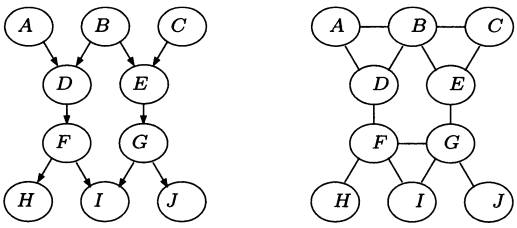

FIGURE 5.29. A Bayesian network (a) with a nontriangulated moral graph (b).

The graph in Figure 5.30 is a triangulated graph extending the moral graph in Figure 5.29. We can use a junction tree for that graph (see Figure 5.31).

5.6.1 *Triangulation of graphs*

It is quite easy to find a triangulated graph extending a graph G. You eliminate the variables in some order, and if you wish to eliminate a node with a

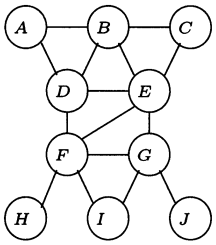

FIGURE 5.30. A triangulated graph extending the moral graph in Figure 5.29.

nonperfect neighbor set, you make it complete by adding fill-ins (the graph in Figure 5.30 is the result of eliminating in the order $A, C, H, I, J, B, G, D, E, F$). The resulting graph has a perfect elimination sequence, and therefore it is triangulated.

There are several different elimination orders, and many of them produce different triangulated graphs. We aim to work with the one yielding the smallest domains.

Definition Let V be a set of variables. For $X \in V$, $n(X)$ denotes the number of states of X. The *size of* V, $sz(V)$, is the product $\prod_V n(X)$. Let BN be a Bayesian network, let G be a triangulated graph extending BN's moral graph, and let $V_1, \ldots, V_n$ be the cliques of G. The *size of* G is the sum $size(G) = \sum_i sz(V_i)$.

Unfortunately, it is NP-hard to determine an elimination sequence yielding a triangulation of minimal size. However, there is a heuristic algorithm that has proven to give fairly good results.

Heuristics Eliminate repeatedly a simplicial node, and if this is not possible, eliminate a node X of minimal $sz(F_X)$.

Example Let the number of states for the variables in Figure 5.29 be as follows: A, B, C, H, I, and J have two states, D has four states, E has five states, F has six states, and G has seven states. After having eliminated the variables A, C, H, I, and J, we eliminate a nonsimplicial node. We have $sz(F_B) = 40, sz(F_D) = 48, sz(F_E) = 70, sz(F_F) = 168$, and $sz(F_G) = 210$. We choose to eliminate B, creating the fill-in (D, E). With this new link, we have new sizes $sz(F_D) = 120$ and $sz(F_E) = 140$. We eliminate D and add the fill-in (E, F). Now, the graph is triangulated. Actually, the triangulation is not optimal (see Exercise 5.23).

For later use, we establish the following proposition.

Proposition 5.6 *Let $A_1, \ldots, A_n$ be an elimination sequence triangulating the graph G, and let A_i and A_j be two nonneighbors in $G (i < j)$. Then, the elimination sequence introduces the fill-in (A_i, A_j) if and only if there is a path $A_i - X - \ldots - A_j$ such that all intermediate nodes are eliminated before A_i.*

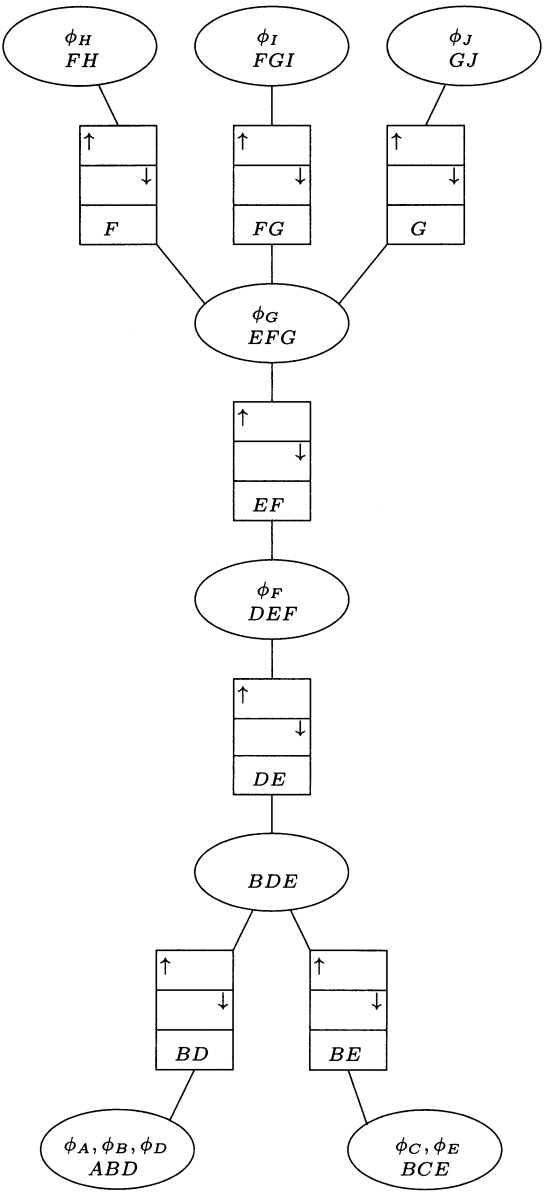

FIGURE 5.31. A junction tree with potentials from the Bayesian network in Figure 5.29. Notation: $\phi_X = P(X \mid pa(X))$.

Proof: Assume fill-ins may be introduced that violate the proposition, and let (A_i, A_j) be such a fill-in with i as small as possible. Let the link be introduced when eliminating the node A_k. Because new fill-ins cannot be attached to A_i when it has been eliminated, we must have $k < i$. One of the links (A_k, A_i) and (A_k, A_i) when eliminating A_k must be a fill-in. Let it be (A_k, A_i). Then, there is a path $A_k - X - \ldots - A_i$ such that all intermediate nodes are eliminated before A_k (see Figure 5.32). If also (A_k, A_j) is a fill-in, the same must hold. Connecting these two paths yields a path $A_i - X - \ldots - A_j$ such that all intermediate nodes are eliminated before A_i, a contradiction.

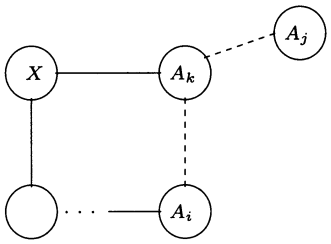

FIGURE 5.32. A path connecting A_i and A_j through nodes eliminated before A_i.

Next, assume that we have a path $A_i - X - \ldots - A_j$ such that all intermediate nodes are eliminated before A_i. Let A_k be any node on the path to be eliminated, and let Y and Z be the neighbors on the path. After the elimination of A_k, there is a link (Y, Z), and there is still a path $A_i - X - \ldots - A_j$ such that all intermediate nodes are eliminated before A_i, so the property is invariant under elimination. When all the nodes before A_i are eliminated, the path must be the link (A_i, A_j). □

5.6.2 Triangulation of time-stamped models

Return to Exercise 2.20 and consider the model in Figure 2.40. In Figure 5.33, we have folded it out to three time slices.

As you have probably experienced when solving Exercise 2.20, your computer ran out of memory when you tried to compile the model folded out to four or five time slices. The reason is that the cliques become too large.

A conceptually simple way of considering propagation in time-stamped models is that information is transmitted from one time slice to the next (if the task is forecasting) or to the previous time slice (if the task is to find out what happened in the past). In other words, probability potentials describing time slice i are transmitted from time slice i to time slice $i + 1$ or to time slice $i - 1$.

Let us consider forward passing from time slice i to time slice $i + 1$, and let W be the set of variables with a child in slice $i + 1$. We wish to

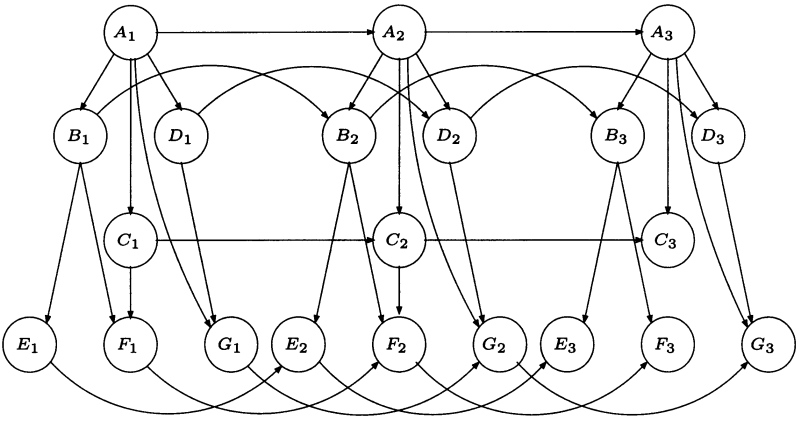

FIGURE 5.33. Three time slices of the model in Figure 2.40.

pass potentials representing the joint probability of W. For the model in Figure 5.33, we pass the information from slice 1 to slice 2 by eliminating all nodes in slice 1 before any node from slice 2 is eliminated. Now, consider any pair of nodes (X_2, Y_2). If there is a path in slice 1 connecting them, then they will be linked after the elimination of slice 1 (Proposition 5.6). Because the moral graph for slice 1 is connected, and all nodes in slice 2 have a parent in slice 1, the entire slice 2 will be a subset of a clique if slice 1 is eliminated before any node from slice 2. If you only process two time slices, you may avoid this clique explosion by using another elimination sequence. However, it will inevitably arrive when you extend the number of time slices to process. Some cliques will contain all variables with a child in the next slice or will contain all variables with a parent in the previous slice.

This situation is not reserved for models with connected time slices. Consider the model in Figure 5.34. If the model is folded out to four time slices, and the first three slices are eliminated before any node from slice four, then slice four becomes a complete set. Figure 5.35 shows the moral graph for four slices of the model. The reader can check that all pairs of nodes in slice four have a connecting path through the past slices.

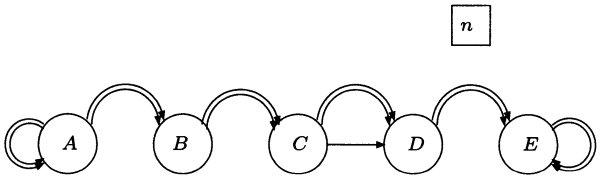

FIGURE 5.34. A time-stamped model with very sparse connection inside the slices.

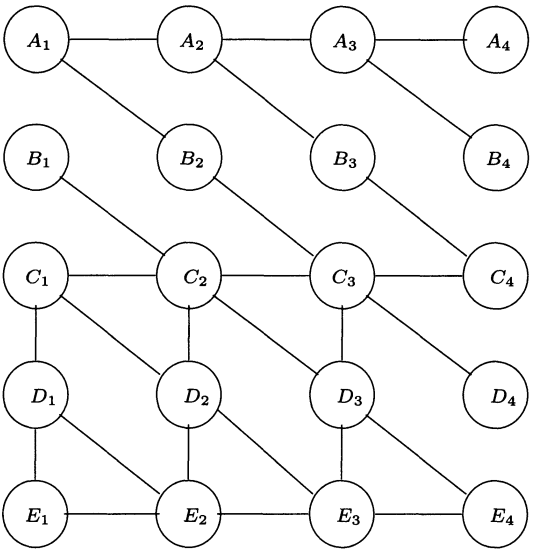

FIGURE 5.35. The moral graph for four time slices of the model in Figure 5.34.

5.7 Stochastic Simulation

The propagation method can require tables for the cliques in the triangulated graph. These cliques may be very large, and it happens that the space requirements cannot be met by the hardware available. In that case, an approximate method would be satisfactory.

In this section, we give a flavor of an approximate method called *stochastic simulation.* The idea behind the simulation is that the causal model is used to simulate the flow of impact. When impact from a set of variables to a variable A is simulated, a random generator is used to decide the state of A.

To illustrate the technique, consider the Bayesian network in Figure 5.36 with the conditional probabilities specified in Table 5.1. The idea is now to draw a random configuration of the variables (A, B, C, D, E) and to do this a sufficient number of times.

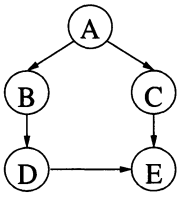

FIGURE 5.36. An example network. All variables have the states y and n.

	A	
B	y	n
y	0.3	0.8
n	0.7	0.2

$P(B \mid A)$

	A	
C	y	n
y	0.7	0.4
n	0.3	0.6

$P(C \mid A)$

	B	
D	y	n
y	0.5	0.1
n	0.5	0.9

$P(D \mid B)$

	C	
D	y	n
y	(0.9, 0.1)	(0.999, 0.001)
n	(0.999, 0.001)	(0.999, 0.001)

$P(E \mid C, D)$

TABLE 5.1. The conditional probabilities for the example network. $P(A) = (0.4, 0.6)$.

A random configuration is selected by successively sampling the states of the variables. First, the state of A is sampled. A random generator (with even distribution) is asked to give a real number between 0 and 1. If the number is less than 0.4, the state is y; otherwise the state is n. Assume that the result is y. From the conditional probability table $P(B \mid A)$, we have that $P(B \mid y) = (0.3, 0.7)$. The random generator is asked again, and if the number is less than 0.3, the state of B is y. This procedure is repeated to get the state of C, D, and E, and a configuration is determined.

The next configuration is sampled through the same procedure, and the procedure is repeated until m configurations are sampled. In Table 5.2, an example set of configurations is given. The probability distributions for the

	CDE							
AB	yyy	yyn	yny	ynn	nyy	nyn	nny	nnn
yy	4	0	5	0	1	0	2	0
yn	2	0	16	0	1	0	8	0
ny	9	1	10	0	14	0	16	0
nn	0	0	4	0	0	0	7	0

TABLE 5.2. A set of 100 configurations of (A, B, C, D, E) sampled from the network in Figure 5.36 and Table 5.1.

variables are calculated by counting in the sample set (see Exercise 5.27). For 39 of the samples in Table 5.2, the first state is y, and this gives an estimated probability $P(A) = (0.39, 0.61)$.

The preceding method, called *forward sampling*, does not require a triangulation of the network, and it is not necessary to store the sampled configurations (like Table 5.2). It is enough to store the counts for each variable.

Whenever a sampled configuration has been determined, the counts of all variables are updated, and the sample can be discarded. The method saves very much space, and each configuration is determined in time linear to the number of variables. The cost is accuracy and time.

So far, only the initial probabilities have been calculated. When evidence arrives, it can be handled by simply discarding the configurations that do not conform to it. In other words, a new series of stochastic simulations is started, and whenever a state of an observed variable is drawn, you stop simulating if the state drawn is not the one observed.

Unfortunately, this method has a serious drawback. Assume in the preceding example that the observations for the network are $B = n$ and $E = n$. The probability for $(B = n, E = n)$ is 0.00282. It means that in order to get 100 configurations, you should for this tiny example expect to perform more than 35,000 stochastic simulations.

Methods have been constructed for dealing with this problem. A widely used method is *Gibbs sampling*. In Gibbs sampling, you start with some configuration consistent with the evidence (for example determined by forward sampling), and then you change randomly the state of the variables in causal order. In one sweep through the variables, you determine a new configuration, and then you use this configuration for a new sweep, and so on.

In the example, let $B = n$ and $E = n$ be the evidence, and let the starting configuration be *ynyyn*. Now, calculate the probability of A given the other states of that configuration, that is, $P(A \mid B = n, C = y, D = y, E = n)$. From the network, we see that it is sufficient to calculate $P(A \mid B = n, C = y)$. It is easily done by Bayes' rule; it is (0.8, 0.2). We draw a number from the random generator, and let us assume that the number is 0.456, resulting in $A = y$. The next free variable is C. We calculate

$$\begin{aligned} P(C \mid A = y, B = n, D = y, E = n) &= P(C \mid A = y, D = y, E = n) \\ &= (0.996, 0.04). \end{aligned}$$

We draw from the random generator, and assume that we keep $C = y$.

In general, the calculation proceeds as follows. Let A be a variable in a Bayesian network BN, let $B_1, \ldots, B_n$ be the remaining variables, and let $b^* = (b_1, \ldots, b_n)$ be a configuration of $(B_1, \ldots, B_n)$. Then, $P(A, b^*)$ is the product of all potentials of BN with B_i instantiated to b_i. Therefore, $P(A, b^*)$ is proportional to the product of the potentials involving A, and $P(A \mid b^*)$ is the result of normalizing this product. Note that the calculation of $P(A \mid b^*)$ is a local task.

To return to the example, the next variable is D, and we follow the same procedure. Assume that the result is $D = y$. Then, the configuration from the first sweep is unaltered *ynyyn*.

The next sweep follows the same procedure. Assume the result for A is that the state is changed to n. Then, we calculate $P(C \mid A = n, D = y, E = n)$ and so forth.

In this way, a large sample of configurations consistent with the observations is produced. The question is whether the sample is representative of the probability distribution. It is not always so. It may be that the initial configuration is rather improbable, and therefore the first samples likewise are out of the mainstream. Therefore, you usually discard the first 5-10% of the samples. It is called *burn-in.*

Another problem is that you may be stuck in certain "areas" of the configurations. Perhaps there is a set of very likely configurations, but in order to reach them from the one you are in, a variable should change to a state that is highly improbable given the remaining configuration (see Exercise 5.28).

A third serious problem is that it may be very hard to find a starting configuration. In fact, it is *NP*-hard (see Exercise 5.29).

We will not deal with these problems but refer the interested reader to the literature.

5.8 Bibliographical Notes

A version of probability updating in singly connected DAGs through message passing was presented by Kim and Pearl (1983). In (Pearl 1986), conditioning was used to reduce propagation in multiply connected networks to propagation in singly connected networks. Shachter (1986) introduces arc reversal and uses it for a probability updating procedure in the bucket elimination style. Two versions of join tree propagation were presented in the late 1980s. Shafer and Shenoy (1990) proposed the method presented in this book. They did not exploit lazy evaluation but worked with multiplied potentials. Lauritzen and Spiegelhalter (1988) and Jensen et al. (1990) proposed what is now called the Hugin method. It also works with multiplied potentials, but the potentials in the cliques are changed dynamically. This, together with a division operation in the separators, reduced the calculation substantially for join trees with branching higher than three. A detailed study of the similarities and differences of the two methods is reported in (Shafer 1996). Lazy propagation (Madsen and Jensen 1999b) dissolves the difference between Shafer–Shenoy and Hugin propagation.

The concepts of triangulated graphs and join trees have been discovered and rediscovered with various names. In (Bertele and Brioschi 1972), they are used for dynamic programming, and Beeri et al. (1983) uses them for database management. A good reference on triangulated graphs is (Golumbic 1980).

Forward sampling was proposed by Henrion (1988). Gibbs sampling was originally introduced for image restoration by Geman and Geman (1984). Gilks et al. (1994) have developed a system, BUGS, for Gibbs sampling in Bayesian networks.

5.9 Exercises

Exercise 5.1 *BN* has the potentials in Table 5.3.

	A	
B	y	n
y	0.2	0.6
n	0.8	0.4

$P(B \mid A)$

	B	
C	y	n
y	0.3	0.2
n	0.7	0.8

$P(C \mid B)$

	C	
D	y	n
y	0.9	0.6
n	0.1	0.4

$P(D \mid C)$

TABLE 5.3. Potentials for Exercise 5.1. $P(A) = (0.2, 0.8)$.

(*i*) Calculate $P(A \mid D = y)$.
(*ii*) Calculate $P(C \mid D = y)$.

Exercise 5.2 *BN* has the potentials in Table 5.4.

	A	
B	y	n
y	0.2	0.6
n	0.8	0.4

$P(B \mid A)$

	B	
C	y	n
y	0.1	0.5
n	0.9	0.5

$P(C \mid B)$

	B	
D	y	n
y	0.7	0.4
n	0.3	0.6

$P(D \mid B)$

TABLE 5.4. Potentials for Exercise 5.2. $P(A) = (0.2, 0.8)$.

(*i*) Calculate $P(A \mid C = y, D = y)$.
(*ii*) Calculate $P(A \mid D = y)$.

Exercise 5.3 *BN* has the potentials in Table 5.5.
(*i*) Calculate $P(A \mid D = y), P(B \mid D = y), P(C \mid D = y)$.
(*ii*) Calculate $P(B \mid C = y)$.

Exercise 5.4 Consider the Bayesian network in Figure 5.37. All variables have three states.
(*i*) Calculate the size of the table $P(A, B, C, D, E, F, G = g_1, H = h_1)$.
(*ii*) In the calculation of $P(A \mid G = g_1, H = h_1)$, the variables have been marginalized in the following order: B, F, D, E, C. Calculate the size

	A	
B	y	n
y	0.2	0.6
n	0.8	0.4

$P(B \mid A)$

	A	
C	y	n
y	0.1	0.5
n	0.9	0.5

$P(C \mid A)$

	B	
C	y	n
y	(0.3, 0.7)	(0.2, 0.8)
n	(0.9, 0.1)	(0.6, 0.4)

$P(D \mid B, C)$

TABLE 5.5. Potentials for Exercise 5.3. $P(A) = (0.2, 0.8)$.

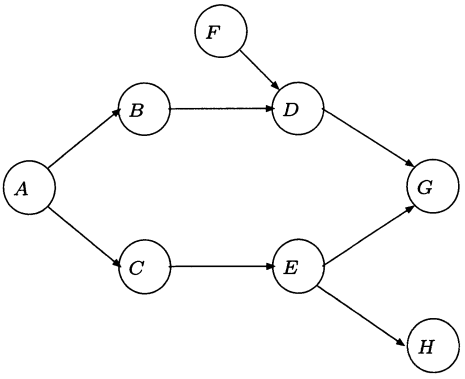

FIGURE 5.37. The network for Exercise 5.4.

of each table produced in the process, and compare the sum with the result of (*i*).

(*iii*) Determine an elimination order yielding a sum smaller than the one from (*ii*).

Exercise 5.5 We have the potentials $\phi_1(A_1, A_2, A_3)$, $\phi_2(A_2, A_3, A_5)$, $\phi_3(A_1, A_3, A_4)$, $\phi_4(A_5, A_6)$ over the universe $\{A_1, A_2, A_3, A_4, A_5, A_6\}$.

(*i*) Determine the domain graph.

(*ii*) Eliminate A_3.

(*iii*) Determine the domain graph for the resulting set of potentials.

Exercise 5.6 We have the potentials $\phi_1(A_1, A_2, A_3)$, $\phi_2(A_2, A_4, A_5)$, $\phi_3(A_4, A_6, A_7)$, $\phi_4(A_1, A_6, A_8)$ over the universe $\{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8\}$.

(*i*) Determine the domain graph.

(*ii*) Eliminate A_1.

(*iii*) Determine the domain graph for the resulting set of potentials.

Exercise 5.7 Consider the domain graph for the potentials in Exercise 5.5. Determine a perfect elimination sequence ending with A_1.

Exercise 5.8 Consider the domain graph for the potentials in Exercise 5.6. Does the graph have a perfect elimination sequence?

Exercise 5.9 Consider the Bayesian network in Figure 5.37.

(*i*) Determine the domain graph.

(*ii*) Does the domain graph have a perfect elimination sequence?

Exercise 5.10 Consider the graph in Figure 5.38.

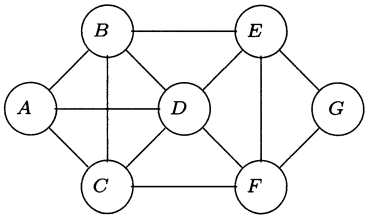

FIGURE 5.38. The graph for Exercise 5.10.

(*i*) Determine the simplicial nodes.

(*ii*) Is the graph triangulated?

Exercise 5.11 Consider the graph in Figure 5.39.

(*i*) Determine the simplicial nodes.

(*ii*) Is the graph triangulated?

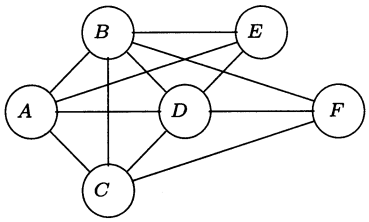

FIGURE 5.39. The graph for Exercise 5.11.

Exercise 5.12 Definition Let G be an undirected graph with node set U. A *path* in G is a sequence $A_1, \ldots, A_n$ of distinct nodes where A_i and A_{i+1} are neighbors. A *cycle* is a path where $A_1 = A_n$, and all other nodes are distinct. A *chord* in a cycle $A_1, \ldots, A_n$ is a link between two nodes A_i and A_j, where i and j are not consecutive numbers. G is *chord-saturated* if any cycle of length > 3 has a chord.

(*i*) Prove that any triangulated graph is chord-saturated. (Hint: Use induction and the fact that any cycle through a simplicial node must have a chord.)

(*ii*) Prove the following decomposition lemma. Let G be a noncomplete chord-saturated graph with at least three nodes and with node set U. Then, there is a complete subset S of U such that $G \setminus S$ is disconnected. (Hint: Let A and B be two nonadjacent nodes, and let S be a minimal set of nodes such that any path connecting A and B contains a node from S. Use chord saturation and minimality of S to prove that S is complete.)

(*iii*) Prove that any chord-saturated graph is triangulated. (Hint: Use (*ii*) to prove that any noncomplete chord-saturated graph with at least two nodes has at least two simplicial nodes.)

Exercise 5.13 Consider the domain graph from Exercise 5.5.

(*i*) Determine the cliques.

(*ii*) Construct a join tree for the graph.

Exercise 5.14 Consider the graph in Figure 5.39.

(*i*) Determine the cliques.

(*ii*) Construct a join tree for the graph.

Exercise 5.15 Consider the Bayesian network in Figure 5.40. Construct a join tree.

Exercise 5.16 A directed acyclic graph is *singly connected* if the graph you get by dropping the directions of the links is a tree (the graph in Figure 5.40 is singly connected).

(*i*) Prove that the moral graph of a singly connected graph is triangulated. (Hint: If you eliminate a node with exactly one parent and no children

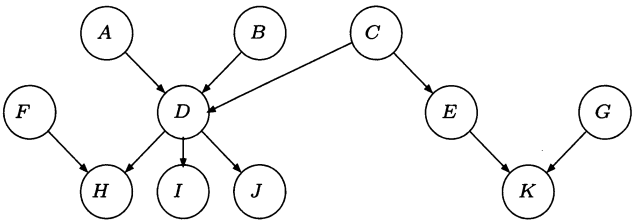

FIGURE 5.40. The Bayesian network for Exercise 5.15.

or with no parents and exactly one child, then the result is a moral graph for a singly connected graph.)

(*ii*) Prove that the separators in a join tree for a singly connected graph consist of exactly one node. (Hint: If the neighbors A and B share the neighbors C and D, then C and D are neighbors.)

Exercise 5.17 Consider the Bayesian network in Exercise 5.15.

Indicate the potentials to communicate in a full lazy propagation with evidence $F = f$, $I = i$, $E = e$.

Exercise 5.18 Consider the Bayesian network in Figure 5.41.

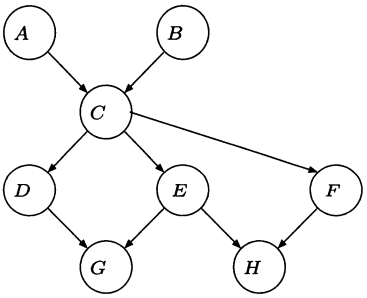

FIGURE 5.41. The Bayesian network for Exercise 5.18.

(*i*) Construct a junction tree.

(*ii*) Indicate the potentials to communicate in a full lazy propagation without evidence.

(*iii*) Indicate the potentials to communicate with evidence $D = d$ and $H = h$.

Exercise 5.19 Prove Proposition 5.5. (Hint: Assume a deadlock (no triggered nodes).)

Exercise 5.20 Show that any asynchronous full order of message passing corresponds to a CollectEvidence(R) followed by a DistributeEvi-

DENCE(R) for some node R. (Hint: Look at the first node that receives all its messages.)

Exercise 5.21 Triangulate the domain graph from Exercise 5.6.

Exercise 5.22

(i) Construct a junction tree for the Bayesian network in Figure 5.42 by using the elimination order F, J, B, A, I, K, E.

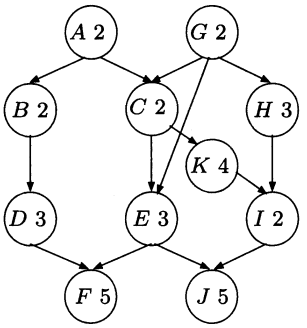

FIGURE 5.42. The Bayesian network for Exercise 5.22.

(ii) The numbers inside the nodes indicate the number of states. Use the heuristics from Section 5.6.1 to construct a junction tree.

Exercise 5.23 Consider the Bayesian network in Figure 5.29, and let the number of states be as listed in Section 5.6.1. Find a better triangulation than the one obtained by using the heuristics from Section 5.6.1.

Exercise 5.24 (Conditioning) Propagation methods for DAGs without multiple paths have existed for a long time. A propagation method for multiply connected DAGs consists in reducing a DAG to a set of singly connected DAGs.

(i) Consider the DAG (a) in Figure 5.43 with $P(A), P(B \mid A), P(C \mid A)$, and $P(D \mid B, C)$ given. Assume that $A = a$. Show that the DAG is reduced to the DAG (b) with $P(B, a), P(C, a)$, and $P(D \mid B, C)$ given. (Hint: Use the chain rule.) Calculate $P(B, a)$ and $P(C, a)$.

(ii) Show that $P(D, a) = \sum_{B,C} P(D \mid B, C)P(B, a)P(C, a)$.

(iii) Assume that for all states a of A we have a reduced DAG as in (i). Let evidence e be entered and propagated in all the reduced DAGs, yielding $P(B, a, e), P(C, a, e), P(D, a, e)$ for all a. Calculate $P(B, e)$ and $P(A, e)$.

This procedure is called *conditioning on* A.

(iv) Reduce the DAG by conditioning on B. Show that the tables are $P(A, b), P(C \mid A)$, and $P(D \mid C, b)$.

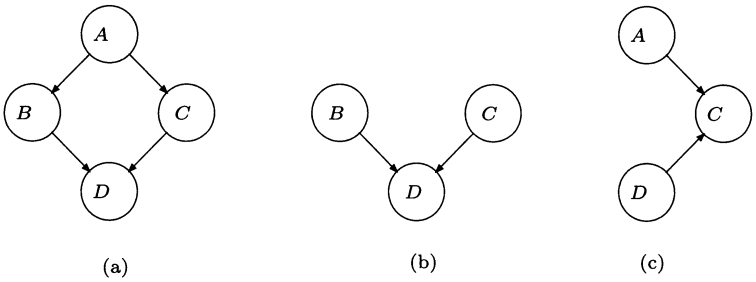

FIGURE 5.43. Figure for Exercise 5.24 (i)–(v).

(v) Show that conditioning on D does not result in a singly connected DAG.

Conditioning over several variables can be performed stepwise.

(vi) Determine a minimal set of conditioning variables for the DAG in Figure 5.44 to reduce it to singly connected DAGs.

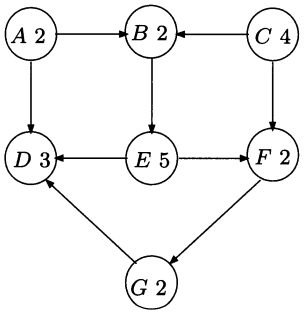

FIGURE 5.44. Figure for Exercise 5.24 (vi)–(vii).

(vii) The numbers attached to the variables indicate the number of states. Determine a conditioning resulting in a minimal number of singly connected DAGs.

Exercise 5.25 Let $\mathcal{C}$ be the set of cliques from a triangulated graph. A *pre-$\mathcal{J}$-tree* is a tree over $\mathcal{C}$ with separators $S = V \cap W$ for adjacent cliques V, W. The *weight* of a pre-$\mathcal{J}$-tree is the sum of the number of variables in the separators.

(i) Prove that a join tree is a pre-$\mathcal{J}$-tree of maximal weight.

(ii) Prove that any pre-$\mathcal{J}$-tree of maximal weight is a join tree.

Exercise 5.26

(i) Consider the graph in Figure 5.35. Determine a triangulation such that no clique contains more than four nodes.

(*ii*) Expand the model in Figure 5.34 to six time slices. Can this model be triangulated such that no clique contains more than four nodes?

Exercise 5.27 Calculate the marginals from the sample in Table 5.2 and compare the result with the exact marginals.

Exercise 5.28 The binary variables A and B are parents of the binary variable C. $P(A) = P(B) = (0.5, 0.5)$, and the conditional probability table is an exclusive *OR* table ($C = y$ if and only if exactly one of A and B is in the state y). Show that Gibbs sampling on this structure will give either $P(C = y) = 1$ or $P(C = n) = 1$.

Exercise 5.29 Given a Bayesian network over U with evidence e entered, show that it is *NP*-hard to find a configuration U^* such that $P(U^*, e) > 0$. (Hint: Look at Exercise 2.21.)

6
Bayesian Network Analysis Tools

The main reason for building a Bayesian network is to estimate the state of certain variables given some evidence. In Chapter 5, we gave methods that made it easy to access $P(A \mid e)$ for any variable A. However, this may not be sufficient. It may be crucial to establish the joint probability for a *set of variables*. Section 6.2 gives a general method for calculating $P(V \mid e)$ for any set V of variables.

Another typical request is to ask for the most probable configuration. We give a method for this in Section 6.3. Section 6.5 deals with methods for analyzing whether the evidence entered into the network is coherent – for example, to trace flawed data.

A very important tool for a decision support system is *explanation*: a tool to explain to the user how the system came to its conclusions. A part of explanation is *sensitivity to evidence*. How sensitive is the conclusion to (small) changes in the evidence? Which parts of the evidence are crucial and/or sufficient for the conclusion? This is the subject of Section 6.6.

Finally, we present methods for analyzing the sensitivity of posterior probabilities to changes in the numbers specified in the model.

The procedures in this chapter are based on lazy propagation as presented in Chapter 5, but most of them are also valid using other propagation methods. In lazy propagation, you work with sets of potentials representing the product. Often, you will perform the product of the union of two sets of potentials. We call this operation to "take the product of the two sets" and unless necessary for the exposition we do not bother whether this is done by taking the union of the two sets or by actually taking all potentials in the two sets and multiplying them together.

6.1 IEJ trees

Let e_X be a finding of the form "only the states $x'_1, \ldots, x'_q$ of the variable X are possible." If you know $P(X)$, then $P(e_X)$ is easy to calculate, namely as the sum of the probabilities for the states declared possible.

We will in several connections need $P(e)$ for a set of findings, e, and therefore we repeat Theorem 5.2 in condensed form.

Proposition 6.1 *Let BN be a Bayesian network, and let $e = \{e_1, \ldots, e_m\}$ be evidence. When e has been entered and a full propagation has been performed, then $P(e)$ can be calculated in the following way: take any separator S, multiply the two messages in the mail boxes (to get $P(S, e)$), and marginalize all variables out of the product.*

Proposition 6.1 can be used for more than calculation of probabilities of evidence. Assume that to some Bayesian network we have received evidence e, and we want to calculate the probability of the configuration $\underline{c} = (A = a, B = b, C = c)$ given e; that is, we want $P(\underline{c}|e)$. Proposition 6.1 yields $P(e)$. If we now enter $\underline{c}$ as further evidence and perform an extra propagation, then Proposition 6.1 yields $P(\underline{c}, e)$, and the fundamental rule gives

$$P(\underline{c}|e) = \frac{P(\underline{c}, e)}{P(e)}.$$

This technique can, for example, solve the question from Section 2.2.4 with the model in Figure 2.17: the sequence *baaca* is received; what is the probability that the transmitted word is **baaba**?

Sometimes, we may want to calculate $P(e')$ for various subsets $e' \subseteq e$. To do this, we can work with two copies of the junction tree. In the first copy, we have performed an initial propagation, and the appropriate messages are placed in the mailboxes. In the second copy, we have entered and propagated evidence, and the messages from this propagation are stored in the mailboxes. To be precise, we can work with junction trees where the separators have four mailboxes (see Figure 6.1). We call these junction trees *IEJ trees* (for Initial Evidence Junction).

The separator S in Figure 6.1 divides the evidence into two sets: the evidence e_V entered at the left of S and e_W entered at the right of S. From Proposition 6.1, we have that $P(S)$ is the product of Φ^V and Φ^W, and $P(S, e)$ is the product of Φ_e^V and Φ_e^W. Now, look at the pair (Φ^V, Φ_e^W). This pair is the pair of messages we would have had we entered e_W only. Therefore, the product must be $P(S, e_W)$, and we can easily calculate $P(e_W)$ as well as $P(S \mid e_W)$ and similarly for $P(S, e_V)$ and $P(S \mid e_V)$.

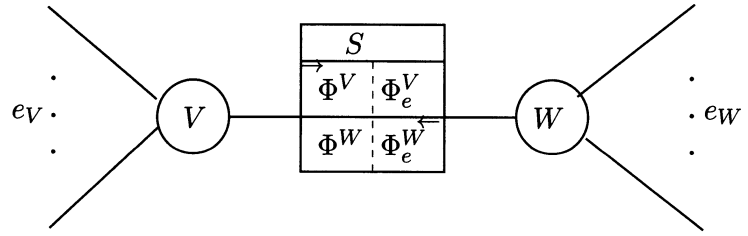

FIGURE 6.1. The separators in an IEJ tree contain four mailboxes, two for each direction. One of the mailboxes contains the message from the initial propagation, and the other contains the message from a propagation of evidence.

6.2 Joint Probabilities and A-Saturated Junction Trees

When dealing with utility functions over several variables and in various other connections, we have been faced with a request for the joint probability of several variables.

Take, for example, the stud farm example from Section 2.2 and the situation in Figure 2.15, and assume that the farmer must decide on a new mating among the horses Fred, Dorothy, Eric, and Gwenn. Which pair should be chosen to minimize the risk of getting a carrier as offspring?

If the set requested is a subset of a node in the junction tree, then you have the joint distribution directly. Otherwise, the technique from Section 6.1 can be used by entering and propagating all configurations, but it is troublesome.

A better technique is to perform propagation without eliminating variables from the requested set. This technique is called *variable propagation*.

Example Assume that we request $P(A, B, C, D, E)$ from the junction tree in Figure 6.2. Then, collect to (DEH) and in the operations do not marginalize A, B, and C. In Figure 6.3, the functions communicated in the operation are indicated. Note that the "sending" of functions does not mean that the functions are moved. What is sent is a pointer to a table for the function, and because variable propagation involves fewer marginalizations than normal propagation, it may be faster. However, when the incoming messages are finally multiplied, we must work with a considerably larger domain.

6.2.1 A-saturated junction trees

Sometimes, a variable A may be of particular interest. It may be a hypothesis variable, and you may be interested in investigating $P(A \mid X)$ for many different variables X. You may enter each state of X and propagate, but it requires one propagation for each state of each variable. Instead, you can

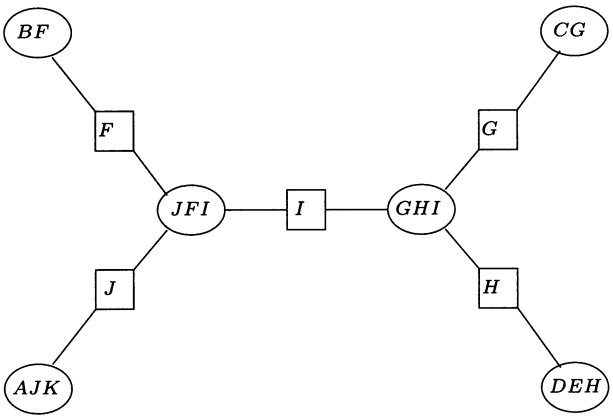

FIGURE 6.2. A junction tree from which we request $P(A, B, C, D, E)$.

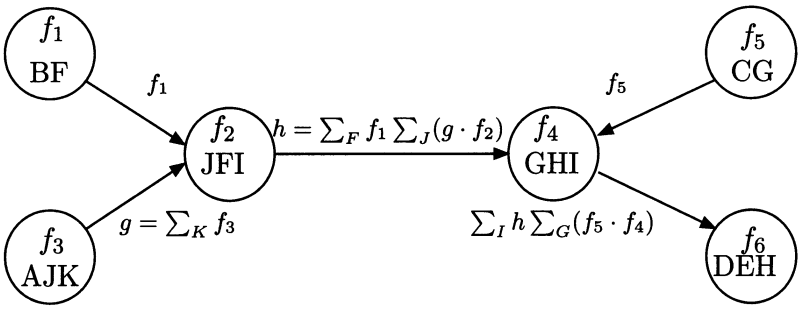

FIGURE 6.3. The messages passed when performing variable propagation for the calculation of $P(A, B, C, D, E)$. We assume that each clique holds one function (over its domain).

make A present in the entire junction tree: perform a full propagation but do not eliminate A. The result is called an *A-saturated junction tree*.

If W is a set of variables, we can do the same, and the result is a *W-saturated junction tree*A-saturated. The propagation in the preceding example is the COLLECTEVIDENCE part of establishing a (A, B, C, D, E)-saturated junction tree. Notice that the work requested for establishing a W-saturated junction tree does not exceed the work required for a normal propagation, but more space may be required.

Proposition 6.2 *Let T be a W-saturated junction tree with evidence e, and let X be any variable. Then, $P(W \mid X, e)$ is calculated through the following procedure*

1. *Choose any node V or separator S in T containing X.*
2. *$P(V \cup W, e)$ is the product of V's set of potentials with the incoming messages ($P(S \cup W, e)$ is the product of the two messages.)*
3. $P(W, X, e) = \sum_{V \setminus (W \cup \{X\})} P(V \cup W, e)$.
4. $P(X, e) = \sum_W P(W, X, e)$.
5. $P(W \mid X, e) = \frac{P(W,X,e)}{P(X,e)}$.

Note that in a W-saturated junction tree you can for each X get $P(W \mid X, e)$ through one local calculation. On the other hand, this local calculation is more complex than in the case of a normal junction tree. In the extreme, W may be very close to the universe, and the "local" calculation is extremely demanding.

We will deal later with W-saturated IEJ trees. They contain four messages in each separator, and they can be used for easy calculations of $P(W|e')$ for various subsets of the evidence.

6.3 Configuration of Maximal Probability

In the example in Section 2.2.4 concerning transmission of symbol strings, the immediate task is to find out which symbol string most probably has been transmitted. Using propagation of variables, the joint probabilities for all possible strings can be calculated, and thereby the most probable string can be found. This may require an intractably large table. There is, however, a much more efficient method.

Example Consider the small system consisting of the variables A, B, and C with the joint probability determined by the conditional probabilities specified in Table 6.1, and suppose that we want to find out which configuration of (A, B, C) has maximal probability.

	a_1	a_2
b_1	0.6	0.2
b_2	0.4	0.8

$P(B|A)$

	b_1	b_2
c_1	0.2	0.7
c_2	0.8	0.3

$P(C|B)$

TABLE 6.1. Probability tables for a small system, $P(A) = (0.4, 0.6)$.

Let us start calculating the probability α of the most probable configuration. α is the largest number in the joint probability table $P(A, B, C)$.

$$\begin{aligned}
\alpha &= \max_{A,B,C} P(A,B,C) = \max_{A,B,C} P(A)P(B|A)P(C|B) \\
&= \max_{A}\left(\max_{B}(\max_{C}(P(A)P(B|A)P(C|B)))\right) \\
&= \max_{A}\left(\max_{B}(P(A)P(B|A)\max_{C}P(C|B))\right) \\
&= \max_{A}\left(P(A)\max_{B}(P(B|A)\max_{C}P(C|B))\right).
\end{aligned}$$

In the preceding equations, we first used the chain rule for Bayesian networks and next the distributive law for the operation max. First, we determine $\max_C P(C|B)$. It is a distribution over B, and from Table 6.1 we get the distribution $(0.8, 0.7)$. Next, this distribution is multiplied by $P(B|A)$ (see Table 6.2).

$B \setminus A$	a_1	a_2
b_1	0.48	0.16
b_2	0.28	0.56

TABLE 6.2. $P(B|A)\max_C P(C|B)$.

When maximizing Table 6.2 over B, we get the distribution $(0.48, 0.56)$ over A. It is multiplied with the prior distribution $(0.4, 0.6)$, and we get $(0.192, 0.336)$. From this, we can conclude that the most probable configuration has probability 0.336, and the A component of it must be a_2.

To get the B component, return to Table 6.2. Because we know that $A = a_2$, we have the state of maximal value for B is b_2. Actually, when the value 0.56 in Table 6.2 is multiplied by the prior, 0.6, for a_2, we get the maximal value 0.336. In the same way, the C state is determined from $P(C \mid B)$ to c_1.

Let U be the universe for a Bayesian network. The general task of determining the configuration of maximal probability is to determine the X component for each $X \in U$. In fact, there may be several configurations

of maximal probability. We will leave this problem for a short while and assume that there is exactly one configuration of maximal probability, so the general task is for each variable X to get the distribution resulting from maximizing the remaining variables out. To help in the calculation, we have the following proposition.

Proposition 6.3 (*The distributive law for max*)

$$\max_Z f(X,Y)g(Y,Z) = f(X,Y)\max_Z g(Y,Z).$$

So the task is very similar to the task from Chapter 5, except that the operation is "max" instead of "$\sum$." Because the distributive law holds for max also, the propagation methods from Chapter 5 can be applied by substituting "max" for "$\sum$." This is called *max-propagation*, and accordingly we may use the term *sum-propagation* for the methods in Chapter 5. The result of maximizing variables out of a function f is called a *max-marginal* of f.

Theorem 6.1 *Let BN be a Bayesian network representing $P(U)$, and let T be a junction tree corresponding to BN. Let e be the evidence represented by the functions $\{\underline{e}_1, \ldots, \underline{e}_m\}$, and assume that the evidence functions are attached to appropriate nodes in the junction tree.*

After a full round of (lazy) max-propagation in T, we have

(i) *for each separator S, $\max_{U\setminus S} P(U,e)$ is the product of the two messages in S's mail boxes;*

(ii) *for each node V, $\max_{U\setminus V} P(U,e)$ is the product of the potential set attached to V and the incoming messages.*

Proof: Repeat the consideration from Chapter 5 with "max" instead of "$\sum$." Because the potential sets attached to the nodes in the junction tree are never changed, you can always change between max-propagation and sum-propagation. □

Several configurations of maximal probability

When there is exactly one configuration of maximal probability, then we can for each variable X read the component by taking the state of maximal probability in the max-marginalized distribution for X. However, if there are several configurations of maximal probability, then for some variables $\{Y_1, \ldots, Y_m\}$ there are several states of maximal probability in their max-marginalized distributions. Unfortunately, it does not hold that all combinations of these max-probable states form a configuration of maximal probability. If you request one of them, you can enter a max-probable state as evidence and perform a new max-propagation. If there are still several max-probable states in some of the remaining variables, you can repeat this operation until all variables have only one max-probable configuration.

6.4 Axioms for Propagation in Junction Trees

As shown in Section 6.3, the propagation algorithm can be used for tasks other than probability updating. Therefore, a general framework and a set of axioms for propagation in junction trees have been established. The framework and the axioms look very much like the properties listed in Section 1.3.5, and we will state them in a more general form here.

We have a set ϑ of valuations. Each $v \in \vartheta$ has a set dom $v \subseteq U$ attached. U is called the *universe*. Valuations can be *combined* through a binary operation $\otimes$, and for each $V \subseteq U$ there is a *projection operator* $v^{\downarrow V}$.

Axioms

(*i*) $\mathrm{dom}(\mathrm{v}_1 \otimes \mathrm{v}_2) \subseteq \mathrm{dom}(\mathrm{v}_1) \cup \mathrm{dom}(\mathrm{v}_2)$.

(*ii*) $\mathrm{dom}(\mathrm{v}^{\downarrow \mathrm{V}}) \subseteq \mathrm{V}$.

(*iii*) Combination is associative: $(v_1 \otimes v_2) \otimes v_3 = v_1 \otimes (v_2 \otimes v_3)$.

(*iv*) Combination is commutative: $v_1 \otimes v_2 = v_2 \otimes v_1$.

(*v*) $(v^{\downarrow V})^{\downarrow W} = v^{\downarrow V \cap W}$.

(*vi*) The distributive law: If dom $v_1 \subseteq V$, then $(v_1 \otimes v_2)^{\downarrow V} = v_1 \otimes (v_2)^{\downarrow V}$.

(*vii*) $v^{\downarrow \emptyset}$ is a neutral element with respect to combination, and it is denoted **1**.

The axiom (*vii*) is not needed, but it is customary to include it as an assumption.

With respect to probability updating in Bayesian networks, combination corresponds to multiplication, and projection corresponds to marginalizing out (see Section 1.3.5). Because the expression $v^{\downarrow V \cap W}$ is symmetric in V and W, axiom (*v*) includes the property that marginalization is commutative.

In the case of determining the most probable configuration, projection corresponds to maximizing out.

If you have a valuation framework satisfying the preceding axioms, you can calculate $(\bigotimes_i v_i)^{\downarrow X}$ for all $X \in U$ through junction tree propagation. We will not prove it here, but the interested reader may reread Chapter 5 and check that only the preceding axioms are used.

6.5 Data Conflict

A Bayesian network represents a closed world with a finite set of variables and causal relations. The causal relations are not universal but reflect relations under certain constraints. Take, for example, a diagnostic system

that on the basis of blood analysis monitors pregnancy. Only diseases and relations relevant for pregnant women are represented in the model, so if the blood originates from a man, the case is not covered by the model. It may happen that findings from male blood are impossible given the model. If so, the inconsistency is easy to detect: the probability of the evidence is 0. However, most often a set of findings is possible in the given model, and the system will not object to it. It will yield posterior probability distributions that may look rather harmless. The same also happens if test results are flawed. In a diagnostic situation, a single flawed test result may lead the investigation in a completely wrong direction (such flawed pieces of information are called *red herrings*).

6.5.1 Insemination

Consider the insemination example from Section 2.1.3, and assume that the farmer also has a scanning test. The model is given in Figure 6.4. To make things easy, assume that all tests have 2% false positives as well as false negatives (the prior for Pr is (0.87, 0.13)).

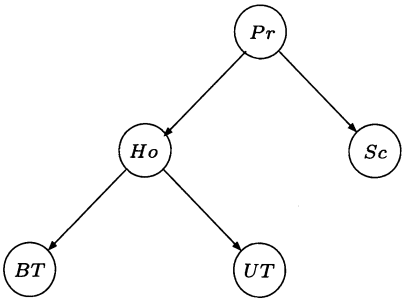

FIGURE 6.4. Insemination extended with a scanning test.

Assume that we get the evidence $UT = n$ and $Sc = n$ but $BT = y$. From our knowledge of the network model, we would say that the findings are in conflict. However, a propagation of the evidence does not disclose it. The posterior probabilities for Pr are (0.12, 0.88). Because the test results are consistent, we may be facing a rare case, but it may also be the case that the blood test is flawed or that the case is not covered by the model (a bull may have sneaked into the laboratory). We do not really have tools to distinguish between these situations, but it would be good to have a tool that gives the warning "it seems that the evidence is conflicting."

6.5.2 The conflict measure conf

Several approaches for analyzing data for conflicts have been made. In this section, we present a measure that only requires two propagations and gives

an indication of a possible conflict. The idea behind the measure is that correct findings originating from a coherent case covered by the model conform to certain expected patterns laid down in the model. In other words, the findings should be positively correlated. If $e = \{e_1, \ldots, e_m\}$ is a set of findings, we would expect $P(e)$ to exceed the probability for independent findings: $P(e_1) \cdot \ldots \cdot P(e_m)$. Hence, we define the *conflict measure* as

$$\text{conf}(\{e_1, \ldots, e_m\}) = \log_2 \frac{P(e_1) \cdot \ldots \cdot P(e_m)}{P(e)}.$$

The reason for the $\log_2$ is sheer convenience; some formulas look nicer. You can take any logarithm. A positive $\text{conf}(e)$ is an indicator of a possible conflict. For the insemination case, the conf value is 3.1.

To get the required probabilities, you start by performing a propagation without evidence entered. From this, you get $P(X)$ for all X in U. If e_i is a finding on X, then $P(e_i)$ can be calculated from $P(X)$ as explained in Section 6.1. To compute $P(e)$, you use Proposition 6.1.

6.5.3 *Conflict or rare case*

It may happen that typical data from a very rare case cause a high conf value. In the insemination case, a very rare blood type may have the effect of always causing BT to give a positive result (see Figure 6.5).

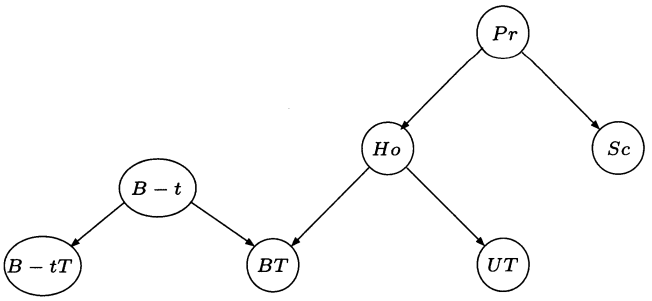

FIGURE 6.5. A rare blood type (frequency 0.001) causes BT to always give a positive test result. $B-tT$ is a test for blood type. $B-tT$ has 0.1% false positives and negatives.

For this extension of the model, we get $\text{conf}(UT = n, Sc = n, BT = y) = 3.1$. It is still indicating a possible conflict. The reason is that although the evidence is perfectly coherent, if the cow has this particular blood type, then it is very rare. Now, assume that the blood type test gives the result y. This resolves the conflict because the conf of the new set of evidence is -1.34.

The preceding problem calls for a method for pointing out whether a positive conf value may be explained as a rare case covered by the model.

Let $e = \{e_1, \ldots, e_m\}$ be findings for which $\text{conf}(e) > 0$, and let h be a hypothesis that could explain the findings: $\text{conf}(\{e_1, \ldots, e_m, h\}) \leq 0$.

We have

$$\begin{aligned} \text{conf}(\{e_1, \ldots, e_m, h\}) &= \log_2 \frac{P(e_1) \cdot \ldots \cdot P(e_m)P(h)}{P(e, h)} \\ &= \text{conf}(e) + \log_2 \frac{P(h)}{P(h \mid e)}. \end{aligned}$$

This means that if

$$\log_2 \frac{P(h \mid e)}{P(h)} \geq \text{conf}(e), \tag{6.1}$$

then h can explain away the conflict. In the blood example, the value of the left-hand side of (6.1) is 5.4, with $h =$ "$B - t = y$."

The fraction $\frac{P(h|e)}{P(h)}$ is used in various ways, and it is called the *normalized likelihood*. Note that by Bayes' rule

$$\frac{P(h \mid e)}{P(h)} = \frac{P(e \mid h)}{P(e)}.$$

Normalized likelihoods can be monitored automatically for all variables. Therefore, when analyzing for conflict/rare case, it is easy to detect whether a conflict may be due to a particular variable being in a very rare state.

6.5.4 *Tracing of conflicts*

After the conflict measure has been found positive, a further task would then be to find out whether a possible conflict is due to flawed findings and, if so, to trace them.

Let us return to the insemination problem with evidence $e = \{e_S =$ "$Sc = n$," $e_U =$ "$UT = n$," $e_B =$ "$BT = y$"$\}$. We have $\text{conf}(e) = 3.1$. We want to trace the origin of the conflict.

The evidence e is in the network communicated to Pr in two sets, $e' = \{e_B, e_U\}$ and $e'' = \{e_S\}$. A further investigation could therefore be to see whether e' contains an internal conflict. To do that, we need $P(e')$, which is 0.0196. We get

$$\text{conf}(e') = 3.16,$$

and not surprisingly a conflict is detected in e'.

Another possibility could be that the two sets e' and e'' are conflicting. We define

$$\text{conf}(e', e'') = \log_2 \frac{P(e')P(e'')}{P(e)} = -0.001,$$

which indicates that the two sets of findings are not conflicting, and we conclude that e'' is flawed.

To deal with tracing of conflicts, we can use IEJ trees. Using an IEJ tree, we can easily calculate the *local conflict* (see Figure 6.1)

$$\text{conf}(e_V, e_W) = \log_2 \frac{P(e_V)P(e_W)}{P(e)}.$$

The local conflict is a measure of whether the two sets of evidence e_V and e_W are in conflict.

We can also calculate the *partial conflicts* $\text{conf}(e_V)$ and $\text{conf}(e_W)$. The partial conflicts give an indication of possible internal conflicts in the sets e_V and e_W.

For each separator, we can get the probability of the evidence entered to the left and the probability of the evidence entered to the right. We can calculate the local and partial conflicts, and they are used for tracing the origin of the global conflict.

Using the IEJ tree in Figure 6.6, we calculate the following local and internal conflicts: $\text{conf}(e_B, e_U) = 3.16$, $\text{conf}(e_B, e_S) = 2.55$, $\text{conf}(e_S, e_U) = -1.93$, $\text{conf}(\{e_B, e_U\}, e_S) = -0.001$, $\text{conf}(\{e_B, e_S\}, e_U) = 0.615$, $\text{conf}(\{e_S, e_U\}, e_B) = 5.1$. These conflict measures point clearly at e_B as the dubious finding.

To round off this section, we give the following proposition, which relates the three kinds of conflicts.

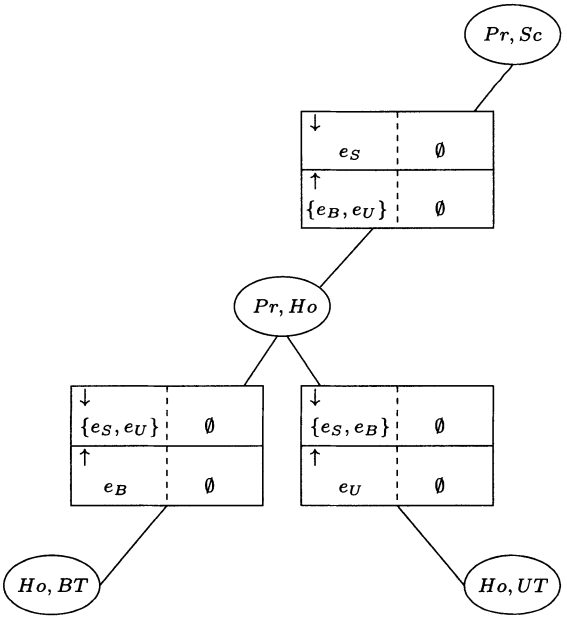

FIGURE 6.6. The IEJ tree for the insemination example. The various sets of evidence held are indicated in the mailboxes.

Proposition 6.4 *The global conflict is the sum of the local conflict and the partial conflicts.*

$$\text{conf}(e) = \text{conf}(e_V, e_W) + \text{conf}(e_V) + \text{conf}(e_W).$$

Proof: Exercise 6.4. □

6.5.5 Other approaches to conflict detection

The conf measure is not the only way of dealing with conflict detection. An approach to the problem would be to incorporate sources of surprise directly in the model. This can be done by entering variables modeling probabilities for malfunctioning of sensors and by extending causal variables such as disease variables with the state *other*. This approach, however, has the problem that it is difficult to model malfunctions or *other* unless the types of malfunction and *other* are known. Also, with *other* you can only handle discrepancies that are local in the network.

Another approach is to calculate a so-called *surprise index* for the set of findings. If the findings e are statements of the variables $A, \ldots, B$, the surprise index is the sum of probabilities for all configurations of $(A, \ldots, B)$ with a probability no higher than $P(e)$. If the surprise index is less than 0.1, this should be an indication of a possible conflict. In the insemination case, the surprise index is 0.06. Unfortunately, the calculation of a surprise index is exponential in the number of findings, and it must be considered intractable in general.

6.6 SE analysis

Evidence e has been entered into a network, and some hypotheses $h_1, \ldots, h_n$ are the focus of interest. Sensitivity analysis applied to evidence gives answers to questions such as:

- Which evidence is in favor of/against/irrelevant for h_i?
- Which evidence discriminates h_i from h_j?

We will in short denote this kind of analysis as *SE analysis*.

6.6.1 Example and definitions

The following example is used for illustration.

> In the morning, when Mr. Holmes leaves his house, he realizes that his grass is wet. He wonders whether it has rained during

the night or whether he has forgotten to turn off his sprinkler. He looks at the grass of his neighbors, Dr. Watson and Mrs. Gibbon. Both lawns are dry, and he concludes that he must have forgotten to turn off his sprinkler.

The network for Holmes' reasoning is shown in Figure 6.7, and the initial probabilities are given in Table 6.3.

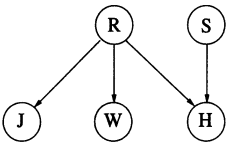

FIGURE 6.7. Network for the wet grass example. Holmes can inspect both Dr. Watson's and Mrs. Gibbon's grass.

	$R=y$	$R=n$
y	0.99	0.1
n	0.01	0.9

	$R=y$	$R=n$
$S=y$	(1, 0)	(0.9, 0.1)
$S=n$	(0.99, 0.01)	(0, 1)

$P(J \mid R) = P(W \mid R)$ $\qquad$ $P(H \mid R, S)$

TABLE 6.3. Tables for the wet grass example. $P(R) = (0.1, 0.9) = P(S)$.

The evidence e consists of the three findings e_H, e_W, e_J, and the hypothesis in focus is h_s : "$S = y$." We have $P(h_s) = 0.1$ and $P(h_s \mid e) = 0.9999$. We have $P(h_s \mid e_H) = 0.51, P(h_s \mid e_W) = 0.1 = P(h_s \mid e_J)$ [1], so neither e_W nor e_J alone have any impact on the hypothesis, but e_H is not sufficient for the conclusion. Therefore, the immediate conclusion that e_W and e_J are irrelevant for the hypothesis is not correct. We must conclude that evidence in combination may have a larger impact than the "sum" of the individual impacts.

To investigate further, we must consider the impacts of subsets of the evidence. We have

$$P(h_s \mid e_W, e_J) = 0.1, P(h_s \mid e_H, e_J) = 0.988 = P(h_s \mid e_W, e_H).$$

To relate the preceding probabilities to their impact on the hypothesis, we can divide them by the prior probability $P(h_s)$ to get the normalized likelihood.

[1] A d-separation analysis could yield some of the results; however, that is not the point here.

Other measures can be used, for example *Bayes factors*,

$$\frac{P(e \mid h)}{P(e \mid \neg h)},$$

or the fraction of achieved probability,

$$\frac{P(h \mid e')}{P(h \mid e)}.$$

The various normalized likelihoods are given in Table 6.4.

$W = n$	$J = n$	$H = y$	$\frac{P(e' \mid h_s)}{P(e)}$
1	1	1	9.999
1	1	0	1
1	0	1	9.88
1	0	0	1
0	1	1	9.88
0	1	0	1
0	0	1	5.1
0	0	0	1

TABLE 6.4. Normalized likelihoods for the subsets in the example. A "1" in the table indicates that the finding is an element of e'.

From Table 6.4, we can conclude that no single finding is sufficient for the conclusion. Also, although (e_W, e_J) alone has no impact on h_s, these two findings cannot both be removed. Moreover, we see that the subsets (e_H, e_J) and (e_H, e_W) can account for almost all the change in the probability for h_s.

Definitions Let e be evidence and h a hypothesis. Suppose that we want to investigate how sensitive the result $P(h \mid e)$ is to the particular set e.

We say that evidence $e' \subseteq e$ is *sufficient* if $P(h \mid e')$ is almost equal to $P(h \mid e')$. We then also say that $e \setminus e'$ is *redundant*. The term *almost equal* can be made precise by selecting a threshold θ_1 and requiring that $\left|\frac{P(h|e')}{P(h|e)} - 1\right| < \theta_1$. Note that $\frac{P(h|e')}{P(h|e)}$ is the fraction between the two likelihood ratios.

e' is *minimal sufficient* if it is sufficient, but no proper subset of e' is so.

e' is *crucial* if it is a subset of any sufficient set.

e' is *important* if the probability of h changes too much without it – to be more precise, if $\left|\frac{P(h|e\setminus e')}{P(h|e)} - 1\right| > \theta_2$, where θ_2 is some chosen threshold.

In the preceding example, put $\theta_2 = 0.2, \theta_1 = 0.05$. Then, (e_H, e_J) and (e_H, e_W) are minimal sufficient, (e_W, e_J) is important, and e_H is crucial.

In Holmes' universe, there is another possible hypothesis, namely h_r : "$R = y$." To find out which findings discriminate between the two hypotheses, an analysis of h_r can be performed. $P(h_r \mid e')$ is calculated for each subset of e', and the ratio between the two (normalized) likelihoods is used. The ratios are shown in Table 6.5.

$W = n$	$J = n$	$H = y$	$\frac{P(e' \mid h_s)}{P(e' \mid h_r)}$
1	1	1	6622
1	1	0	7300
1	0	1	74
1	0	0	81
0	1	1	74
0	1	0	81
0	0	1	0.92
0	0	0	1

TABLE 6.5. Likelihood ratios for the hypotheses h_s and h_r.

Table 6.5 shows that e_W and e_J are good discriminators between the two hypotheses.

As we have just illustrated, the heart of sensitivity analysis is the calculation of $P(h \mid e')$ for each $e' \subseteq e$. Since the number of subsets grows exponentially with the number of findings, the job may become very heavy, particularly when $P(h \mid e')$ must be calculated through a propagation in a large network.

Note that when $P(h \mid e')$ and $P(h)$ are available, then Bayes factors also can be calculated:

$$\frac{P(e' \mid h)}{P(e' \mid \neg h)} = \frac{P(h \mid e')P(\neg h)}{P(h)P(\neg h \mid e')} = \frac{P(h \mid e')(1 - P(h))}{P(h)(1 - P(h \mid e'))}.$$

6.6.2 *h-saturated junction trees and SE analysis*

A-saturated junction trees — sometimes extended to IEJ trees — can be of great help for SE analysis. If a particular state h of the hypothesis variable H is the focus of interest, another type of junction tree will suffice. Let e be the evidence. After propagating e, we insert $H = h$ in an appropriate node R and perform a DISTRIBUTEEVIDENCE from R. The messages from this propagation are stored in the separators also (see Figure 6.8). This type of junction tree we call an *h*-saturated junction tree.

The specific approach to SE analysis dependens greatly on the type of hypothesis, the type and size of the evidence, the topology of the network, and other factors, and in the following we only give some hints on how the tasks may be approached.

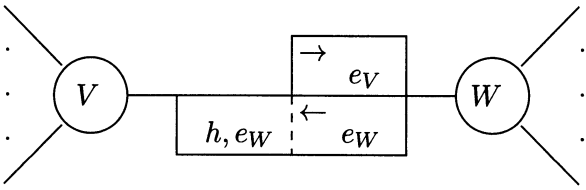

FIGURE 6.8. Part of an h-saturated junction tree where the hypothesis $H = h$ is entered to the right. The evidence handled is indicated.

What-if?

Assume that we want to investigate the impact on H if the finding e_X is removed or changed to e'_X.

Use an H-saturated junction tree. Go to the node V where e_X is placed, and you can perform the analysis through purely local calculations. The same H-saturated junction tree can be used for all findings. What-if? analysis can, for example, sort out redundant findings.

If you have a single state h in focus, you can use an h-saturated junction tree. Go to the node V where e_X is placed. Local to V you have messages for all evidence, and substituting e_X with e'_X (e'_X may be empty) will give you $P(e \setminus \{e_X\} \cup \{e'_X\})$. You also have messages involving e together with "$H = h$." Substituting e_X with e'_X will give you $P(e \setminus \{e_X\} \cup \{e'_X\}, h)$. From this, you get $P(H = h \mid e \setminus \{e_X\} \cup \{e'_X\})$. With that kind of what-if? analysis you can also determine the findings acting for or against h.

Note that this technique also allows you to investigate the effect of evidence on a variable for which you have not yet received evidence.

It is tempting to remove all redundant findings for further SE analysis. However, this would not be sound. You may remove one redundant finding, but by doing this another finding may switch from being redundant to being crucial. A finding that can safely be removed is called *irrelevant.* A finding is irrelevant if it has no impact on h in combination with any subset of e.

Crucial findings

Assume that $P(h \mid e)$ is high, and we want to determine the set of crucial findings.

Use an h-saturated junction tree. It may happen that some findings are evidence against h, but they are overwritten by the entire set. We assume that findings acting against h have been sorted out (for example, through what-if? analysis as before).

For the remaining evidence, we assume monotonity: no nonsufficient set contains a sufficient subset.

Then, e_X is crucial if and only if $e \setminus \{e_X\}$ is not sufficient. Using an h-saturated junction tree, it is easy to determine the crucial findings.

Minimal sufficient sets

It will be natural to continue the preceding procedure and repeatedly remove findings from sufficient sets. However, h-saturated junction trees only allow you to remove findings inserted in the same node. If they are not inserted in the same node, new propagations are required.

Using an h-saturated IEJ tree can speed up the search (that is, five mailboxes for each separator; see Figure 6.9). An h-saturated junction tree gives you access to $P(h \mid e')$ for a large family of subsets $e' \subset e$ (see Table 6.6). From this family, you choose the minimal sufficient subsets and continue the search for each of them by establishing a new h-saturated IEJ tree.

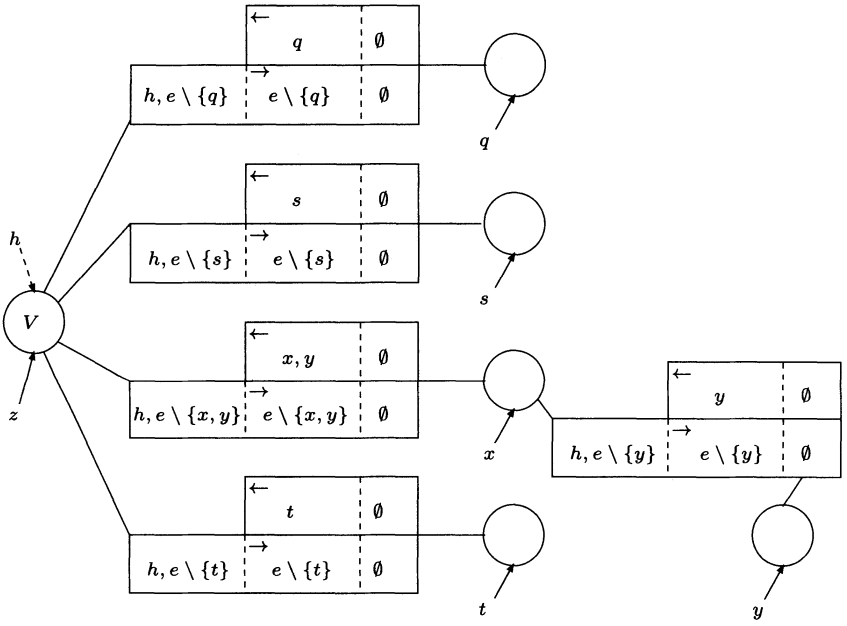

FIGURE 6.9. An h-saturated IEJ tree. The evidence "communicated" is indicated in the separators. It is assumed that h is inserted in V. The subsets of the evidence accessed are listed in Table 6.6.

As described in Section 6.2.1, the separators can be used to achieve $P(h \mid e')$ for the sets e' "communicated" to them. A similar procedure can be used for the nodes. Take, for example, the node V in Figure 6.8. By selecting appropriate messages from the neighbors, we can handle any union of sets communicated to a separator. This yields, for example, a way of calculating $P(h \mid q, t)$. A full list is given in Table 6.6. Note that some subsets are not in the list, such as $\{t, y\}$.

e	$\emptyset$	$\{t\}$	$e \setminus \{t\}$	$\{s\}$	$e \setminus \{s\}$
$\{q\}$	$e \setminus \{q\}$	$\{y\}$	$e \setminus \{y\}$	$\{x\}$	$e \setminus \{x\}$
$\{z\}$	$e \setminus \{z\}$	$\{x, y\}$	$e \setminus \{x, y\}$	$\{z, t\}$	$e \setminus \{z, t\}$
$\{z, s\}$	$e \setminus \{z, s\}$	$\{z, q\}$	$e \setminus \{z, q\}$	$\{t, s\}$	$e \setminus \{t, s\}$
$\{t, q\}$	$e \setminus \{t, q\}$	$\{s, q\}$	$e \setminus \{s, q\}$	$\{x, y\}$	$e \setminus \{x, y\}$
$\{z, x, y\}$	$\{t, s, q\}$	$\{x, y, q\}$	$\{z, t, s\}$	$\{x, y, s\}$	$\{z, t, q\}$
$\{x, y, t\}$	$\{z, s, q\}$				

TABLE 6.6. A list of sets of evidence e' for which the h-saturated IEJ tree in Figure 6.8 yields $P(h \mid e')$ through a local computation.

6.7 Sensitivity to Parameters

We have a Bayesian network BN with evidence e. Assume that we have a single hypothesis variable H, and let a particular state h of H be the focus of interest. Let $\mathbf{t}$ be a set of parameters for BN. We are interested in how $P(h \mid e)$ varies with $\mathbf{t}$. As mentioned in Section 3.4.2, the functional dependencies are rather simple.

Theorem 6.2 *Let BN be a Bayesian network over the universe U. Let t be a parameter and let e be evidence entered in BN. Then, assuming proportional scaling, we have*

$$P(e)(t) = \alpha t + \beta,$$

where α and β are real numbers.

Before proving Theorem 6.2, we need a lemma.

Lemma 6.1 Let $\mathrm{Pot}(V)$ be a potential over the variables V. Let $A \in V$ and let $\mathbf{v}^*$ be a configuration over $V \setminus \{A\}$. Let all entries be real-valued except for $\mathrm{Pot}(A, \mathbf{v}^*)$, which has the form $(\alpha_1 t + \beta_1, \ldots, \alpha_k t + \beta_k)$. Then,

$$\sum_V \mathrm{Pot}(V) = \alpha t + \beta,$$

where α and β are real numbers.

Proof: Let us first look at the example in Table 6.7.

		B		
		b_1	b_2	b_3
C	c_1	(1, 2, 3)	(2, 4, 7)	(4, 1, 2)
	c_2	(5, 2, 1)	$(t+1, -2t+2, 5t-2)$	(1, 1, 1)
	c_3	(2, 2, 1)	(3, 1, 4)	(2, 2, 2)

TABLE 6.7. $\mathrm{Pot}(A, B, C)$.

To calculate $\sum_V \text{Pot}(A, B, C)$, first take the sum of all numbers in the entries with $B \neq b_2$ and $C \neq c_2$. The result is 56. Then, add the expressions in the (b_2, c_2) entry, and you get $4t + 57$.

In general, let V^* be all configurations in V except for the $(A, \mathbf{v}^*)$ configurations. Then,

$$\sum_V \text{Pot}(V) = \sum_{V^*} \text{Pot}(V) + \sum_A \text{Pot}(A, \mathbf{v}^*).$$

The first term is a real number β^*, and the second is $(\alpha_1 t + \beta_1) + \ldots + (\alpha_k t + \beta_k)$. Hence,

$$\sum_V \text{Pot}(V) = \left(\sum_i \alpha_i\right) t + \left(\sum_i \beta_i\right) + \beta^*.$$

□

Proof: We hereby prove Theorem 6.2. Let $U = \{A\} \cup \{A_1, \ldots, A_n\}$. Put $fa(A) = \{A\} \cup pa(A)$ and let π be a parent configuration for which

$$P(A \mid \pi) = (t, \gamma_2(1-t), \ldots, \gamma_k(1-t)).$$

(We have without loss of generality assumed that the parameter t is attached to the first state of A.) Let the evidence potentials be $\underline{e}_1, \ldots, \underline{e}_m$. Now,

$$\begin{aligned} P(e) &= \sum_U P(U, e) = \sum_U P(A \mid pa(A)) \prod_i P(A_i \mid pa(A_i)) \prod_j \underline{e}_j \\ &= \sum_{fa(A)} P(A \mid pa(A)) \sum_{U \setminus fa(A)} \prod_i P(A_i \mid pa(A_i)) \prod_j \underline{e}_j. \end{aligned}$$

The factor $\sum_{U \setminus fa(A)} \prod_i P(A_i \mid pa(A_i)) \prod_j \underline{e}_j$ is a potential, $\text{Pot}(fa(A))$, with only real-numbered values, and we have

$$P(e) = \sum_{fa(A)} P(A \mid pa(A)) \text{Pot}(fa(A)).$$

The product $P(A \mid pa(A)) \text{Pot}(fa(A))$ is a potential satisfying the conditions in Lemma 6.1, and we can conclude that

$$P(e) = \alpha t + \beta.$$

□

Notation Let $\mathbf{t} = (t_1, \ldots, t_m)$ be a set of variables, and let $pol(\mathbf{t})$ be a polynomial over $\mathbf{t}$. $pol(\mathbf{t})$ is said to be *multilinear* if all exponents in the expression are of at most degree 1. If so, it has a term for each subset of $\mathbf{t}$.

Corollary 6.1 Let BN be a Bayesian network over the universe U. Let $\mathbf{t}$ be a set of parameters for different distributions, and let e be evidence entered into BN. Then, assuming proportional scaling, $P(e)(\mathbf{t})$ is a multilinear polynomial over $\mathbf{t}$.

Proof: For the sake of notational convenience, let $t = (x, y)$. From Theorem 6.2, we have

$$P(e)(x, y) = \alpha_x(y)x + \beta_x(y) = \alpha_y(x)y + \beta_y(x).$$

Inserting $x = 0$ yields

$$\beta_x(y) = \alpha_y(0)y + \beta_y(0); \tag{6.2}$$

that is, $\beta_x(y)$ is a linear function.

Inserting $x = 1$ yields

$$\alpha_x(y) + \beta_x(y) = \alpha_y(1)y + \beta_y(1).$$

Using (6.2), we get

$$\begin{aligned} \alpha_x(y) &= \alpha_y(1)y + \beta_y(1) - \alpha_y(0)y - \beta_y(0) \\ &= (\alpha_y(1) - \alpha_y(0))y + \beta_y(1) - \beta_y(0); \end{aligned} \tag{6.3}$$

that is, $\alpha_x(y)$ is a linear function. Combining (6.2) and (6.3), we get

$$P(e)(x, y) = ((\alpha_y(1) - \alpha_y(0))y + \beta_y(1) - \beta_y(0))x + \alpha_y(0)y + \beta_y(0),$$

which is of the form $\alpha xy + \beta x + \gamma y + \delta$.

If we have more than two parameters, we let $\mathbf{t} = (x, \mathbf{y})$, where $\mathbf{y}$ is a set of parameters. The earlier reasoning then yields that $\beta_x(\mathbf{y})$ and $\alpha_x(\mathbf{y})$ are multilinear polynomials over $\mathbf{y}$, and we repeat the arguments on $\beta_x(\mathbf{y})$ and $\alpha_x(\mathbf{y})$.

□

Corollary 6.2 Let BN be a Bayesian network over the universe U. Let $\mathbf{t}$ be a set of parameters for different distributions. Let a be a state of $A \in U$ and let e be evidence. Then, $P(a \mid e)(\mathbf{t})$ is a fraction of two multilinear polynomials over $\mathbf{t}$.

Proof: Corollary 6.1 and $P(a \mid e) = \frac{P(a,e)}{P(e)}$. □

6.7.1 One-way sensitivity analysis

Let $\mathbf{t}$ be parameters for BN and let e be evidence. Let $\mathbf{t}_0$ be the initial values of $\mathbf{t}$. In one-way sensitivity analysis, we wish to determine $P(e)$ as a function of each parameter s for all other parameters fixed to their initial values. Let $\mathbf{t}$ be modeled explicitly as described in Section 3.4.3, and let us for convenience assume that none of the priors for the parameter variables are extreme; that is, none of the initial values are 0 or 1.

Consider the junction tree with e propagated. Let S be the variable for parameter s. For the prior of S, we have $P(S = 1) = s_0$. After propagation, we have

$$P(S, e)(\mathbf{t}_0) = (x, y) = (P(S = 1, e), P(S = 0, e)).$$

We have

$$P(e)(s) = \alpha s + \beta,$$

and

$$\beta = P(e)(0) = P(e \mid S = 0) = \frac{P(S = 0, e)}{P(S = 0)} = \frac{y}{1 - S_o}.$$

Then,

$$\alpha = P(e)(1) - \beta = \frac{P(S = 1, e)}{P(S = 1)} - \beta = \frac{x}{s_o} - \beta.$$

To determine $P(h, e)$, you can perform a new propagation, but you may also establish an h-saturated junction tree as described in Section 6.6. This only requires half a new propagation, so through one and a half propagations (and easy local calculations), you can perform one-way sensitivity analysis for all parameters.

6.7.2 Two-way sensitivity analysis

Let s and u be two parameters. Then, $P(e)(s, r) = \alpha sr + \beta s + \gamma r + \delta$, and we wish to determine the coefficients. From the propagation described in Section 6.7.1, we have the value of $P(e)(s_0, r_0)$. By working locally in the node containing S, we get the value of $P(e)(0, r_0)$ and $P(e)(1, r_0)$, and by working locally in the node containing R, we get $P(e)(s_0, 0)$ and $P(e)(s_0, 1)$. In other words, we have five equations with four unknowns, and we can determine $(\alpha, \beta, \gamma, \delta)$ provided we can pick four equations with an invertible coefficient matrix. Unfortunately, the equations are of rank 3, and we need extra information. By entering a new value s_1 and propagating, we get sufficient information to locally compute all two-way sensitivity analyses involving s.

To calculate three-way sensitivity analyses is much more demanding, and the number of propagations grows exponentially with the number of parameters considered. The complexity of the local computations also increases exponentially. We will not treat this further.

6.8 Bibliographical Notes

Max-propagation was proposed by Dawid (1992). The axioms for propagation were formulated by Shafer and Shenoy (1990), and Lauritzen and Jensen (1997) extended them to cover Hugin propagation. A measure for calculating data conflict (surprise index) was first proposed by Habbema (1976). The method presented here is due to Jensen et al. (1991). The approach has been extended by Kim and Valtorta (1995). SE analysis is part of *explanation*, which was systematically studied by Suermondt (1992). The presentation here is an extension of (Jensen et al. 1995). Theorem 6.2 establishing the linearity of $P(e)(t)$ was independently proved by Castillo et al. (1997) and Coupé and van der Gaag (1998), and the method described here is based on (Kjærulff and van der Gaag 2000).

6.9 Exercises

Exercise 6.1 Construct the IEJ tree for the Bayesian network from Exercise 5.2 with evidence "$D = y$."

Exercise 6.2 Construct the IEJ tree for the Bayesian network from Exercise 5.3 with the evidence "$C = y$."

Exercise 6.3 ^{E} This exercise concerns the stud farm from Section 2.2 and the situation in Figure 2.15.

(*i*) The farmer must decide on a new mating among the horses Fred, Dorothy, Eric, and Gwenn. Which pair should be chosen to minimize the risk of getting a carrier as offspring?

(*ii*) What is the most probable configuration of genotypes of all horses? Does this correspond to the most probable genotype for each horse?

(*iii*) The prior frequencies λ_L and λ_K of the a-gene for the outside horses L and K are parameters. Determine intervals for both parameters for which Dorothy and Gwenn have a risk above 0.70 of being a carrier.

(*iv*) Assume that the farmer gets the evidence that Ann is pure, Brian is pure, and Cecily is a carrier. Perform a data conflict analysis.

(*v*) Assume that a horse is taken out if the probability of its being a carrier is above 0.60. The evidence "John=aa" is double checked and considered certain. Perform an SE analysis of the evidence from (*iv*) for the grandparents of John.

Exercise 6.4 ^{E} This exercise concerns the transmission of symbol strings from Section 2.2.4 and Exercise 2.10 (*i*).

(*i*) The sequence *baaca* is received. What is the most probable word transmitted?

(*ii*) Perform a data conflict analysis of the evidence.

(*iii*) Consider the parameters $t = P(T_4 = a \mid T_3 = a)$, $s = P(R_4 = c \mid T_4 = a)$, and $u = P(R_4 = c \mid T_4 = b)$. Perform an analysis of the sensitivity of the conclusion "the word transmitted is **baaba**." A one-way analysis could, for example, determine the minimal distance to a number where the conclusion changes.

(*iv*) The parameters s and u are common for all R-variables. Perform a sensitivity analysis as in (*iii*).

Exercise 6.5 ^{E} Consider the poker model from Exercise 2.11 (*ii*). Assume that you have seen your opponent change two cards first and then no cards. You have a flush. You know that your opponent sometimes changes no cards in the second round, no matter her hand. Let the frequency of this be t, and let your initial estimate be $t_0 = 0.001$. Analyze the sensitivity of the conclusion with respect to t and determine the value for which you have the best hand with probability 0.67.

Exercise 6.6 Prove Proposition 6.4.

7

Algorithms for Influence Diagrams

An influence diagram has three types of nodes, *chance nodes*, *decision nodes*, and *utility nodes*. The set of chance nodes is denoted U_C, the set of decision nodes is denoted U_D, and the set of utility nodes is denoted U_V. The *universe* is $U = U_C \cup U_D$. We also refer to the members of U as the variables of the influence diagram.

The decision nodes have a temporal order, $D_1, \ldots, D_n$, and the chance nodes are partitioned according to when they are observed: I_0 is the set of chance nodes observed prior to any decision, ... I_i is the set of chance nodes observed after D_i is taken and before the decision D_{i+1} is taken. I_n is the set of chance nodes never observed or observed too late to have an impact on any decision. In other words, we have a partial temporal ordering $I_0 < D_1 < I_1 < \ldots < D_n < I_n$. Note that an influence diagram is constructed so that if $A < D_i$, then there is a directed path from A to D_i.

We will in this chapter use the influence diagram DI in Figure 7.1 as a standard example. In order not to make things unnecessarily complicated, all variables in DI are binary. For DI, we have $I_0 = \emptyset, I_1 = \{T\}, I_2 = \{A, B, C\}$.

The graphical representation of influence diagrams as in Bayesian networks allows analysis of conditional independence. d-separation for influence diagrams is performed slightly differently from the way it is done for Bayesian networks. First, ignore the utility nodes. Also, the links into decision nodes are information links or precedence links, and they are ignored.

For the DI example, we can perform d-separation analysis on Figure 7.1. We get, for example, that C is d-separated from T given B (note that you need not condition on D_2 because the link from T to D_2 is ignored). Also,

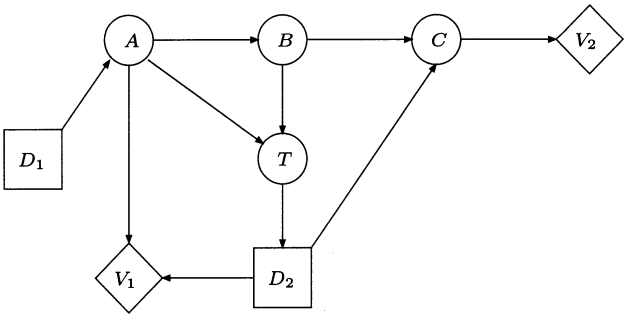

FIGURE 7.1. The example influence diagram, DI.

A and T are d-separated from D_2. This means that if I *perform an action* from D_2, then this action has no impact on T. Note that this is different from "if I am told what action from D_2 was performed, what can I infer on T?" For example, if I know that the performer maximizes expected utilities, I may be able to infer a lot on T.

Decision variables play a different role from chance variables. For chance variables, you ask whether information on node A may change your belief of node B. For decision variables, the question is whether an action from D will have consequences for the node B. Although the two concepts are different, they are in the case of influence diagrams not conflicting. In general, effects of decisions cannot "go back in time."

Proposition 7.1 *Let $A \in I_i$ and let D_j be a decision variable with $i < j$. Then,*

(i) *A and D_j are d-separated and hence*

$$P(A \mid D_j) = P(A).$$

(ii) *Let W be any set of variables prior to D_j in the temporal ordering. Then, A and D_j are d-separated given W and hence*

$$P(A \mid D_j, W) = P(A \mid W).$$

Proof:

(*i*) Because D_j has no parents, any impact from D_j must follow the direction of a link from D_j. The only way the impact can start going in the opposite direction of a link is if it meets a converging connection at a chance variable B, and then it can only do so if either B or one of its children C have received evidence. D_j is the only evidence, so this cannot happen. Hence, if D_j and A are not d-separated, there must be a directed path from D_j to A. Because $A < D_j$ in the temporal ordering, there is a directed path from A to D_j, and because the graph is acyclic, there cannot be a directed path from D_j to A.

(*ii*) We argue in the same way as for (*i*). By following directions of links from D_j, we can only start going opposite to the direction by meeting evidence. Because all evidence is prior in the temporal ordering, we know from (*i*) that we cannot meet it.

□

7.1 The Chain Rule for Influence Diagrams

For Bayesian networks, we have that $P(U)$ is the product of all probability potentials attached. For the influence diagram, we have a similar theorem. Again, decision variables act differently from chance variables. Because $P(D)$ for a decision variable under my control has no meaning, I do not give decision nodes prior probabilities. Also, it has no meaning to attach a probability distribution to a chance variable A effected by a decision variable D unless a decision has been taken and the action performed. Therefore, in Figure 7.1 it has no meaning to consider $P(A)$ or $P(A, D)$. What is meaningful is $P(A \mid d)$ for some $d \in D$, and we may bunch the probabilities for all decisions of D together in the expression $P(A \mid D)$.

Theorem 7.1 *Let ID be an influence diagram with universe $U = U_C \cup U_D$. Then,*

$$P(U_C \mid U_D) = \prod_{X \in U_C} P(X \mid pa(X)).$$

Proof: Let us first look at the influence diagram DI.

From the fundamental rule, we have

$$\begin{aligned} & P(C, T, B, A \mid D_1, D_2) \\ &= P(C \mid T, B, A, D_1, D_2) P(T, B, A \mid D_1, D_2) \\ &= P(C \mid T, B, A, D_1, D_2) P(T \mid B, A, D_1, D_2) \\ & \quad P(B \mid A, D_1, D_2) P(A \mid D_1, D_2). \end{aligned} \tag{7.1}$$

Because C is d-separated from A, T, and D_1 given B and D_2, we have

$$P(C \mid T, B, A, D_1, D_2) = P(C \mid B, D_2).$$

We also have

$$\begin{aligned} P(T \mid B, A, D_1, D_2) &= P(T \mid B, A) \\ P(B \mid A, D_1, D_2) &= P(B \mid A) \\ P(A \mid D_1, D_2) &= P(A \mid D_1). \end{aligned}$$

Substituting in (7.1) yields

$$P(C, B, T, A \mid D_1, D_2) = P(C \mid B, D_2) P(T \mid B, A) P(B \mid A) P(A \mid D_1,),$$

which is the product of the probability potentials for DI.

A general proof can follow another line of reasoning. Let $\underline{d}$ be a particular configuration of decisions. By inserting them into the influence diagram ID, you get a Bayesian network representing $P(U_C \mid \underline{d})$, the joint probability of U_C, under the condition that the decisions $\underline{d}$ are taken. Using the chain rule for Bayesian networks, you infer that $P(U_C \mid \underline{d})$ is the product of all probability potentials attached with the decision variables instantiated to $\underline{d}$. Because this holds for all instantiations of U_D, you get the result. □

7.2 Strategies and Expected Utilities

To solve an influence diagram, you may unfold it to a decision tree and solve it. In Figure 7.2, we have unfolded DI from Figure 7.1.

When solving the decision tree in Figure 7.2, we start in the leaves. Consider the path (d_1^1, t_1). We wish to compute the expected utility of performing action d_1^2 given (d_1^1, t_1). We have

$$EU(d_1^2 \mid d_1^1, t_1) = \sum_{A,C} P(A, C \mid d_1^1, t_1, d_1^2)(V_1(A, d_1^2) + V_2(C)).$$

For the action d_2^2, we have

$$EU(d_2^2 \mid d_1^1, t_1) = \sum_{A,C} P(A, C \mid d_1^1, t_1, d_2^2)(V_1(A, d_2^2) + V_2(C)).$$

Taken together, we write

$$EU(D_2 \mid d_1^1, t_1) = \sum_{A,C} P(A, C \mid d_1^1, t_1, D_2)(V_1(A, D_2) + V_2(C)).$$

We choose the action of maximal expected utility, and we get a decision rule for D_2 with $D_1 = d_1^1$ and $T = t_1$,

$$\sigma_2(d_1^1, t_1) = \text{argmax}_{D_2} EU(D_2 \mid d_1^1, t_1),$$

where "argmax_{D_2}" denotes "the action from D_2 maximizing." If there are several decisions yielding the maximum, any of them can do. The maximal expected utility from D_2 given (d_1^1, t_1) is

$$\rho_2(d_1^1, t_1) = \max_{D_2} \sum_{A,C} P(A, C \mid d_1^1, t_1, D_2)(V_1(A, D_2) + V_2(C)).$$

Generalizing these two formulas to any path over D_1, T, we get a policy for D_2

$$\begin{aligned} \sigma_2(D_1, T) &= \text{argmax}_{D_2} EU(D_2 \mid D_1, T) \\ &= \text{argmax}_{D_2} \sum_{A,C} P(A, C \mid D_1, T, D_2)(V_1(A, D_2) + V_2(C)), \end{aligned}$$

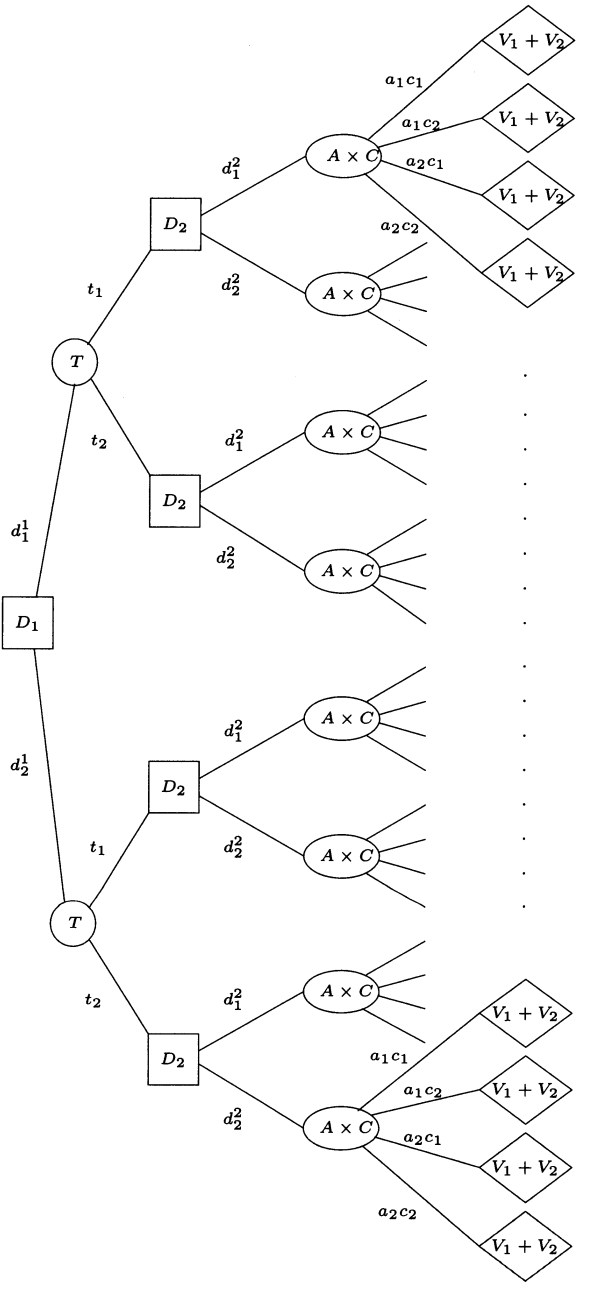

FIGURE 7.2. *DI* from Figure 7.1 unfolded into a decision tree. Note that the last chance node in all paths is the Cartesian product of A and B and that the utilities in the leaves are the sum of V_1 and V_2.

and a new utility function

$$\rho_2(D_1, T) = \max_{D_2} \sum_{A,C} P(A, C \mid D_1, T, D_2)(V_1(A, D_2) + V_2(C)). \quad (7.2)$$

$\rho_2(D_1, T)$ gives the expected utilities when we know the values of (D_1, T). The decision tree in Figure 7.2 can now be reduced to the one in Figure 7.3.

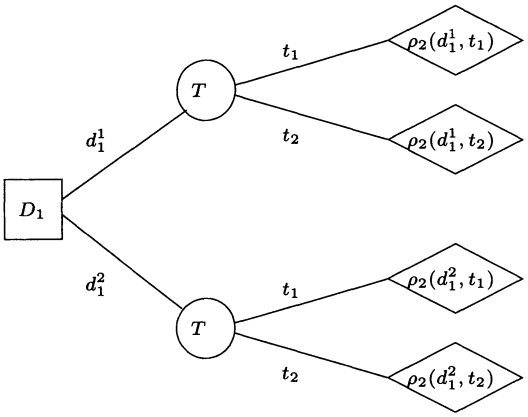

FIGURE 7.3. The decision tree from Figure 7.2 with D_2 replaced by a utility function reflecting that the policy σ_2 for D_2 is followed.

Next, look at the decision D_1 as in Figure 7.3. If we take the action d_1^1, we get the expected utility

$$EU(d_1^1) = P(t_1 \mid d_1^1)\rho_2(d_1^1, t_1) + P(t_2 \mid d_1^1)\rho_2(d_1^1, t_2),$$

which can also be written

$$EU(D_1) = \sum_T P(T \mid D_1)\rho_2(D_1, T).$$

The policy for D_1 is

$$\sigma_1 = \text{argmax}_{D_1} \sum_T P(T \mid D_1)\rho_2(D_1, T),$$

and the expected utility from performing optimal decisions is

$$\rho_1 = \max_{D_1} \sum_T P(T \mid D_1)\rho_2(D_1, T). \quad (7.3)$$

So far, we have written various formulas without really connecting them to potentials of the influence diagram. In principle, all probabilities in the formulas can be taken from the influence diagram by inserting evidence

and propagating, but let us take a closer look at (7.3) for ρ_1. Combining (7.2) and (7.3), we get

$$\begin{aligned}
\rho_1 &= \max_{D1} \sum_T P(T \mid D_1) \max_{D_2} \\
&\quad \sum_{A,C} P(A, C \mid D_1, T, D_2)(V_1(A, D_2) + V_2(C)) \\
&= \max_{D_1} \sum_T \max_{D_2} \\
&\quad \sum_{A,C} P(T \mid D_1) P(A, C \mid D_1, T, D_2)(V_1(A, D_2) + V_2(C)) \\
&= \max_{D_1} \sum_T \max_{D_2} \\
&\quad \sum_{A,C} P(T \mid D_1, D_2) P(A, C \mid D_1, T, D_2)(V_1(A, D_2) + V_2(C)) \\
&= \max_{D_1} \sum_T \max_{D_2} \\
&\quad \sum_{A,C} P(A, C, T \mid D_1, D_2)(V_1(A, D_2) + V_2(C)) \\
&= \max_{D_1} \sum_T \max_{D_2} \\
&\quad \sum_{A,B,C} P(A, B, C, T \mid D_1, D_2)(V_1(A, D_2) + V_2(C)) \\
&= \max_{D_1} \sum_T \max_{D_2} \\
&\quad \sum_{A,B,C} P(U_C \mid U_D)(V_1(A, D_2) + V_2(C)).
\end{aligned}$$

The formula for σ_1 is

$$\sigma_1 = \operatorname{argmax}_{D_1} \sum_T \max_{D_2} \sum_{A,B,C} P(U_C \mid U_D)(V_1(A, D_2) + V_2(C)).$$

For the policy σ_2, we have

$$\sigma_2(D_1, T) = \operatorname{argmax}_{D_2} \sum_{A,C} P(A, C \mid D_1, T, D_2)(V_1(A, D_2) + V_2(C)).$$

We can multiply inside "argmax_{D_2}" with anything not varying with D_2:

$$\begin{aligned}
&\sigma_2(D_1, T) \\
&= \text{argmax}_{D_2} P(T \mid D_1) \sum_{A,C} P(A, C \mid D_1, T, D_2)(V_1(A, D_2) + V_2(C)) \\
&= \text{argmax}_{D_2} \sum_{A,C} P(T \mid D_1, D_2) P(A, C \mid D_1, T, D_2)(V_1(A, D_2) + V_2(C)) \\
&= \text{argmax}_{D_2} \sum_{A,C} P(A, T, C \mid D_1, D_2)(V_1(A, D_2) + V_2(C)) \\
&= \text{argmax}_{D_2} \sum_{A,B,C} P(U_C \mid U_D)(V_1(A, D_2) + V_2(C)),
\end{aligned}$$

and the similarity with the formula for σ_1 is transparent. Similar calculations yield for ρ_2

$$\rho_2(D_1, T) = \frac{1}{P(T \mid D_1)} \text{max}_{D_2} \sum_{A,B,C} P(U_C \mid U_D)(V_1(A, D_2) + V_2(C)).$$

Definitions A *policy* for decision D_i is a mapping σ_i, which for any configuration of the past of D_i yields a decision for D_i; that is,

$$\sigma_i(I_0, D_1, \ldots, D_{i-1}, I_{i-1}) \in D_i.$$

A *strategy* for an influence diagram is a set of policies, one for each decision.

A *solution* to an influence diagram is a strategy maximizing the expected utility.

Theorem 7.2 *Let ID be an influence diagram over $U = U_C \cup U_D$ and $U_V = \{V_i\}$. Let the temporal order of the variables be described as $I_0 < D_1 < I_1 < \ldots < D_n < I_n$ and let $V = \sum_i V_i$. Then,*

(i) *an optimal policy for D_i is*

$$\begin{aligned}
&\sigma_i(I_0, D_1, \ldots, I_{i-1}) \\
&\quad = \text{argmax}_{D_i} \sum_{I_i} \text{max}_{D_{i+1}} \ldots \text{max}_{D_n} \sum_{I_n} P(U_C \mid U_D) V.
\end{aligned}$$

(ii) *the expected utility from following the policy σ_i (and acting optimally in the future) is*

$$\begin{aligned}
&\rho_i(I_0, D_1, \ldots, I_{i-1}) \\
&\quad = \frac{1}{P(I_0, \ldots, I_{i-1} \mid D_1, \ldots, D_{i-1})} \text{max}_{D_i} \\
&\quad \sum_{I_i} \text{max}_{D_{i+1}} \ldots \text{max}_{D_n} \sum_{I_n} P(U_C \mid U_D) V,
\end{aligned}$$

and the strategy for ID consisting of an optimal policy for each decision yields maximum expected utility

$$MEU(ID) = \sum_{I_0} \max_{D_1} \sum_{I_1} \max_{D_2} \ldots \max_{D_n} \sum_{I_n} P(U_C \mid U_D)V.$$

Proof: We start with the last decision D_n. We have for the expected utility given the past

$$\begin{aligned}
&EU(D_n \mid I_0, D_1, \ldots, D_{n-1}, I_{n-1}) \\
&\quad = \sum_{I_n} P(I_n \mid I_0, D_1, \ldots, D_{n-1}, I_{n-1}, D_n)V \\
&\quad = \sum_{I_n} \frac{1}{P(I_0, \ldots, I_{n-1} \mid D_1, \ldots, D_n)} P(I_n, I_0, \ldots, I_{n-1} \mid D_1, \ldots, D_n)V \\
&\quad = \frac{1}{P(I_0, \ldots, I_{n-1} \mid D_1, \ldots, D_{n-1})} \sum_{I_n} P(U_C \mid U_D)V.
\end{aligned}$$

In the last expression, we used $P(I_0, \ldots, I_{n-1} \mid D_1, \ldots, D_n) = P(I_0, \ldots, I_{n-1} \mid D_1, \ldots, D_{n-1})$. We get

$$\begin{aligned}
&\rho_n(I_0, D_1, \ldots, I_{n-1}) \\
&\quad = \frac{1}{P(I_0, \ldots, I_{i-1} \mid D_1, \ldots, D_{n-1})} \max_{D_n} \sum_{I_n} P(U_C \mid U_D)V,
\end{aligned}$$

and

$$\begin{aligned}
&\sigma_n(I_0, D_1, \ldots, I_{n-1}) \\
&\quad = \operatorname{argmax}_{D_n} \frac{1}{P(I_0, \ldots, I_{n-1} \mid D_1, \ldots, D_{n-1})} \sum_{I_n} P(U_C \mid U_D)V \\
&\quad = \operatorname{argmax}_{D_n} \sum_{I_n} P(U_C \mid U_D)V.
\end{aligned}$$

Next, assume that the theorem holds for $i+1, \ldots, n$ and consider decision D_i. We have

$$
\begin{aligned}
& EU(D_i \mid I_0, D_1, \ldots, D_{n-1}, I_{i-1}) \\
&= \sum_{I_i} P(I_i \mid I_0, D_1, \ldots, D_{i-1}, I_{i-1}, D_i) \rho_{i+1}(I_0, D_1, \ldots, I_i) \\
&= \sum_{I_i} \frac{1}{P(I_0, \ldots, I_{i-1} \mid D_1, \ldots, D_i)} P(I_i, I_0, \ldots, I_{i-1} \mid D_1, \ldots, D_i) \\
&\quad \frac{1}{P(I_0, \ldots, I_i \mid D_1, \ldots, D_i)} \max_{D_{i+1}} \\
&\quad \sum_{I_{i+1}} \ldots \max_{D_n} \sum_{I_n} P(U_C \mid U_D) V \\
&= \sum_{I_i} \frac{1}{P(I_0, \ldots, I_{i-1} \mid D_1, \ldots, D_i)} \max_{D_{i+1}} \\
&\quad \sum_{I_{i+1}} \ldots \max_{D_n} \sum_{I_n} P(U_C \mid U_D) V \\
&= \frac{1}{P(I_0, \ldots, I_{i-1} \mid D_1, \ldots, D_{i-1})} \sum_{I_i} \max_{D_{i+1}} \\
&\quad \sum_{I_{i+1}} \ldots \max_{D_n} \sum_{I_n} P(U_C \mid U_D) V,
\end{aligned}
$$

and we get the formulas in (i) and (ii).

As we repeatedly have determined a policy maximizing expected utility no matter the past, no other set of policies can give a higher expected utility. The formula for $MEU(ID)$ is the formula from (ii) for ρ_0. It is calculated by taking $\rho_1(D_1)$, multiplying by $P(I_0)$, and marginalizing I_0 out.

$$
\begin{aligned}
& MEU(ID) \\
&= \sum_{I_0} P(I_0) \rho_1(I_0) \\
&= \sum_{I_o} P(I_0) \frac{1}{P(I_0)} \max_{D_1} \sum_{I_1} \max_{D_2} \ldots \max_{D_n} \sum_{I_n} P(U_C \mid U_D) V \\
&= \sum_{I_0} \max_{D_1} \sum_{I_1} \max_{D_2} \ldots \max_{D_n} \sum_{I_n} P(U_C \mid U_D) V.
\end{aligned}
$$

□

Because $P(U_C \mid U_D)$ is the product of all probability potentials attached to ID, we have a method for calculating ρ_i as well as σ_i. The formulas allow you to start with the product of all probability potentials and then

marginalize out in reverse temporal order where chance variables are sum-marginalized and decision variables are max-marginalized. Each time an I_i is marginalized out, the result is used to determine a policy for D_i.

The method has the same problem as the method for Bayesian networks, namely that $P(U_C \mid U_D)$ may be an intractably large table, and we must look for methods that allow us to deal with smaller domains. We will deal with this problem in detail in Section 7.3.

7.2.1 *The example DI*

The influence diagram DI in Figure 7.1 has the potentials in Table 7.1.

$A \setminus D_1$	d_1^1	d_2^1
y	0.2	0.8
n	0.8	0.2

$P(A \mid D_1)$

$B \setminus A$	y	n
y	0.8	0.2
n	0.2	0.8

$P(B \mid A)$

$A \setminus B$	y	n
y	(0.9, 0.1)	(0.5, 0.5)
n	(0.5, 0.5)	(0.1, 0.9)

$P(T \mid A, B)$

$B \setminus D_2$	d_1^2	d_2^2
y	(0.9, 0.1)	(0.5, 0.5)
n	(0.5, 0.5)	(0.9, 0.1)

$P(C \mid D_2, B)$

$A \setminus D_2$	d_1^2	d_2^2
y	3	0
n	0	2

$V(A, D_2)$

$V_2(C) = (10, 0)$

TABLE 7.1. Potentials for DI.

Using Theorem 7.2 (running ID in a system processing influence diagrams), we get $\sigma_2(D_1, T)$ and $\rho_2(D_1, T)$ as listed in Table 7.2.

$T \setminus D_1$	d_1^1	d_2^1
y	d_1^2	d_1^2
n	d_2^2	d_2^2

$\sigma_2(D_1, T)$

$T \setminus D_1$	d_1^1	d_2^1
y	9.51	11.29
n	10.34	8.97

$\rho_1(D_1, T)$

TABLE 7.2. $\sigma_2(D_1, T)$ and $\rho_2(D_1, T)$ for DI.

Finally, we get $\sigma_1 = d_2^1$ and $MEU(DI) = 10.58$. Note that $\sigma_2(D_1, T)$ has the property that the state of T alone determines the decision to choose, and

we can remove D_1 from the domain of σ_2. This phenomenon can sometimes be determined from the d-separation properties of the influence diagram (see Figure 7.4), and we say that this part of the past is not *requisite*. For DI, it cannot be deduced from the structure; the potentials happened to cause it.

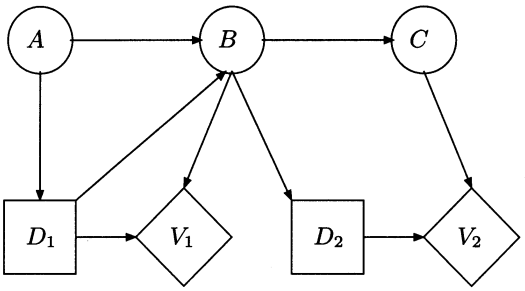

FIGURE 7.4. An influence diagram where D_1 is not requisite for D_2.

7.3 Variable Elimination

The method for solving influence diagrams has many similarities to the propagation algorithm for Bayesian networks: you start off with a set of potentials, and you eliminate one variable at a time. There are differences. First, the elimination order is constrained by the temporal order. Because max-marginalization and sum-marginalization do not commute, you must do it in an order where you first sum-marginalize I_n, then max-marginalize D_n, sum-marginalize I_{i-1}, and so on. This type of elimination order is called a *strong elimination order*. Furthermore, you deal with two types of potentials. Also, you only need to eliminate in one direction; this corresponds to COLLECTEVIDENCE.

We will first analyze the calculations when eliminating a variable. Let Φ be a set of probability potentials and Ψ a set of utility potentials. The two sets represent the expression $\prod \Phi(\sum \Psi)$, the product of all probability potentials multiplied by the sum of all utility potentials.

Now, assume that we calculate $\sum_X \prod \Phi(\sum \Psi)$ for some chance variable X. Divide Φ into two sets: Φ_X, which is the set of potentials with X in the domain, and $\Phi^* = \Phi \setminus \Phi_X$. The set Ψ is in the same way divided into the two sets Ψ_X and Ψ^*. Put $\phi_X = \sum_X \prod \Phi_X$ and $\psi_X = \sum_X \prod \Phi_X(\sum \Psi_X)$. Using the distributive law, we get

$$\begin{aligned}
\sum_X \prod \Phi \left(\sum \Psi \right) &= \prod \Phi^* \sum_X \left(\prod \Phi_X \left(\sum \Psi^* + \sum \Psi_X \right) \right) \\
&= \prod \Phi^* \left(\left(\sum \Psi^* \right) \sum_X \left(\prod \Phi_X \right) + \sum_X \prod \Phi_X \left(\sum \Psi_X \right) \right) \\
&= \prod \Phi^* \left(\left(\sum \Psi^* \right) \phi_X + \psi_X \right) \\
&= \prod \Phi^* \phi_X \left(\sum \Psi^* + \frac{\psi_X}{\phi_X} \right).
\end{aligned}$$

We see that the result of eliminating the chance variable X is that Φ_X is removed from the set of probability potentials and substituted with ϕ_X. For the set of utility potentials, Ψ_X is removed and $\frac{\psi_X}{\phi_X}$ is added.

Let D be a decision variable. We divide again the potentials into Φ_D, Φ^* and Ψ_D, Ψ^*. Because all variables coming after D in the temporal ordering have been eliminated when we are about to eliminate D, then $\prod \Phi_D$ does not vary with D (see Exercise 7.2). Therefore, taking $\max_D$ of $\prod \Phi_D$ is an almost empty operation; it only removes D from the domain. Using the distributive law for max, putting $\phi_D = \max_D \prod \Phi_D$ and $\psi_D = \max_D \prod \Phi_D (\sum \Psi_D)$, and exploiting that $\prod \Phi_D (\sum \Psi^*)$ does not vary with D, we get

$$\begin{aligned}
&\max_D \prod \Phi \left(\sum \Psi \right) \\
&= \prod \Phi^* \max_D \left(\prod \Phi_D \left(\sum \Psi^* + \sum \Psi_D \right) \right) \\
&= \prod \Phi^* \left(\max_D \prod \Phi_D \left(\sum \Psi^* \right) + \max_D \prod \Phi_D \left(\sum \Psi_D \right) \right) \\
&= \prod \Phi^* \left(\phi_D \left(\sum \Psi^* \right) + \psi_D \right) \\
&= \prod \Phi^* \phi_D \left(\sum \Psi^* + \frac{\psi_D}{\phi_D} \right).
\end{aligned}$$

The result is similar to the result for sum-elimination. We sum up.

Variable elimination for influence diagrams You work with two sets of potentials: Φ, the set of probability potentials; Ψ, the set of utility potentials. When a variable X is eliminated, the potential sets are modified in the following way

1.

$$\begin{aligned}
\Phi_X : &= \{\phi \in \Phi \mid X \in \mathrm{dom}\phi\} \\
\psi_X : &= \{\phi \in \Psi \mid X \in \mathrm{dom}\phi\}.
\end{aligned}$$

2. If X is a chance variable, then

$$\begin{aligned} \phi_X : &= \sum_X \prod \Phi_X \\ \psi_X : &= \sum_X \prod \Phi_X \left(\sum \Psi_X \right). \end{aligned}$$

If X is a decision variable, then

$$\begin{aligned} \phi_X : &= \max_X \prod \Phi_X \\ \psi_X : &= \max_X \prod \Phi_X \left(\sum \Psi_X \right). \end{aligned}$$

3.

$$\begin{aligned} \Phi : &= (\Phi \setminus \Phi_X) \cup \{\phi_X\} \\ \Psi : &= (\Psi \setminus \Psi_X) \cup \left\{ \frac{\psi_X}{\phi_X} \right\}. \end{aligned}$$

The influence diagram is solved by repeatedly eliminating variables in strong elimination order.

7.3.1 Strong junction trees

The considerations on triangulated graphs and junction trees can be applied when the preceding method is used for solving influence diagrams. The considerations will not be repeated here. Consider the influence diagram in Figure 7.5.

When solving the influence diagram, you first establish the moral graph: for each potential you link all variables in the domain. For the graph, it means that you remove precedence and information links, add a link for each pair with a common child (including a common utility node), and finally you remove the directions and the utility nodes. It is done in Figure 7.6 for the influence diagram DIF.

Unlike Bayesian networks, we cannot choose any elimination order for the triangulation. We must follow a strong elimination order: first eliminate I_n (in any order), then eliminate D_n, then I_{n-1}, and so on (if some I_i is empty, we may permute the elimination of D_{i+1} and D_i). The resulting triangulation is called a *strong triangulation.* Figure 7.7 shows the strong triangulation resulting from eliminating the moral graph of DIF through the strong elimination order $A, L, I, J, K, H, C, D, D_4, G, D_3, D_2, E, F, D_1, B$.

If you use the method for constructing the join trees from Section 5.3.1, a strong triangulation will give a strong junction tree with the last clique constructed in the strong elimination order as a strong root. A junction

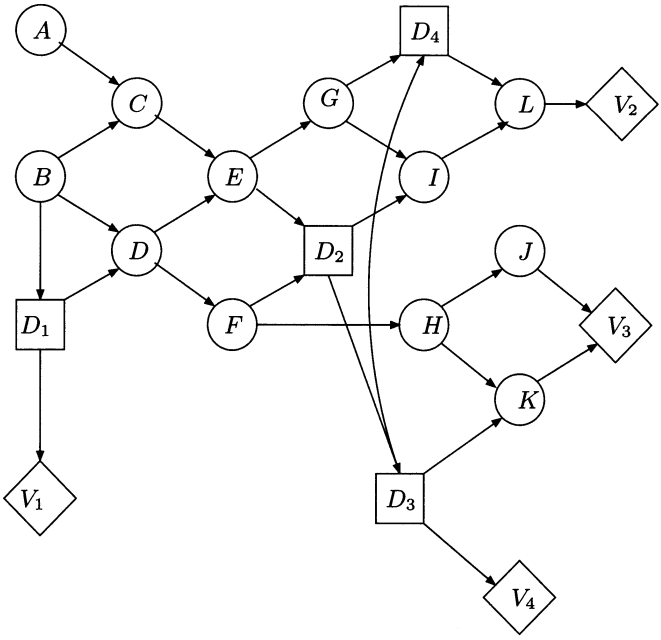

FIGURE 7.5. The influence diagram DIF from Figure 4.29.

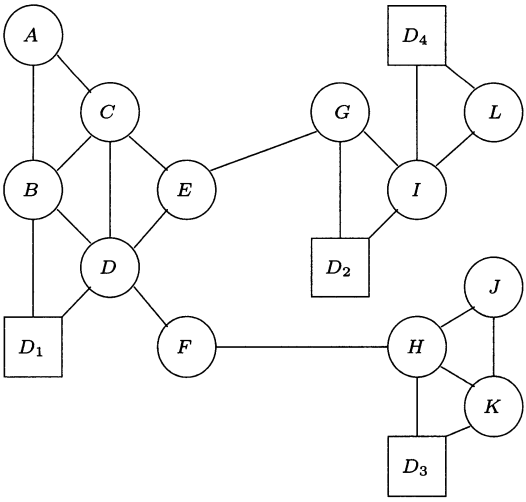

FIGURE 7.6. The moral graph for DIF.

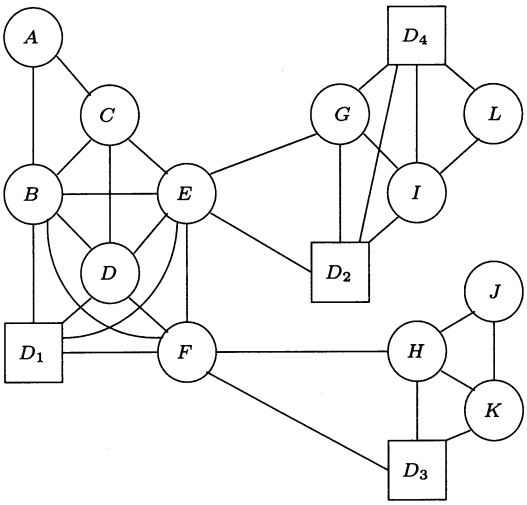

FIGURE 7.7. A strong triangulation of the graph in Figure 7.6.

tree with strong root R has the following property: for any two neighboring cliques C, C' with separator S and C' closest to R, it holds that the variables of S precede all variables of $C \setminus S$ in the temporal order. This property ensures that when CollectEvidence(R) is called, then whenever a variable is eliminated the appropriate potentials are present. Figure 7.8 shows a strong junction tree for the graph in Figure 7.7.

Note Although the influence diagram prescribes a specific order of the decisions, it happens that some decisions are independent such that the order may be altered without changing the strategy or the MEU. This is sometimes detected when constructing a strong junction tree. In other words, if you follow the method from Section 5.3.1, you may get a tree with decision nodes eliminated in two different branches (as is the case in Figure 7.8). From the strong junction tree, you can construct elimination sequences that do not meet the temporal constraints (the elimination sequence $J, K, H, D_3, A, C, L, I, D_4, G, D_2, D, F, D_1, B$ is a perfect elimination sequence ending with B, but it does not follow the temporal order). Because the result of CollectEvidence(R) is independent of the actual order of messages sent, all elimination sequences allowed by the strong junction tree give the same result (as long as the elimination order inside each clique obeys the temporal constraints). This means that the strong junction tree in Figure 7.8 discloses that D_3 and D_4 are independent, and the temporal order can be relaxed to a partial order of the decision nodes.

It may also happen that the strong junction tree does not allow for a strong elimination sequence when calling CollectEvidence(R). An example is given in Figure 7.9. However, it is no problem. All eliminations are correct.

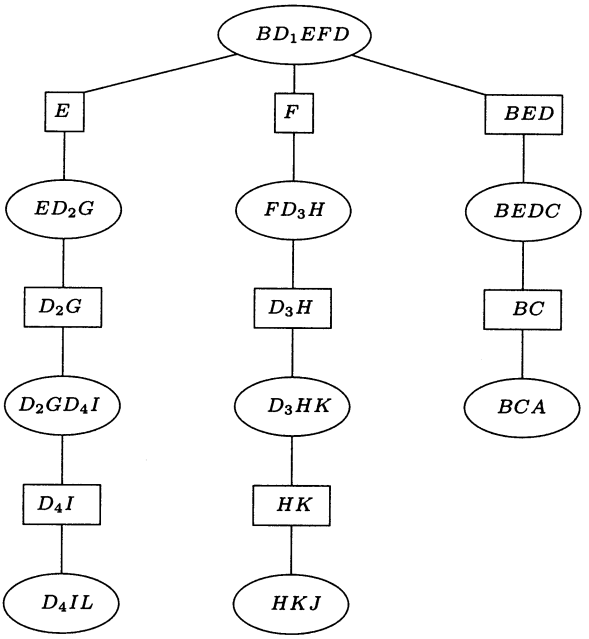

FIGURE 7.8. A strong junction tree for the graph in Figure 7.7.

7.3.2 Relevant past

As noted previously, the domain of a policy for a decision variable D_i is $(I_0, D_1, \ldots, I_{i-1})$. A strong elimination order can reveal reductions of the domain: whenever D_i is eliminated, you only consider the potentials with D_i in the domain. Therefore, the relevant past must be a subset of the union of these domains.

For the strong elimination for $DIF, A, L, I, J, K, H, C, D, D_4, G, D_3, D_2, E, F, D_1, B$, we get the following policies with domains: $\sigma_4(G, D_2), \sigma_3(F), \sigma_2(E), \sigma_1(B)$.

This analysis does not guarantee minimal policy domains, as can be seen from the influence diagram in Figure 7.10.

7.4 Policy Networks

When a strategy for an influence diagram has been determined, we have for each decision node D_i a policy σ_i. The domain of σ_i is $(I_0, D_1, \ldots, I_{i-1})$, but as explained in Section 7.3.2 we may be able to reduce it. Let $Rpa(D_i)$ denote the reduced past as determined by your method. Hence, we have a set of policies $\sigma_i(Rpa(D_i))$.

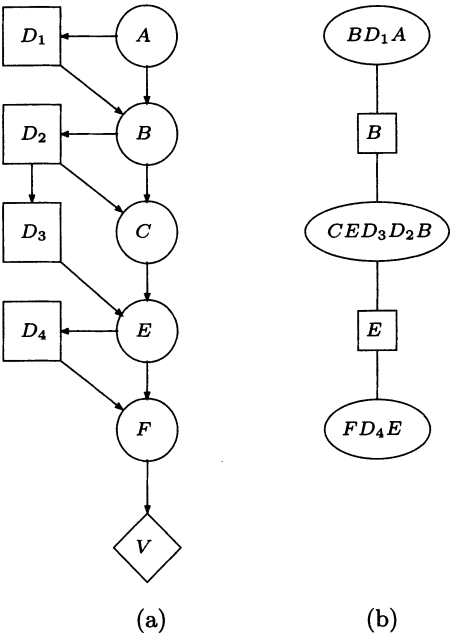

FIGURE 7.9. An influence diagram (a) with a strong junction tree (b) for which CollectEvidence(R) does not initiate a strong elimination sequence meeting the temporal constraints: C should be eliminated before D_4.

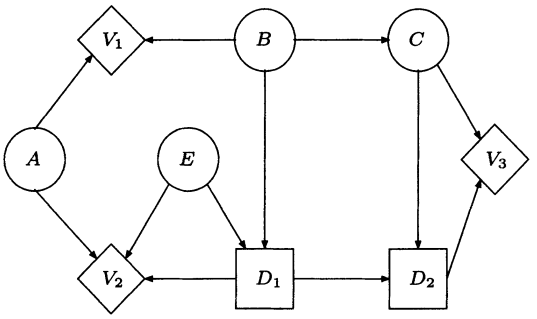

FIGURE 7.10. An influence diagram with policy $\sigma_1(E)$ sufficient for D_1. However, analysis through strong elimination sequences yields a policy $\sigma_1(E, B)$.

Definition Let ID be an influence diagram over $U = U_C \cup U_D$, and let $\sigma_1, \ldots, \sigma_n$ be a set of optimal policies. A *policy network* for ID (denoted ID^*) is a Bayesian network over $U = U_C \cup U_D^*$, where all decision variables D_i have been substituted with a chance variable D_i^*. The probability potentials from ID are kept. Furthermore, each D_i^* is given parents $Rpa(D_i)$ and $P(D_i^* \mid Rpa(D_i))$ is 1 for the chosen decision and 0 for the rest.

Figure 7.11 shows the policy network for the influence diagram DIF with policy domains determined in Section 7.3.2.

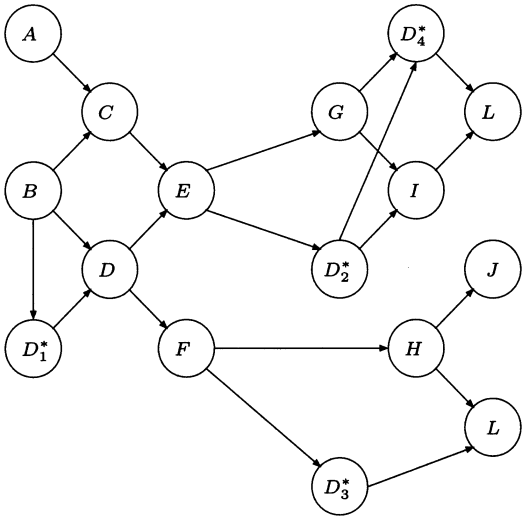

FIGURE 7.11. A policy network for DIF.

Example

A farmer has a wheat field. Twice during the season, he observes the state of the field and decides for a possible treatment with fungicides. Later, he observes the state of the field to decide booking of machinery for harvest. Figure 7.12 shows an influence diagram for his decision scenario.

To make prebooking of machinery and for booking plane tickets for summer vacations, he wishes to know what time of harvest he may eventually decide for.

Based on the influence diagram, an optimal strategy is determined, and the policy network is constructed (see Figure 7.13).

From the policy network, he can read the probabilities of his future decision on time of harvest. After the first observation and decision, he may enter this as evidence and get a new probability distribution for the optimal time of harvesting.

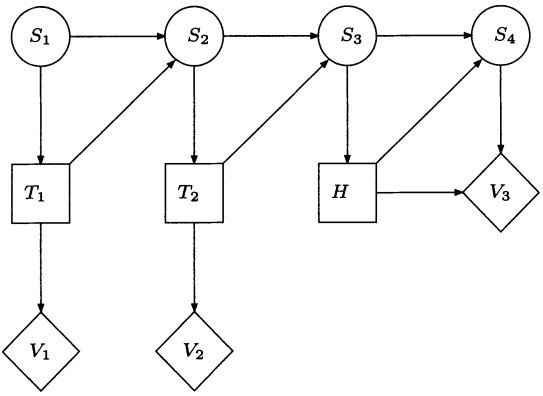

FIGURE 7.12. An influence diagram for treatment and time of harvest.

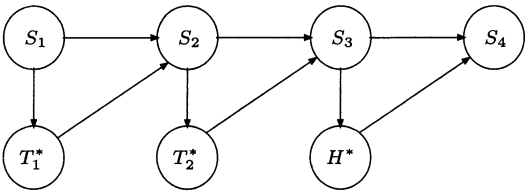

FIGURE 7.13. A policy network for the influence diagram in Figure 7.12.

Policy networks can be used in other ways. Assume that you know the farmer's influence diagram and observe some of his actions. Then, the policy network can give you estimates on what he may have observed or done in the past. Furthermore, policy networks can be used for analyzing the strategy proposed by the system: risk profile (what is the probability to lose $\$X$ or go bankrupt?), probability of success (winning at least $\$X$), variance of the expected utility, and so on.

7.5 Value of Information

As described earlier, influence diagrams require a fixed sequence of observations and decisions. Still, it may be worthwhile to analyze the value of various observations.

For the influence diagram in Figure 7.14, we have that the variable C is observed prior to D_3.

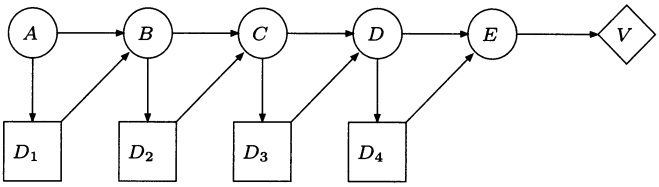

FIGURE 7.14. An influence diagram.

The observation may improve the decision D_3 and yield a higher expected utility. The observation has a cost, but because it does not affect the strategy, it is not part of the model. Assume that we wish to analyze how much the observation actually improves the expected utility. The situation where C is not observed is reflected in the influence diagram in Figure 7.15. If the difference in MEU between the two influence diagrams is smaller than the cost, then it does not pay to perform the test.

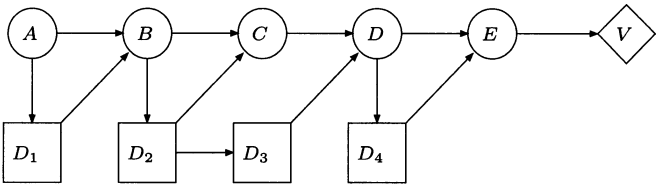

FIGURE 7.15. An influence diagram for the scenario from Figure 7.14 but with C not observed.

If we assume that the cost of observing is not dependent on the timing, the MEU cannot get higher by delaying an observation that must eventu-

ally be performed (see Exercise 7.7). Therefore, the only options we have are either to observe as soon as possible or never observe.

We can introduce a graphical notation indicating options for observations. The chance node C with the observation option is given a triangular parent T_C, and we give T_C a utility child node whose value is the cost of observing C (if the observation is costless, you cannot be better off by not performing the observation).

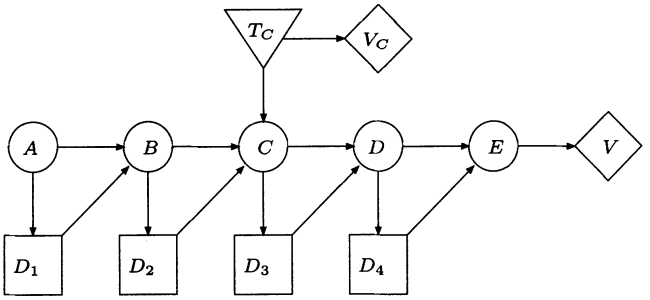

FIGURE 7.16. An influence diagram extended with a node indicating the option of not observing.

Using a method similar to propagation of variables as described in Section 6.2, the calculation of the various MEUs can be joined in one strong junction tree. Perform a strong triangulation for the influence diagram modeling that the observations have not been performed (that is, with the variables as members of I_n) and construct the strong junction tree. When solving the influence diagram corresponding to an observation of the chance node C just before deciding D_i, you use the same strong junction tree. However, you defer the elimination of C until D_i has been eliminated. Figure 7.17 gives an example of the influence diagram DIF, where an observation is optional for several variables. You may check that you can solve all influence diagrams corresponding to all combinations of possible observations through delayed elimination in the strong junction tree in Figure 7.8.

7.6 LIMIDs

The major complexity problem for influence diagrams is that the relevant past for a policy may be intractably large. A way of addressing this problem is to restrict memory. This restriction can be introduced in the form of history variables or information blocking, as described in Section 4.6.3. Another way is to pinpoint explicitly what is remembered when taking a decision. In other words, the no-forgetting assumption when interpreting an

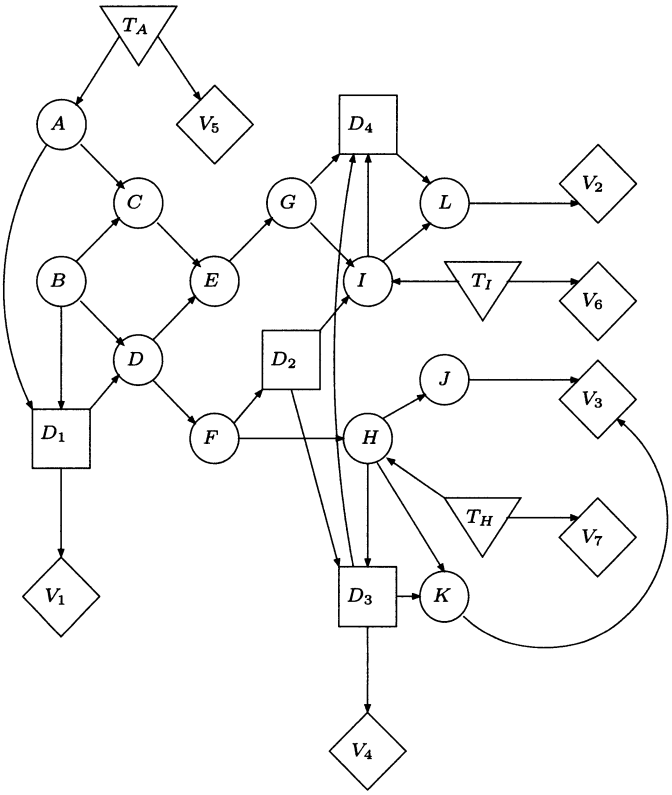

FIGURE 7.17. An influence diagram with the option of not observing A, H, and I.

influence diagram is dropped, and instead memory is represented directly by information links.

Assume that for the fishing example in Figure 4.30 we add the restriction that we (the EU politicians) only remember last year's decision, but we can recall the T observations up to two years back. This can be represented compactly as in Figure 7.18. Figure 7.18 is folded out in Figure 7.19.

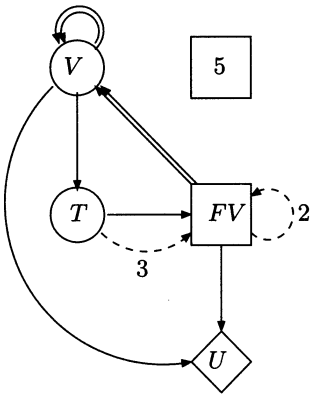

FIGURE 7.18. Figure 4.30 extended with special forgetting links indicating that the information is forgotten m time slices ahead. FV is only remembered in the next time slice. T will be remembered two time slices ahead.

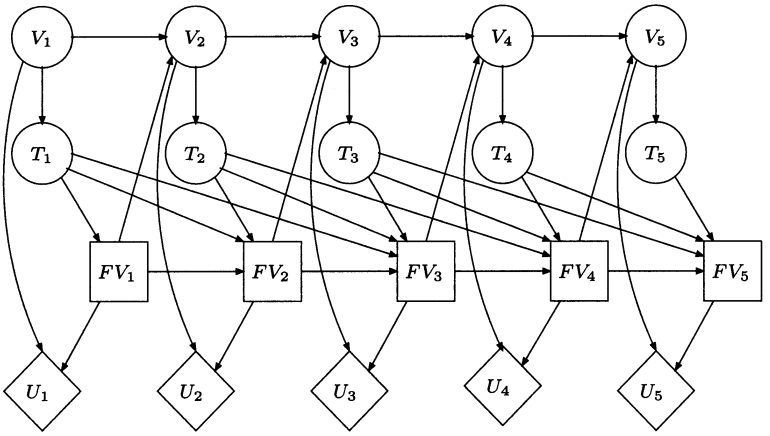

FIGURE 7.19. This spaghetti is a folded-out version of Figure 7.18.

An influence diagram with direct representation of memory is called a *limited memory influence diagram* (LIMID). To stress the difference, influence diagrams can be called *perfect recall influence diagrams.* The ad-

vantage of LIMIDs is that they allow you to work with decision policies with small domains. A solution to a LIMID is an approximation to a solution for the corresponding perfect recall influence diagram. The strong junction tree method automatically constructs cliques containing domains for perfect recall policies, and it is not well-suited for taking advantage of the space reduction offered by LIMIDs.

Instead, policy networks can be used. The specification of memory in a LIMID specifies the domains of the decision policies (we ignore the fact that some informational parents may turn up nonrequisite), and LIMID gives us the structure of the policy network (see Figure 7.20).

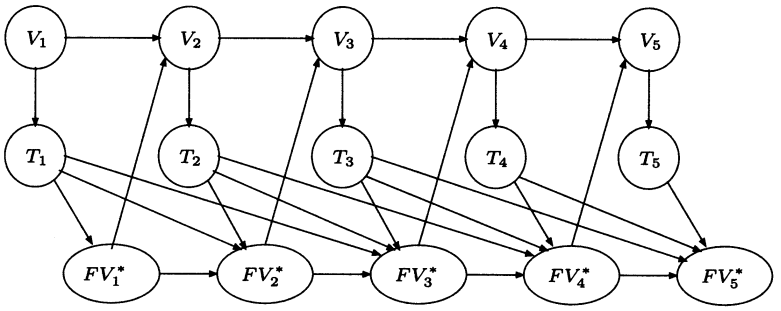

FIGURE 7.20. The policy network for the LIMID in Figure 7.19.

We attach a set of initial policies to the D^* variables. The initial policies need not be deterministic. Next, you change the policy network to a series of one-action networks and solve them as described in Section 4.1. It is natural to start with the last decision. The single-action network for the last decision in the fishing network is shown in Figure 7.21.

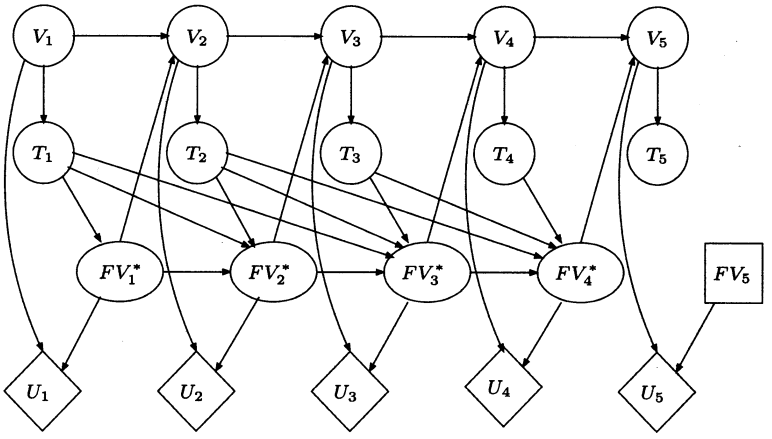

FIGURE 7.21. The single-action network for the last decision in Figure 7.20.

To establish an optimal policy for FV_5, you need $P(V_5 \mid FV_4^*, T_5, T_4, T_3)$. You can use any junction tree for the underlying Bayesian network and exploit variable propagation as described in Section 6.2. Notice that there are no constraints on the elimination order when triangulating.

Next, use the new policy for FV_5 in a policy network and construct the single-action network for FV_4. It is shown in Figure 7.22.

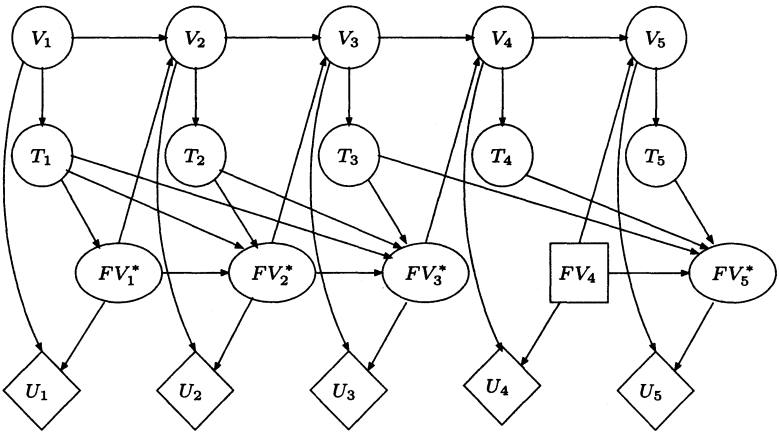

FIGURE 7.22. A single-action network for FV_4.

Through variable propagation, you establish a new policy for FV_4 by calculating $P(V_5 \mid FV_4, FV_3^*, T_4, T_3, T_2)$ and $P(V_4 \mid FV_3^*, T_4, T_3, T_2)$. Continue to FV_3 down to FV_1.

The initial policies for FV_1, FV_2, FV_3, and FV_4 were used when determining a new policy for FV_5. These initial policies had an impact on $P(V_5 \mid FV_4^*, T_5, T_4, T_3)$, and you must repeat the process based on the new policies; that is, the procedure is iterative. It can be shown that the procedure converges, and that it converges to an optimal strategy for the LIMID. However, this need not be an optimal strategy for the perfect recall influence diagram, and it is an issue of research to establish bounds on the distance between the LIMID optimal strategy and the perfect recall optimal strategy.

The repeated construction of single-action networks and variable propagation can be performed in a unified framework saving a large number of repetitions of the same calculation. We will not treat this further but refer the interested reader to the literature.

7.7 Bibliographical Notes

Various methods for solving influence diagrams have been constructed. Olmsted (1983) and Shachter (1986) used arc-reversal; Shenoy (1992), Cowell (1994), Jensen et al. (1994), and Ndilikilikesha (1994) use elimination and direct manipulation of potentials. The presentation here is based on (Madsen and Jensen 1999a). Cooper (1988) presents a method which works well for scenarios with one decision variable. It substitutes the decision variable and the utility variables with chance variables and uses Bayesian network propagation. Zhang (1998) exploits Cooper's method to full influence diagrams. Policy networks were introduced by Nilsson and Jensen (2000). Value of information for influence diagrams has been treated by Dittmer and Jensen (1997) and Shachter (2000). LIMIDs is proposed in (Nilsson and Lauritzen 2000).

7.8 Exercises

Exercise 7.1 Consider the influence diagram DI from Figure 7.1 but without the utility node V_1. Derive the formulas for an optimal strategy.

Exercise 7.2 Let ID be an influence diagram over $U_C \cup U_D$ and with temporal ordering $I_0, D_1, \ldots, D_n, I_n$. Put

$$\phi(U_C, U_D) = P(U_C \mid U_D),$$

$$\phi_i(I_0, D_1, \ldots, I_{i-1}, D_i) = \sum_{I_{i,\ldots,I_n}} \phi(U_C, U_D).$$

Show that

$$\phi_i(I_0, D_1, \ldots, I_{i-1}, d_i) = \phi_i(I_0, D_1, \ldots, I_{i-1}, d_i')$$

for all d_i, d_i' in D_i.

Exercise 7.3 Construct a strong junction tree for the influence diagram in Figure 4.28 and determine the domains of the policies.

Exercise 7.4 Construct strong junction trees for the influence diagrams in Figures 4.31 and 4.33. Compare the clique sizes and the domains of the policies.

Exercise 7.5 Show that any strong triangulation of the influence diagram in Figure 7.10 will place E and B in the clique where D_1 is eliminated.

Exercise 7.6 Construct a strong junction tree for the influence diagram in Figure 7.23.

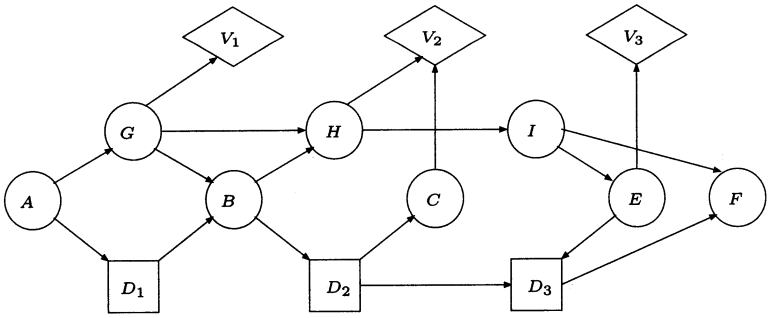

FIGURE 7.23. Figure for Exercise 7.6.

(*i*) Is D_2 requisite for D_3?

(*ii*) Is B requisite for D_3?

(*iii*) Construct a join tree for the policy network and compare the size with the size of the strong junction tree.

Exercise 7.7

(*i*) Let $\{a_{ij}\}$ be an $n \times m$ matrix of reals. Prove that

$$\max_i \sum_j a_{ij} \leq \sum_j \max_i a_{ij}$$

(*ii*) Use (*i*) to show that the *MEU* of an influence diagram will not increase by delaying an observation. (Hint: Look at the formulas for the two elimination orders.)

List of Notation

$\mathrm{Acc}(P, M)$	Measure of acceptance of M as a representation of P
$\mathrm{argmax}_D \rho$	An action from D maximizing ρ
$\mathrm{conf}(e)$	Measure of conflict of evidence e
dist_K	Kullback-Leibler divergence
dist_Q	Euclidean distance
$\mathrm{dom}(\phi)$	The domain of potential ϕ
$\underline{e}$	Evidence e represented as a finding
ECR	Expected cost of repair
$\mathrm{ef}(A)$	Efficiency of action a
EU	Expected utility
grad	Gradient vector
$\max_A$	Max-marginal over variable A
MEU	Maximal expected utility
$P(a)$	Probability of event a
$P(a \mid b)$	Probability of event a given b
$P(a \mid b, c)$	Probability of event a given b and c
$P(A)$	Probability distribution for variable a
$P(A \mid B)$	Probability distributions for variable A given the states of variable B
$P(A, B)$	Joint probability distribution for variables A and B
$pa(A)$	The parent set for variable A
$Size(G)$	The size of a triangulated graph
$Size(M)$	The size of model M
$\mathbf{1}$	The unit potential
$\sum_A$	Summation over variable A

$\Phi^{\downarrow V}$	Projection of set of potentials Φ down to domain V
$\Pi_i \psi_i$	The product of the potentials ϕ_i
$\Pi \Phi$	The product of all potentials in set Φ
Φ^{-X}	The potentials resulting from elimination of variable X from set of potentials Φ
$\otimes$	Combination operator
$\wedge$	Logical and
$\vee$	Logical or

Bibliography

Andreassen, S. (1992). Knowledge representation by extended linear models. In E. Keravnou (Ed.), *Deep Models for Medical Knowledge Engineering*, pp. 129–145. Elsevier Science Publishers.

Andreassen, S., F. V. Jensen, S. K. Andersen, B. Falck, U. Kjærulff, M. Woldbye, A. Sørensen, A. Rosenfalck, and F. Jensen (1989). MUNIN – an expert EMG assistant. In J. E. Desmedt (Ed.), *Computer-Aided Electromyography and Expert Systems*, Chapter 21, pp. 255–277. Elsevier Science Publishers, Amsterdam.

Bangsø, O. and P.-H. Wuillemin (2000). Top-down Construction and Repetitive Structures Representation in Bayesian Networks. In *Proceedings of the Thirteenth International FLAIRS Conference*, Orlando, FL. AAAI Press, Cambridge, MA.

Beeri, C., R. Fagin, D. Maier, and M. Yannakakis (1983). On the desirability of acyclic database schemes. *Journal of the Association for Computing Machinery 30*(3), 479–513.

Ben-Bassat, M. (1978). Myopic policies in sequential classification. *IEEE Transactions of Computing 27*, 170–74.

Bertele, U. and F. Brioschi (1972). *Nonserial Dynamic Programming.* Academic Press, London.

Buntine, W. L. (1994). Operations for learning with graphical models. *Journal of Artificial Intelligence Research 2*, 159–225.

Buntine, W. L. (1996). A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering 8*(2), 195–210.

Castillo, E., J. M. Gutiérrez, and A. S. Hadi (1996). A New Method for Efficient Symbolic Propagation in Discrete Bayesian Networks. *Networks 28*(1), 31–43.

Castillo, E., J. M. Gutiérrez, and A. S. Hadi (1997). Sensitivity analysis in discrete Bayesian networks. In A. P. Sage (Ed.), *IEEE Transactions on Systems, Man, and Cybernetics. Part A: Systems and Humans*, pp. 412–423. IEEE Periodicals, New York.

Cooper, G. F. (1988). A method for using belief networks as influence diagrams. In R. D. Schachter, T. S. Levitt, L. N. Kanal, and J. F. Lemmer (Eds.), *Proceedings of the Fourth Workshop on Uncertainty in Artificial Intelligence*, Minneapolis, MN, pp. 55–63. Elsevier Science Publishers, New York.

Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence 42*(2–3), 393–405.

Cooper, G. F. and E. Herskovits (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning 9*(4), 309–348.

Coupé, V. M. H. and L. C. van der Gaag (1998). Practicable sensitivity analysis of Bayesian belief networks. In M. Hušková, P. Lachout, and J. A. Víšek (Eds.), *Prague Stochastics '98 – Proceedings of the Joint Session of the 6th Prague Symposium of Asymptotic Statistics and the 13th Prague Conference on Information Theory, Statistical Decision Functions and Random Processes*, pp. 81–86. Union of Czech Mathematicians and Physicists, Prague.

Cowell, R. G. (1994). Decision networks: a new formulation for multistage decision problems. Research Report 132, Department of Statistical Science, University College London, London.

Cowell, R. G., A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter (1999). *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Science. Springer-Verlag, New York.

D'Ambrosio, B. (1991). Local expression language for probabilistic dependence: a preliminary report. In B. D'Ambrosio, P. Smets, and P. Bonissone (Eds.), *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, Los Angeles, CA, pp. 95–102. Morgan Kaufmann Publishers, San Mateo, CA.

Dawid, A. P. (1992). Applications of a general propagation algorithm for a probabilistic expert system. *Statistics and Computing 2*, 25–36.

Dawid, A. P. and S. L. Lauritzen (1993). Hyper Markov laws in the statistical analysis of decomposable graphical models. *The Annals of Statistics 21*(3), 1272–1317.

de Dombal, F., D. Leaper, J. Staniland, A. McCann, and J. Harrocks (1972). Computer-aided diagnosis of acute abdominal pain. *British Medical Journal 2*, 9–13.

Dechter, R. (1996). Bucket elimination: a unifying framework for probabilistic inference. In E. Horvitz and F. V. Jensen (Eds.), *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, Portland, OR, pp. 211–219. Morgan Kaufmann Publishers, San Francisco, CA.

Dittmer, S. L. and F. V. Jensen (1997). Myopic value of information in influence diagrams. In D. Geiger and P. Shenoy (Eds.), *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, Providence, RI, pp. 142–149. Morgan Kaufmann Publishers, San Francisco, CA.

Druzdzel, M. and L. van der Gaag (1995). Elicitation of probabilities for belief networks: combining qualitative and quantitative information. In P. Besnard and S. Hanks (Eds.), *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Quebec, Canada, pp. 141–148. Morgan Kaufmann Publishers, San Francisco, CA.

Edwards, D. and T. Havranek (1985). A fast procedure for model search in multidimensional contingency tables. *Biometrika 72*, 339–351.

Fung, R. and S. Crawford (1990). CONSTRUCTOR – a system for induction of probabilistic models. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, pp. 762–779. AAAI Press, Cambridge, MA.

Geman, S. and D. Geman (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 6*(6), 721–741.

Gilks, W., A. Thomas, and D. Spiegelhalter (1994). A language and a program for complex Bayesian modelling. *The Statistician 43*, 169–178.

Golumbic, M. C. (1980). *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, London.

Gorry, G. and G. Barnett (1968). Experience with a model of sequential diagnosis. *Computers and Biomedical Research 1*, 490–507.

Habbema, J. D. F. (1976). Models diagnosis and detection of diseases. In F. de Dombal et al. (Eds.), *Decision Making and Medical Care*, pp. 399–411. Elsevier Science Publishers, Amsterdam.

Heckerman, D. (1990). Probabilistic similarity networks. *Networks 20*, 607–636.

Heckerman, D., J. Breese, and K. Rommelse (1995). Decision-theoretic troubleshooting. *Communications of the ACM 38*(3), 49–56.

Heckerman, D., E. Horvitz, and B. Nathwani (1992). Towards normative expert systems: Part I, the Pathfinder project. *Methods of Information in Medicine 31*, 90–105.

Henrion, M. (1988). Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In J. F. Lemmer and L. M. Kanal (Eds.), *Uncertainty in Artificial Intelligence 2*, pp. 149–163. Elsevier Science Publishers.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes.* MIT Press, Cambridge, MA.

Howard, R. A. (1966). Information value theory. *IEEE Transactions on Systems Science and Cybernetics 2*, 22–26.

Howard, R. A. (1984). The used car buyer. In R. A. Howard and J. E. Matheson (Eds.), *Readings on The Principles and Applications of Decision Analysis*, Volume 2, Chapter 36, pp. 691–718. Strategic Decisions Group, Menlo Park, CA. Original copyright 1962.

Howard, R. A. and J. E. Matheson (1984). Influence diagrams. In R. A. Howard and J. E. Matheson (Eds.), *Readings on the Principles and Applications of Decision Analysis*, Volume 2, pp. 719–762. Strategic Decisions Group, Menlo Park, CA.

Jensen, F., F. V. Jensen, and S. L. Dittmer (1994). From influence diagrams to junction trees. In R. L. de Mantaras and D. Poole (Eds.), *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, Seattle, WA, pp. 367–373. Morgan Kaufmann Publishers, San Francisco, CA.

Jensen, F. V. (1999). Gradient descent training of Bayesian networks. In A. Hunter and S. Parson (Eds.), *Proceedings of the Fifth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, Volume 1638 of *Lecture Notes in Artificial Intelligence*, London, UK, pp. 190–200. Springer, Berlin.

Jensen, F. V., S. Aldenryd, and K. B. Jensen (1995). Sensitivity analysis in Bayesian networks. In C. Froidevaux and J. Kohlas (Eds.), *Proceedings of ECSQARU'95*, Volume 946 of *Lecture Notes in Artificial Intelligence*, Fribourg, Switzerland, pp. 243–250. Springer, Berlin.

Jensen, F. V., B. Chamberlain, T. Nordahl, and F. Jensen (1991). Analysis in HUGIN of data conflict. In P. Bonnisone et al. (Eds.), *Uncertainty in Artificial Intelligence 6*, pp. 519–528. Elsevier Science Publishers.

Jensen, F. V., S. L. Lauritzen, and K. G. Olesen (1990). Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly 4*, 269–282.

Jordan, M. I. (Ed.) (1998). *Learning in Graphical Models*, Volume 89 of *Nato ASI Series, Series D: Behavioural and Social Sciences*. Kluwer Academic Publishers, Dordrecht, The Netherlands. Published in cooperation with NATO Scientific Affairs Division.

Kalagnanam, J. and M. Henrion (1990). A comparison of decision analysis and expert rules for sequential analysis. In P. Besnard and S. Hanks (Eds.), *Uncertainty in Artificial Intelligence 4*, pp. 271–281. Elsevier Science Publishers.

Kim, J. H. and J. Pearl (1983). A computational model for causal and diagnostic reasoning in inference systems. In A. Bundy (Ed.), *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, pp. 190–193. Morgan Kaufmann, Los Altos, CA.

Kim, Y. and M. Valtorta (1995). On the detection of conflicts in diagnostic Bayesian networks using abstraction. In P. Besnard and S. Hanks (Eds.), *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 362–367. Morgan Kaufmann Publishers, San Francisco.

Kjærulff, U. (1995). HUGS: Combining exact inference and Gibbs sampling in junction trees. In P. Besnard and S. Hanks (Eds.), *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Quebec, Canada, pp. 368–375. Morgan Kaufmann Publishers, San Francisco, CA.

Kjærulff, U. and L. C. van der Gaag (2000). Making sensitivity analysis computationally efficient. In C. Boutilier and M. Goldszmidt (Eds.), *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, Stanford, CA, pp. 317–325. Morgan Kaufmann Publishers, San Francisco, CA.

Lauritzen, S. L. (1996). *Graphical Models*. Oxford University Press, Oxford.

Lauritzen, S. L. and F. V. Jensen (1997). Local computation with valuations from a commutative semigroup. *Annals of Mathematics and Artificial Intelligence 21*, 51–69.

Lauritzen, S. L. and D. J. Spiegelhalter (1988). Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society, Series B 50*, 157–224.

Lindley, D. V. (1971). *Making Decisions*. John Wiley & Sons, New York.

Madsen, A. L. and F. V. Jensen (1999a). Lazy evaluation of symmetric bayesian decision problems. In K. Laskey and H. Prade (Eds.), *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, Stockholm, Sweden, pp. 382–390. Morgan Kaufmann Publishers, San Francisco, CA.

Madsen, A. L. and F. V. Jensen (1999b). Lazy propagation: A junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence 113*, 203–245.

Mitchell, T. M. (1997). *Machine Learning.* McGraw–Hill Book Co., New York.

Ndilikilikesha, P. (1994). Potential influence diagrams. *International Journal of Approximate Reasoning 10*, 251–285.

Nilsson, D. and F. V. Jensen (2000). Probabilities of future decisions. In B. Bouchon-Meunier, R. R. Yager, and L. A. Zadeh (Eds.), *Information, Uncertainty and Fusion*, pp. 161–171. Kluwer Academic Publishers, Dordrecht, the Netherlands.

Nilsson, D. and S. Lauritzen (2000). Evaluating influence diagrams using LIMIDs. In C. Boutilier and M. Goldszmidt (Eds.), *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, Stanford, CA, pp. 436–445. Morgan Kaufmann Publishers, San Francisco, CA.

Olesen, K. G., S. L. Lauritzen, and F. V. Jensen (1992). aHUGIN: A system creating adaptive causal probabilistic networks. In D. Dubois, M. P. Wellman, B. D'Ambrosio, and P. Smets (Eds.), *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, Stanford, CA, pp. 223–229. Morgan Kaufmann Publishers, San Francisco, CA.

Olmsted, S. M. (1983). *On Representing and Solving Decision Problems.* Ph. D. thesis, Department of Engineering-Economic Systems, Stanford University, Stanford, CA.

Pearl, J. (1982). Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach. In D. L. Waltz (Ed.), *National Conference on Artificial Intelligence*, Pittsburgh, PA, pp. 133–136. AAAI Press, Menlo Park, CA.

Pearl, J. (1986). Fusion, propagation, and structuring in belief networks. *Artificial Intelligence 29*(3), 241–288.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Series in Representation and Reasoning. Morgan Kaufmann Publishers, San Francisco, CA.

Puterman, M. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, Chichester, UK.

Raiffa, H. and R. Schlaifer (1961). *Applied Statistical Decision Theory.* MIT Press, Cambridge, MA.

Raiffa, M. (1968). *Decision Analysis: Introductory Lectures on Choices under Uncertainty.* Addison–Wesley, Reading, MA.

Russell, S. J., J. Binder, D. Koller, and K. Kanazawa (1995). Local learning in probabilistic networks with hidden variables. In C. S. Mellish (Ed.), *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Canada, pp. 1146–1152. Morgan Kaufmann Publishers, San Mateo, CA.

Shachter, R. D. (1986). Evaluating influence diagrams. *Operations Research 34*(6), 871–882.

Shachter, R. D. (2000). Efficient value of information computation. In K. B. Laskey and H. Prade (Eds.), *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, Stanford, CA, pp. 594–601. Morgan Kaufmann Publishers, San Francisco, CA.

Shafer, G. (1996). *Probabilistic Expert Systems.* Society for Industrial and Applied Mathematics, Philadelphia, PA.

Shafer, G. and P. Shenoy (1990). Probability propagation. *Annals of Mathematics and Artificial Intelligence 2*, 327–352.

Shenoy, P. P. (1992). Valuation-based systems for Bayesian decision analysis. *Operations Research 40*(3), 463–484.

Sochorová, M. and J. Vomlel (2000). Troubleshooting: NP-hardness and solutions methods. In J. Vejnarová (Ed.), *Proceedings of the Fifth Workshop on Uncertainty Processing*, Jindrichuv Hradec, Czech Republic, pp. 198–212. University of Economics, Prague.

Spiegelhalter, D. and S. L. Lauritzen (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks 20*, 579–605.

Spiegelhalter, D. J. and R. P. Knill-Jones (1984). Statistical and knowledge-based approaches to clinical decision-support systems. *Journal of the Royal Statistical Society, Series A 147*, 35–77.

Spohn, W. (1980). Stochastic independence, causal independence and shieldability. *Journal of Philosophical Logic 9*, 73–99.

Suermondt, H. J. (1992). *Explanation in Bayesian Belief Networks.* Ph. D. thesis, Knowledge Systems Laboratory, Medical Computer Science, Stanford University, CA. Report No. STAN-CS-92-1417.

Verma, T. S. (1987). Causal networks: Semantics and expressiveness. In L. N. Kanal, T. S. Levitt, and J. F. Lemmer (Eds.), *Proceedings of the Third Workshop on Uncertainty in Artificial Intelligence*, pp. 352–359. Elsevier Science Publishers, New York.

von Neumann, J. and O. Morgenstein (1953). *Theory of Games and Economic Behavior.* John Wiley & Sons, New York.

Winterfeldt, D. and W. Edwards (1986). *Decision Analysis and Behavioral Research.* Cambridge University Press, Cambridge, UK.

Zhang, N. L. W. (1998). Probabilistic Inference in Influence Diagrams. In G. F. Cooper and S. Moral (Eds.), *Proceedings of the fourteenth Conference on Uncertainty in Artificial Intelligence*, Wisconsin, pp. 514–522. Morgan Kaufmann Publishers, San Francisco, CA.

Index