Probabilistic Reasoning

(Probabilistisch Redeneren)

authors: Linda van der Gaag
Silja Renooij

Fall 2013

Universiteit Utrecht

Preface

In artificial-intelligence research, the probabilistic-network, or (Bayesian) belief-network framework for automated reasoning with uncertainty is rapidly gaining in popularity. The framework provides a powerful formalism for representing a joint probability distribution on a set of statistical variables. In addition, it offers algorithms for efficient probabilistic inference. At present, more and more knowledge-based systems employing the framework are being developed for various domains of application, ranging from probabilistic information retrieval to medical diagnosis. This syllabus provides a tutorial introduction to the probabilistic-network framework and highlights some issues of ongoing research in applying the framework for problem solving in real-life domains. Each chapter includes a number of exercises, and answers or hints to some of them (indicated by a *) are provided at the end.

This syllabus was first written in the late 1990s by L.C. van der Gaag and has been continuously under development eversince. Since 2001, adaptions and extensions have been made mostly by S. Renooij. The syllabus is by no means devoid from imperfections and any useful comments on its contents are greatly appreciated by the authors.

For the 2006 edition, several references to relevant recent research have been added to Chapters 4, 5, and 6. In addition, material on the subject of sensitivity analysis has been extended. For the 2009 edition, some small errors were corrected and relevant references were added. Similar updates were done for the 2013 edition, for which also an example in Chapter 6 was extended.

Linda van der Gaag
Silja Renooij
Utrecht University
July 2013

Contents

Chapter 1

Introduction

This chapter gives some historical background about the use of probability theory and other uncertainty formalisms for reasons of decision support. It briefly motivates the emergence of Probabilistic networks (or: (Bayesian) belief networks) and the reason why probabilistic network applications historically have often concerned the medical domain.

Over the past few decades interest in the results of artificial-intelligence research has been growing to an increasing extent. Especially the area of *knowledge-based systems* has attracted much attention. The phrase knowledge-based system, or *expert system*, is generally employed to denote computer systems in which some symbolic representation of human knowledge is incorporated and applied [Lucas & Van der Gaag, 1991, Jackson, 1990]. Knowledge-based systems are typically designed to deal with real-life problems that require considerable human knowledge and expertise for their solution; examples range from medical diagnosis and technical trouble shooting to financial advice and product design. It is their ability to capture and reason with (specialised) human knowledge that allows knowledge-based systems to arrive at a performance comparable to that of human experts. Knowledge-based systems by now have found their way from academic laboratories to the industrial world and are being integrated into conventional software environments.

As more and more knowledge-based systems are being developed for a large variety of problems, it becomes apparent that the knowledge required to solve these problems often is not precisely defined but instead is of an imprecise nature. In fact, many real-life problem domains are fraught with uncertainty. Human experts in these domains typically are able to form judgements and take decisions based on uncertain, incomplete, and sometimes even contradictory information. To be of practical use, a knowledge-based system has to deal with such information at least equally well. For this purpose, a knowledge-based system employs a formalism for representing uncertainty and an associated algorithm for manipulating uncertain information. The major research topic in artificial intelligence of *reasoning with uncertainty*, or *plausible reasoning*, addresses the design of such formalisms and algorithms [Shafer & Pearl, 1990].

As probability theory is a mathematically well-founded theory about uncertainty, having a long and outstanding tradition of research and experience, it is not surprising that this theory takes a prominent place in research on reasoning with uncertainty in knowledge-based systems. Unfortunately, applying probability theory in a knowledge-based context is not as easy as it may seem at first sight. Straightforward application of the basic concepts from probability theory leads to insuperable problems of computational complexity: explicit representation of a joint probability distribution requires exponential space (exponential in the number of variables discerned), and even if the distribution could be represented more economically, computing probabilities of interest by the basic rules of marginalisation and conditioning would have an exponential time complexity. The rich history of applying probability theory for reasoning with uncertainty in knowledge-based systems shows various attempts to settle these problems.

In this chapter, we sketch the historical background of applying probability theory in a knowledge-based system. We would like to note that our intention is not to be complete, but merely to give an impression of the problems encountered by researchers pioneering in automated probabilistic inference. In our sketch, we focus on the task of (medical) diagnosis. For a given problem domain, we discern a set of possible *hypotheses* $H = \{h_1, \ldots, h_n\}$, $n \geq 1$, and a set of pieces of *evidence* $E = \{e_1, \ldots, e_m\}$, $m \geq 1$, that may be observed in relation with these hypotheses. For ease of exposition, we assume that each of the hypotheses is either *true* or *false*; equally, we assume that each of the pieces of evidence is either *true* or *false*. A *diagnostic problem* in this domain now is a set of pieces of evidence $e \subseteq E$ that is actually observed and needs to be explained in terms of the hypotheses discerned. A *diagnosis* for a problem e under consideration is a set of hypotheses $h \subseteq H$ that best explains e.

As early as in the 1960s several research efforts on automated reasoning with uncertainty for diagnostic applications were undertaken [Warner *et al.*, 1961, Gorry & Barnett, 1968, De Dombal *et al.*, 1972]. The systems constructed in this period were based to a large extent on application of Bayes' Theorem; in the sequel, we will refer to the approach taken in these early systems as the *naive-Bayesian approach*. In this approach, the basic idea of computing a diagnosis for a set of actually observed pieces of evidence $e \subseteq E$ is to compute for all sets of hypotheses $h \subseteq H$ the conditional probability $\Pr(h \mid e)$ from the distribution Pr on the domain at hand, and then select a set $h' \subseteq H$ with highest probability. Since for real-life applications the conditional probabilities $\Pr(e \mid h)$ often are easier to come by than the conditional probabilities $\Pr(h \mid e)$, generally Bayes' Theorem is used for computing the required probabilities:

$$\Pr(h \mid e) \quad = \quad \frac{\Pr(e \mid h) \cdot \Pr(h)}{\Pr(e)}$$

It will be evident that this approach is quite expensive from a computational point of view: because a diagnosis may be composed of several different hypotheses, the number of probabilities to be computed equals $2^n - 1$. To overcome this problem of *time complexity*, a simplifying assumption is made: it is assumed that all hypotheses are *mutually exclusive* and *collectively exhaustive*. With this assumption only the n singleton hypothesis sets $\{h_i\}$ have to be considered as possible diagnoses. So, only the probabilities $\Pr(h_i \mid e)$ (writing h_i instead of $\{h_i\}$) for all $h_i \in H$ have to be computed. To this end, once more Bayes' Theorem is used:

$$\Pr(h_i \mid e) \quad = \quad \frac{\Pr(e \mid h_i) \cdot \Pr(h_i)}{\Pr(e)} \quad = \quad \frac{\Pr(e \mid h_i) \cdot \Pr(h_i)}{\sum_{k=1}^{n} \Pr(e \mid h_k) \cdot \Pr(h_k)}$$

For automated application of Bayes' Theorem in this form, several probabilities are required from the joint probability distribution Pr on the domain at hand. In fact, conditional probabilities $\Pr(e \mid h_k)$, $k = 1, \ldots, n$, for *every* combination of pieces of evidence $e \subseteq E$, have to be available. Apart from the fact that it is hardly likely that these probabilities will be readily available in a real-life problem domain, this means storing exponentially many probabilities. To overcome this problem of *space complexity*, a second simplifying assumption is made: it is assumed that all pieces of evidence are *conditionally independent* given any of the hypotheses discerned.

The two simplifying assumptions taken together allow for computing the probabilities $\Pr(h_i \mid e)$ for all $h_i \in H$ given observed evidence $e = \{e_{j_1}, \ldots, e_{j_p}\}$, $1 \leq p \leq m$, from

$$\begin{aligned}\Pr(h_i \mid e_{j_1} \wedge \cdots \wedge e_{j_p}) &= \frac{\Pr(e_{j_1} \wedge \cdots \wedge e_{j_p} \mid h_i) \cdot \Pr(h_i)}{\sum_{k=1}^{n} \Pr(e_{j_1} \wedge \cdots \wedge e_{j_p} \mid h_k) \cdot \Pr(h_k)} = \\ &= \frac{\Pr(e_{j_1} \mid h_i) \cdots \Pr(e_{j_p} \mid h_i) \cdot \Pr(h_i)}{\sum_{k=1}^{n} \Pr(e_{j_1} \mid h_k) \cdots \Pr(e_{j_p} \mid h_k) \cdot \Pr(h_k)}\end{aligned}$$

It will be evident that for any diagnostic problem e now only $n-1$ probabilities have to be computed, and that for this purpose only $m \cdot n$ conditional probabilities and $n-1$ prior ones have to be stored.

The systems for automated reasoning with uncertainty constructed in the 1960s were rather small-scaled: they were devised for clear-cut problem domains with only a small number of hypotheses and restricted evidence. For these small systems, all probabilities necessary for applying Bayes' Theorem could be acquired from statistical analysis of empirical data. Despite the underlying (over-)simplifying assumptions, these systems performed considerably well [De Dombal *et al.*, 1974]. Nevertheless, interest in this naive Bayesian approach to reasoning with uncertainty faded in the late 1960s and early 1970s. One of the reasons for this decline in interest is that the approach was feasible only for highly restricted problem domains. For larger or more complex domains, the above-mentioned simplifying assumptions often were seriously violated, causing degeneration of system behaviour. In addition, for larger domains the approach inevitably became demanding, either computationally or from an assessment point of view.

At this stage, the first diagnostic knowledge-based systems began to emerge from artificial-intelligence research. These systems mostly use *production rules* for representing human (experiential) knowledge in a modular form closely resembling logical implications — production rules are expressions of the form **if** ⟨*condition*⟩ **then** ⟨*conclusion*⟩. These so-called *rule-based expert systems* exhibit 'intelligent' reasoning behaviour by employing a heuristic reasoning algorithm that use the production rules for selective gathering of evidence and for pruning the search space of possible diagnoses. It is this pruning behaviour that renders the rule-based expert systems capable of dealing with larger and complexer problem domains than the early naive-Bayesian systems are. The best-known rule-based expert system developed in the 1970s is the MYCIN system for assisting physicians in the diagnosis and treatment of bacterial infections [Buchanan & Shortliffe, 1984].

In the context of rule-based expert systems, the naive Bayesian approach to reasoning with uncertainty is no longer feasible due to the large number of probabilities to be computed. Since in a rule-based system during problem solving the search space of possible diagnoses is pruned by heuristic as well as probabilistic criteria, it is necessary to compute probabilities for all intermediate results derived by the production rules in addition to the probabilities of the separate hypotheses. To allow for efficient computation of all these probabilities, a set of computation rules has been designed. These computation rules provide for computing the probability of an (intermediate) result from probabilities associated with the production rules that are used in its derivation; to this end, each production rule is assigned the conditional probability of its conclusion given its condition. Unfortunately, these computation rules do not always accord with the axioms of probability theory and can not even be considered approximation rules for computing probabilities. In the sequel, we will use the phrase *quasi-probabilistic* to refer to this approach. The most well-known illustration of the quasi-probabilistic approach is the *certainty-factor model*, designed originally for dealing with uncertainty in the MYCIN system [Shortliffe & Buchanan, 1984]. The certainty-factor model enjoys widespread use in rule-based expert systems built after MYCIN, even though by now it is widely known that the model is mathematically flawed. The relative success of the model can however be accounted for by its satisfactory behaviour in most applications and by its conceptual and computational simplicity [Van der Gaag, 1994].

Although the quasi-probabilistic approach to reasoning with uncertainty in knowledge-based systems on the one hand met with considerable success in the artificial-intelligence community, it was criticised severely on the other hand because of its ad-hoc character. The incorrectness of the approach from a mathematical point of view even led to a world-wide debate concerning the appropriateness of probability theory for handling uncertainty in a knowledge-based context.

The *adversaries* of probability theory argue that the theory is not expressive enough to cope with the different kinds of uncertainty that are encountered in real-life problem domains and therefore have to be dealt with in knowledge-based systems. As a consequence several other (more or less) mathematical models have been proposed for reasoning with uncertainty. A major trend in plausible reasoning has arisen from the claim that probability theory is not able to capture imprecision or vagueness, notions of uncertainty which are salient in natural language representations. The name of L.A. Zadeh is inseparable from this trend: he was the first to propose *fuzzy set theory* as the point of departure for the development of methods that are able to cope with vague information. *Dempster-Shafer theory* lies at the basis of another major trend in plausible reasoning. The theory was developed by G. Shafer, building on earlier work by A.P. Dempster [Shafer, 1976]. It was motivated by the observation that probability theory is not able to discern between uncertainty and ignorance due to incompleteness of information.

The *advocates* of probability theory, on the other hand, claim that it is provable that probability theory is the only correct way of dealing with uncertainty and that anything that can be done with non-probabilistic methods, can be done equally well using a probability-based method. For this claim often an argument by R.T. Cox is cited [Cox, 1979]: Cox states a simple set of intuitive properties a measure of uncertainty has to satisfy and subsequently shows that the basic axioms of probability theory follow. Here, we will not enter into the debate concerning the appropriateness of probability theory for reasoning with uncertainty in knowledge-based systems; for a wide range of diverging opinions, the reader is referred to [Cheeseman, 1988] with its ensuing discussions.

Although the above-mentioned debate was not in the least subdued, in the mid-1980s the *probabilistic network framework* was introduced as a novel approach to applying probability theory for reasoning with uncertainty in knowledge-based systems

[Pearl, 1988]. The probabilistic network framework is characterised by a powerful formalism for representing domain knowledge and the uncertainties that go with it — more in specific, the formalism provides for a concise representation of a joint probability distribution on a set of statistical variables. Associated with this formalism are algorithms for efficiently computing probabilities of interest and for processing evidence; these algorithms constitute the basic building blocks for reasoning with knowledge represented in the formalism. When compared to the naive-Bayesian approach on the one hand and the quasi-probabilistic approach on the other hand, the *probabilistic network approach* offers advantages over both. In contrast with the quasi-probabilistic approach, the probabilistic network approach has a firm mathematical foundation in probability theory. Contrasting the naive-Bayesian approach, the probabilistic network approach circumvents the need for simplifying assumptions by capturing and reasoning about actual independences among variables. Since its introduction, the probabilistic network framework has rapidly gained in popularity and by now is beginning to illustrate its worth in complex problem domains: practical applications have been and are being developed for example for medical diagnosis and prognosis [Andreassen *et al.*, 1987, Heckerman *et al.*, 1992, Blanco *et al.*, 2005], for probabilistic information retrieval [Bruza & Van der Gaag, 1994], in computer vision [Jensen *et al.*, 1990], in forensic science [Taroni *et al.*, 2006] and various other domains (see [Pourret, Naim & Marcot, 2008]). Whereas earlier applications of probabilistic networks were mostly handcrafted with the help of domain experts, the increasing availability of large data sets has made it much easier to construct applications directly from data [Neapolitan, 2003]. Even large data sets, however, usually do not contain sufficient reliable information to construct reliable networks of general topology; for this reason, network engineers either resort to using various types of classifier [Friedman *et al.*, 1997], or again have to rely on domain expertise to complete the network [Druzdzel & Van der Gaag, 2000].

This syllabus provide a tutorial introduction to the probabilistic network framework and highlight some issues of ongoing research in applying the framework for real-life problem solving. It is organised as follows. Chapter 2 provides some preliminaries from graph theory and from probability theory. In Chapter 3, we discuss the representation of probabilistic independence in graphical models. Chapter 4 introduces the probabilistic network framework: it details the probabilistic network formalism and outlines its associated algorithms. In Chapter 5 we address building probabilistic networks for real-life problem domains. Analysis of and problem solving with probabilistic networks is the topic of Chapter 6. The syllabus is rounded off with some conclusive discussion in Chapter 7.

Chapter 2

Preliminaries

This chapter refreshes the necessary concepts from graph- and probability theory that play a central role in the probabilistic network framework. These concepts can be found in any textbook on graph theory and probability theory. The chapter also introduces the notation that is used throughout the syllabus.

2.1 Graph Theory

In this section, some concepts from graph theory are reviewed. Our review is tailored to the probabilistic network framework and is not meant to be exhaustive; for further information, any introductory textbook on graph theory will suffice.

Generally two types of graph are discerned: undirected and directed ones.

Definition 2.1.1 *An* undirected graph *G is a pair $G = (V(G), E(G))$ where $V(G)$ is a finite set of* vertices *(also called nodes) and $E(G)$ is a set of unordered pairs (V_i, V_j), $V_i, V_j \in V(G)$, called* edges.

A directed graph*, or* digraph *for short, is a pair $G = (V(G), A(G))$ where $V(G)$ is a finite set of vertices and $A(G)$ is a set of ordered pairs (V_i, V_j), $V_i, V_j \in V(G)$, called* arcs. *An arc (V_i, V_j) is often written as $V_i \rightarrow V_j$ or $V_j \leftarrow V_i$.*

For a vertex in a graph, different sets of related vertices can be identified.

Definition 2.1.2 *In a digraph G, vertex V_j is called a* predecessor *(or parent) of vertex V_i if $(V_j, V_i) \in A(G)$; the set of all predecessors of vertex V_i in G is denoted as $\rho_G(V_i)$. Likewise, vertex V_j is called a* successor *(or child) of vertex V_i if $(V_i, V_j) \in A(G)$; the set of all successors of vertex V_i in G is denoted as $\sigma_G(V_i)$. The reflexive transitive closure*[1] *of V_i under the predecessor relation is denoted as $\rho^*_G(V_i)$; an element from $\rho^*_G(V_i)$ is called an* ancestor *of V_i. Similarly, $\sigma^*_G(V_i)$ denotes the* descendants *of V_i.*

The set of neighbours *of vertex V_i is defined as*

$$\nu_G(V_i) = \begin{cases} \sigma_G(V_i) \cup \rho_G(V_i) & \textit{if } G \textit{ is directed;} \\ \{V_j \mid (V_i, V_j) \in E(G)\} & \textit{if } G \textit{ is undirected} \end{cases}$$

[1]The reflexive closure of set A under r is $r^0(A) = A$, the transitive closure is $r^+(A) = r(A) \cup r^+(r(A))$, and both combined gives $r^*(A) = r^0(A) \cup r^+(A)$.

The size of the neighbour-set of a vertex is called its degree*. In case of a vertex in a digraph, we in addition define the* in-degree *to be its number of predecessors and the* out-degree *to be the number of its successors; the incoming and outgoing arcs together are called its* incident arcs*.*

In the sequel, we will often drop the subscript G from ρ_G etc. as long as ambiguity cannot occur.

The following definition introduces several types of vertex sequence for undirected graphs.

Definition 2.1.3 *Let $G = (V(G), E(G))$ be an undirected graph. A* path *from vertex V_0 to vertex V_k in G is a sequence of vertices $V_0, \dots, V_k$, $k \geq 0$, with distinct edges $(V_{i-1}, V_i) \in E(G)$, $i = 1, \dots, k$, between them; k is called the* length *of the path. A path is called* simple *if all vertices are distinct. A* cycle *is a path $V_0, \dots, V_k, V_0$ from V_0 to V_0 of non-zero length. The graph G is called* cyclic *if it contains at least one cycle; otherwise, it is called* acyclic*.*

In undirected graphs, self-loops (an edge (V_0, V_0)) are generally not allowed. The concepts of path and cycle introduced for undirected graphs directly apply to directed graphs by considering arcs rather than edges. Unless stated otherwise, we typically assume paths to be simple.

We now introduce the concept of *underlying graph.* This concept associates an undirected graph with a directed one. We thereby assume that directed graphs do not contain self-loops either, although this is not a convention.

Definition 2.1.4 *Let $G = (V(G), A(G))$ be a digraph. The* underlying graph *H of G is the undirected graph $H = (V(H), E(H))$ where $V(H) = V(G)$ and $E(H)$ is obtained from $A(G)$ by replacing each arc $(V_i, V_j) \in A(G)$ by an edge (V_i, V_j).*

Related to a digraph's underlying graph, we introduce two additional types of vertex sequence for digraphs.

Definition 2.1.5 *Let G be a digraph and let H be its underlying graph. A* chain *from vertex V_0 to vertex V_k in G is a sequence of vertices $V_0, \dots, V_k$, $k \geq 0$, that is a path in the underlying graph H of G; k is called the* length *of the chain. A* loop *in G is a sequence of vertices that is a cycle in the underlying graph H of G.*

Note that, in a digraph, the concept of path takes the directions of the arcs into account, while the concept of chain does not. A digraph is therefore *acyclic* if it contains no directed cycles; an acyclic digraph (or DAG) can contain loops, however.

In a directed graph, two vertices may be connected by a chain. If this property holds for any two vertices in a digraph, we say that the graph is *connected.*

Definition 2.1.6 *A digraph G is* connected *if there exists at least one chain between any two vertices in G; otherwise, it is called* unconnected*.*

We have introduced the concept of connectedness to apply to directed graphs; the concept, however, is easily extended to apply to undirected graphs.

We now distinguish between several types of digraph.

Definition 2.1.7 *A digraph G is called* singly connected *if it does not contain any loops; otherwise, it is called* multiply connected*. A singly connected digraph G is called a* directed tree *if each vertex in G has at most one predecessor.*

Note that in a singly connected digraph, there is at most one chain between any two vertices; this property does not hold for multiply connected digraphs.

To conclude, we introduce the concept of a *subgraph.* The concept is introduced for undirected graphs, but is extended straightforwardly to apply to digraphs.

Definition 2.1.8 *Let $G = (V(G), E(G))$ be an undirected graph. The* subgraph H induced *by $V \subseteq V(G)$ is the undirected graph $H = (V, (V \times V) \cap E(G))$.*

Note that a subgraph induced by a set of variables V takes from the original graph *all* edges existent among the vertices from V.

2.2 Probability Theory

In this section, we provide a brief review of some basic concepts from probability theory. Once more, our review is tailored to the probabilistic network framework and is not meant to be an exhaustive tutorial; for further information, any introductory textbook on probability theory will suffice.

Probability theory is often approached from a set-theoretic point of view. Probability distributions are then defined on *sets* of elements that represent events. All possible outcomes of an experiment (for example, the possible outcomes of rolling a die) are given by the sample space Ω and each event A is a subset of Ω. A probability measure/function/distribution then is a function from events to the $[0, 1]$ interval. As events are sets, combinations of events can be expressed using operations on sets such as union ($\cup$) and intersection ($\cap$). Outcomes of an experiment are often *coded* by using a *random variable* (also: *statistical/ stochastic variable*) which is a function from the sample space to another space (such as reals). By writing probability distributions on statistical variables, the notation suppresses references to the actual sample space. However, as a statement about a statistical variable defines an event, there is no actual difference.

In the Probabilistic Network community statistical variables are taken to be functions from Ω to Ω. For a variable V defined on outcomes *true* and *false*, for example, we therefore have $V(\mathit{true}) = \mathit{true}$ and $V(\mathit{false}) = \mathit{false}$. We now simply say that V can have or take on one of the *values true* and *false*, in which case we write $V = \mathit{true}$ or $V = \mathit{false}$ as possible *value-assignments*. The Probabilistic Network community approaches probability theory from an algebraic point of view by associating probabilities with *logical propositions* instead of sets.

In this syllabus, we consider a set of *statistical variables* $V = \{V_1, \ldots, V_n\}$, $n \geq 1$. For ease of exposition, we will often restrict the discussion to binary variables taking one of the truth values *true* and *false*; the generalisation to variables with more than two discrete values, however, is rather straightforward. For abbreviation, we will use v_i to denote the proposition that the variable V_i takes the value *true*; $V_i = \mathit{false}$ will be denoted as $\neg v_i$. The set of variables V may be looked upon as spanning a *Boolean algebra of propositions* $\mathcal{V}$. Informally speaking, this algebra comprises all logical propositions that are built from value assignments to the variables discerned. More formally, the Boolean algebra of propositions $\mathcal{V}$ spanned by V is the set of logical propositions consisting of the constant propositions True and False[2], the atomic proposition v_i for all $V_i \in V$, and all compound propositions that are constructed from these by applying the binary operators $\wedge$ (*conjunction*), $\vee$ (*disjunction*), and the unary operator $\neg$ (*negation*); the elements of the algebra $\mathcal{V}$ adhere to the usual axioms of propositional logic.

We now define a *joint probability distribution* as a function on a Boolean algebra of propositions that is spanned by a set of statistical variables.

Definition 2.2.1 *Let V be a set of statistical variables and let $\mathcal{V}$ be the Boolean algebra of propositions spanned by V. Let $\Pr : \mathcal{V} \rightarrow [0,1]$ be a function such that*

- $\Pr(x) \geq 0$*, for all $x \in \mathcal{V}$, and* $\Pr(\mathsf{False}) = 0$*, more in specific;*
- $\Pr(\mathsf{True}) = 1$*;*
- $\Pr(x \vee y) = \Pr(x) + \Pr(y)$*, for all $x, y \in \mathcal{V}$ such that $x \wedge y \equiv \mathsf{False}$.*

Then, $\Pr$ *is called a* joint probability distribution *on V. For each $x \in \mathcal{V}$, the function value* $\Pr(x)$ *is termed the* probability *of x.*

A probability $\Pr(x)$ for a logical proposition x expresses the amount of certainty concerning the truth of x. Note that in the previous definition we have associated probabilities with logical propositions instead of with sets, which is the more common view taken in (introductory) literature on probability theory. It can easily be shown, however, that the probability of an event (a set of outcomes) is equivalent to the probability of the truth of the proposition asserting the occurrence of the event [Finetti, 1970].

Example 2.2.2 *Suppose X and Y are statistical variables representing a coin toss. Let A be the event that X = heads and Y = tails then the probability of this event is*

- *from a set-theoretic point of view: the probability of event A, written* $\Pr(A)$*;*
- *from an algebraic point of view: the probability that (X = heads and Y = tails) $\equiv$* True*, written* $\Pr(X = \textit{heads} \wedge Y = \textit{tails})$

If X = heads and Y = tails were considered two separate events A and B, then this would make no difference from the algebraic point of view, but in the set-theoretic approach we should now write $\Pr(A \cap B)$*.*

In the sequel, we will want to single out *strictly positive* joint probability distributions as these have some interesting properties. Strictly positive distributions for example are well-known for their not embedding any functional or logical relationships among their variables.

Definition 2.2.3 *Let V be a set of statistical variables and let $\mathcal{V}$ be the Boolean algebra of propositions spanned by V. Let* $\Pr$ *be a joint probability distribution on V.* $\Pr$ *is* strictly positive *if* $\Pr(x) = 0$ *implies $x \equiv \mathsf{False}$.*

We now introduce the concept of conditional probability.

[2]Note the difference between these *propositions* and the afore mentioned *outcomes/values*!

Definition 2.2.4 *Let V be a set of statistical variables and let $\mathcal{V}$ be the Boolean algebra of propositions spanned by V. Let* Pr *be a joint probability distribution on V. For each $x, y \in \mathcal{V}$ with* $\Pr(y) > 0$*, the* conditional probability *of x given y, denoted as* $\Pr(x \mid y)$*, is defined as*

$$\Pr(x \mid y) = \frac{\Pr(x \wedge y)}{\Pr(y)}$$

The conditional probability $\Pr(x \mid y)$ expresses the amount of certainty concerning the truth of x *given* that the information y is *known with certainty.* Note that a conditional probability $\Pr(x \mid y) = p$ does not mean that whenever y is known to be true, the probability of x equals p: it means that the probability of x equals p if y is known to be true and *nothing else is known that may affect the certainty concerning the truth of x.* In the sequel, we will assume that all conditional probabilities specified are properly defined, that is, for each conditional probability $\Pr(x \mid y)$, we will implicitly assume that $\Pr(y) > 0$. We further state without proof that for a given element $y \in \mathcal{V}$, the conditional probabilities $\Pr(x \mid y)$ for all $x \in \mathcal{V}$ once more constitute a joint probability distribution on V; this probability distribution is called the *conditional probability distribution given y* and will sometimes be denoted as $\Pr^y$. A conditional probability distribution is sometimes referred to as a *posterior* probability distribution; the joint probability distribution it is obtained from then in contrast is referred to as the *prior* distribution.

The following definition introduces the concept of independence of propositions.

Definition 2.2.5 *Let V be a set of statistical variables and let $\mathcal{V}$ be the Boolean algebra of propositions spanned by V. Let* Pr *be a joint probability distribution on V. Then, two propositions $x, y \in \mathcal{V}$ are called* (mutually) independent *in* Pr *if*

$$\Pr(x \wedge y) = \Pr(x) \cdot \Pr(y)$$

otherwise, x and y are called dependent *in* Pr. *Two propositions $x, y \in \mathcal{V}$ are called* conditionally independent *given the proposition $z \in \mathcal{V}$ in* Pr *if*

$$\Pr(x \wedge y \mid z) = \Pr(x \mid z) \cdot \Pr(y \mid z)$$

otherwise, x and y are called conditionally dependent *given z in* Pr.

In the sequel, we will make extensive use of various well-known properties of joint probability distributions. Before stating these properties, we provide some additional concepts and notational conventions. Recall that so far we have built on the Boolean algebra of propositions $\mathcal{V}$ spanned by some set of statistical variables V. In the sequel, we will want to focus on the variables themselves and to refer to (arbitrary) conjunctions of value assignments to all variables from the set V or from some subset of V. We will use C_W to denote the conjunction $C_W = \bigwedge_{V_i \in W} V_i$ of all variables from $W \subseteq V$; for $W = \varnothing$, we take $C_W = \mathsf{True}$. The conjunction of variables C_W is called the *configuration template* of W. A conjunction c_W of value assignments to the variables from W is called a *configuration* of W. A configuration c_W is nothing more than a short-hand notation to indicate that you are considering a proposition that consists of the conjunction of atomic propositions representing *some* value assignment to each variable in the set W. Writing c_W, $W = \{W_1, \ldots W_m\}$, instead of $W_1 =$ *some value* $\wedge$ $W_2 =$ *some value* $\wedge \ldots \wedge$ $W_m =$ *some value* is very convenient, especially if you don't care about the actual values and variables. Note that a configuration template C_W is a short-hand notation for an even more general statement, namely about *all possible* value assignments to the variables in W: any configuration c_W of W of interest can be obtained by *filling in* appropriate values for all variables involved. To avoid abundance of braces, we will often write C_{V_i} and c_{V_i} instead of $C_{\{V_i\}}$ and $c_{\{V_i\}}$, respectively, for singleton sets $\{V_i\}$. Please note that for a single vertex V_i we have that $C_{\{V_i\}} = V_i$ and that $\Pr(V_i)$ has therefore an entirely different meaning than it would from a set-theoretic point of view!

We now state the various properties that we will use in the sequel. We would like to note that in the literature on probability theory these properties are introduced in many different appearances; we have chosen the form that suits our purposes best. The property stated in the following proposition is known as the *chain rule.*

Proposition 2.2.6 *Let $V = \{V_1, \ldots, V_n\}$, $n \geq 1$, be a set of statistical variables and let* Pr *be a joint probability distribution on V. Then,*

$$\Pr(C_V) = \Pr(V_1 \wedge \cdots \wedge V_n) = \Pr(V_n \mid V_1 \wedge \cdots \wedge V_{n-1}) \cdot \ldots \cdot \Pr(V_2 \mid V_1) \cdot \Pr(V_1)$$

In the expression stated in the previous proposition, each V_i is a variable that takes either the value *true* or the value *false*, expressed as v_i and $\neg v_i$, respectively. The expression therefore represents 2^n equalities, one for each configuration of the set of variables V.

The property stated in the following proposition is termed the *marginalisation property.*

Proposition 2.2.7 *Let V be a set of statistical variables and let* Pr *be a joint probability distribution on V. Then,*

$$\Pr(C_X) = \sum_{c_Y} \Pr(C_X \wedge c_Y)$$

for all sets of variables $X, Y \subseteq V$.

We state without proof that for any set of variables $X \subseteq V$, the probabilities $\Pr(c_X)$ for all configurations c_X of X once more constitute a joint probability distribution; this probability distribution is termed the *marginal probability distribution* on X.

The *conditioning property* is stated in the following proposition.

Proposition 2.2.8 *Let V be a set of statistical variables and let* Pr *be a joint probability distribution on V. Then,*

$$\Pr(C_X) = \sum_{c_Y} \Pr(C_X \mid c_Y) \cdot \Pr(c_Y)$$

for all sets of variables $X, Y \subseteq V$.

The following theorem is known as *Bayes' Theorem.*

Theorem 2.2.9 *Let V be a set of statistical variables and let* Pr *be a joint probability distribution on V. Then,*

$$\Pr(C_X \mid C_Y) = \frac{\Pr(C_Y \mid C_X) \cdot \Pr(C_X)}{\Pr(C_Y)}$$

for all sets of variables $X, Y \subseteq V$.

To conclude this section, we once more turn to the concept of (conditional) independence. Recall that so far we have taken the concept of independence to apply to *propositions*. We now introduce the concept of independence of *variables*.

Definition 2.2.10 *Let V be a set of statistical variables and let $X, Y, Z \subseteq V$. Let* $\Pr$ *be a joint probability distribution on V. Then, the set of variables X is called* conditionally independent *of the set of variables Y given the set of variables Z in* $\Pr$ *if*

$$\Pr(C_X \mid C_Y \wedge C_Z) = \Pr(C_X \mid C_Z)$$

otherwise, X is called conditionally dependent *of Y given Z in* $\Pr$.

In qualitative terms, the expression $\Pr(C_X \mid C_Y \wedge C_Z) = \Pr(C_X \mid C_Z)$ indicates that, once information about Z is available, information about Y is irrelevant with respect to X. Note that for X and Y to be independent given Z, *every* pair of configurations of X and Y has to be independent given *every* configuration of Z. Independence of variables therefore implies independence of propositions. The reverse property, however, does *not* hold in general. Also note that the expression from Definition 2.2.10 is asymmetric in X and Y. Using Bayes' Theorem, however, it is easily shown that $\Pr(C_X \mid C_Y \wedge C_Z) = \Pr(C_X \mid C_Z)$ implies $\Pr(C_Y \mid C_X \wedge C_Z) = \Pr(C_Y \mid C_Z)$.

Exercises

Exercise 2.1

Prove the following properties for any joint probability distribution, using only definitions and not the properties from this exercise:

a. the chain rule (stated in Proposition 2.2.6);

b. Bayes' Theorem (stated in Theorem 2.2.9);

c. the marginalisation property (stated in Proposition 2.2.7);

** d. the conditioning property (stated in Proposition 2.2.8).*

Exercise 2.2

Let V be a set of statistical variables and let $\Pr$ *be a joint probability distribution on V. Show that*

$$\Pr(C_X \vee C_Y) = \Pr(C_X) + \Pr(C_Y) - \Pr(C_X \wedge C_Y)$$

for all sets of variables $X, Y \subseteq V$.

Exercise 2.3

Let V be a set of statistical variables and let $\Pr$ *be a joint probability distribution on V. Show that*

$$\Pr(C_X \mid C_Z) = \sum_{c_Y} \Pr(C_X \mid c_Y \wedge C_Z) \cdot \Pr(c_Y \mid C_Z)$$

for all sets of variables $X, Y, Z \subseteq V$.

*** Exercise 2.4**

Let V be a set of statistical variables and let $X, Y, Z \subseteq V$. Let $\Pr$ *be a joint probability distribution on V. Show that the set of variables X is conditionally independent of the set of variables Y given the set of variables Z if and only if* $\Pr(C_X \wedge C_Y \mid C_Z) = \Pr(C_X \mid C_Z) \cdot \Pr(C_Y \mid C_Z)$.

Chapter 3

Independences and Graphical Representations

This chapter formalises two types of independence relation. The first, I_{Pr}, is the type of relation that can be captured by a probability distribution. These independence relations form a proper subset of a more general type of independence relation I that abstracts away from probability distributions. The chapter also discusses different representations of independence relations, most notably (in)directed graphs. An important notion introduced in this chapter is the concept of d-separation. Current research into independence relations is focussed on defining small generating sets [Waal & Van der Gaag, 2005], and on automated construction of graphical representations from them [Baioletti *et al.*, 2011].

The historical background to the framework of probabilistic networks shows various attempts to handle the computational complexity of applying probability theory for reasoning with uncertainty in knowledge-based systems. The concept of (conditional) independence plays a key role in these attempts as knowledge about independences allows for simplifying computations. In this chapter, we address formalisms that allow for a concise representation of an independence relation for effective use in a knowledge-based system.

3.1 The Concept of Independence Revisited

In most introductory literature on probability theory, the concept of (conditional) independence is introduced in terms of numerical quantities: the independence relation of a joint probability distribution is taken to be implicitly embedded in the probabilities involved. Recall for example that in the previous chapter we have defined two sets of variables X and Y to be conditionally independent given a third set of variables Z if $\mathrm{Pr}(C_X \mid C_Y \wedge C_Z) = \mathrm{Pr}(C_X \mid C_Z)$. A definition of independence in terms of numbers suggests that, in order to determine whether two sets of variables are (conditionally) independent, several conditional probabilities have to be computed and several equalities have to be tested; moreover, such a definition suggests that for determining independence a joint probability distribution has to be explicitly available for the variables discerned. In contrast, humans tend to be able to state directly, with conviction and consistency, whether or not two sets of variables are independent. Such statements of independence typically are issued qualitatively, without any reference to numerical manipulation of exact probabilities. Based on these observations, we cannot but conclude that the concept of independence is far more basic to human reasoning than its numerical definition suggests. In fact, the definition of independence in terms of probabilities may be looked upon as a *quantitative* way of capturing the basic concept which is *qualitative* in nature. To formalise properties of the qualitative concept of independence, J. Pearl and his co-researchers have designed an *axiomatic system for independence* [Pearl & Paz, 1985, Pearl & Verma, 1987, Geiger & Pearl, 1988]. In this section, we review this axiomatic system.

3.1.1 Pearl's Axiomatic System for Independence

We begin our review of Pearl's axiomatic system for independence by introducing some new terminology and notational convention.

Definition 3.1.1 *Let V be a set of statistical variables and let* $\Pr$ *be a joint probability distribution on V. Then, the* independence relation $I_{\Pr} \subseteq \mathcal{P}(V) \times \mathcal{P}(V) \times \mathcal{P}(V)$ *of* $\Pr$ *is defined by $(X, Z, Y) \in I_{\Pr}$ if and only if* $\Pr(C_X \mid C_Y \wedge C_Z) = \Pr(C_X \mid C_Z)$ *for all sets of variables $X, Y, Z \subseteq V$.*

In the sequel, we will write $I_{\Pr}(X, Z, Y)$ to denote $(X, Z, Y) \in I_{\Pr}$ and $\neg I_{\Pr}(X, Z, Y)$ to denote $(X, Z, Y) \notin I_{\Pr}$. A statement $I_{\Pr}(X, Z, Y)$ of a joint probability distribution's independence relation $I_{\Pr}$ is termed an *independence statement.* In qualitative terms, an independence statement $I_{\Pr}(X, Z, Y)$ expresses that in the context of information about Z information about Y is irrelevant with respect to X. We note that the above definition allows for stating some trivial but convenient independence statements, such as $I_{\Pr}(X, X, Y)$, which holds iff $\Pr(C_X \mid C_Y \wedge C_X) = \Pr(C_X \mid C_X)$, i.e. $1 = 1$; its symmetric version $I_{\Pr}(Y, X, X)$ is also trivially true since $I_{\Pr}(Y, X, X)$ iff $\Pr(C_Y \mid C_X \wedge C_X) = \Pr(C_Y \mid C_X)$.

In designing his axiomatic system for independence, Pearl builds on a set of properties that are satisfied by any joint probability distribution's independence relation; Theorem 3.1.2 reviews these properties.

Theorem 3.1.2 *Let V be a set of statistical variables. Let* $\Pr$ *be a joint probability distribution on V and let $I_{\Pr}$ be its independence relation. Then, $I_{\Pr}$ satisfies the properties*

- $I_{\Pr}(X, Z, Y) \rightarrow I_{\Pr}(Y, Z, X)$; *(symmetry)*
- $I_{\Pr}(X, Z, Y \cup W) \rightarrow I_{\Pr}(X, Z, Y) \wedge I_{\Pr}(X, Z, W)$; *(decomposition)*
- $I_{\Pr}(X, Z, Y \cup W) \rightarrow I_{\Pr}(X, Z \cup W, Y)$; *(weak union)*
- $I_{\Pr}(X, Z, Y) \wedge I_{\Pr}(X, Z \cup Y, W) \rightarrow I_{\Pr}(X, Z, Y \cup W)$; *(contraction)*

for all mutually disjoint sets of variables $X, Y, Z, W \subseteq V$. If the distribution $\Pr$ *is strictly positive, then $I_{\Pr}$ satisfies the additional property*

- $I_{\Pr}(X, Z \cup W, Y) \wedge I_{\Pr}(X, Z \cup Y, W) \rightarrow I_{\Pr}(X, Z, Y \cup W)$; *(intersection)*

for all mutually disjoint sets of variables $X, Y, Z, W \subseteq V$.

The properties stated in the previous theorem are easily verified from the basic axioms of probability theory. We would like to note that we have closely followed Pearl by stating the properties in the theorem to hold for *mutually disjoint* sets of variables only [Pearl, 1988]. These properties, however, also hold for overlapping sets of variables [Van der Gaag & Meyer, 1998].

Pearl now takes the properties stated in Theorem 3.1.2 as *axioms* for the qualitative concept of independence [Pearl, 1988]. Following the properties for I_{Pr}, Pearl assumed for each axiom that the sets of variables involved are mutually disjoint. Given the insight that the properties also hold for overlapping sets, we will lift the assumption of mutual disjointness in the next and all following definitions involving independence relations. The following now defines informational independence:

Definition 3.1.3 *Let V be a set of statistical variables. A* semi-graphoid independence relation *on V is a ternary relation $I \subseteq \mathcal{P}(V) \times \mathcal{P}(V) \times \mathcal{P}(V)$ such that I satisfies the properties*

- $I(X, Z, Y) \rightarrow I(Y, Z, X)$;
- $I(X, Z, Y \cup W) \rightarrow I(X, Z, Y) \wedge I(X, Z, W)$;
- $I(X, Z, Y \cup W) \rightarrow I(X, Z \cup W, Y)$;
- $I(X, Z, Y) \wedge I(X, Z \cup Y, W) \rightarrow I(X, Z, Y \cup W)$;

for all sets of variables $X, Y, Z, W \subseteq V$. A graphoid independence relation *I on V is a semi-graphoid independence relation on V such that I satisfies the additional property*

- $I(X, Z \cup W, Y) \wedge I(X, Z \cup Y, W) \rightarrow I(X, Z, Y \cup W)$;

for all sets of variables $X, Y, Z, W \subseteq V$.

The properties described in the previous definition with each other convey the idea that learning irrelevant information does not alter the independences among the variables discerned [Pearl, 1988]. We consider the qualitative meanings of the various properties separately.

The property

$$I(X, Z, Y) \rightarrow I(Y, Z, X)$$

for all sets of variables $X, Y, Z \subseteq V$, states that if information about Y is deemed irrelevant with respect to X in the context of some information about Z, then information about X must be irrelevant with respect to Y in this context; this property is called the *symmetry axiom*.

The property

$$I(X, Z, Y \cup W) \rightarrow I(X, Z, Y) \wedge I(X, Z, W)$$

for all sets of variables $X, Y, Z, W \subseteq V$, asserts that if information about both Y and W is judged irrelevant with respect to X, then both information about Y and information about W must be irrelevant with respect to X separately; this property is known as the *decomposition axiom.* We would like to note that the decomposition axiom may be reformulated as $I(X, Z, Y \cup W) \rightarrow I(X, Z, Y)$; we have chosen, however, to use Pearl's original formulation because it conveys the idea of decomposition more clearly.

The property

$$I(X, Z, Y \cup W) \rightarrow I(X, Z \cup W, Y)$$

for all sets of variables $X, Y, Z, W \subseteq V$, states that learning information about W that is known to be irrelevant with respect to X cannot help irrelevant information about Y to become relevant with respect to X; this property is known as the *weak union axiom*.

The property

$$I(X, Z, Y) \wedge I(X, Z \cup Y, W) \rightarrow I(X, Z, Y \cup W)$$

for all sets of variables $X, Y, Z, W \subseteq V$, states that if we judge information about W to be irrelevant with respect to X after learning some irrelevant information about Y, then the information about W must have been irrelevant with respect to X before we learned Y; this property is known as the *contraction axiom*. Note that the contraction axiom can be reformulated as $I(X, Z, Y) \rightarrow (I(X, Z \cup Y, W) \rightarrow I(X, Z, Y \cup W))$. From this reformulation, it is seen that the axiom can be looked upon as a *conditional* reverse of the weak union axiom.

We now consider the property

$$I(X, Z \cup W, Y) \wedge I(X, Z \cup Y, W) \rightarrow I(X, Z, Y \cup W)$$

for all sets of variables $X, Y, Z, W \subseteq V$, for graphoid independence relations. This property states that if, in the context of some information about Z, learning information about W renders information about Y irrelevant with respect to X and learning Y renders W irrelevant with respect to X, then the information about both Y and W must be irrelevant with respect to X given Z; this property is known as the *intersection axiom*.

From Definition 3.1.3 and Theorem 3.1.2, we observe that every independence relation that is embedded in a joint probability distribution is a semi-graphoid independence relation; this property is stated more formally in the following corollary.

Corollary 3.1.4 *Let V be a set of statistical variables. Let* $\Pr$ *be a joint probability distribution on V and let $I_{\Pr}$ be its independence relation. Then, $I_{\Pr}$ is a semi-graphoid independence relation. Furthermore, if* $\Pr$ *is strictly positive, then $I_{\Pr}$ is a graphoid independence relation.*

Unfortunately, although any probability distribution's independence relation is a semi-graphoid independence relation, the reverse property does *not* hold. There exist semi-graphoid independence relations for which there do not exist joint probability distributions embedding them; for details, we refer to [Van der Gaag & Meyer, 1996, Studený, 1989]. We would like to note that it has been shown that a finite axiomatisation of the concept of probabilistic independence does not exist [Studený, 1992].

3.1.2 Properties of Independence Relations

Using the definition of informational independence, we derive some convenient properties of (semi-graphoid and graphoid) independence relations. The following lemma shows that the symmetry and contraction axioms are easily generalised to bi-implications.

Lemma 3.1.5 *Let V be a set of statistical variables. Furthermore, let I be a semi-graphoid independence relation on V. Then,*

- $I(X, Z, Y) \leftrightarrow I(Y, Z, X)$;
- $I(X, Z, Y) \wedge I(X, Z \cup Y, W) \leftrightarrow I(X, Z, Y \cup W)$;

for all sets of variables $X, Y, Z, W \subseteq V$.

Proof. We begin our proof by observing that since I is a semi-graphoid independence relation, it obeys the first four axioms stated in Definition 3.1.3. The first property stated in the lemma now follows directly from the symmetry axiom. For the second property, we observe that $I(X, Z, Y) \wedge I(X, Z \cup Y, W) \rightarrow I(X, Z, Y \cup W)$ coincides with the contraction axiom and therefore trivially holds for the relation I. We will now prove that $I(X, Z, Y \cup W) \rightarrow I(X, Z, Y) \wedge I(X, Z \cup Y, W)$. We have

$$I(X, Z, Y \cup W) \quad \Rightarrow \quad I(X, Z, Y) \wedge I(X, Z, W) \quad \Rightarrow \quad I(X, Z, Y)$$

by the decomposition axiom. In addition, we have

$$I(X, Z, Y \cup W) \quad \Rightarrow \quad I(X, Z \cup Y, W)$$

by weak union. The property stated in the lemma now follows directly. □

For graphoid independence relations we have that the intersection axiom can also be generalised to a bi-implication.

Lemma 3.1.6 *Let V be a set of statistical variables. Furthermore, let I be a graphoid independence relation on V. Then,*

$$I(X, Z \cup W, Y) \wedge I(X, Z \cup Y, W) \leftrightarrow I(X, Z, Y \cup W)$$

for all sets of variables $X, Y, Z, W \subseteq V$.

Proof. We will only prove that $I(X, Z, Y \cup W) \rightarrow I(X, Z \cup W, Y) \wedge I(X, Z \cup Y, W)$; the reverse property coincides with the intersection axiom and therefore trivially holds for the independence relation I. We have

$$I(X, Z, Y \cup W) \Rightarrow I(X, Z \cup W, Y)$$

and

$$I(X, Z, Y \cup W) \Rightarrow I(X, Z \cup Y, W)$$

by the weak union axiom. The property stated in the lemma now follows directly. □

In the sequel, we will use the phrase *independence relation* to denote a semi-graphoid independence relation, unless stated otherwise.

3.2 Graphical Representations of Independence

One of the problems in applying probability theory for automated reasoning with uncertainty in a knowledge-based system is the space complexity of representing a joint probability distribution. Since the concept of independence plays a key role in solving this problem, a formalism for representing joint probability distributions should allow for efficiently modeling independences. There are various ways of representing an independence relation. One way, for example, is to enumerate the separate statements of an independence relation explicitly. Such a representation clearly is impractical as the number of tuples in an independence relation can be astronomical. Another way is to make use of the axioms from Definition 3.1.3: only the statements from an appropriate subset of the independence relation are enumerated explicitly and all its other statements are defined implicitly by this set and the defining axioms. Although exploiting the axioms allows for a far more economical representation of an independence relation than explicit enumeration, it can still require exponential space. In this section, we consider more concise representations of independence relations, building on the idea of *graphical encoding*. In Section 3.2.1 we address modeling an independence relation in an undirected graph; in Section 3.2.2 we consider the representation of an independence relation in the formalism of directed graphs.

3.2.1 Undirected Graphs

Undirected graphs have no probabilistic meaning by themselves. For representing an independence relation in an undirected graph, therefore, a probabilistic meaning has to be assigned to the topological properties of such a graph, that is, we have to assign a probabilistic meaning to the vertices of the graph and to assign a probabilistic meaning to its edges. Informally speaking, we choose to encode an independence relation in an undirected graph by modelling the *variables* of the relation as *vertices* and by representing its *independence statements* by *absence of edges*. To formally capture this meaning, we begin by defining a graphical criterion for reading from a graph sets of vertices that allow for *blocking* all paths between two given sets of vertices; this graphical criterion is termed the *separation criterion* for undirected graphs.

Definition 3.2.1 *Let $G = (V(G), E(G))$ be an undirected graph. Let $X, Y, Z \subseteq V(G)$ be sets of vertices in G. The set of vertices Z is said to* separate *the sets of vertices X and Y in G, denoted as $\langle X \mid Z \mid Y\rangle_G$, if for each vertex $V_i \in X$ and each vertex $V_j \in Y$ every simple path from V_i to V_j in G contains at least one vertex from Z.*

We look upon a separating set of variables as effectively *blocking* influence: if a set of variables Z separates two sets of variables X and Y, then Z is looked upon as blocking any flow of information or influence between X and Y. Two sets of variables X and Y that are thus separated by a set Z now are taken to be conditionally independent given Z.

Definition 3.2.2 *Let V be a set of statistical variables and let I be an independence relation on V. Furthermore, let $G = (V(G), E(G))$ be an undirected graph with $V(G) = V$. Then,*

- *the graph G is called an* undirected dependence map, *or* D-map *for short, for I, if for all sets of variables $X, Y, Z \subseteq V$, we have: if $I(X, Z, Y)$ then $\langle X \mid Z \mid Y\rangle_G$;*

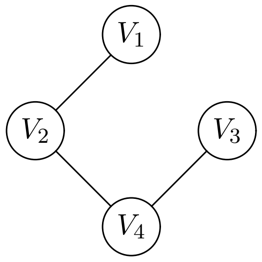


Figure 3.1: An Example Undirected D-map.

- *the graph G is called an* undirected independence map*, or* I-map *for short, for I, if for all sets of variables $X, Y, Z \subseteq V$, we have: if $\langle X \mid Z \mid Y\rangle_G$ then $I(X, Z, Y)$;*
- *the graph G is called an* undirected perfect map*, or* P-map *for short, for I, if G is both an undirected D-map and an undirected I-map for I.*

From the previous definition we have that in an undirected D-map for an independence relation any pair of neighbouring vertices represents a pair of variables that are dependent; however, not every pair of dependent variables needs be represented as a pair of neighbouring vertices. In an undirected D-map, therefore, a pair of non-neighbouring vertices may represent either a pair of dependent variables or a pair of independent variables.

Example 3.2.3 Let $V = \{V_1, V_2, V_3, V_4\}$ be a set of statistical variables. We consider the independence relation I on V that is defined by the independence statements $I(\{V_1\}, \{V_2, V_3\}, \{V_4\})$ and $I(\{V_2\}, \{V_1, V_4\}, \{V_3\})$. The undirected graph G shown in Figure 3.1 is an example of an undirected D-map for I, since for each independence statement $I(X, Z, Y)$, $X, Y, Z \subseteq V$, from the relation I, there exists a matching separation statement $\langle X \mid Z \mid Y\rangle_G$. For example, for the independence statement $I(\{V_1\}, \{V_2, V_3\}, \{V_4\})$ we find that the only path from V_1 to V_4 comprises the vertex V_2 which is included in the set $\{V_2, V_3\}$, that is, $\langle\{V_1\} \mid \{V_2, V_3\} \mid \{V_4\}\rangle_G$. □

In an undirected I-map, any pair of non-neighbouring vertices represents a pair of variables that are independent; however, not every pair of independent variables is represented as a pair of non-neighbouring vertices. In an undirected I-map, therefore, a pair of neighbouring vertices may represent either a pair of dependent variables or a pair of independent variables.

Example 3.2.4 Consider once more the independence relation I from Example 3.2.3. The graph G shown in Figure 3.2 is an example of an undirected I-map for the relation I, since for each separation statement $\langle X \mid Z \mid Y\rangle_G$, $X, Y, Z \subseteq V$, read from G, there exists a matching independence statement $I(X, Z, Y)$: for the single separation statement $\langle\{V_2\} \mid \{V_1, V_4\} \mid \{V_3\}\rangle_G$ read from the graph, we find the statement $I(\{V_2\}, \{V_1, V_4\}, \{V_3\})$ in the relation I. □

An undirected P-map for an independence relation faithfully represents all independences as well as all dependences from the relation.

Example 3.2.5 Consider once more the independence relation I from Example 3.2.3. The graph G shown in Figure 3.3 is the only undirected P-map for G. Note that G is an undirected D-map for I as well as an undirected I-map for I. □

From the above observations, we conclude that the various types of map provide for reading different types of probabilistic information from an undirected graph. If the graph is an undirected I-map for an independence relation, we can read independence statements from it as any separation statement is guaranteed to correspond with an independence statement; from an undirected D-map we can read dependence statements, and from an undirected P-map we can read both independence and dependence statements.

We now investigate the expressive power of the formalism of undirected graphs for representing independence relations.

Lemma 3.2.6 *For every independence relation, there exist an undirected D-map and an undirected I-map.*

Proof. We first show that for every independence relation there exists an undirected D-map. To this end, we show that the edgeless undirected graph $G = (V, \varnothing)$ is an undirected D-map for any independence relation I on the set of variables V. Since the graph G is edgeless, we have that the property: if $I(X, Z, Y)$ then $\langle X \mid Z \mid Y \rangle_G$, trivially holds for all sets of variables $X, Y, Z \subseteq V$. Informally speaking, as the graph G does not represent any dependences, it cannot represent any dependence incorrectly.

We now show that for every independence relation there exists an undirected I-map. To this end, we show that the complete undirected graph $G' = (V, V \times V)$ is an undirected I-map for any independence relation I on V. Since in this graph every pair of vertices is connected by an edge, we have that the property: if $\langle X \mid Z \mid Y \rangle_G$ then $I(X, Z, Y)$, trivially holds for all sets of variables $X, Y, Z \subseteq V$. Informally speaking, as the graph G' does not represent any independences, it cannot represent any independence incorrectly. $\square$

From the previous lemma, we have that every independence relation has an undirected D-map and an undirected I-map; in fact, an independence relation may have *several* undirected D-maps and I-maps. Unfortunately, a similar property does not hold for undirected P-maps: not every independence relation has an undirected P-map.

Example 3.2.7 Consider an independence relation I on a set of variables V such that $I(X, Z_1, Y)$ for some sets of variables $X, Y, Z_1 \subseteq V$ and $\neg I(X, Z_1 \cup Z_2, Y)$ for some set $Z_2 \subseteq V$ with $Z_2 \neq \varnothing$ and X, Y, Z_1, Z_2 are disjoint. The independence relation I is an example of an independence relation comprising an *induced dependence*: the sets of variables X and Y are independent given Z_1, but become dependent if information about Z_2 becomes available. Now, for any undirected graph G to be an undirected

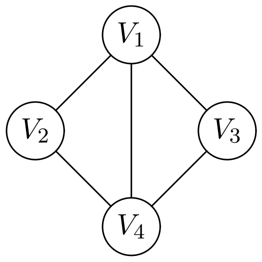


Figure 3.2: An Example Undirected I-map.

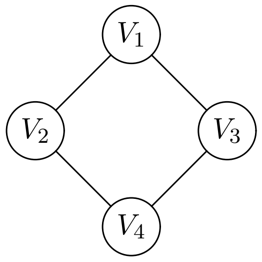


Figure 3.3: An Example Undirected P-map.

P-map for this independence relation I, it should be an undirected D-map as well as an undirected I-map for I. For G to be an undirected D-map for I, it should display the set Z_1 as a separating set for X and Y; on the other hand, for G to be an undirected I-map, it should display $Z_1 \cup Z_2$ as *not* separating X and Y. We observe, however, that for any undirected graph G the property: if $\langle X \mid Z_1 \mid Y \rangle_G$ then $\langle X \mid Z_1 \cup Z_2 \mid Y \rangle_G$ holds by definition. More in general, if two sets of vertices X and Y are separated by a third set of vertices Z in an undirected graph G, then X and Y are separated by any superset of Z in G as well. From this observation, it will be evident that no undirected graph can satisfy the two requirements mentioned above simultaneously and, hence, that there does not exist an undirected P-map for I. $\square$

Although not every independence relation has an undirected P-map, there are independence relations that can indeed be represented faithfully. An independence relation for which an undirected P-map exists is termed *undirected graph-isomorphic.*

Definition 3.2.8 *An independence relation I is said to be* undirected graph-isomorphic *if there exists an undirected graph G such that G is an undirected P-map for I.*

We would like to note that, if an independence relation is undirected graph-isomorphic, then it allows one and only one undirected P-map. To conclude, we would like to mention that a necessary and sufficient condition has been identified for an independence relation to be undirected graph-isomorphic [Pearl, 1988]; this condition allows for testing whether a given independence relation lends itself to representation in an undirected graph. Here, we will not elaborate any further on this observation.

3.2.2 Directed Graphs

In the previous section, we have addressed the representation of an independence relation in the formalism of undirected graphs. We have seen that not every independence relation can be represented faithfully in this formalism. In this section, we investigate the formalism of *directed graphs (digraphs)* and its expressive power for representing independence relations; we will consider *acyclic* digraphs only.

Just like undirected graphs, do directed graphs not have a probabilistic meaning by themselves and therefore have to be assigned one. For this purpose, we build on the same idea as outlined in the previous section for undirected graphs. We once more model the *variables* of an independence relation as *vertices* in a digraph and represent its *independence statements* by *absence of arcs.* For undirected graphs, this basic idea has been formalised in the separation criterion. For digraphs, we will now formulate a similar criterion, called the *d-separation criterion.* Before defining this criterion, however, we introduce the concept of *blocking* influence among variables. The definition provided here is an enhancemnet of the original definition by Pearl, based upon more recent insights [Van der Gaag & Meyer, 1998].

Definition 3.2.9 *Let $G = (V(G), A(G))$ be an acyclic digraph and let s be a chain in G between $V_i \in V(G)$ and $V_j \in V(G)$. The chain s is* blocked *by a (possibly empty) set of vertices $W \subseteq V(G)$ if $V_i \in W$ or $V_j \in W$, or s contains three consecutive vertices X_1, X_2, X_3, for which (at least) one of the following conditions hold:*

a. arcs (X_2, X_1) and (X_2, X_3) are on the chain s, and $X_2 \in W$;

b. arcs (X_1, X_2) and (X_2, X_3) are on the chain s, and $X_2 \in W$;

c. arcs (X_1, X_2) and (X_3, X_2) are on the chain s, and $\sigma^*(X_2) \cap W = \varnothing$.

In defining the concept of a *blocked chain*, we have distinguished three graphical conditions. Figure 3.4 serves as a reference for these conditions; in the two chains representing the conditions *b.* and *c.*, vertex X_2 is drawn with shading to indicate that it is comprised in the blocking set W for the chain at hand.

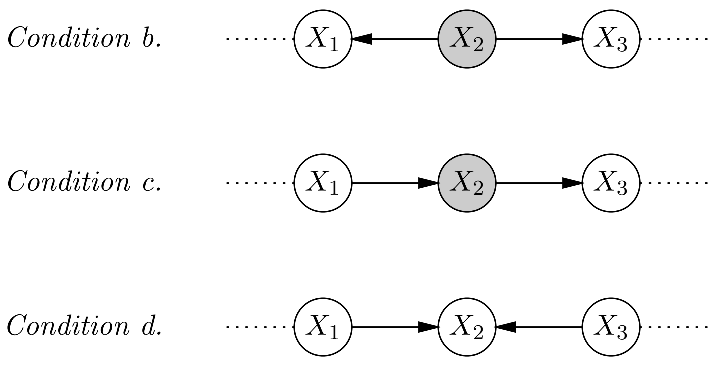


Figure 3.4: Chain Blocking; Shaded Vertices are Members of Blocking-set W.

Building on the concept of blocking of single chains, we now define the d-separation criterion for reading from a digraph sets of vertices that allow for blocking *all* chains between two given sets of vertices.

Definition 3.2.10 *Let $G = (V(G), A(G))$ be an acyclic digraph. Let $X, Y, Z \subseteq V(G)$ be sets of vertices in G. The set of vertices Z is said to* d-separate *the sets of vertices X and Y in G, denoted as $\langle X \mid Z \mid Y \rangle^d_G$, if for each vertex $V_i \in X$ and each vertex $V_j \in Y$ every chain from V_i to V_j in G is blocked by Z.*

We would like to note that in assigning a meaning to the topological properties of a directed graph, we want to distinguish between conditional independences and dependences, that is, between *two* alternatives only. The formalism of directed graphs, however, allows for distinguishing between *three* alternatives since there are three different ways in which two arcs between three vertices can be directed (up to renaming of vertices). This observation accounts for two conditions of the concept of blocking having been assigned the same meaning; these are the first two conditions depicted in Figure 3.4. Note that the third condition models an induced dependence.

The following definition now relates the d-separation criterion to the concept of independence.

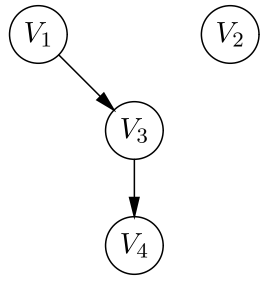


Figure 3.5: An Example Directed D-map.

Definition 3.2.11 *Let V be a set of statistical variables and let I be an independence relation on V. Furthermore, let $G = (V(G), A(G))$ be an acyclic digraph with $V(G) = V$. Then,*

- *the graph G is called a* directed dependence map*, or* D-map *for short, for I, if for all sets of variables $X, Y, Z \subseteq V$, we have: if $I(X, Z, Y)$ then $\langle X \mid Z \mid Y \rangle^d_G$;*
- *the graph G is called a* directed independence map*, or* I-map *for short, for I, if for all sets of variables $X, Y, Z \subseteq V$, we have: if $\langle X \mid Z \mid Y \rangle^d_G$ then $I(X, Z, Y)$;*
- *the graph G is called a* directed perfect map*, or* P-map *for short, for I, if G is both a directed D-map and a directed I-map for I.*

The different types of directed map have the same meaning as the different types of undirected map we discerned in Section 3.2.1. In a directed D-map for an independence relation, any pair of neighbouring vertices again represents a pair of dependent variables; however, not every pair of dependent variables needs be represented as a pair of neighbouring vertices.

Example 3.2.12 Let $V = \{V_1, V_2, V_3, V_3\}$ be a set of statistical variables. We consider the independence relation I on V that is defined by the independence statements $I(\{V_1\}, \varnothing, \{V_2\})$ and $I(\{V_1, V_2\}, \{V_3\}, \{V_4\})$. The digraph G shown in Figure 3.5 is an example of a directed D-map for I, since for each independence statement $I(X, Z, Y)$, $X, Y, Z \subseteq V$, from the relation I, there exists a matching d-separation statement $\langle X \mid Z \mid Y \rangle^d_G$. For example, for the independence statement $I(\{V_1\}, \varnothing, \{V_2\})$, we find that $\langle \{V_1\} \mid \varnothing \mid \{V_2\} \rangle^d_G$ as there does not exist any chain between the vertices V_1 and V_2 in G. $\square$

In a directed I-map, any pair of non-neighbouring vertices once more represents a pair of variables that are independent; however, not every pair of independent variables is represented as a pair of non-neighbouring vertices.

Example 3.2.13 Consider once more the independence relation I from Example 3.2.12. The digraph G shown in Figure 3.6 is an example of a directed I-map for the relation I, since for each d-separation statement $\langle X \mid Z \mid Y \rangle^d_G$, $X, Y, Z \subseteq V$, read from G, there exists a matching independence statement $I(X, Z, Y)$ in I. For example, for the d-separation statement $\langle \{V_1\} \mid \{V_3\} \mid \{V_4\} \rangle^d_G$ read from the digraph, we find the statement $I(\{V_1\}, \{V_3\}, \{V_4\})$ in the independence relation. $\square$

A directed P-map for an independence relation once more faithfully captures all independences and dependences from the relation.

Example 3.2.14 Consider once more the independence relation I from Example 3.2.12. The digraph G shown in Figure 3.7 is a directed P-map for G. Note that G is a directed D-map as well as a directed I-map for I. □

From the above observations, we once more conclude that, if a digraph is a directed I-map for an independence relation, we can read independence statements from it; from a directed D-map we can read dependence statements, and from a directed P-map we can read both independence and dependence statements.

Just as we have done for undirected graphs, we investigate the expressive power of the formalism of directed graphs for representing independence relations. It can easily be shown that every independence relation has a directed D-map and a directed I-map.

Lemma 3.2.15 *For every independence relation, there exist a directed D-map and a directed I-map.*

Unfortunately, a similar property does not hold for directed P-maps: not every independence relation has a directed P-map.

Example 3.2.16 Consider an independence relation I on a set of variables V such that $I(\{V_1\}, \{V_2, V_3\}, \{V_4\})$ and $I(\{V_2\}, \{V_1, V_4\}, \{V_3\})$ for some variables $V_1, V_2, V_3, V_4 \in V$. No acyclic digraph can model these independence statements simultaneously. □

Although not every independence relation has a directed P-map, there are independence relations that can indeed be represented faithfully. An independence relation for which a directed P-map exists is termed *directed graph-isomorphic.*

Definition 3.2.17 *An independence relation I is said to be* directed graph-isomorphic *if there exists an acyclic digraph G such that G is a directed P-map for I.*

We would like to note that, if an independence relation is directed graph-isomorphic, it may allow several different directed P-maps, that is, a directed P-map does not need to be unique. To conclude, we would like to mention that a necessary condition has been identified for an independence relation to be directed graph-isomorphic [Pearl, 1988]. Here, we will not elaborate any further on this observation.

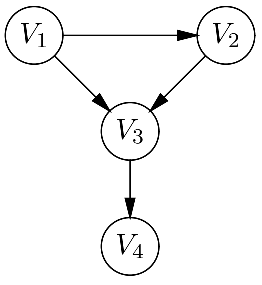


Figure 3.6: An Example Directed I-map.

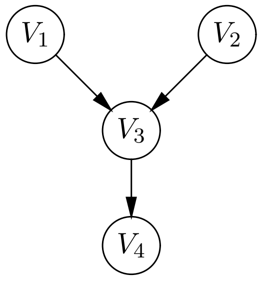


Figure 3.7: An Example Directed P-map.

3.2.3 Choosing a Graphical Representation

In the previous sections we have investigated two different graphical formalisms for representing independence relations: the formalism of undirected graphs and the formalism of directed graphs. Both formalisms allow for a *space-efficient* representation of an independence relation: all graphical representations considered are polynomial in terms of the number of variables of the relation at hand. In addition, the two formalisms provide for an *explicit* representation of independence that allows for efficiently verifying independence statements without requiring numerical computations. These properties render graphical formalisms highly suitable for representing independence relations in knowledge-based systems. Unfortunately, while the graphical formalisms allow for capturing some independence relations to accuracy, they do not allow for a faithful representation for every such relation. The correspondence between independence relations on the one hand and graphical representations on the other hand is shown schematically in Figure 3.8. This figure depicts that there are independence relations that are undirected graph-isomorphic yet not directed graph-isomorphic, and vice versa; also, there are independence relations that are both undirected and directed graph-isomorphic and independence relations that are not graph-isomorphic at all. This property negatively affects the suitability of graphical formalisms for representing independence relations. The efficiency of representation, however, generally is considered to outweigh the lack of expressive power of these formalisms. We would like to note that it is possible to mix directed and undirected graphs; such mixed graphs are called *chain graphs* [Studený, 1998] and are capable of representing a broader class of independence relations. Chain graphs are, however, hardly used for practical applications as a result of their more complex semantics.

In the probabilistic network framework, therefore, a graphical formalism is used for representing independences. As experience learns that the independence relations that are typically encountered in practical problem domains often are best represented by a directed graph rather than by an undirected graph, the formalism of directed graphs is employed in the framework. There does exist an undirected counterpart of the probabilistic network framework, called a *Markov network*.

In real-life problem domains, independence relations may be encountered that are not graph-isomorphic. For such an independence relation, it is not possible to faithfully represent all independences as well as all dependences. The relation therefore has to be represented in either a directed I-map or a directed D-map. Now observe that since we are interested in exploiting *in*dependences for simplifying computations, we have to make sure that independences that can be read from the graphical representation of an independence relation actually do hold in the relation at hand; otherwise, we would

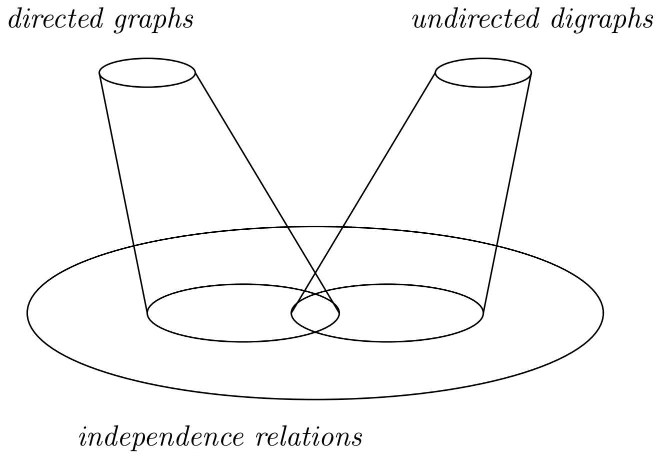


Figure 3.8: Independence Relations and Graphical Representations.

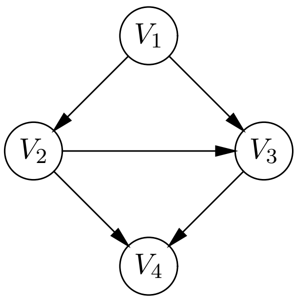


Figure 3.9: An Example Minimal Directed I-map.

assume independences where there are none and thereby introduce errors in inference. An independence relation that is not directed graph-isomorphic therefore is best represented by a directed *I-map*. Furthermore, acknowledging that some independences will escape representation, we settle for a representation in which *as many* of the independences of the relation as possible are modelled, that is, we insist that the number of unrepresented independences is kept at a minimum. A directed I-map that does not contain any superfluous arcs is called *minimal*.

Definition 3.2.18 *Let V be a set of statistical variables and let I be an independence relation on V. Furthermore, let $G = (V(G), A(G))$ be an acyclic digraph with $V(G) = V$. Then, the digraph G is called a* minimal *directed I-map for I if G is a directed I-map for I and no proper subgraph of G is a directed I-map for I.*

We would like to note that an independence relation may allow several minimal I-maps, that is, a minimal directed I-map need not be unique.

Example 3.2.19 Let $V = \{V_1, V_2, V_3, V_4\}$ be a set of statistical variables. We consider once more the independence relation I on V that is defined by the statements $I(\{V_1\}, \{V_2, V_3\}, \{V_4\})$ and $I(\{V_2\}, \{V_1, V_4\}, \{V_3\})$. In Example 3.2.16 we have argued that the relation I is not directed graph-isomorphic. The digraph G shown in Figure 3.9 now is an example of a minimal directed I-map for I. Note that, whereas the independence statement $I(\{V_1\}, \{V_2, V_3\}, \{V_4\})$ is portrayed by G, the statement $I(\{V_2\}, \{V_1, V_4\}, \{V_3\})$ has escaped representation. □

Exercises

Exercise 3.1

Let V be a set of statistical variables. Let $\Pr$ *be a joint probability distribution on V and let $I_{\Pr}$ be its independence relation. Show that $I_{\Pr}$ satisfies the properties*

* *a.* $I_{\Pr}(X, Z, Y) \rightarrow I_{\Pr}(Y, Z, X)$*;*

* *b.* $I_{\Pr}(X, Z, Y \cup W) \rightarrow I_{\Pr}(X, Z, Y) \wedge I_{\Pr}(X, Z, W)$*;* (quite difficult!)

* c. $I_{\Pr}(X, Z, Y \cup W) \rightarrow I_{\Pr}(X, Z \cup W, Y)$;

d. $I_{\Pr}(X, Z, Y) \wedge I_{\Pr}(X, Z \cup Y, W) \rightarrow I_{\Pr}(X, Z, Y \cup W)$;

for all sets of variables $X, Y, Z, W \subseteq V$.

* Exercise 3.2

Let V be a set of statistical variables and let I be a semi-graphoid independence relation on V. Show that

$$I(X, Z, Y \cup W) \wedge I(Y, Z, W) \rightarrow I(X \cup W, Z, Y)$$

for all sets of variables $X, Y, Z, W \subseteq V$.

* Exercise 3.3

Let V be a set of statistical variables and let I be a semi-graphoid independence relation on V. Show that

$$I(X, Y \cup Z, U \cup W) \wedge I(Y, Z \cup U, X) \rightarrow I(X, Z \cup U, Y \cup W)$$

for all sets of variables $X, Y, Z, U, W \subseteq V$.

Exercise 3.4

Let $V = \{V_1, V_2, V_3, V_4\}$ be a set of statistical variables. Furthermore, let I be the following (semi-graphoid) independence relation on V:

$$\begin{array}{lll}
I(\{V_1\}, \varnothing, \{V_4\}) & I(\{V_1\}, \varnothing, \{V_2\}) & I(\{V_1, V_3\}, \{V_2\}, \{V_4\}) \\
I(\{V_2\}, \varnothing, \{V_4\}) & I(\{V_2\}, \varnothing, \{V_1\}) & I(\{V_4\}, \{V_2\}, \{V_1\}) \\
I(\{V_3\}, \varnothing, \{V_4\}) & I(\{V_1, V_4\}, \varnothing, \{V_2\}) & I(\{V_4\}, \{V_2\}, \{V_3\}) \\
I(\{V_4\}, \varnothing, \{V_1\}) & I(\{V_2\}, \varnothing, \{V_1, V_4\}) & I(\{V_4\}, \{V_2\}, \{V_1, V_3\}) \\
I(\{V_4\}, \varnothing, \{V_2\}) & I(\{V_2, V_4\}, \varnothing, \{V_1\}) & I(\{V_1\}, \{V_3\}, \{V_4\}) \\
I(\{V_4\}, \varnothing, \{V_3\}) & I(\{V_1\}, \varnothing, \{V_2, V_4\}) & I(\{V_2\}, \{V_3\}, \{V_4\}) \\
I(\{V_1, V_2\}, \varnothing, \{V_4\}) & I(\{V_2\}, \{V_1\}, \{V_4\}) & I(\{V_1, V_2\}, \{V_3\}, \{V_4\}) \\
I(\{V_1, V_3\}, \varnothing, \{V_4\}) & I(\{V_3\}, \{V_1\}, \{V_4\}) & I(\{V_1\}, \{V_4\}, \{V_2\}) \\
I(\{V_2, V_3\}, \varnothing, \{V_4\}) & I(\{V_2, V_3\}, \{V_1\}, \{V_4\}) & I(\{V_2\}, \{V_4\}, \{V_1\}) \\
I(\{V_4\}, \varnothing, \{V_1, V_2\}) & I(\{V_4\}, \{V_1\}, \{V_2\}) & I(\{V_3\}, \{V_1, V_2\}, \{V_4\}) \\
I(\{V_4\}, \varnothing, \{V_1, V_3\}) & I(\{V_4\}, \{V_1\}, \{V_3\}) & I(\{V_4\}, \{V_1, V_2\}, \{V_3\}) \\
I(\{V_4\}, \varnothing, \{V_2, V_3\}) & I(\{V_4\}, \{V_1\}, \{V_2, V_3\}) & I(\{V_2\}, \{V_1, V_3\}, \{V_4\}) \\
I(\{V_1, V_2, V_3\}, \varnothing, \{V_4\}) & I(\{V_1\}, \{V_2\}, \{V_4\}) & I(\{V_4\}, \{V_1, V_3\}, \{V_2\}) \\
I(\{V_4\}, \varnothing, \{V_1, V_2, V_3\}) & I(\{V_3\}, \{V_2\}, \{V_4\}) & I(\{V_1\}, \{V_2, V_3\}, \{V_4\}) \\
 & & I(\{V_4\}, \{V_2, V_3\}, \{V_1\})
\end{array}$$

Show that each statement $I(X, Z, Y)$, $X, Y, Z \subseteq V$, of the independence relation I can be derived from the statements $I(\{V_1, V_2, V_3\}, \varnothing, \{V_4\})$ and $I(\{V_1\}, \varnothing, \{V_2\})$ by the independence axioms from Definition 3.1.3.

* Exercise 3.5

Let $V = \{V_1, V_2, V_3, V_4\}$ be a set of statistical variables. Let I be the independence relation on V that is defined by the statements $I(\{V_1\}, \{V_2, V_3\}, \{V_4\})$ and $I(\{V_2\}, \{V_1, V_4\}, \{V_3\})$.

a. Give all undirected D-maps for the independence relation I;

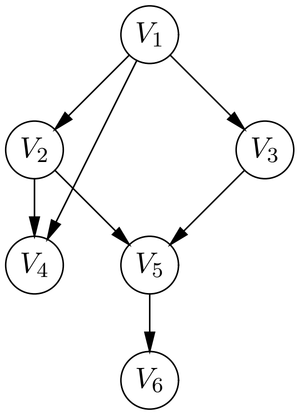
b. Give all undirected I-maps for the independence relation I.

Exercise 3.6

Recall that ν_G is the neighbour set(see Definition 2.1.2). Show that for any undirected graph $G = (V(G), E(G))$ the following property holds: for any vertex $V_i \in V(G)$ and any vertex $V_j \in V(G) \setminus (\{V_i\} \cup \nu_G(V_i))$, we have that $\langle \{V_i\} \mid \nu_G(V_i) \mid \{V_j\} \rangle_G$.

* Exercise 3.7

Let G be the following acyclic digraph:



Examine for each of the following statements whether or not it holds in G:

a. $\langle \{V_1\} \mid \{V_2, V_3\} \mid \{V_6\} \rangle^d_G$;

b. $\langle \{V_2\} \mid \varnothing \mid \{V_3\} \rangle^d_G$;

c. $\langle \{V_2\} \mid \{V_1\} \mid \{V_3\} \rangle^d_G$;

d. $\langle \{V_4\} \mid \{V_1\} \mid \{V_3\} \rangle^d_G$;

e. $\langle \{V_2\} \mid \{V_3, V_4\} \mid \{V_6\} \rangle^d_G$;

f. $\langle \{V_3\} \mid \varnothing \mid \{V_1\} \rangle^d_G$.

*** Exercise 3.8**

Let $V = \{V_1, V_2, V_3, V_4\}$ be a set of statistical variables. Let I be the independence relation on V that is defined by the statements $I(\{V_1\}, \varnothing, \{V_2\})$ and $I(\{V_1, V_2\}, \{V_3\}, \{V_4\})$.

a. Give some directed D-maps for the independence relation I;

b. Give some directed I-maps for the independence relation I.

*** Exercise 3.9**

Show that for every independence relation there exists a directed D-map and a directed I-map.

*** Exercise 3.10**

Given an example of an independence relation that has more than one directed P-map.

Exercise 3.11

Let $V = \{V_1, V_2, V_3, V_4\}$ be a set of statistical variables. Let I be the independence relation on V that is defined by the statements $I(\{V_1\}, \{V_2, V_3\}, \{V_4\})$ and $I(\{V_2\}, \{V_1, V_4\}, \{V_3\})$. Give some minimal directed I-maps for the relation I.

Exercise 3.12

Show that for any acyclic directed graph $G = (V(G), A(G))$ the following property holds: for any vertex $V_i \in V(G)$ and any vertex $V_j \in V(G) \setminus (\sigma^*_G(V_i) \cup \rho_G(V_i))$, we have that $\langle\{V_i\} \mid \rho_G(V_i) \mid \{V_j\}\rangle^d_G$.

*** Exercise 3.13**

Let V be a set of statistical variables. Let I be an independence relation on V and let G be a directed I-map for I. Now, let H be the underlying graph of G. Is H an undirected I-map for I ?

*** Exercise 3.14**

a. Give an example of an independence relation that has both an undirected P-map and a directed P-map.

b. Give an example of an independence relation that has an undirected P-map but no directed P-map.

c. Give an example of an independence relation that has a directed P-map but no undirected P-map.

d. Give an example of an independence relation that has no undirected P-map nor a directed P-map.

Chapter 4

The Probabilistic Network Framework

This chapter defines the concept of probabilistic network and reconstructs the algorithm for exact probabilistic inference as designed by Judea Pearl; for ease of exposition only binary variables are considered. Other, and often more efficient, algorithms for exact inference exist such as *jointree propagation* (S.L. Lauritzen, D.J. Spiegelhalter, F.V. Jensen, P.P Shenoy, G. Shafer) and *variable elimination* (R. Dechter, G.F. Cooper) methods. The reason for discussing Pearl's algorithm is that it explicitly exploits the graphical structure of the network without transforming it, and therefore seems the one easiest to explain and comprehend.

Current research focuses on finding still more efficient algorithms, both for exact and approximate inference, and also for dealing with networks that include continuous variables. Also, now and again, researchers look into better algorithms for loop cutset conditioning (A. Becker, D. Geiger), the extension to Pearl's algorithm that is required to do inference in multiply connected graphs.

The probabilistic network framework is characterised by a powerful and intuitively appealing formalism for representing a joint probability distribution on a set of statistical variables, for use in a knowledge-based system. Associated with this formalism are algorithms for efficiently computing probabilities of interest and for processing evidence; these algorithms constitute the basic building blocks for reasoning with knowledge represented in the formalism. In this chapter we introduce the probabilistic network formalism and detail some of the most well-known associated algorithms.

4.1 The Probabilistic Network Formalism

The probabilistic network formalism provides for a concise representation of a joint probability distribution on a set of statistical variables; such a representation is called a *Bayesian belief network*, a *belief network* for short, or a *probabilistic network*. The conciseness of representation is arrived at by explicit separation of knowledge of the independences holding in a distribution and the numerical quantities involved. To this end, a probabilistic network comprises two parts: a *qualitative* part and a *quantitative* part. The qualitative part of a probabilistic network is a graphical representation of the independences holding among the variables in the probability distribution that is

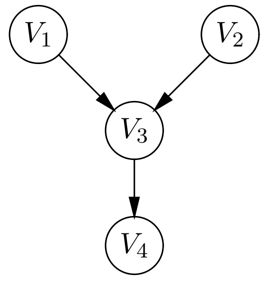


Figure 4.1: The Digraph of an Example Probabilistic Network.

being represented; more in specific, the qualitative part of a probabilistic network is a (minimal) directed I-map for the independence relation of the distribution. Associated with the qualitative part of a probabilistic network is a set of functions representing numerical quantities from the distribution; with each vertex in the digraph is associated a *probability assessment function* which basically is a set of (conditional) probabilities describing the influence of the values of the vertex' predecessors on the probabilities of the values of this vertex itself. These probability assessment functions with each other constitute the quantitative part of the probabilistic network.

We define the concept of a probabilistic network more formally.

Definition 4.1.1 *A* probabilistic network *is a tuple $B = (G, \Gamma)$ where*

- *$G = (V(G), A(G))$ is an acyclic digraph with vertices $V(G) = \{V_1, \ldots, V_n\}$, $n \geq 1$, and arcs $A(G)$;*
- *$\Gamma = \{\gamma_{V_i} \mid V_i \in V(G)\}$ is a set of real-valued non-negative functions*

  $$\gamma_{V_i} \colon \{c_{V_i}\} \times \{c_{\rho_G(V_i)}\} \to [0, 1]$$

  called (conditional) probability assessment functions*, such that for each configuration $c_{\rho_G(V_i)}$ of the set $\rho_G(V_i)$ of (immediate) predecessors of vertex V_i in G, we have that $\gamma_{V_i}(\neg v_i \mid c_{\rho_G(V_i)}) = 1 - \gamma_{V_i}(v_i \mid c_{\rho_G(V_i)})$, $i = 1, \ldots, n$.*

Note that in the previous definition V_i is viewed as a vertex from the digraph and as a statistical variable, alternatively.

Example 4.1.2 We consider the digraph G shown in Figure 4.1. With the digraph G, we associate a set $\Gamma = \{\gamma_{V_i} \mid i = 1, \ldots, 4\}$ of probability assessment functions γ_{V_i}. For example, for the vertices V_1 and V_2, the probability assessment functions γ_{V_1} and γ_{V_2} may be defined as

$$\begin{array}{llcll} \gamma_{V_1}(v_1) & = 0.25 & \text{and} & \gamma_{V_2}(v_2) & = 0.5 \\ \gamma_{V_1}(\neg v_1) & = 0.75 & & \gamma_{V_2}(\neg v_2) & = 0.5 \end{array}$$

For vertex V_4, the probability assessment function γ_{V_4} is defined as

$$\begin{array}{llll} \gamma_{V_4}(v_4 \mid v_3) & = 0.8 & \gamma_{V_4}(\neg v_4 \mid v_3) & = 0.2 \\ \gamma_{V_4}(v_4 \mid \neg v_3) & = 0 & \gamma_{V_4}(\neg v_4 \mid \neg v_3) & = 1.0 \end{array}$$

For vertex V_3, the probability assessment function γ_{V_3} is defined by the values

$$\begin{aligned}
\gamma_{V_3}(v_3 \mid v_1 \wedge v_2) &= 0.75 \\
\gamma_{V_3}(v_3 \mid \neg v_1 \wedge v_2) &= 0.4 \\
\gamma_{V_3}(v_3 \mid v_1 \wedge \neg v_2) &= 0.25 \\
\gamma_{V_3}(v_3 \mid \neg v_1 \wedge \neg v_2) &= 0.2
\end{aligned}$$

and their *complementary* values

$$\begin{aligned}
\gamma_{V_3}(\neg v_3 \mid v_1 \wedge v_2) &= 0.25 \\
\gamma_{V_3}(\neg v_3 \mid \neg v_1 \wedge v_2) &= 0.6 \\
\gamma_{V_3}(\neg v_3 \mid v_1 \wedge \neg v_2) &= 0.75 \\
\gamma_{V_3}(\neg v_3 \mid \neg v_1 \wedge \neg v_2) &= 0.8
\end{aligned}$$

The pair $B = (G, \Gamma)$ is a probabilistic network. □

The probability assessment functions of a probabilistic network with each other provide all information necessary for uniquely defining a joint probability distribution on the variables discerned that respects the independence relation portrayed by the qualitative part of the network; henceforth, we will call this distribution the joint probability distribution *defined* by the network.

Proposition 4.1.3 *Let $B = (G, \Gamma)$ be a probabilistic network. Then,*

$$\Pr(C_{V(G)}) = \prod_{V_i \in V(G)} \gamma_{V_i}(V_i \mid C_{\rho_G(V_i)})$$

defines a joint probability distribution Pr *on $V(G)$ such that G is a directed I-map for the independence relation $I_{\Pr}$ of* Pr*.*

Proof. Since the digraph G of the probabilistic network B is acyclic, it allows a total ordering of its vertices such that any successor of a vertex in the digraph follows it in the ordering; such an ordering is termed a *topological order* of the digraph's vertices. Note that any topological order of the vertices of G constitutes an ordering of the corresponding statistical variables. We take ι_G to be such a topological order; for ease of exposition, we assume that $\iota_G(V_i) = i$ for all $V_i \in V(G)$. We now consider an arbitrary joint probability distribution P on $V(G)$ such that G is a directed I-map for the independence relation of P. To the expression $P(C_{V(G)})$ describing the distribution P, we apply the chain rule from probability theory, such that every variable V_i from $V(G)$ is conditioned on the variables $V_1, \ldots, V_{i-1}$, preceding it in the ordering ι_G, that is,

$$P(C_{V(G)}) = \prod_{V_i \in V(G)} P(V_i \mid V_1 \wedge \cdots \wedge V_{i-1})$$

From the digraph G, we read, by means of the d-separation criterion, that $\langle \{V_i\} \mid \rho_G(V_i) \mid \{V_1, \ldots, V_{i-1}\} \setminus \rho_G(V_i)\rangle_G^d$ for every vertex $V_i \in V(G)$. Since G is a directed I-map for the distribution P, we have for every V_i that $I_P(\{V_i\}, \rho_G(V_i), \{V_1, \ldots, V_{i-1}\} \setminus \rho_G(V_i))$. From this observation, we have that

$$P(V_i \mid V_1 \wedge \cdots \wedge V_{i-1}) = P(V_i \mid C_{\rho_G(V_i)})$$

for every vertex $V_i \in V(G)$. By exploiting this property, we find that

$$P(C_{V(G)}) = \prod_{V_i \in V(G)} P(V_i \mid C_{\rho_G(V_i)})$$

Since P has been chosen arbitrarily, we conclude that any joint probability distribution such that G is a directed I-map for its independence relation, satisfies this property. By reversing the argument, we find that there exists a (unique) joint probability distribution Pr on $V(G)$ such that G is a directed I-map for the independence relation of Pr and $\Pr(V_i \mid C_{\rho_G(V_i)}) = \gamma_{V_i}(V_i \mid C_{\rho_G(V_i)})$ for each variable $V_i \in V(G)$. □

We illustrate the basic idea of the previous proposition by means of an example.

Example 4.1.4 We consider once more the probabilistic network $B = (G, \Gamma)$ from Example 4.1.2. From the property stated in Proposition 4.1.3, we have that from the expression

$$\Pr(V_1 \wedge V_2 \wedge V_3 \wedge V_4) \quad = \gamma_{V_1}(V_1) \cdot \gamma_{V_2}(V_2) \cdot \gamma_{V_3}(V_3 \mid V_1 \wedge V_2) \cdot \gamma_{V_4}(V_4 \mid V_3)$$

any probability of interest can be computed. For example, we have that

$$\begin{aligned}\Pr(v_1 \wedge \neg v_2 \wedge v_3 \wedge v_4) &= \gamma_{V_1}(v_1) \cdot \gamma_{V_2}(\neg v_2) \cdot \gamma_{V_3}(v_3 \mid v_1 \wedge \neg v_2) \cdot \gamma_{V_4}(v_4 \mid v_3) = \\ &= 0.25 \cdot 0.5 \cdot 0.25 \cdot 0.8 = 0.025\end{aligned}$$

and

$$\begin{aligned}\Pr(\neg v_1 \wedge \neg v_2 \wedge v_3 \wedge v_4) &= \gamma_{V_1}(\neg v_1) \cdot \gamma_{V_2}(\neg v_2) \cdot \gamma_{V_3}(v_3 \mid \neg v_1 \wedge \neg v_2) \cdot \gamma_{V_4}(v_4 \mid v_3) = \\ &= 0.06\end{aligned}$$

By marginalisation we can now compute the probability $\Pr(\neg v_2 \wedge v_3 \wedge v_4)$:

$$\begin{aligned}\Pr(\neg v_2 \wedge v_3 \wedge v_4) &= \Pr(v_1 \wedge \neg v_2 \wedge v_3 \wedge v_4) + \Pr(\neg v_1 \wedge \neg v_2 \wedge v_3 \wedge v_4) = \\ &= 0.085\end{aligned}$$

□

From Proposition 4.1.3 it is readily seen that the function values of a probabilistic network's probability assessment functions can be interpreted as conditional probabilities.

Corollary 4.1.5 *Let $B = (G, \Gamma)$ be a probabilistic network and let* Pr *be the joint probability distribution defined by B. Then, for each vertex $V_i \in V(G)$, we have that*

$$\Pr(V_i \mid C_{\rho_G(V_i)}) = \gamma_{V_i}(V_i \mid C_{\rho_G(V_i)})$$

4.2 Probabilistic Inference

A probabilistic network generally is used for *probabilistic inference*, that is, for making probabilistic statements concerning the variables that are represented in the network. For this purpose, the property stated in Proposition 4.1.3 can be exploited: from the proposition we have that, since the digraph of a probabilistic network and its associated probability assessment functions with each other uniquely define a joint probability distribution, any (prior or posterior) probability of interest can be computed from the network. Explicitly generating the represented probability distribution as indicated by the proposition and then using the basic rules of marginalisation and conditioning, however, is computationally infeasible. In addition, such a straightforward approach would not exploit the independences that are represented by the qualitative part of the network at hand. More efficient algorithms for probabilistic inference have been designed that do exploit the represented independences. The best known of these are the algorithm of J. Pearl [Pearl, 1988] and the algorithm of S.L. Lauritzen and D.J. Spiegelhalter [Lauritzen & Spiegelhalter, 1988]. In the sequel, we will focus on Pearl's algorithm.

The basic idea of Pearl's algorithm for probabilistic inference is best explained from an object-centered point of view. The digraph of a probabilistic network is looked upon as a *computational architecture*: the vertices of the digraph are *autonomous objects* and its arcs are *bi-directional communication channels.* Each vertex has a local processor that is capable of performing simple probabilistic computations and a local memory in which its associated probability assessment function is stored. Through the communication channels the vertices send each other *parameters* providing information about the joint probability distribution that is represented by the network — that is, information determined from the network's probability assessments and the independences portrayed in the digraph — and about the evidence that has been entered and processed so far. Each vertex is now able to compute the probabilities of its values from its own probability assessment function and the information it receives from its neighbours.

Initially, a probabilistic network is in an *equilibrium* state: recomputation of the various parameters that the vertices send one another, does not result in a change in any of them. Now, when a piece of evidence is entered into the network for some vertex, this equilibrium is perturbed, since the probability distribution over the network's variables under consideration has now changed: the distribution should be conditioned on the (additional) evidence obtained. Once the value of a variable is known with certainty, the probability distribution for the variable degenerates and can no longer change; entering evidence for a variable into a probabilistic network thus basically amounts to adding its corresponding vertex to the (initially empty) blocking-set W for the digraph, thereby changing the independences that have to be taken into account by the inference algorithm. Therefore, to process the evidence entered for some vertex, the parameters this vertex sends to its neighbours are updated. After receiving updated parameters, these neighbours in turn compute new parameters to send to their neighbours, and so on. The impact of the evidence thus spreads throughout the network by *message-passing* between neighbouring vertices.

In this section, we detail the computations involved in Pearl's algorithm. In doing so, we distinguish between various types of graph. In Section 4.2.1, we focus on directed trees. In Section 4.2.2 we address singly connected digraphs more in general. In Section 4.2.3 we discuss probabilistic inference with probabilistic networks comprising a multiply connected digraph.

4.2.1 Directed Trees

In discussing Pearl's algorithm, we begin by focusing on probabilistic inference with a probabilistic network comprising a *directed tree* for its qualitative part, that is, in the network's digraph a vertex may have several successors but at most one predecessor. Before detailing the computations involved in inference with such a network, we intro-

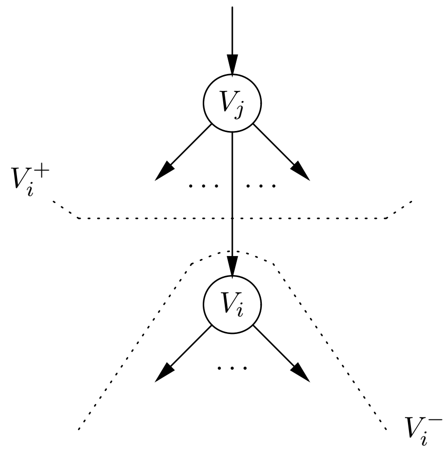


Figure 4.2: A Fragment of a Directed Tree.

duce some more terminology and notational convention that we will use in the sequel. Let G be the digraph of a probabilistic network, and let V be (a subset of) its set of vertices. A vertex $V_i \in V$ is called *instantiated* if its value is known with certainty; otherwise, it is called *uninstantiated.* Now, let $X \subseteq V$ be the set of instantiated vertices from V. The configuration c_X of X that is known with certainty is called a *partial configuration* of V and will be denoted as $\tilde{c}_V$. Note that the notation $\tilde{c}_V$ provides for referring to the subset of instantiated vertices in a probabilistic network without having to specify this subset explicitly.

At any time during probabilistic inference with a probabilistic network, the probabilities of the values of a vertex of interest are dependent upon all evidence entered so far into the network.

Lemma 4.2.1 *Let $B = (G, \Gamma)$ be a probabilistic network where $G = (V(G), A(G))$ is a directed tree, and let* Pr *be the joint probability distribution defined by B. Let $V_i \in V(G)$ be a vertex in G, and let $V_i^- = \sigma^*(V_i)$ and $V_i^+ = V(G) \setminus V_i^-$. Then,*

$$\Pr(V_i \mid \tilde{c}_{V(G)}) = \alpha \cdot \Pr(\tilde{c}_{V_i^-} \mid V_i) \cdot \Pr(V_i \mid \tilde{c}_{V_i^+})$$

where $\tilde{c}_{V(G)} = \tilde{c}_{V_i^-} \wedge \tilde{c}_{V_i^+}$ and α is a normalisation constant.

Proof. For the probabilities $\Pr(V_i \mid \tilde{c}_{V(G)})$ of the values of vertex V_i, we have

$$\Pr(V_i \mid \tilde{c}_{V(G)}) = \frac{\Pr(\tilde{c}_{V(G)} \mid V_i) \cdot \Pr(V_i)}{\Pr(\tilde{c}_{V(G)})} = \frac{\Pr(\tilde{c}_{V_i^-} \wedge \tilde{c}_{V_i^+} \mid V_i) \cdot \Pr(V_i)}{\Pr(\tilde{c}_{V_i^-} \wedge \tilde{c}_{V_i^+})}$$

by Bayes' Theorem. Now consider Figure 4.2 showing a fragment of the directed tree G of the network. Using the d-separation criterion, we observe that $\langle X | \{V_i\} | Y \rangle_G^d$ for all sets of vertices $X \subseteq V_i^-$ and $Y \subseteq V_i^+$. Since G is a directed I-map for the joint probability distribution Pr, we conclude that $I_{\Pr}(X, \{V_i\}, Y)$ for all sets $X \subseteq V_i^-$,

$Y \subseteq V_i^+$. Exploiting this observation, we find

$$\begin{aligned}\Pr(V_i \mid \tilde{c}_{V(G)}) &= \frac{\Pr(\tilde{c}_{V_i^-} \mid V_i) \cdot \Pr(\tilde{c}_{V_i^+} \mid V_i) \cdot \Pr(V_i)}{\Pr(\tilde{c}_{V_i^-} \wedge \tilde{c}_{V_i^+})} = \\ &= \frac{\Pr(\tilde{c}_{V_i^-} \mid V_i) \cdot \Pr(V_i \mid \tilde{c}_{V_i^+})}{\Pr(\tilde{c}_{V_i^-} \mid \tilde{c}_{V_i^+})}\end{aligned}$$

Now observe that the factor $(\Pr(\tilde{c}_{V_i^-} \mid \tilde{c}_{V_i^+}))^{-1}$ in the expression depends on the *location* of vertex V_i in the digraph and, if instantiated, on the value of its instantiation, but *not* on the value of V_i under consideration. It therefore is a constant with respect to V_i. In the sequel, this constant will universally be denoted as α. The constant α is generally referred to as a *normalisation constant* because it can be computed from $\Pr(v_i \mid \tilde{c}_{V(G)}) + \Pr(\neg v_i \mid \tilde{c}_{V(G)}) = 1$. The property stated in the lemma now follows by substitution. □

The previous lemma shows that the probabilities of the values of a vertex of interest can be expressed in terms of two factors describing the influence of evidence entered for this vertex' descendants and for all other vertices, separately. The following definition introduces some new terminology for these separate factors.

Definition 4.2.2 *Let $B = (G, \Gamma)$ be a probabilistic network where $G = (V(G), A(G))$ is a directed tree, and let* Pr *be the joint probability distribution defined by B. Let $V_i \in V(G)$ be a vertex in G, and let V_i^- and V_i^+ be as before. The* compound causal parameter π_{V_i} *for vertex V_i is the function $\pi_{V_i} \colon \{C_{V_i}\} \rightarrow [0,1]$ defined by*

$$\pi_{V_i}(V_i) = \Pr(V_i \mid \tilde{c}_{V_i^+})$$

The compound diagnostic parameter λ_{V_i} *for V_i is the function $\lambda_{V_i} : \{C_{V_i}\} \rightarrow [0,1]$ defined by*

$$\lambda_{V_i}(V_i) = \Pr(\tilde{c}_{V_i^-} \mid V_i)$$

Note that the compound diagnostic parameter for a vertex describes the combined influence on this vertex' probabilities of all evidence that has been entered for its descendants; the compound causal parameter for the vertex describes the combined influence of evidence entered for all other vertices in the digraph.

We take a closer look at the previous definition for some special cases. We begin by addressing *uninstantiated* vertices having either no incoming or no outgoing arcs. A directed tree has one vertex W without any incoming arcs; this vertex is the *root* of the tree. We observe that for W the set W^+ is empty. So, $\tilde{c}_{W^+} = \mathsf{True}$. The compound causal parameter π_W for W therefore equals $\pi_W(W) = \Pr(W)$. The directed tree may further include several vertices having no outgoing arcs; these vertices are the *leaves* of the tree. For a leaf V, we observe that the set V^- consists of V only. From V being uninstantiated, we have that $\tilde{c}_{V^-} = \mathsf{True}$. The compound diagnostic parameter λ_V for V therefore equals $\lambda_V(V) = 1$. To conclude, we consider *instantiated* vertices. For a vertex V_i for which the evidence $V_i = \mathit{true}$ has been entered, we find $\pi_{V_i}(v_i) = \Pr(v_i \mid \tilde{c}_{V_i^+})$ and $\pi_{V_i}(\neg v_i) = \Pr(\neg v_i \mid \tilde{c}_{V_i^+})$, and $\lambda_{V_i}(v_i) = \Pr(\tilde{c}_{V_i^-} \mid v_i)$ and $\lambda_{V_i}(\neg v_i) = 0$; an analogous observation holds for the case where the evidence $V_i = \mathit{false}$ has been entered.

Using the definition of the compound causal and diagnostic parameters for a vertex, the property stated in Lemma 4.2.1 can now be reformulated: for each vertex V_i, we have that

$$\Pr(V_i \mid \tilde{c}_{V(G)}) = \alpha \cdot \pi_{V_i}(V_i\,) \cdot \lambda_{V_i}(V_i\,)$$

where α is a normalisation constant. In this form, the lemma is known as the *data fusion lemma* [Pearl, 1988]. Note that the data fusion lemma implies that the compound and diagnostic parameters for a vertex provide it with enough information for computing the probabilities of its values, that is, no further knowledge of the joint probability distribution is needed.

The compound diagnostic parameter for a vertex specifies probabilistic information from *all* its descendants combined; an analogous observation applies to the compound causal parameter for the vertex. To be able to exploit the digraph of a probabilistic network as a computational architecture as outlined before, the compound parameters for a vertex have to be computed from separate parameters originating from its various neighbours. The following definition introduces such separate parameters; the Lemmas 4.2.4 and 4.2.5 will show the computation of the compound parameters from these separate ones.

Definition 4.2.3 *Let $B = (G, \Gamma)$ be a probabilistic network where $G = (V(G), A(G))$ is a directed tree, and let Pr be the joint probability distribution defined by B. For each vertex $V \in V(G)$, let V^- and V^+ be as before. Now, let V_i be a vertex with a successor V_k in G. The* causal parameter $\pi^{V_i}_{V_k}$ from V_i to V_k *is the function* $\pi^{V_i}_{V_k} : \{C_{V_i}\} \rightarrow [0,1]$ *defined by*

$$\pi^{V_i}_{V_k}(V_i\,) = \Pr(V_i \mid \tilde{c}_{V_k^+})$$

Now, let V_i be a vertex having the predecessor V_j in G. The diagnostic parameter $\lambda^{V_j}_{V_i}$ from V_i to V_j *is the function* $\lambda^{V_j}_{V_i} : \{C_{V_j}\} \rightarrow [0,1]$ *defined by*

$$\lambda^{V_j}_{V_i}(V_j\,) = \Pr(\tilde{c}_{V_i^-} \mid V_j\,)$$

Note that a causal parameter is a parameter that a vertex sends to a successor to provide this successor with information concerning its non-descendants. A diagnostic parameter is a parameter that a vertex sends to its predecessor to provide this predecessor with information concerning the vertices located in the subtree rooted at the vertex at hand. The separate causal and diagnostic parameters are the messages the vertices send each other through the communication channels of the computational architecture and, hence, may be looked upon as associated with the arcs of the directed tree of the probabilistic network at hand.

We take a closer look at the previous definition for some special cases. We observe that for the root of a directed tree no diagnostic parameter is defined because it does not have a predecessor. For the leaves of the tree no causal parameters are defined as these vertices do not have any successors. In addition, we note that for a vertex V_i for which the evidence $V_i = true$ is observed and entered into the network, we find that $\pi^{V_i}_{V_k}(v_i) = 1$ and $\pi^{V_i}_{V_k}(\neg v_i) = 0$ for any successor V_k of V_i; an analogous observation holds for the case where $V_i = false$ is observed.

The following lemma now shows that a vertex can compute its compound causal parameter from the causal parameter it receives from its predecessor and its own probability assessment function.

Lemma 4.2.4 *Let $B = (G, \Gamma)$ be a probabilistic network where $G = (V(G), A(G))$ is a directed tree. Let $V_i \in V(G)$ be a vertex with the predecessor V_j in G. Let π_{V_i} be the compound causal parameter for vertex V_i, and let $\pi_{V_i}^{V_j}$ be the causal parameter from V_j to V_i. Then,*

$$\pi_{V_i}(V_i) = \sum_{c_{V_j}} \gamma_{V_i}(V_i \mid c_{V_j}) \cdot \pi_{V_i}^{V_j}(c_{V_j})$$

Proof. Let Pr be the joint probability distribution defined by the probabilistic network B. For vertex V_i, let V_i^+ be as before. Then, by definition we have that

$$\pi_{V_i}(V_i) = \Pr(V_i \mid \tilde{c}_{V_i^+})$$

By conditioning on the values of V_i's predecessor V_j, we find

$$\pi_{V_i}(V_i) = \Pr(V_i \mid v_j \wedge \tilde{c}_{V_i^+}) \cdot \Pr(v_j \mid \tilde{c}_{V_i^+}) + \Pr(V_i \mid \neg v_j \wedge \tilde{c}_{V_i^+}) \cdot \Pr(\neg v_j \mid \tilde{c}_{V_i^+})$$

Now consider once more Figure 4.2 showing a fragment of the directed tree G of the network. Using the d-separation criterion, we observe that $\langle \{V_i\} | \{V_j\} | X \rangle_G^d$ for all sets of vertices $X \subseteq V_i^+$. Since G is a directed I-map for the distribution Pr, we have that $I_{\Pr}(\{V_i\}, \{V_j\}, X)$ for all sets $X \subseteq V_i^+$. Exploiting this observation, we find

$$\pi_{V_i}(V_i) = \Pr(V_i \mid v_j) \cdot \Pr(v_j \mid \tilde{c}_{V_i^+}) + \Pr(V_i \mid \neg v_j) \cdot \Pr(\neg v_j \mid \tilde{c}_{V_i^+})$$

The probabilities $\Pr(V_i \mid v_j)$ and $\Pr(V_i \mid \neg v_j)$ have been specified as function values of the probability assessment function γ_{V_i} and therefore are directly available to vertex V_i. In addition, vertex V_i receives the probabilities $\Pr(v_j \mid \tilde{c}_{V_i^+})$ and $\Pr(\neg v_j \mid \tilde{c}_{V_i^+})$ from its predecessor V_j as function values of the causal parameter $\pi_{V_i}^{V_j}$. The property stated in the lemma now follows by substitution. □

A vertex can further compute its compound diagnostic parameter from the separate diagnostic parameters it receives from its successors in the digraph. This property is stated more formally in the following lemma.

Lemma 4.2.5 *Let $B = (G, \Gamma)$ be a probabilistic network where $G = (V(G), A(G))$ is a directed tree. Let $V_i \in V(G)$ be an uninstantiated vertex with the successors $\sigma(V_i) = \{V_{i_1}, \ldots, V_{i_m}\}$, $m \geq 1$, in G. Furthermore, let λ_{V_i} be the compound diagnostic parameter for vertex V_i and, for each $V_{i_j} \in \sigma(V_i)$, let $\lambda_{V_{i_j}}^{V_i}$ be the diagnostic parameter from V_{i_j} to V_i. Then,*

$$\lambda_{V_i}(V_i) = \prod_{j=1,\ldots,m} \lambda_{V_{i_j}}^{V_i}(V_i)$$

Proof. Let Pr be the joint probability distribution defined by the probabilistic network B. For each vertex $V \in V(G)$, let V^- be as before. Since V_i is an uninstantiated vertex, we have that $\tilde{c}_{V_i^-} = \tilde{c}_{V_{i_1}^-} \wedge \cdots \wedge \tilde{c}_{V_{i_m}^-}$. So,

$$\lambda_{V_i}(V_i) \; = \; \Pr(\tilde{c}_{V_i^-} \mid V_i) \; = \; \Pr(\tilde{c}_{V_{i_1}^-} \wedge \cdots \wedge \tilde{c}_{V_{i_m}^-} \mid V_i)$$

Now consider Figure 4.3 showing a fragment of the directed tree G of the network. We

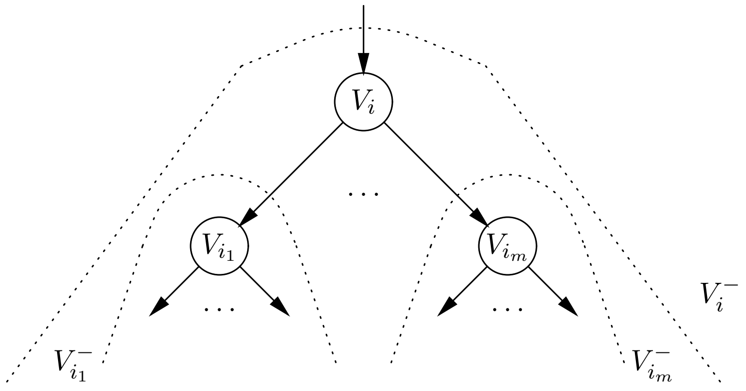


Figure 4.3: Exploiting d-Separation for Computing $\lambda_{V_i}(V_i)$.

observe that $\langle X | \{V_i\} | Y \rangle_G^d$ for all sets of vertices $X \subseteq V_{i_j}^-$ and $Y \subseteq \bigcup_{k=1,\ldots,m,k\neq j} V_{i_k}^-$, $j = 1, \ldots, m$. Since G is a directed I-map for the distribution Pr, we have that $I_{\Pr}(X, \{V_i\}, Y)$ for all sets $X \subseteq V_{i_j}^-$, $Y \subseteq \bigcup_{k=1,\ldots,m,k\neq j} V_{i_k}^-$, $j = 1, \ldots, m$. It follows that

$$\lambda_{V_i}(V_i) = \Pr(\tilde{c}_{V_{i_1}^-} \mid V_i) \cdot \ldots \cdot \Pr(\tilde{c}_{V_{i_m}^-} \mid V_i)$$

The probabilities $\Pr(\tilde{c}_{V_{i_j}^-} \mid V_i)$ are sent to vertex V_i by its successor V_{i_j} as function values of the diagnostic parameter $\lambda_{V_{i_j}}^{V_i}$, $j = 1, \ldots, m$. The property stated in the lemma now follows by substitution. $\Box$

Note that the previous lemma applies to *uninstantiated* vertices only. However, the property mentioned in the lemma can be taken to hold for an *instantiated* vertex V_i as well, if entering evidence for V_i into the network is modelled by adding a 'dummy' successor D for vertex V_i that sends an appropriate diagnostic parameter to V_i. For the evidence $V_i = \textit{true}$, this 'dummy' successor sends the diagnostic parameter $\lambda_D^{V_i}$ with $\lambda_D^{V_i}(v_i) = 1$ and $\lambda_D^{V_i}(\neg v_i) = 0$ to V_i; an analogous observation holds for the evidence $V_i = \textit{false}$.

So far, we have shown that a vertex can compute the probabilities of its values from its own probability assessment function and the causal and diagnostic parameters it receives from its neighbours. Now observe that this vertex in turn has to compute parameters to send to its neighbours. The following lemma shows that a vertex can compute the diagnostic parameter to send to its predecessor from its own probability assessment function and the diagnostic parameters it receives from its successors. In other words, for this purpose it combines its own information about the joint probability distribution with the information it receives concerning the evidence entered so far for its descendants.

Lemma 4.2.6 *Let $B = (G, \Gamma)$ be a probabilistic network where $G = (V(G), A(G))$ is a directed tree. Let $V_i \in V(G)$ be a vertex with the predecessor V_j in G. Let λ_{V_i} be the compound diagnostic parameter for vertex V_i and let $\lambda_{V_i}^{V_j}$ be the diagnostic parameter from V_i to V_j. Then,*

$$\lambda_{V_i}^{V_j}(V_j) = \sum_{c_{V_i}} \lambda_{V_i}(c_{V_i}) \cdot \gamma_{V_i}(c_{V_i} \mid V_j)$$

Proof. Let Pr be the joint probability distribution defined by the probabilistic network B. For vertex V_i, let V_i^- be as before. Then, by definition we have that

$$\lambda_{V_i}^{V_j}(V_j) = \Pr(\tilde{c}_{V_i^-} \mid V_j)$$

By conditioning on the values of V_i, we find

$$\lambda_{V_i}^{V_j}(V_j) = \Pr(\tilde{c}_{V_i^-} \mid v_i \wedge V_j) \cdot \Pr(v_i \mid V_j) + \Pr(\tilde{c}_{V_i^-} \mid \neg v_i \wedge V_j) \cdot \Pr(\neg v_i \mid V_j)$$

Now consider once more Figure 4.2 showing a fragment of the directed tree G of the network. Using the d-separation criterion, we observe that $\langle X | \{V_i\} | \{V_j\} \rangle_G^d$ for all sets of vertices $X \subseteq V_i^-$. Since G is a directed I-map for the distribution Pr, it follows that $I_{\Pr}(X, \{V_i\}, \{V_j\})$ for all sets $X \subseteq V_i^-$. So,

$$\lambda_{V_i}^{V_j}(V_j) = \Pr(\tilde{c}_{V_i^-} \mid v_i) \cdot \Pr(v_i \mid V_j) + \Pr(\tilde{c}_{V_i^-} \mid \neg v_i) \cdot \Pr(\neg v_i \mid V_j)$$

The probabilities $\Pr(v_i \mid V_j)$ and $\Pr(\neg v_i \mid V_j)$ have been specified as function values of the probability assessment function γ_{V_i} associated with vertex V_i and hence are directly available to V_i. In addition, V_i computes the probabilities $Pr(\tilde{c}_{V_i^-} \mid v_i)$ and $Pr(\tilde{c}_{V_i^-} \mid \neg v_i)$ as function values of its compound diagnostic parameter λ_{V_i}. The property stated in the lemma now follows by substitution. □

Similarly, a vertex can compute the causal parameter to send to a successor from its compound causal parameter and the diagnostic parameters it receives from its other successors. The following lemma states this property more formally.

Lemma 4.2.7 *Let $B = (G, \Gamma)$ be a probabilistic network where $G = (V(G), A(G))$ is a directed tree. Let $V_i \in V(G)$ be an uninstantiated vertex with the successors $\sigma(V_i) = \{V_{i_1}, \ldots, V_{i_m}\}$, $m \geq 1$, in G. Furthermore, let π_{V_i} be the compound causal parameter for vertex V_i; for each $V_{i_j} \in \sigma(V_i)$, let $\pi_{V_{i_j}}^{V_i}$ be the causal parameter from V_i to V_{i_j} and let $\lambda_{V_{i_j}}^{V_i}$ be the diagnostic parameter from V_{i_j} to V_i. Then,*

$$\pi_{V_{i_j}}^{V_i}(V_i) = \alpha \cdot \pi_{V_i}(V_i) \cdot \prod_{k=1,\ldots,m, k \neq j} \lambda_{V_{i_k}}^{V_i}(V_i)$$

where α is a normalisation constant.

Proof. Let Pr be the joint probability distribution defined by the probabilistic network B. For each vertex $V \in V(G)$, let V^+ and V^- be as before. Then, by definition we have that

$$\pi_{V_{i_j}}^{V_i}(V_i) = \Pr(V_i \mid \tilde{c}_{V_{i_j}^+})$$

Using Bayes' Theorem, we find

$$\pi_{V_{i_j}}^{V_i}(V_i) = \frac{\Pr(\tilde{c}_{V_{i_j}^+} \mid V_i) \cdot \Pr(V_i)}{\Pr(\tilde{c}_{V_{i_j}^+})}$$

Now consider Figure 4.4 showing a fragment of the directed tree G of the network. Since V_i is an uninstantiated vertex, we have that $\tilde{c}_{V_{i_j}^+} = \tilde{c}_{V_i^+} \wedge (\bigwedge_{k=1,\ldots,m,k \neq j} \tilde{c}_{V_{i_k}^-})$. So,

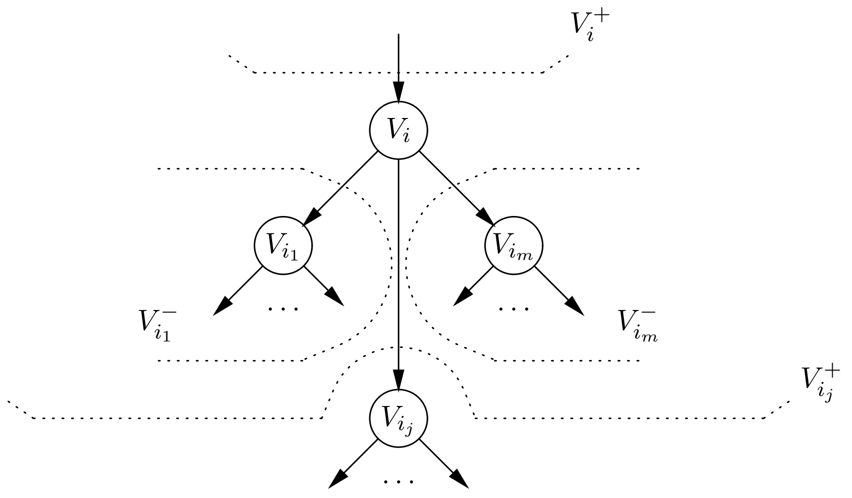


Figure 4.4: Exploiting d-Separation for Computing $\pi^{V_i}_{V_{i_j}}(V_i)$.

$$\pi^{V_i}_{V_{i_j}}(V_i) = \frac{\Pr(\tilde{c}_{V_i^+} \wedge (\bigwedge_{k=1,\ldots,m,k\neq j} \tilde{c}_{V_{i_k}^-}) \mid V_i) \cdot \Pr(V_i)}{\Pr(\tilde{c}_{V_{i_j}^+})}$$

Furthermore, using the d-separation criterion, we observe that $\langle X \mid \{V_i\} \mid Y\rangle^d_G$ for all sets of vertices $X \subseteq V_i^+$ and $Y \subseteq V_{i_k}^-$, $k = 1,\ldots,m$, and for all sets of vertices $X \subseteq V_{i_k}^-$ and $Y \subseteq V_{i_l}^-$, $k = 1,\ldots,m$, $l = 1,\ldots,m$, $k \neq l$. Exploiting this observation, we find

$$\begin{aligned}\pi^{V_i}_{V_{i_j}}(V_i) &= \frac{\Pr(\tilde{c}_{V_i^+} \mid V_i) \cdot \prod_{k=1,\ldots,m,k\neq j} \Pr(\tilde{c}_{V_{i_k}^-} \mid V_i) \cdot \Pr(V_i)}{\Pr(\tilde{c}_{V_{i_j}^+})} = \\ &= \frac{\Pr(V_i \mid \tilde{c}_{V_i^+}) \cdot \prod_{k=1,\ldots,m,k\neq j} \Pr(\tilde{c}_{V_{i_k}^-} \mid V_i) \cdot \Pr(\tilde{c}_{V_i^+})}{\Pr(\tilde{c}_{V_{i_j}^+})}\end{aligned}$$

The probabilities $\Pr(V_i \mid \tilde{c}_{V_i^+})$ equal the function values of the compound causal parameter π_{V_i} for V_i. The probabilities $Pr(\tilde{c}_{V_{i_k}^-} \mid V_i)$ equal the function values of the diagnostic parameter $\lambda^{V_i}_{V_{i_k}}$ vertex V_i receives from its successor V_{i_k}. In addition, we observe that the factor

$$\frac{\Pr(\tilde{c}_{V_i^+})}{\Pr(\tilde{c}_{V_{i_j}^+})} = \frac{1}{\Pr(\tilde{c}_{V_{i_j}^+} \mid \tilde{c}_{V_i^+})}$$

is dependent on the *location* of the variables V_i and V_{i_j} in the digraph but *not* on their values. This factor may therefore be looked upon as a normalisation constant for V_i and V_{i_j}, and will henceforth be denoted as α; it can be computed from $\Pr(v_i \mid \tilde{c}_{V_{i_j}^+}) + \Pr(\neg v_i \mid \tilde{c}_{V_{i_j}^+}) = 1$. The property stated in the lemma now follows by substitution. □

The previous lemma applies to uninstantiated vertices only; the lemma, however, can be taken to hold for instantiated vertices as described before.

The data fusion lemma and the four computation rules provided by the Lemmas 4.2.4, 4.2.5, 4.2.6, and 4.2.7 with each other constitute Pearl's algorithm for probabilistic inference with a probabilistic network comprising a directed tree for its qualitative part. Note that Pearl's algorithm provides for computing probabilities as well as for processing evidence in such a probabilistic network. The computation rules for the separate causal and diagnostic parameters enable a vertex to pass on the impact of a piece of evidence correctly, and allow for the evidence to spread throughout the network. Close examination of the computation rules from the Lemmas 4.2.6 and 4.2.7 further reveals that a vertex that sends an updated parameter will not receive a new parameter originating from the same evidence. A causal parameter or a diagnostic parameter *to* a vertex is not affected by the diagnostic parameter or the causal parameter, respectively, *from* that vertex. In addition, the topological property that in a directed tree there is at most one chain between any two vertices prohibits the process of parameter updating that originates from some vertex to reach this vertex along another chain. These properties with each other guarantee that feedback and circular reasoning are prevented and that evidence is propagated throughout the network in a single pass.

We state an additional property concerning the compound diagnostic parameter for a vertex that is useful for investigating the spreading of evidence.

Lemma 4.2.8 *Let $B = (G, \Gamma)$ be a probabilistic network where $G = (V(G), A(G))$ is a directed tree. For each vertex $V_i \in V(G)$, let $\lambda_{V_i}(V_i)$ be the compound diagnostic parameter for V_i. If $\tilde{c}_{V(G)} = \mathsf{True}$, then $\lambda_{V_i}(V_i) = 1$ for all $V_i \in V(G)$.*

Proof. The property stated in the lemma is proven by (reverse) induction on the depth of the directed tree G. Let n be the maximal depth of the tree.

Induction Basis
The property holds for every leaf of the tree at depth n by definition.

Induction Hypothesis
For a specific $d \leq n$, we assume that $\lambda_{V_i}(V_i) = 1$ for all vertices V_i at depth $d, d+1, \ldots, n$.

Induction Step
Now consider a vertex V_i at depth $d-1$ in the tree. We distinguish between two cases. If V_i is a leaf of the tree, then $\lambda_{V_i}(V_i) = 1$ by definition. Now, suppose that V_i has m successors $V_{i_1}, \ldots, V_{i_m}$, $m \geq 1$. From $\tilde{c}_{V(G)} = \mathsf{True}$, it follows that V_i is an uninstantiated vertex. Therefore, it follows from the property stated in Lemma 4.2.5 that

$$\lambda_{V_i}(V_i) = \prod_{j=1,\ldots,m} \lambda^{V_i}_{V_{i_j}}(V_i)$$

where $\lambda^{V_i}_{V_{i_j}}$ is the diagnostic parameter from V_{i_j} to V_i, $j = 1, \ldots, m$. For each parameter $\lambda^{V_i}_{V_{i_j}}$ we have from the property stated in Lemma 4.2.6 that

$$\lambda^{V_i}_{V_{i_j}}(V_i) = \sum_{c_{V_{i_j}}} \lambda_{V_{i_j}}(c_{V_{i_j}}) \cdot \gamma_{V_{i_j}}(c_{V_{i_j}} \mid V_i)$$

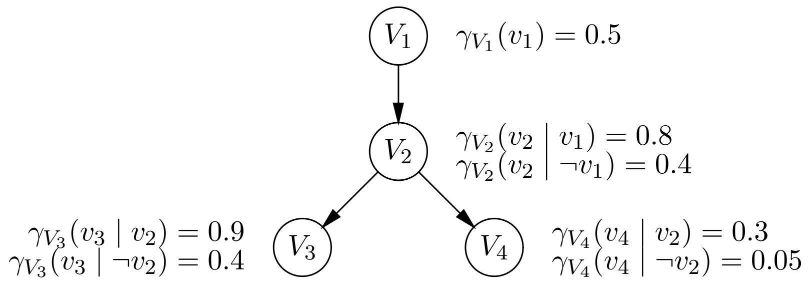

V_1
$\gamma_{V_1}(v_1) = 0.5$
V_2
$\gamma_{V_2}(v_2 \mid v_1) = 0.8$
$\gamma_{V_2}(v_2 \mid \neg v_1) = 0.4$
$\gamma_{V_3}(v_3 \mid v_2) = 0.9$
$\gamma_{V_3}(v_3 \mid \neg v_2) = 0.4$
V_3
V_4
$\gamma_{V_4}(v_4 \mid v_2) = 0.3$
$\gamma_{V_4}(v_4 \mid \neg v_2) = 0.05$

Figure 4.5: An Example Probabilistic Network.

Since vertex V_{i_j} is a successor of V_i, it is located at depth d in the directed tree G. From the induction hypothesis we have that $\lambda_{V_{i_j}}(V_{i_j}) = 1$. So,

$$\lambda^{V_i}_{V_{i_j}}(V_i) = \sum_{c_{V_{i_j}}} \gamma_{V_{i_j}}(c_{V_{i_j}} \mid V_i) = 1$$

From $\lambda^{V_i}_{V_{i_j}}(V_i) = 1$ for all successors V_{i_j} of V_i, it follows that $\lambda_{V_i}(V_i) = 1$. Since vertex V_i has been chosen arbitrarily, it follows that for each vertex $V_i \in V(G)$, we have $\lambda_{V_i}(V_i) = 1$. $\square$

It will be evident that the previous lemma can be taken to apply to subtrees of a directed tree as well.

We conclude our discussion of Pearl's algorithm for probabilistic inference so far with an example.

Example 4.2.9 We consider the probabilistic network $B = (G, \Gamma)$ shown in Figure 4.5. Let Pr be the joint probability distribution defined by B. We address the computation of the probabilities and different parameters for the various vertices in the network.

Vertex V_1 sets out to compute the probabilities of its values by application of the data fusion lemma:

$$\begin{aligned} \Pr(v_1) &= \alpha \cdot \pi_{V_1}(v_1) \cdot \lambda_{V_1}(v_1) \\ \Pr(\neg v_1) &= \alpha \cdot \pi_{V_1}(\neg v_1) \cdot \lambda_{V_1}(\neg v_1) \end{aligned}$$

To be able to compute the probabilities of interest from the lemma, vertex V_1 needs to compute its compound parameters π_{V_1} and λ_{V_1}. Since no evidence has been entered into the network as yet, we have from the property stated in Lemma 4.2.8 that the values of the compound diagnostic parameter λ_{V_1} equal

$$\begin{aligned} \lambda_{V_1}(v_1) &= 1 \\ \lambda_{V_1}(\neg v_1) &= 1 \end{aligned}$$

Vertex V_1 now turns to the computation of its compound causal parameter π_{V_1}. Using the property stated in Lemma 4.2.4, it finds the values of this parameter to be

$$\begin{aligned} \pi_{V_1}(v_1) &= 0.5 \\ \pi_{V_1}(\neg v_1) &= 0.5 \end{aligned}$$

Substitution of the values of the compound parameters into the data fusion lemma and subsequent elimination of the normalisation constant α yields

$$\begin{aligned} \Pr(v_1) &= 0.5 \\ \Pr(\neg v_1) &= 0.5 \end{aligned}$$

Now observe that vertex V_1 not just computes its probabilities, it also computes the parameter to send to its neighbour in the tree. Using the property stated in Lemma 4.2.7, it computes the values of the causal parameter $\pi_{V_2}^{V_1}$ for its successor V_2 to be

$$\begin{aligned}\pi_{V_2}^{V_1}(v_1) &= 0.5\\ \pi_{V_2}^{V_1}(\neg v_1) &= 0.5\end{aligned}$$

By sending the thus computed parameter to vertex V_2, vertex V_1 provides V_2 with sufficient information about the represented probability distribution to enable vertex V_2 to compute its probabilities.

Just as vertex V_1, vertex V_2 sets out to compute the probabilities of its values by application of the data fusion lemma:

$$\begin{aligned}\Pr(v_2) &= \alpha\cdot\pi_{V_2}(v_2)\cdot\lambda_{V_2}(v_2)\\ \Pr(\neg v_2) &= \alpha\cdot\pi_{V_2}(\neg v_2)\cdot\lambda_{V_2}(\neg v_2)\end{aligned}$$

Vertex V_2 now needs to compute its compound parameters π_{V_2} and λ_{V_2}. Since no evidence has been entered into the network as yet, we once more have from the property stated in Lemma 4.2.8 that the values of the compound diagnostic parameter equal

$$\begin{aligned}\lambda_{V_2}(v_2) &= 1\\ \lambda_{V_2}(\neg v_2) &= 1\end{aligned}$$

Using the property stated in Lemma 4.2.4, vertex V_2 now computes the compound causal parameter π_{V_2} from its own probability assessment function γ_{V_2} and the causal parameter $\pi_{V_2}^{V_1}$ it receives from its predecessor. For the values $\pi_{V_2}(v_2)$ and $\pi_{V_2}(\neg v_2)$, it thus finds

$$\begin{aligned}\pi_{V_2}(v_2) &= \gamma_{V_2}(v_2\mid v_1)\cdot\pi_{V_2}^{V_1}(v_1)+\gamma_{V_2}(v_2\mid\neg v_1)\cdot\pi_{V_2}^{V_1}(\neg v_1)=\\ &= 0.8\cdot 0.5+0.4\cdot 0.5=0.6\\ \pi_{V_2}(\neg v_2) &= \gamma_{V_2}(\neg v_2\mid v_1)\cdot\pi_{V_2}^{V_1}(v_1)+\gamma_{V_2}(\neg v_2\mid\neg v_1)\cdot\pi_{V_2}^{V_1}(\neg v_1)=\\ &= 0.2\cdot 0.5+0.6\cdot 0.5=0.4\end{aligned}$$

Substitution of the values of the compound parameters into the data fusion lemma and subsequent elimination of the normalisation constant α yields

$$\begin{aligned}\Pr(v_2) &= 0.6\\ \Pr(\neg v_2) &= 0.4\end{aligned}$$

In addition to computing its own probabilities, vertex V_2 computes the parameters to send to its neighbours in the tree. For its predecessor V_1, vertex V_2 computes the diagnostic parameter $\lambda_{V_2}^{V_1}$; the values of this parameter equal

$$\begin{aligned}\lambda_{V_2}^{V_1}(v_1) &= 1\\ \lambda_{V_2}^{V_1}(\neg v_1) &= 1\end{aligned}$$

Vertex V_2 further computes a causal parameter for every one of its successors in the tree. Using the property stated in Lemma 4.2.7, it computes the values of the causal parameter $\pi_{V_3}^{V_2}$ for vertex V_3 from

$$\begin{aligned}\pi_{V_3}^{V_2}(v_2) &= \alpha\cdot\pi_{V_2}(v_2)\cdot\lambda_{V_4}^{V_2}(v_2)\\ \pi_{V_3}^{V_2}(\neg v_2) &= \alpha\cdot\pi_{V_2}(\neg v_2)\cdot\lambda_{V_4}^{V_2}(\neg v_2)\end{aligned}$$

It is readily seen that the values of the diagnostic parameter $\lambda_{V_4}^{V_2}$ that vertex V_2 receives from vertex V_4 both equal one. Substitution of the various parameter values and subsequent elimination of the normalisation constant α now yields

$$\begin{aligned}
\pi_{V_3}^{V_2}(v_2) &= 0.6 \\
\pi_{V_3}^{V_2}(\neg v_2) &= 0.4
\end{aligned}$$

For vertex V_4, vertex V_2 equally computes

$$\begin{aligned}
\pi_{V_4}^{V_2}(v_2) &= 0.6 \\
\pi_{V_4}^{V_2}(\neg v_2) &= 0.4
\end{aligned}$$

By sending the thus computed parameters to its respective neighbours, vertex V_2 enables these neighbours to compute their probabilities in turn. Vertex V_3 now computes the probabilities of its values to be

$$\begin{aligned}
\Pr(v_3) &= 0.7 \\
\Pr(\neg v_3) &= 0.3
\end{aligned}$$

Vertex V_4 computes its probabilities to be

$$\begin{aligned}
\Pr(v_4) &= 0.2 \\
\Pr(\neg v_4) &= 0.8
\end{aligned}$$

Now, suppose that the evidence $V_4 = \mathit{true}$ is observed and entered into the probabilistic network B. We address the computation of the posterior probabilities and the different parameters for the various vertices in the network using Pearl's algorithm. Vertex V_4 once more sets out to compute the probabilities of its values by application of the data fusion lemma:

$$\begin{aligned}
\Pr^{v_4}(v_4) &= \alpha \cdot \pi_{V_4}(v_4) \cdot \lambda_{V_4}(v_4) \\
\Pr^{v_4}(\neg v_4) &= \alpha \cdot \pi_{V_4}(\neg v_4) \cdot \lambda_{V_4}(\neg v_4)
\end{aligned}$$

Since the evidence $V_4 = \mathit{true}$ has been entered for vertex V_4, it finds the values of its compound diagnostic parameter λ_{V_4} to be

$$\begin{aligned}
\lambda_{V_4}(v_4) &= 1 \\
\lambda_{V_4}(\neg v_4) &= 0
\end{aligned}$$

resulting in

$$\begin{aligned}
\Pr^{v_4}(v_4) &= 1 \\
\Pr^{v_4}(\neg v_4) &= 0
\end{aligned}$$

as expected. In addition to updating its own probabilities, vertex V_4 computes a new parameter to send to its neighbour in the tree. Using the property stated in Lemma 4.2.6, vertex V_4 computes the values of the diagnostic parameter $\lambda_{V_4}^{V_2}$ for its predecessor V_2 to be

$$\begin{aligned}
\lambda_{V_4}^{V_2}(v_2) &= \gamma_{V_4}(v_4 \mid v_2) \cdot \lambda_{V_4}(v_4) + \gamma_{V_4}(\neg v_4 \mid v_2) \cdot \lambda_{V_4}(\neg v_4) = \\
&= 0.3 \cdot 1 + 0.7 \cdot 0 = 0.3 \\
\lambda_{V_4}^{V_2}(\neg v_2) &= \gamma_{V_4}(v_4 \mid \neg v_2) \cdot \lambda_{V_4}(v_4) + \gamma_{V_4}(\neg v_4 \mid \neg v_2) \cdot \lambda_{V_4}(\neg v_4) = \\
&= 0.05 \cdot 1 + 0.95 \cdot 0 = 0.05
\end{aligned}$$

By sending the thus computed parameter to vertex V_2, vertex V_4 informs V_2 of the newly entered evidence, enabling it to update its probabilities.

As before, vertex V_2 sets out to compute the probabilities of its values by application of the data fusion lemma:

$$\begin{aligned}\Pr^{v_4}(v_2) &= \alpha \cdot \pi_{V_2}(v_2) \cdot \lambda_{V_2}(v_2)\\ \Pr^{v_4}(\neg v_2) &= \alpha \cdot \pi_{V_2}(\neg v_2) \cdot \lambda_{V_2}(\neg v_2)\end{aligned}$$

Since no evidence has been entered for vertex V_2's non-descendants, it once more computes the values of its compound causal parameter π_{V_2} to be

$$\begin{aligned}\pi_{V_2}(v_2) &= 0.6\\ \pi_{V_2}(\neg v_2) &= 0.4\end{aligned}$$

Using the property stated in Lemma 4.2.5, vertex V_2 computes the values of its compound diagnostic parameter to be

$$\begin{aligned}\lambda_{V_2}(v_2) &= \lambda^{V_2}_{V_3}(v_2) \cdot \lambda^{V_2}_{V_4}(v_2)\\ \lambda_{V_2}(\neg v_2) &= \lambda^{V_2}_{V_3}(\neg v_2) \cdot \lambda^{V_2}_{V_4}(\neg v_2)\end{aligned}$$

It is readily seen that the values of the diagnostic parameter $\lambda^{V_2}_{V_3}$ that vertex V_2 receives from vertex V_3 both equal one. Substitution of the various parameter values now yields

$$\begin{aligned}\lambda_{V_2}(v_2) &= 0.3\\ \lambda_{V_2}(\neg v_2) &= 0.05\end{aligned}$$

Vertex V_2 substitutes the values of its compound parameters π_{V_2} and λ_{V_2} into the data fusion lemma to find

$$\begin{aligned}\Pr^{v_4}(v_2) &= \alpha \cdot 0.6 \cdot 0.3 = \alpha \cdot 0.18\\ \Pr^{v_4}(\neg v_2) &= \alpha \cdot 0.4 \cdot 0.05 = \alpha \cdot 0.02\end{aligned}$$

After eliminating the normalisation constant α it finds

$$\begin{aligned}\Pr^{v_4}(v_2) &= 0.9\\ \Pr^{v_4}(\neg v_2) &= 0.1\end{aligned}$$

In addition to updating its own probabilities, vertex V_2 computes new parameters to send to its other neighbours than vertex V_4. Using the property stated in Lemma 4.2.7, vertex V_2 computes the values of the causal parameter $\pi^{V_2}_{V_3}$ for its successor V_3 to be

$$\begin{aligned}\pi^{V_2}_{V_3}(v_2) &= \alpha \cdot \pi_{V_2}(v_2) \cdot \lambda^{V_2}_{V_4}(v_2) = \alpha \cdot 0.6 \cdot 0.3 = 0.9\\ \pi^{V_2}_{V_3}(\neg v_2) &= \alpha \cdot \pi_{V_2}(\neg v_2) \cdot \lambda^{V_2}_{V_4}(\neg v_2) = \alpha \cdot 0.4 \cdot 0.05 = 0.1\end{aligned}$$

Using the property stated in Lemma 4.2.6, vertex V_2 computes the values of the diagnostic parameter $\lambda^{V_1}_{V_2}$ for its predecessor V_1 to be

$$\begin{aligned}\lambda^{V_1}_{V_2}(v_1) &= \gamma_{V_2}(v_2 \mid v_1) \cdot \lambda_{V_2}(v_2) + \gamma_{V_2}(\neg v_2 \mid v_1) \cdot \lambda_{V_2}(\neg v_2) =\\ &= 0.8 \cdot 0.3 + 0.2 \cdot 0.05 = 0.25\\ \lambda^{V_1}_{V_2}(\neg v_1) &= \gamma_{V_2}(v_2 \mid \neg v_1) \cdot \lambda_{V_2}(v_2) + \gamma_{V_2}(\neg v_2 \mid \neg v_1) \cdot \lambda_{V_2}(\neg v_2) =\\ &= 0.4 \cdot 0.3 + 0.6 \cdot 0.05 = 0.15\end{aligned}$$

By sending the thus computed parameters to vertex V_3 and vertex V_1, respectively, vertex V_2 enables these vertices to compute their updated probabilities. Vertex V_3 computes the updated probabilities of its values to be

$$\begin{aligned}\Pr^{v_4}(v_3) &= 0.85\\ \Pr^{v_4}(\neg v_3) &= 0.15\end{aligned}$$

Vertex V_1 computes its updated probabilities to be

$$\begin{aligned}\Pr^{v_4}(v_1) &= 0.625\\ \Pr^{v_4}(\neg v_1) &= 0.375\end{aligned}$$

□

4.2.2 Singly Connected Digraphs

So far, we have only discussed Pearl's algorithm for probabilistic inference for probabilistic networks comprising a directed tree for their qualitative part. In this section, we extend the algorithm to apply to probabilistic networks of which the qualitative part is a *singly connected digraph* more in general. Before detailing the computations involved in the extended algorithm, we introduce some new terminology and notational conventions that we will use in the sequel. We consider a singly connected digraph G. For this graph G, we observe that removal of any arc splits the graph into two separate components. From this property we have that in the digraph G we can identify for a vertex V_i with m neighbours, m subgraphs of G each containing a neighbour of V_i such that, after removal of V_i and all its incoming and outgoing arcs, there does not exist a path from one such subgraph to another. The following definition introduces these subgraphs more formally; Figure 4.6 illustrates the basic idea.

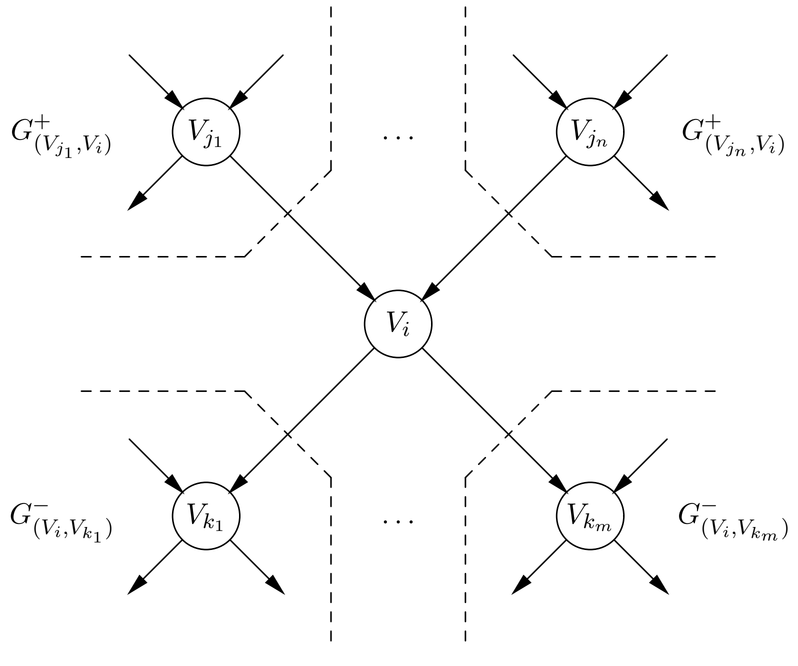


Figure 4.6: Upper and Lower Graphs.

Definition 4.2.10 *Let $G = (V(G), A(G))$ be a singly connected directed graph. For each arc $(V_i, V_j) \in A(G)$, let $G_{(V_i,V_j)} = (V(G), A(G) \setminus \{(V_i, V_j)\})$ be the digraph that results after removing arc (V_i, V_j) from G. Now, let $V_i \in V(G)$ be a vertex in G. For each predecessor $V_j \in \rho(V_i)$ of V_i, let $G^+_{(V_j,V_i)}$ be the component of $G_{(V_j,V_i)}$ that includes V_j in its vertex set; $G^+_{(V_j,V_i)}$ is called an* upper graph *of vertex V_i. For each successor $V_k \in \sigma(V_i)$ of V_i, let $G^-_{(V_i,V_k)}$ be the component of $G_{(V_i,V_k)}$ that includes V_k in its vertex set; $G^-_{(V_i,V_k)}$ is called a* lower graph *of V_i.*

In the previous section, we have detailed Pearl's computation rules for probabilistic inference with a probabilistic network comprising a directed tree: we recall that these rules address the computation of the different parameters for the various vertices in the tree. The proofs of the lemmas that we have presented show that these computation rules derive from *exploiting independences*. These independences are read from the qualitative part of a probabilistic network by *local inspection of a vertex' incoming and outgoing arcs only*. The computation rules therefore make use explicitly of the property that in the network's digraph there is at most one chain between any two vertices. Since singly connected digraphs share this property with directed trees, Pearl's tree algorithm is extended straightforwardly to apply to probabilistic networks comprising a singly connected digraph for their qualitative part. In fact, only the computation rules for the compound causal parameter and for the separate diagnostic parameters are adapted to account for a vertex having multiple predecessors; all other computation rules remain unaltered. For the sake of completeness, we will review all computation rules involved.

We begin by addressing the computation of probabilities from a probabilistic network comprising a singly connected digraph. It will be evident that the probabilities of the values of a vertex of interest are dependent upon all evidence entered into the network so far. As we have seen for the vertices in a directed tree, the probabilities of the values of a vertex in a singly connected digraph can be written in terms of two factors. These factors describe the combined influence on the vertex' probabilities of evidence entered for the vertices in its upper graphs and of evidence entered for the vertices in its lower graphs, respectively. We redefine the compound causal and diagnostic parameters for a vertex to capture these factors in view of singly connected digraphs.

Definition 4.2.11 *Let $B = (G, \Gamma)$ be a probabilistic network where $G = (V(G), A(G))$ is a singly connected digraph, and let* Pr *be the joint probability distribution defined by B. Let $V_i \in V(G)$ be a vertex in G, and let $V_i^+ = \bigcup_{V_j \in \rho(V_i)} V(G^+_{(V_j,V_i)})$ and $V_i^- = V(G) \setminus V_i^+$. The* compound causal parameter *π_{V_i} for vertex V_i is the function $\pi_{V_i} : \{C_{V_i}\} \to [0,1]$ defined by*

$$\pi_{V_i}(V_i) = \Pr(V_i \mid \tilde{c}_{V_i^+})$$

The compound diagnostic parameter *λ_{V_i} for V_i is the function $\lambda_{V_i} : \{C_{V_i}\} \to [0,1]$ defined by*

$$\lambda_{V_i}(V_i) = \Pr(\tilde{c}_{V_i^-} \mid V_i)$$

Note that this redefinition of the compound parameters differs from Definition 4.2.2 with respect to the sets V_i^+ and V_i^- for a vertex V_i. The basic idea of the parameters, however, remains unaltered. The compound causal parameter for a vertex describes the combined influence on its probabilities of all evidence that has been entered 'above' it in the digraph; the compound diagnostic parameter for the vertex describes the combined influence of all evidence that has been entered 'below' it in the digraph. Also note that, for a vertex W without any incoming arcs, we once more find $\pi_W(W) = \Pr(W)$; for a vertex V without any outgoing arcs, we again have $\lambda_V(V) = 1$.

Using the redefinition of the compound causal and diagnostic parameters for a vertex, the *data fusion lemma* now applies to a probabilistic network comprising a singly connected digraph for its qualitative part: for each vertex V_i in the network, we once more have

$$\Pr(V_i \mid \tilde{c}_{V(G)}) = \alpha \cdot \pi_{V_i}(V_i) \cdot \lambda_{V_i}(V_i)$$

where α is a normalisation constant.

The compound diagnostic and causal parameters for a vertex specify probabilistic information from its lower graphs combined and from all its upper graphs combined, respectively. We observe that to be able to exploit the qualitative part of a probabilistic network as a computational architecture, these compound parameters once again have to be computed from separate causal and diagnostic parameters originating from the various neighbours of the vertex. We redefine these separate parameters before addressing the computation of the compound ones.

Definition 4.2.12 *Let $B = (G, \Gamma)$ be a probabilistic network where $G = (V(G), A(G))$ is a singly connected digraph, and let* Pr *be the joint probability distribution defined by B. Let V_i be a vertex with a successor V_k in G and let $G^+_{(V_i,V_k)}$ be as before. The* causal parameter $\pi^{V_i}_{V_k}$ from V_i to V_k *is the function $\pi^{V_i}_{V_k} : \{C_{V_i}\} \to [0,1]$ defined by*

$$\pi^{V_i}_{V_k}(V_i) = \Pr(V_i \mid \tilde{c}_{V(G^+_{(V_i,V_k)})})$$

Now, let V_i be a vertex having a predecessor V_j in G and let $G^-_{(V_j,V_i)}$ be as before. The diagnostic parameter $\lambda^{V_j}_{V_i}$ from V_i to V_j *is the function $\lambda^{V_j}_{V_i} : \{C_{V_j}\} \to [0,1]$ defined by*

$$\lambda^{V_j}_{V_i}(V_j) = \Pr(\tilde{c}_{V(G^-_{(V_j,V_i)})} \mid V_j)$$

The separate causal and diagnostic parameters defined above once again are the messages the vertices send one another through the communication channels of the computational architecture and, hence, may again be looked upon as associated with the arcs of the digraph of the probabilistic network at hand.

The following lemma now shows that a vertex can compute its compound causal parameter from its own probability assessment function and the causal parameters it receives from its various predecessors.

Lemma 4.2.13 *Let $B = (G, \Gamma)$ be a probabilistic network where $G = (V(G), A(G))$ is a singly connected digraph. Let $V_i \in V(G)$ be a vertex with the predecessors $\rho(V_i) = \{V_{j_1}, \ldots, V_{j_n}\}$, $n \geq 1$, in G. Furthermore, let π_{V_i} be the compound causal parameter for vertex V_i and, for each $V_{j_k} \in \rho(V_i)$, let $\pi^{V_{j_k}}_{V_i}$ be the causal parameter from V_{j_k} to V_i. Then,*

$$\pi_{V_i}(V_i) = \sum_{c_{\rho(V_i)}} \gamma_{V_i}(V_i \mid c_{\rho(V_i)}) \cdot \prod_{k=1,\ldots,n} \pi^{V_{j_k}}_{V_i}(c_{V_{j_k}})$$

where $c_{\rho(V_i)} = \bigwedge_{k=1,\ldots,n} c_{V_{j_k}}$.

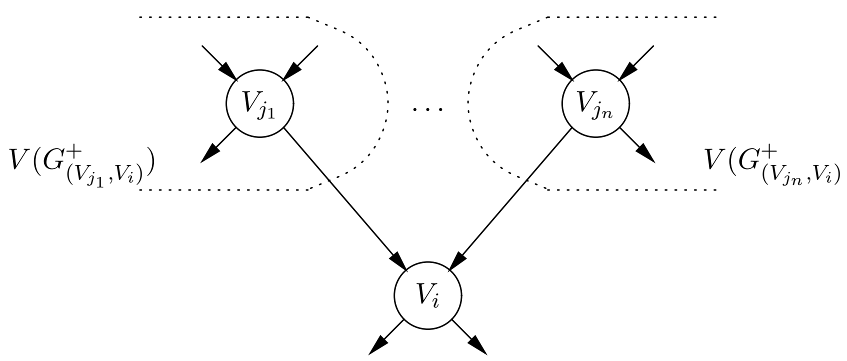


Figure 4.7: Exploiting d-Separation for Computing $\pi_{V_i}(V_i)$.

Proof. Let Pr be the joint probability distribution defined by the probabilistic network B. For vertex V_i, let $V_i^+ = \bigcup_{k=1,\ldots,n} V(G^+_{(V_{j_k},V_i)})$ be as before. Then, by definition we have that

$$\pi_{V_i}(V_i) \;=\; \Pr(V_i \mid \tilde{c}_{V_i^+}) \;=\; \Pr(V_i \mid \tilde{c}_{V(G^+_{(V_{j_1},V_i)})} \wedge \cdots \wedge \tilde{c}_{V(G^+_{(V_{j_n},V_i)})})$$

By conditioning on the various configurations of the set $\rho(V_i)$ of V_i's predecessors, we find

$$\begin{aligned}\pi_{V_i}(V_i) \;=\; & \sum_{c_{\rho(V_i)}} \Pr(V_i \mid c_{\rho(V_i)} \wedge \tilde{c}_{V(G^+_{(V_{j_1},V_i)})} \wedge \cdots \wedge \tilde{c}_{V(G^+_{(V_{j_n},V_i)})}) \cdot \\ & \cdot \Pr(c_{\rho(V_i)} \mid \tilde{c}_{V(G^+_{(V_{j_1},V_i)})} \wedge \cdots \wedge \tilde{c}_{V(G^+_{(V_{j_n},V_i)})})\end{aligned}$$

Now consider Figure 4.7 showing a fragment of the singly connected digraph G of the network. Using the d-separation criterion, we observe that $\langle \{V_i\} | \rho(V_i) | X \rangle^d_G$ for all sets of vertices $X \subseteq V_i^+$. Since G is a directed I-map for the distribution Pr, it follows that $I_{\Pr}(\{V_i\}, \rho(V_i), X)$ for all sets $X \subseteq V_i^+$. Exploiting this observation, we find

$$\pi_{V_i}(V_i) = \sum_{c_{\rho(V_i)}} \Pr(V_i \mid c_{\rho(V_i)}) \cdot \Pr(c_{\rho(V_i)} \mid \tilde{c}_{V(G^+_{(V_{j_1},V_i)})} \wedge \cdots \wedge \tilde{c}_{V(G^+_{(V_{j_n},V_i)})})$$

By once more using the d-separation criterion, we find that $I_{\Pr}(X, \{V_{j_k}\}, Y)$ for all sets of vertices $X \subseteq V(G^+_{(V_{j_k},V_i)})$, $Y \subseteq V(G^+_{(V_{j_l},V_i)})$, $l = 1,\ldots,n$, $k = 1,\ldots,n$, $k \neq l$. Exploiting this observation, we find

$$\pi_{V_i}(V_i) = \sum_{c_{\rho(V_i)}} \Pr(V_i \mid c_{\rho(V_i)}) \cdot \Pr(c_{V_{j_1}} \mid \tilde{c}_{V(G^+_{(V_{j_1},V_i)})}) \cdot \ldots \cdot \Pr(c_{V_{j_n}} \mid \tilde{c}_{V(G^+_{(V_{j_n},V_i)})})$$

where $c_{\rho(V_i)} = \bigwedge_{k=1,\ldots,n} c_{V_{j_k}}$. The probabilities $\Pr(V_i \mid c_{\rho(V_i)})$ have been specified as function values of the probability assessment function γ_{V_i} and therefore are directly available to vertex V_i. In addition, V_i receives the probabilities $\Pr(c_{V_{j_k}} \mid \tilde{c}_{V(G^+_{(V_{j_k},V_i)})})$ from its predecessor V_{j_k} as function values of the causal parameter $\pi^{V_{j_k}}_{V_i}$, $k = 1,\ldots,n$. The property stated in the lemma now follows by substitution. □

A vertex can further compute its compound diagnostic parameter from the separate diagnostic parameters it receives from its successors in the digraph. This property is analogous to the property stated in Lemma 4.2.5, that is, for an uninstantiated vertex $V_i \in V(G)$ with successors $\sigma(V_i) = \{V_{i_1}, \ldots, V_{i_m}\}$, $m \geq 1$, in the digraph, we have that

$$\lambda_{V_i}(V_i) = \prod_{j=1,\ldots,m} \lambda^{V_i}_{V_{i_j}}(V_i)$$

The property can be taken to apply to instantiated vertices as outlined before.

A vertex in turn has to compute parameters to send to its various neighbours. A vertex V_i can compute the diagnostic parameter to send to a predecessor V_{j_k} from its own probability assessment function, its compound diagnostic parameter, and the causal parameters it receives from its other predecessors.

Lemma 4.2.14 *Let $B = (G, \Gamma)$ be a probabilistic network where $G = (V(G), A(G))$ is a singly connected graph. Let $V_i \in V(G)$ be a vertex with predecessors $\rho(V_i) = \{V_{j_1}, \ldots, V_{j_n}\}$, $n \geq 1$, in G. Let λ_{V_i} be the compound diagnostic parameter for vertex V_i, and let $\pi^{V_{j_l}}_{V_i}$ be the causal parameter from $V_{j_l} \in \rho(V_i)$ to V_i. Futhermore, for each $V_{j_k} \in \rho(V_i)$, let $\lambda^{V_{j_k}}_{V_i}$ be the diagnostic parameter from V_i to V_{j_k}. Then,*

$$\begin{aligned}\lambda^{V_{j_k}}_{V_i}(V_{j_k}) &= \alpha \cdot \sum_{c_{V_i}} \lambda_{V_i}(c_{V_i}) \cdot \Bigg[\sum_{c_{\rho(V_i)\setminus\{V_{j_k}\}}} \Big(\gamma_{V_i}(c_{V_i} \mid c_{\rho(V_i)\setminus\{V_{j_k}\}} \wedge V_{j_k}) \cdot \\ &\qquad \cdot \prod_{l=1,\ldots,n, l\neq k} \pi^{V_{j_l}}_{V_i}(c_{V_{j_l}}) \Big) \Bigg]\end{aligned}$$

where $c_{\rho(V_i)\setminus\{V_{j_k}\}} = \bigwedge_{l=1,\ldots,n,l\neq k} c_{V_{j_l}}$ and α is a normalisation constant.

Proof. Let Pr be the joint probability distribution defined by the probabilistic network B. Let V_i^-, G^- and G^+ be as before. Then, by definition we have that

$$\lambda^{V_{j_k}}_{V_i}(V_{j_k}) \;=\; \Pr(\tilde{c}_{V(G^-_{(V_{j_k},V_i)})} \mid V_{j_k}) \;=\; \Pr(\tilde{c}_{V_i^-} \wedge \tilde{c}_{V_i^{+\setminus k}}) \mid V_{j_k}),$$

where $V_i^{+\setminus k} = \bigcup_{h=1,\ldots,n,h\neq k} V(G^+_{(V_{j_h},V_i)})$ captures the vertices in the upper graphs of all predecessors of V_i except V_{j_k}.

By conditioning on the values of V_i, we now find

$$\lambda^{V_{j_k}}_{V_i}(V_{j_k}) = \sum_{c_{V_i}} \Pr(\tilde{c}_{V_i^-} \wedge \tilde{c}_{V_i^{+\setminus k}} \mid c_{V_i} \wedge V_{j_k}) \cdot \Pr(c_{V_i} \mid V_{j_k})$$

Using d-separation, it now follows that given V_i, V_i^- is independent of $V \setminus V_i^-$. So,

$$\begin{aligned}\lambda^{V_{j_k}}_{V_i}(V_{j_k}) &= \sum_{c_{V_i}} \Pr(\tilde{c}_{V_i^-} \mid c_{V_i} \wedge V_{j_k}) \cdot \Pr(\tilde{c}_{V_i^{+\setminus k}} \mid c_{V_i} \wedge V_{j_k}) \cdot \Pr(c_{V_i} \mid V_{j_k}) = \\ &= \sum_{c_{V_i}} \Pr(\tilde{c}_{V_i^-} \mid c_{V_i}) \cdot \Pr(c_{V_i} \mid V_{j_k}) \cdot \frac{\Pr(c_{V_i} \mid \tilde{c}_{V_i^{+\setminus k}} \wedge V_{j_k}) \cdot \Pr(\tilde{c}_{V_i^{+\setminus k}} \mid V_{j_k})}{\Pr(c_{V_i} \mid V_{j_k})}\end{aligned}$$

Vertex V_i computes the probabilities $Pr(\tilde{c}_{V_i^-} \mid c_{V_i})$ as function values of its compound diagnostic parameter λ_{V_i}. d-Separation tells us that all predecessors of V_i are independent of each other, so

$$\Pr(\tilde{c}_{V_i^{+\setminus k}} \mid V_{j_k}) = \prod_{h=1,\ldots,n,h\neq k} \Pr(\tilde{c}_{V(G^+_{(V_{j_h},V_i)})})$$

The latter factor is constant for V_i and V_{i_j} and may therefore be looked upon as a normalisation constant, which we denote by α.[1]

We now focus on the term $\Pr(c_{V_i} \mid \tilde{c}_{V_i^{+\setminus k}} \wedge V_{j_k})$ and condition on the values of all predecessors of V_i except V_{j_k}:

$$\sum_{c_{\rho(V_i)\setminus\{V_{j_k}\}}} \Pr(c_{V_i} \mid \tilde{c}_{V_i^{+\setminus k}} \wedge V_{j_k} \wedge c_{\rho(V_i)\setminus\{V_{j_k}\}}) \cdot \Pr(c_{\rho(V_i)\setminus\{V_{j_k}\}} \mid \tilde{c}_{V_i^{+\setminus k}} \wedge V_{j_k})$$

Using d-separation, it now follows that given V_i's predecessors, V_i is independent of $V \setminus V_i^-$; in addition, recall that all predecessors of V_i are independent of each other. So the above term reduces to

$$\begin{aligned} &\sum_{c_{\rho(V_i)\setminus\{V_{j_k}\}}} \Pr(c_{V_i} \mid V_{j_k} \wedge c_{\rho(V_i)\setminus\{V_{j_k}\}}) \cdot \prod_{l=1,\ldots,n,l\neq k} \Pr(c_{V_{j_l}} \mid \tilde{c}_{V_i^{+\setminus k}} \wedge V_{j_k}) \\ = &\sum_{c_{\rho(V_i)\setminus\{V_{j_k}\}}} \Pr(c_{V_i} \mid V_{j_k} \wedge c_{\rho(V_i)\setminus\{V_{j_k}\}}) \cdot \prod_{l=1,\ldots,n,l\neq k} \Pr(c_{V_{j_l}} \mid \tilde{c}_{V(G^+_{(V_{j_l},V_i)})}) \end{aligned}$$

The probabilities $\Pr(c_{V_i} \mid V_{j_k} \wedge c_{\rho(V_i)\setminus\{V_{j_k}\}})$ have been specified as function values of the probability assessment function γ_{V_i} associated with vertex V_i and hence are directly available to V_i. In addition, V_i receives the probabilities $\Pr(c_{V_{j_l}} \mid \tilde{c}_{V(G^+_{(V_{j_l},V_i)})})$ from each of its predecessors $\rho(V_i) \setminus \{V_{j_k}\}$ as function values of their causal parameters $\pi_{V_i}^{V_{j_l}}$. The property stated in the lemma now follows by substitution. □

Finally, if we consider a vertex V_i with successors $\sigma(V_i) = \{V_{i_1}, \ldots, V_{i_m}\}$ in the digraph, then the causal parameter vertex V_i has to send to a successor, is computed from its compound causal parameter and the diagnostic parameters it receives from its other successors; this property is analogous to the property stated in Lemma 4.2.7:

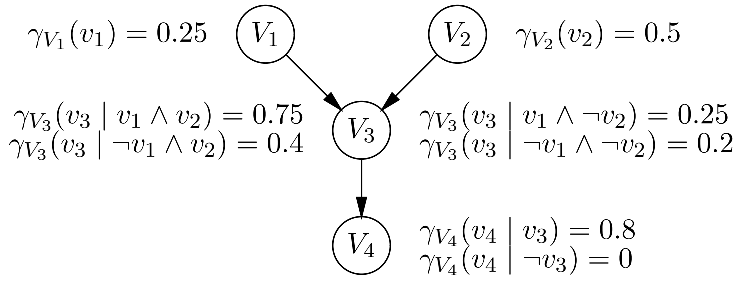
$$\pi_{V_{i_j}}^{V_i}(V_i) = \alpha \cdot \pi_{V_i}(V_i) \cdot \prod_{k=1,\ldots,m,k\neq j} \lambda_{V_{i_k}}^{V_i}(V_i)$$

where α once more is a normalisation constant.

The data fusion lemma and the four computation rules mentioned above with each other constitute Pearl's algorithm for probabilistic inference with a probabilistic network comprising a singly connected digraph for its qualitative part. Once more, these computation rules provide for computing probabilities as well as for processing evidence in such a probabilistic network. The computation rules for the separate causal and diagnostic parameters enable a vertex to correctly pass on the impact of a piece of evidence to its neighbours. As feedback and circular reasoning once more are excluded, evidence is thus propagated throughout the network in a single pass.

Example 4.2.15 We consider once more the probabilistic network $B = (G, \Gamma)$ from Example 4.1.2; for ease of reference, the network is reproduced in Figure 4.8. Note that the digraph G of B is not a directed tree, yet is singly connected.

[1]Note that in this case α cannot be trivially computed, since $\lambda_{V_i}^{V_{j_k}}(v_{j_k}) + \lambda_{V_i}^{V_{j_k}}(\neg v_{j_k})$ typically does not equal 1. For this reason, in applying Pearl's algorithm, normalisation is usually postponed to the final step of data-fusion.



$\gamma_{V_1}(v_1) = 0.25$
V_1
V_2
$\gamma_{V_2}(v_2) = 0.5$
$\gamma_{V_3}(v_3 \mid v_1 \wedge v_2) = 0.75$
$\gamma_{V_3}(v_3 \mid \neg v_1 \wedge v_2) = 0.4$
V_3
$\gamma_{V_3}(v_3 \mid v_1 \wedge \neg v_2) = 0.25$
$\gamma_{V_3}(v_3 \mid \neg v_1 \wedge \neg v_2) = 0.2$
V_4
$\gamma_{V_4}(v_4 \mid v_3) = 0.8$
$\gamma_{V_4}(v_4 \mid \neg v_3) = 0$

Figure 4.8: The Example Probabilistic Network Reproduced.

Let Pr be the joint probability distribution defined by the network B. We address the computation of the probabilities $\Pr(v_3)$ and $\Pr(\neg v_3)$ using Pearl's algorithm. Vertex V_3 sets out to compute the probabilities of interest by application of the data fusion lemma:

$$\begin{aligned}\Pr(v_3) &= \alpha \cdot \pi_{V_3}(v_3) \cdot \lambda_{V_3}(v_3)\\ \Pr(\neg v_3) &= \alpha \cdot \pi_{V_3}(\neg v_3) \cdot \lambda_{V_3}(\neg v_3)\end{aligned}$$

Vertex V_3 now needs to compute its compound parameters π_{V_3} and λ_{V_3}. Since no evidence has been entered into the network as yet, we have that the values of the compound diagnostic parameter λ_{V_3} equal

$$\begin{aligned}\lambda_{V_3}(v_3) &= 1\\ \lambda_{V_3}(\neg v_3) &= 1\end{aligned}$$

Vertex V_3 computes the values of its compound causal parameter π_{V_3} from

$$\begin{aligned}\pi_{V_3}(v_3) &= \gamma_{V_3}(v_3 \mid v_1 \wedge v_2) \cdot \pi^{V_1}_{V_3}(v_1) \cdot \pi^{V_2}_{V_3}(v_2) +\\ &\quad + \gamma_{V_3}(v_3 \mid \neg v_1 \wedge v_2) \cdot \pi^{V_1}_{V_3}(\neg v_1) \cdot \pi^{V_2}_{V_3}(v_2) +\\ &\quad + \gamma_{V_3}(v_3 \mid v_1 \wedge \neg v_2) \cdot \pi^{V_1}_{V_3}(v_1) \cdot \pi^{V_2}_{V_3}(\neg v_2) +\\ &\quad + \gamma_{V_3}(v_3 \mid \neg v_1 \wedge \neg v_2) \cdot \pi^{V_1}_{V_3}(\neg v_1) \cdot \pi^{V_2}_{V_3}(\neg v_2)\\ \pi_{V_3}(\neg v_3) &= \gamma_{V_3}(\neg v_3 \mid v_1 \wedge v_2) \cdot \pi^{V_1}_{V_3}(v_1) \cdot \pi^{V_2}_{V_3}(v_2) +\\ &\quad + \gamma_{V_3}(\neg v_3 \mid \neg v_1 \wedge v_2) \cdot \pi^{V_1}_{V_3}(\neg v_1) \cdot \pi^{V_2}_{V_3}(v_2) +\\ &\quad + \gamma_{V_3}(\neg v_3 \mid v_1 \wedge \neg v_2) \cdot \pi^{V_1}_{V_3}(v_1) \cdot \pi^{V_2}_{V_3}(\neg v_2) +\\ &\quad + \gamma_{V_3}(\neg v_3 \mid \neg v_1 \wedge \neg v_2) \cdot \pi^{V_1}_{V_3}(\neg v_1) \cdot \pi^{V_2}_{V_3}(\neg v_2)\end{aligned}$$

It is readily seen that vertex V_3 receives from its predecessor V_1 the causal parameter $\pi^{V_1}_{V_3}$ with the values

$$\begin{aligned}\pi^{V_1}_{V_3}(v_1) &= 0.25\\ \pi^{V_1}_{V_3}(\neg v_1) &= 0.75\end{aligned}$$

From its predecessor V_2, vertex V_3 receives the causal parameter $\pi^{V_2}_{V_3}$ with the values

$$\begin{aligned}\pi^{V_2}_{V_3}(v_2) &= 0.5\\ \pi^{V_2}_{V_3}(\neg v_2) &= 0.5\end{aligned}$$

Substitution now yields the following values for V_3's compound causal parameter:

$$\begin{aligned}\pi_{V_3}(v_3) &= 0.75 \cdot 0.25 \cdot 0.5 + 0.4 \cdot 0.75 \cdot 0.5 + 0.25 \cdot 0.25 \cdot 0.5 + 0.2 \cdot 0.75 \cdot 0.5 =\\ &= 0.35\\ \pi_{V_3}(\neg v_3) &= 0.25 \cdot 0.25 \cdot 0.5 + 0.6 \cdot 0.75 \cdot 0.5 + 0.75 \cdot 0.25 \cdot 0.5 + 0.8 \cdot 0.75 \cdot 0.5 =\\ &= 0.65\end{aligned}$$

Subsequent substitution of the values of the compound parameters into the data fusion lemma and elimination of the normalisation constant α yields

$$\begin{aligned}\Pr(v_3) &= 0.35\\ \Pr(\neg v_3) &= 0.65\end{aligned}$$

Now, suppose that the evidence $V_3 = \mathit{true}$ is observed and entered into the probabilistic network. We address the computation of the posterior probabilities $\Pr^{v_3}(v_1)$ and $\Pr^{v_3}(\neg v_1)$ using Pearl's algorithm. Vertex V_1 sets out to compute the probabilities of interest by application of the data fusion lemma:

$$\begin{aligned}\Pr^{v_3}(v_1) &= \alpha \cdot \pi_{V_1}(v_1) \cdot \lambda_{V_1}(v_1)\\ \Pr^{v_3}(\neg v_1) &= \alpha \cdot \pi_{V_1}(\neg v_1) \cdot \lambda_{V_1}(\neg v_1)\end{aligned}$$

Vertex V_1 now has to compute its compound parameters π_{V_1} and λ_{V_1}. For the compound causal parameter, it finds the values

$$\begin{aligned}\pi_{V_1}(v_1) &= 0.25\\ \pi_{V_1}(\neg v_1) &= 0.75\end{aligned}$$

The values of the compound diagnostic parameter λ_{V_1} are computed from

$$\begin{aligned}\lambda_{V_1}(v_1) &= \lambda_{V_3}^{V_1}(v_1)\\ \lambda_{V_1}(\neg v_1) &= \lambda_{V_3}^{V_1}(\neg v_1)\end{aligned}$$

From its successor V_3, vertex V_1 receives the diagnostic parameter $\lambda_{V_3}^{V_1}$ with the values

$$\begin{aligned}\lambda_{V_3}^{V_1}(v_1) = {} & \lambda_{V_3}(v_3) \cdot \left[\gamma_{V_3}(v_3 \mid v_1 \wedge v_2) \cdot \pi_{V_3}^{V_2}(v_2) + \gamma_{V_3}(v_3 \mid v_1 \wedge \neg v_2) \cdot \pi_{V_3}^{V_2}(\neg v_2)\right]\\ & + \lambda_{V_3}(\neg v_3) \cdot \left[\gamma_{V_3}(\neg v_3 \mid v_1 \wedge v_2) \cdot \pi_{V_3}^{V_2}(v_2) + \gamma_{V_3}(\neg v_3 \mid v_1 \wedge \neg v_2) \cdot \pi_{V_3}^{V_2}(\neg v_2)\right]\\ = {} & \gamma_{V_3}(v_3 \mid v_1 \wedge v_2) \cdot \pi_{V_3}^{V_2}(v_2) + \gamma_{V_3}(v_3 \mid v_1 \wedge \neg v_2) \cdot \pi_{V_3}^{V_2}(\neg v_2) =\\ = {} & 0.75 \cdot 0.5 + 0.25 \cdot 0.5 = 0.5\\ \lambda_{V_3}^{V_1}(\neg v_1) = {} & \lambda_{V_3}(v_3) \cdot \left[\gamma_{V_3}(v_3 \mid \neg v_1 \wedge v_2) \cdot \pi_{V_3}^{V_2}(v_2) + \gamma_{V_3}(v_3 \mid \neg v_1 \wedge \neg v_2) \cdot \pi_{V_3}^{V_2}(\neg v_2)\right]\\ & + \lambda_{V_3}(\neg v_3) \cdot \left[\gamma_{V_3}(\neg v_3 \mid \neg v_1 \wedge v_2) \cdot \pi_{V_3}^{V_2}(v_2) + \gamma_{V_3}(\neg v_3 \mid \neg v_1 \wedge \neg v_2) \cdot \pi_{V_3}^{V_2}(\neg v_2)\right]\\ = {} & \gamma_{V_3}(v_3 \mid \neg v_1 \wedge v_2) \cdot \pi_{V_3}^{V_2}(v_2) + \gamma_{V_3}(v_3 \mid \neg v_1 \wedge \neg v_2) \cdot \pi_{V_3}^{V_2}(\neg v_2)\\ = {} & 0.4 \cdot 0.5 + 0.2 \cdot 0.5 = 0.3\end{aligned}$$

From the diagnostic parameter $\lambda_{V_3}^{V_1}$ it receives from its successor V_3, vertex V_1 now finds the values of its compound diagnostic parameter to be

$$\begin{aligned}\lambda_{V_1}(v_1) &= 0.5\\ \lambda_{V_1}(\neg v_1) &= 0.3\end{aligned}$$

Substitution of the values of the compound parameters into the data fusion lemma yields

$$\begin{aligned}\Pr^{v_3}(v_1) &= \alpha \cdot 0.25 \cdot 0.5 = \alpha \cdot 0.125\\ \Pr^{v_3}(\neg v_1) &= \alpha \cdot 0.75 \cdot 0.3 = \alpha \cdot 0.225\end{aligned}$$

After elimination of the normalisation constant α (note that we postponed all normalisation until this last step!), vertex V_1 finds

$$\begin{aligned}\Pr^{v_3}(v_1) &= 0.357\\ \Pr^{v_3}(\neg v_1) &= 0.643\end{aligned}$$

□

To conclude our discussion of Pearl's algorithm so far, we take a (superficial) look at the algorithm's computational complexity. An analysis of the computation rules involved in the algorithm serves to show that it has a *worst-case* computational time complexity that is *exponential* in the number of vertices in a network's digraph. We consider, in a probabilistic network with n vertices, a vertex V_i with $O(n)$ predecessors and $O(n)$ successors. For this vertex, computing the compound causal parameter requires at most $O(2^n)$ computations; computing its compound diagnostic parameter requires at most $O(n)$ computations. Vertex V_i can therefore compute the probabilities of its values in at most $O(2^n)$ time. The computation of a single diagnostic parameter from V_i requires at most $O(2^n)$ computations; the computation of a single causal parameter requires only constant time. Vertex V_i can therefore compute all parameters to send to its various neighbours in at most $O(n \cdot 2^n)$ time. Propagating a single piece of evidence throughout the network may thus require $O(n^2 \cdot 2^n)$ computations. The exponential factor in the computational time complexity of Pearl's algorithm arises solely from the number of predecessors a vertex can have in a network's digraph: if the predecessor sets are bounded in size by a constant, then the algorithm has a *linear* runtime complexity.

4.2.3 Multiply Connected Digraphs

In the previous section, we have detailed Pearl's algorithm for probabilistic inference with a probabilistic network comprising a singly connected digraph for its qualitative part. We will now extend the algorithm to apply to probabilistic networks of which the qualitative part is a *multiply connected digraph*.

We recall from our discussion of Pearl's algorithm so far, that the computation rules for probabilistic inference with a probabilistic network comprising a singly connected digraph derive from exploiting independences that are read from the network's graph by local inspection of a vertex' incoming and outgoing arcs. These computation rules therefore make use explicitly of the property that in a singly connected digraph there is at most one chain between any two vertices. We observe that this property does not hold in a multiply connected digraph as in such a graph there may be multiple chains between vertices. Local inspection of a vertex' incident arcs now no longer suffices for reading independences from a probabilistic network's digraph. In fact, application of Pearl's algorithm discussed so far to a probabilistic network comprising a multiply connected digraph inevitably leads to various problems due to the presence of loops in the network's digraph [Suermondt & Cooper, 1990]: vertices may indefinitely send newly updated messages, originating from the same evidence, to their neighbours, causing the network never to reach a new equilibrium, and even if the network *does* reach an equilibrium, it is not guaranteed to reflect the correct updated joint probability distribution. Pearl's algorithm, therefore, cannot be extended based on similar ideas as before to apply to networks with a multiply connected digraph: the algorithm has to be supplemented with an additional method for coping with the loops in such a digraph.

The best-known method for coping with loops in probabilistic inference is the method of *loop cutset conditioning* [Pearl, 1988]. The idea underlying the method of loop cutset conditioning is that of *reasoning by assumption*. For a multiply connected digraph of a probabilistic network, vertices are selected that, upon instantiation, with each other effectively 'cut' all loops of the digraph and cause it to 'behave' as if it were singly connected; the selected vertices are said to constitute a *loop cutset* for the network's digraph. During probabilistic inference with the network, each configuration of the selected loop cutset is looked upon as a (compound) assumption on which reasoning is performed: for each vertex, the probabilities of its values are computed by conditioning successively on all possible configurations of the loop cutset and subsequently weighting the results obtained with the probabilities of these configurations. Since upon instantiation of the loop cutset the digraph behaves as if it were singly connected, the probabilities of a vertex' values can simply be computed from the network using Pearl's basic algorithm for probabilistic networks with a singly connected digraph.

Loop Cutsets

We begin our discussion of the method of loop cutset conditioning by detailing the concept of a loop cutset for a probabilistic network's multiply connected digraph. A loop cutset has to provide, upon instantiation, for the multiply connected digraph to behave as if it were singly connected, regardless of the evidence that has been entered into the network at hand. Note that a loop cutset then serves to prevent, at any time, probabilistic information originating at a single vertex in the digraph to reach another vertex along multiple chains. For this purpose, a loop cutset has to cut, or *block*, every cyclic chain, or loop, in the network's digraph. Note that it does not suffice to block a cyclic chain by the empty set as this chain may become unblocked as further evidence is entered into the network. From these observations, we find that a loop cutset has to block all cyclic chains in the digraph in such a way that each such chain contains at least one vertex from the loop cutset that has an outgoing arc on the chain.

Definition 4.2.16 *Let $G = (V(G), A(G))$ be an acyclic multiply connected digraph. A set of vertices $Cs(G) \subseteq V(G)$ is called a* loop cutset *for G if each loop s in G contains three consecutive vertices X_1, X_2, X_3, for which one of the following conditions holds:*

- *arcs (X_2, X_1) and (X_2, X_3) are on the loop s, and $X_2 \in Cs(G)$;*
- *arcs (X_1, X_2) and (X_2, X_3) are on the loop s, and $X_2 \in Cs(G)$.*

Note that a probabilistic network's multiply connected digraph G may allow several different loop cutsets; in fact, any superset of a loop cutset for G is also a loop cutset for G. Also note that any loop cutset for G is a *non-empty* set of vertices.

Example 4.2.17 We consider the multiply connected digraph G shown in Figure 4.9. The sets of vertices $\{V_2, V_6, V_9\}$, $\{V_3, V_4, V_9\}$, and $\{V_2, V_8, V_{10}, V_{11}\}$ are example loop cutsets for G. $\square$

Loop Cutset Conditioning

After having detailed the concept of a loop cutset, we now turn to the method of loop cutset conditioning itself. To this end, we consider a probabilistic network $B = (G, \Gamma)$ where $G = (V(G), A(G))$ is an acyclic multiply connected digraph; let Pr be the joint probability distribution defined by B. Furthermore, suppose that we have the loop cutset $Cs(G)$ for the digraph G of the network, as defined above. We now address computing the probabilities of the values of some vertex V_i in G, that is, we consider the computation of the probabilities $\Pr(V_i \mid \tilde{c}_{V(G)})$. As the digraph G is multiply connected, these probabilities cannot be computed directly from the network using Pearl's algorithm. Since the loop cutset has been chosen so as to block, upon

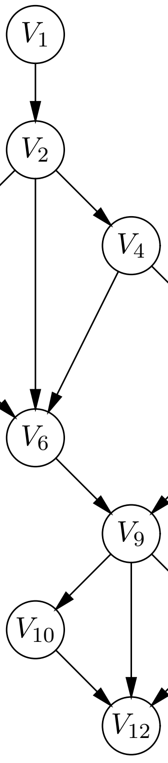


Figure 4.9: An Example Multiply Connected Digraph.

instantiation, all cyclic chains in the digraph, we observe that any probability that is conditioned on a configuration of (at least)the loop cutset can be computed from the network directly. Now, by conditioning the probabilities $\Pr(V_i \mid \tilde{c}_{V(G)})$ of interest on the various configurations of the loop cutset $Cs(G)$ for G, we find

$$\Pr(V_i \mid \tilde{c}_{V(G)}) = \sum_{c_{Cs(G)}} \Pr(V_i \mid \tilde{c}_{V(G)} \wedge c_{Cs(G)}) \cdot \Pr(c_{Cs(G)} \mid \tilde{c}_{V(G)})$$

The probabilities $\Pr(V_i \mid \tilde{c}_{V(G)} \wedge c_{Cs(G)})$ are now conditioned on a configuration of (at least) the loop cutset and therefore can be computed directly from the probabilistic network using Pearl's basic algorithm. The probabilities $\Pr(c_{Cs(G)} \mid \tilde{c}_{V(G)})$, however, cannot be computed from the probabilistic network directly. For computing these probabilities, the following recursive computation rules are used:

- the prior probabilities $\Pr(c_{CS(G)})$ are computed from the joint probability distribution Pr using the property stated in Proposition 4.1.3 [Suermondt & Cooper, 1991].
- for the first piece of evidence c_1 entered into the probabilistic network, the (updated) probabilities of the configurations of the loop cutset are computed from

$$\Pr(c_{Cs(G)} \mid c_1) = \alpha \cdot \Pr(c_1 \mid c_{Cs(G)}) \cdot \Pr(c_{Cs(G)})$$

where α is a normalisation constant. The probabilities $\Pr(c_1 \mid c_{Cs(G)})$ are computed directly from the network using Pearl's basic algorithm for singly connected digraphs; the probabilities $\Pr(c_{Cs(G)})$ have been computed before and, hence, are already available.

- for the j-th piece of evidence c_j entered into the network, the (updated) probabilities of the configurations of the loop cutset are computed from

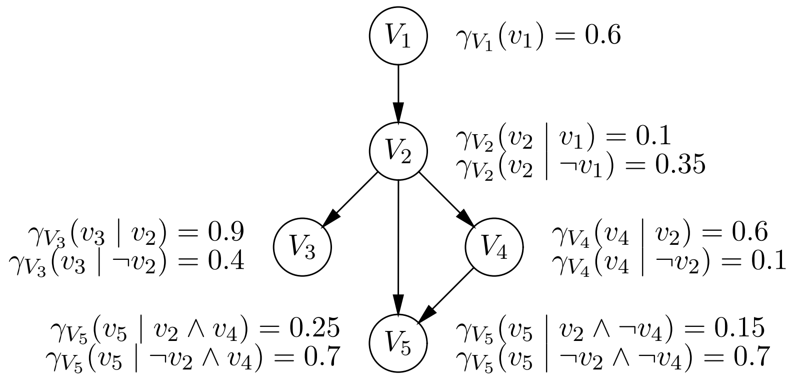
$$\begin{aligned}\Pr(c_{Cs(G)} \mid c_1 \wedge \cdots \wedge c_j) &= \alpha \cdot \Pr(c_j \mid c_{Cs(G)} \wedge c_1 \wedge \cdots \wedge c_{j-1}) \cdot \\ &\quad \cdot \Pr(c_{Cs(G)} \mid c_1 \wedge \cdots \wedge c_{j-1})\end{aligned}$$

  where α once more is a normalisation constant. The probabilities $\Pr(c_j \mid c_{Cs(G)} \wedge c_1 \wedge \cdots \wedge c_{j-1})$ are computed directly from the probabilistic network using Pearl's basic algorithm; the probability $\Pr(c_{Cs(G)} \mid c_1 \wedge \cdots \wedge c_{j-1})$ has been computed before in the previous step of the recursion and need not be re-computed.

Note that the probabilities of the configurations of a digraph's loop cutset involve information from the probability distribution concerning various vertices combined. These probabilities therefore cannot be computed from the network by local message passing between neighbouring vertices alone. For this purpose, the computational architecture is enhanced with a *global supervisor* that is capable of performing the recursive computation rules outlined above.

We briefly illustrate the method of loop cutset conditioning by means of an example.

Example 4.2.18 We consider the probabilistic network $B = (G, \Gamma)$ shown in Figure 4.10. Let Pr be the joint probability distribution defined by B. We address the computation of the probabilities $\Pr(v_4)$ and $\Pr(\neg v_4)$ using Pearl's algorithm.



V_1
$\gamma_{V_1}(v_1) = 0.6$
V_2
$\gamma_{V_2}(v_2 \mid v_1) = 0.1$
$\gamma_{V_2}(v_2 \mid \neg v_1) = 0.35$
$\gamma_{V_3}(v_3 \mid v_2) = 0.9$
$\gamma_{V_3}(v_3 \mid \neg v_2) = 0.4$
V_3
V_4
$\gamma_{V_4}(v_4 \mid v_2) = 0.6$
$\gamma_{V_4}(v_4 \mid \neg v_2) = 0.1$
$\gamma_{V_5}(v_5 \mid v_2 \wedge v_4) = 0.25$
$\gamma_{V_5}(v_5 \mid \neg v_2 \wedge v_4) = 0.7$
V_5
$\gamma_{V_5}(v_5 \mid v_2 \wedge \neg v_4) = 0.15$
$\gamma_{V_5}(v_5 \mid \neg v_2 \wedge \neg v_4) = 0.7$

Figure 4.10: An Example Probabilistic Network.

We begin by observing that the digraph G of the probabilistic network is multiply connected. To compute the probabilities of interest from the network, we enhance Pearl's algorithm with the method of loop cutset conditioning. Suppose that for this purpose we choose for the digraph G the loop cutset $Cs(G) = \{V_2\}$. The global supervisor of the architecture now starts the computation by computing the probabilities of the configurations of the loop cutset. To this end, it employs the property stated in Proposition 4.1.3 to find that

$$\begin{aligned}\Pr(v_2) &= \Pr(v_1 \wedge v_2) + \Pr(\neg v_1 \wedge v_2) = \\ &= \gamma_{V_2}(v_2 \mid v_1) \cdot \gamma_{V_1}(v_1) + \gamma_{V_2}(v_2 \mid \neg v_1) \cdot \Pr(\neg v_1) = \\ &= 0.1 \cdot 0.6 + 0.35 \cdot 0.4 = 0.2 \\ \Pr(\neg v_2) &= \Pr(v_1 \wedge \neg v_2) + \Pr(\neg v_1 \wedge \neg v_2) = \\ &= \gamma_{V_2}(\neg v_2 \mid v_1) \cdot \gamma_{V_1}(v_1) + \gamma_{V_2}(\neg v_2 \mid \neg v_1) \cdot \Pr(\neg v_1) = \\ &= 0.9 \cdot 0.6 + 0.65 \cdot 0.4 = 0.8\end{aligned}$$

The probabilities of interest are now computed by conditioning on the different configurations of the chosen loop cutset:

$$\begin{aligned}\Pr(v_4) &= \Pr(v_4 \mid v_2) \cdot \Pr(v_2) + \Pr(v_4 \mid \neg v_2) \cdot \Pr(\neg v_2)\\ \Pr(\neg v_4) &= \Pr(\neg v_4 \mid v_2) \cdot \Pr(v_2) + \Pr(\neg v_4 \mid \neg v_2) \cdot \Pr(\neg v_2)\end{aligned}$$

The global supervisor *provisionally* enters the value *true* for vertex V_2 into the network to enable vertex V_4 to compute the probabilities $\Pr(v_4 \mid v_2)$ and $\Pr(\neg v_4 \mid v_2)$. Vertex V_4 computes these probabilities to be

$$\begin{aligned}\Pr(v_4 \mid v_2) &= 0.6\\ \Pr(\neg v_4 \mid v_2) &= 0.4\end{aligned}$$

The global supervisor now retracts the value *true* for vertex V_2 and subsequently enters the value *false* for the vertex. It now requests vertex V_4 to compute the probabilities $\Pr(v_4 \mid \neg v_2)$ and $\Pr(\neg v_4 \mid \neg v_2)$. Vertex V_4 returns

$$\begin{aligned}\Pr(v_4 \mid \neg v_2) &= 0.1\\ \Pr(\neg v_4 \mid \neg v_2) &= 0.9\end{aligned}$$

The global supervisor substitutes the probabilities returned by vertex V_4 and the probabilities it has itself computed for the loop cutset, into the conditioning formula to find

$$\begin{aligned}\Pr(v_4) &= 0.6 \cdot 0.2 + 0.1 \cdot 0.8 = 0.2\\ \Pr(\neg v_4) &= 0.4 \cdot 0.2 + 0.9 \cdot 0.8 = 0.8\end{aligned}$$

Now suppose that the evidence $V_3 = \textit{true}$ is observed and entered into the probabilistic network B. We address the computation of the updated probabilities of the values of vertex V_4. The global supervisor begins with updating the probabilities of the configurations of the loop cutset:

$$\begin{aligned}\Pr(v_2 \mid v_3) &= \alpha \cdot \Pr(v_3 \mid v_2) \cdot \Pr(v_2)\\ \Pr(\neg v_2 \mid v_3) &= \alpha \cdot \Pr(v_3 \mid \neg v_2) \cdot \Pr(\neg v_2)\end{aligned}$$

where α is a normalisation constant. The supervisor requests from vertex V_3 the probabilities $\Pr(v_3 \mid v_2)$ and $\Pr(v_3 \mid \neg v_2)$; to enable vertex V_3 to compute these probabilities, it provisionally enters and retracts the values *true* and *false* for vertex V_2, respectively. Vertex V_3 computes the requested probabilities to be

$$\begin{aligned}\Pr(v_3 \mid v_2) &= 0.9\\ \Pr(v_3 \mid \neg v_2) &= 0.4\end{aligned}$$

After substitution and subsequent elimination of the normalisation constant α, the global supervisor finds

$$\begin{aligned}\Pr(v_2 \mid v_3) &= 0.36\\ \Pr(\neg v_2 \mid v_3) &= 0.64\end{aligned}$$

The probabilities of interest are now computed by conditioning as before:

$$\begin{aligned}\Pr^{v_3}(v_4) &= \Pr(v_4 \mid v_2 \wedge v_3) \cdot \Pr(v_2 \mid v_3) + \Pr(v_4 \mid \neg v_2 \wedge v_3) \cdot \Pr(\neg v_2 \mid v_3) =\\ &= 0.6 \cdot 0.36 + 0.1 \cdot 0.64 = 0.28\\ \Pr^{v_3}(\neg v_4) &= \Pr(\neg v_4 \mid v_2 \wedge v_3) \cdot \Pr(v_2 \mid v_3) + \Pr(\neg v_4 \mid \neg v_2 \wedge v_3) \cdot \Pr(\neg v_2 \mid v_3) =\\ &= 0.4 \cdot 0.36 + 0.9 \cdot 0.64 = 0.72\end{aligned}$$

□

To conclude our discussion of the enhancement of Pearl's algorithm for probabilistic inference with the method of loop cutset conditioning, we take a (superficial) look at the algorithm's computational complexity. We recall that analysis of Pearl's algorithm for probabilistic networks comprising a singly connected digraph has revealed that a vertex can compute the probabilities of its values in at most $O(2^n)$ time, where n is the number of vertices in the network's digraph. Propagating a piece of evidence throughout the network may require as many as $O(n^2 \cdot 2^n)$ computations. An analysis of the method of loop cutset conditioning now serves to show that the method adds a factor to the basic algorithm's computational complexity that is *exponential* in the size of the loop cutset employed. Initialising the probabilities for the loop cutset's configurations requires $O(2^l)$ computations, where l is the number of vertices in the loop cutset. Computing the probabilities of a vertex' values now takes $O(2^{n+l})$ time, since these probabilities have to be computed conditioned on every possible configuration of the loop cutset. Propagating a piece of evidence requires at most $O(n^2 \cdot 2^{n+l})$ computations.

Finding a Loop Cutset

In the foregoing, we have argued that the computational complexity of Pearl's algorithm enhanced with loop cutset conditioning, for probabilistic inference with probabilistic networks comprising a multiply connected digraph, is exponential in the size of the loop cutset employed. From a computational point of view, therefore, the best loop cutset to use in practical applications is a loop cutset having the smallest number of vertices. A loop cutset with the smallest number of vertices is called an *optimal* loop cutset; we define the concept of an optimal loop cutset more formally.

Definition 4.2.19 *Let G be an acyclic multiply connected digraph. A loop cutset $Cs(G)$ for G is called* minimal *for G if no proper subset of $Cs(G)$ is a loop cutset for G; the loop cutset $Cs(G)$ is called* optimal *for G if for any other loop cutset $Cs'(G)$ for G we have that $|Cs(G)| \leq |Cs'(G)|$.*

Note that an optimal loop cutset for a multiply connected digraph also is a minimal loop cutset for this digraph; the reverse in general is not true. Also note that an optimal loop cutset for a digraph need not be unique.

Example 4.2.20 We consider once more the multiply connected digraph G shown in Figure 4.9. The sets of vertices $\{V_2, V_6, V_9\}$ and $\{V_3, V_4, V_9\}$ are optimal loop cutsets for G. The set of vertices $\{V_2, V_8, V_{10}, V_{11}\}$ is a minimal loop cutset for G that is not optimal. The set of vertices $\{V_2, V_4, V_8, V_9\}$ is a loop cutset that is neither optimal nor minimal for G. $\square$

Unfortunately, the problem of finding for an acyclic multiply connected digraph an optimal loop cutset is known to be NP-hard [Suermondt & Cooper, 1990]; hence, it is not likely that a generally applicable polynomial-time algorithm for this problem will be found. For the purpose of finding a good loop cutset for a given multiply connected digraph, therefore, generally a *heuristic algorithm* is used. The best known among these algorithms is a heuristic algorithm designed by H.J. Suermondt and G.F. Cooper [Suermondt & Cooper, 1990].

The loop-cutset algorithm of Suermondt and Cooper takes an acyclic multiply connected digraph G for its input and yields a loop cutset $Cs(G)$ for G as its output.

The algorithm is composed of two basic phases that are alternated recursively. In the first phase, the algorithm deletes from the digraph G as many vertices as possible that are not included in any loop; such vertices need not be considered as candidates for the loop cutset as upon instantiation they will never block any loops in G. For this purpose, all vertices that have *at most one* incident arc are removed recursively from G, along with their incident arcs. The second phase of the algorithm is entered with a reduced digraph G' in which each vertex has at least two incident arcs. From among this digraph's vertices, a single vertex is selected for inclusion in the loop cutset in the making. The selection of this vertex is based on a *heuristic criterion* that aims at selecting a vertex that upon instantiation blocks as many loops from the reduced digraph G' as possible. This heuristic criterion is based on the idea that a vertex with a larger number of incident arcs in G' is likely to be included in more loops than a vertex with a smaller number of incident arcs; however, a vertex with two incoming arcs on a loop does not serve to block this loop. The heuristic criterion therefore selects a vertex V_i with highest degree having at most one predecessor. The thus selected vertex is included in the loop cutset in the making. The vertex V_i is subsequently removed from the digraph, along with all its incident arcs. Note that in the thus reduced digraph all loops that are blocked by vertex V_i are effectively cut, leaving only loops that are yet unblocked. The algorithm then returns to the first phase. The two phases of the algorithm are thus alternated recursively until no more vertices remain to be considered in the reduced graph. Figure 4.11 summarises the heuristic loop-cutset algorithm of Suermondt and Cooper in pseudo-code.

```
procedure loop-cutset($G$, $Cs(G)$)
    $Cs(G) := \varnothing$;
    while $V(G) \neq \varnothing$ do
        if there is a vertex $V \in V(G)$ with $degree(V) \leq 1$
        then select vertex $V$
        else $K := \{V_i \mid V_i \in V(G)$ and $V_i$ has $in - degree(V_i) \leq 1\}$;
             select from $K$ a vertex $V$ with $degree(V) \geq degree(V_i)$ for all $V_i \in K$;
             $Cs(G) := Cs(G) \cup \{V\}$
        fi;
        $V(G) := V(G) \setminus \{V\}$;
        $A(G) := A(G) \cap (V(G) \times V(G))$
    od
end
```

Figure 4.11: The Loop-cutset Algorithm of Suermondt and Cooper.

We illustrate the heuristic loop-cutset algorithm outlined above by means of an example.

Example 4.2.21 We consider once again the multiply connected digraph G from Figure 4.9 and address the construction of a loop cutset for G using Suermondt and Cooper's heuristic algorithm.

The algorithm sets out by initialising the loop cutset in the making with the empty set:

$Cs(G) = \varnothing$

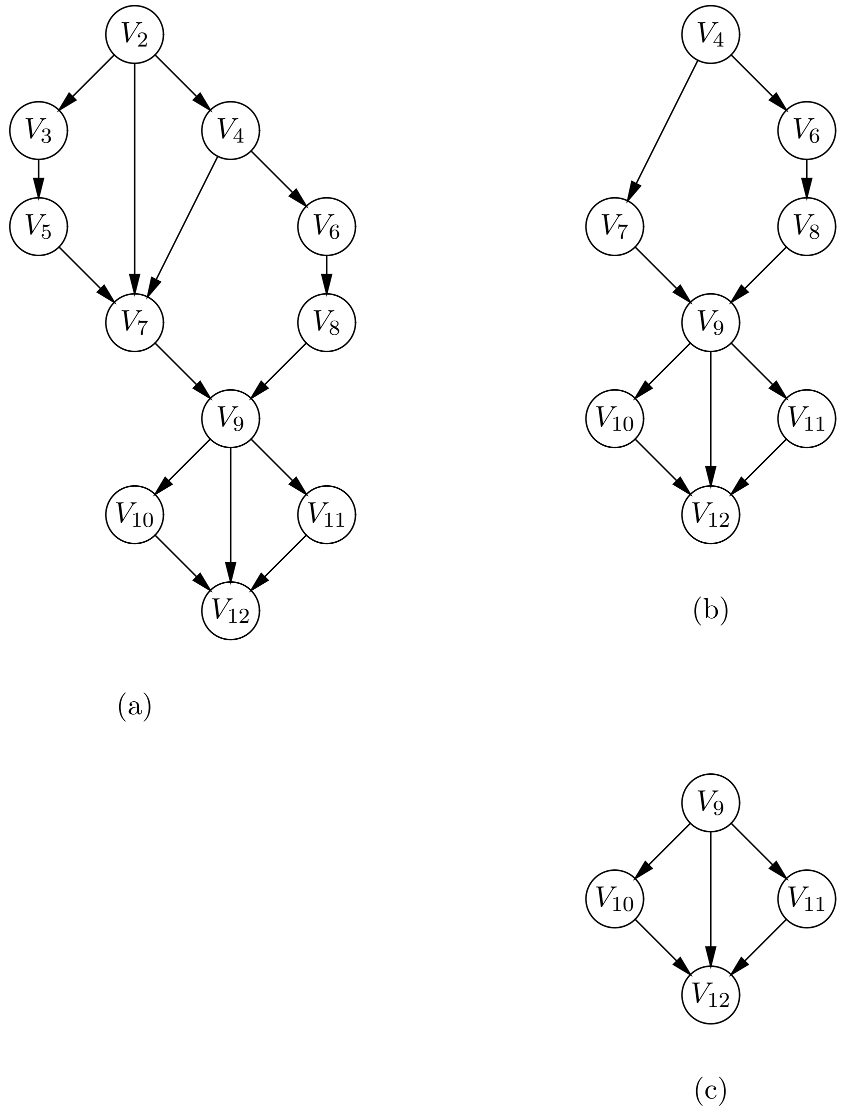


Figure 4.12: Application of the Loop-cutset Algorithm.

The algorithm now notes that the digraph G comprises a single vertex with at most one incident arc — this is the vertex V_1. The algorithm selects this vertex and removes it, along with its incident arc, from G. The thus reduced digraph G_1 is shown in Figure 4.12 (a). From the digraph G_1, the algorithm selects all vertices with at most one incoming arc. These vertices constitute the set of candidates for inclusion in the loop cutset in the making:

$$K_1 = \{V_2, V_3, V_4, V_5, V_6, V_8, V_{10}, V_{11}\}$$

From among these candidates, the algorithm selects a vertex with highest degree. We suppose that the algorithm selects vertex V_2 to include it in the loop cutset in the making:

$$Cs(G) = \{V_2\}$$

The algorithm proceeds by removing from G_1 the selected vertex along with its incident arcs. After subsequently removing all vertices with at most one incident arc, the reduced digraph G_2 shown in Figure 4.12 (b) results. From the digraph G_2, the algorithm once more selects the set of candidates for inclusion in the loop cutset:

$$K_2 = \{V_4, V_6, V_7, V_8, V_{10}, V_{11}\}$$

From among these candidates, the algorithm selects a vertex with highest degree. We suppose that it selects vertex V_6. Vertex V_6 is now added to the loop cutset in the making:

$$Cs(G) = \{V_2, V_6\}$$

After removal of vertex V_6 and its incident arcs from G_2, and recursive removal of all vertices with at most one incident arc, the reduced digraph G_3 shown in Figure 4.12 (c) is yielded. From the digraph G_3, the algorithm selects the set of candidates to be

$$K_3 = \{V_9, V_{10}, V_{11}\}$$

From among these candidates, the algorithm selects vertex V_9 for inclusion in the loop cutset as it has the highest degree:

$$Cs(G) = \{V_2, V_6, V_9\}$$

After removing vertex V_9 with its incident arcs from G_3, subsequent removal of all vertices with at most one incident arc results in the empty graph. The algorithm halts and yields the loop cutset $Cs(G) = \{V_2, V_6, V_9\}$ for the digraph G. Note that the loop cutset yielded by the algorithm is an optimal loop cutset for G. □

The heuristic loop-cutset algorithm of Suermondt and Cooper is correct in the sense that, for any acyclic multiply connected digraph, the algorithm will halt and yield a set of vertices that is a loop cutset of the digraph [Suermondt & Cooper, 1990]. The heuristic algorithm, however, is not guaranteed to always yield an optimal loop cutset. Experimental results have shown that for randomly generated acyclic multiply connected digraphs, the algorithm finds an optimal loop cutset in approximately 70% of the studied digraphs [Suermondt & Cooper, 1990].

4.2.4 Other Algorithms for Probabilistic Inference

In the previous sections, we have discussed Pearl's basic algorithm and its enhancement for general probabilistic inference with a probabilistic network. Pearl's algorithm, however, is not the only algorithm designed for this purpose: several different algorithms have been proposed in the course of the preceding decade. In general, an algorithm for probabilistic inference with a probabilistic network is built from two basic procedures:

- a procedure for (efficiently) computing probabilities of interest from a probabilistic network, and
- a procedure for processing evidence, that is, for entering evidence into the network and subsequently (efficiently) computing the updated joint probability distribution given the evidence.

Note that in Pearl's algorithm these basic procedures are combined into a single set of computation rules where they cannot easily be distinguished. In most other algorithms, however, these basic procedures are more readily discernible. The algorithms proposed further have two important properties in common: the qualitative part of a probabilistic network is exploited more or less directly as a computational architecture, and after a piece of evidence has been processed in the network, again a probabilistic network results. Note that the latter property allows for recursive processing of evidence.

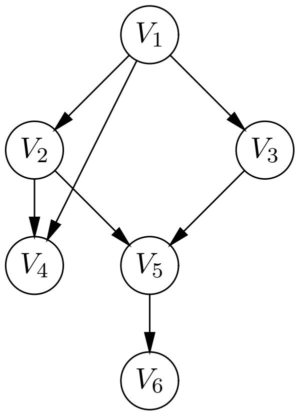
Although all algorithms proposed for probabilistic inference build on the same notion of a probabilistic network, they differ considerably with respect to their underlying concepts. To support this observation, we briefly review another algorithm for probabilistic inference with a probabilistic network, designed by S.L. Lauritzen and D.J. Spiegelhalter [Lauritzen & Spiegelhalter, 1988]. Lauritzen and Spiegelhalter have observed that, after a piece of evidence has become available, updating the joint probability distribution generally entails going against the 'direction' of the initially assessed conditional probabilities. From this observation they have concluded that the *directed* qualitative part of a probabilistic network is not suitable as an architecture for probabilistic inference directly. Their algorithm therefore builds on an *undirected* representation of a joint probability distribution. Prior to probabilistic inference, the original probabilistic network is transformed into a so-called *decomposable* probabilistic network. A decomposable probabilistic network again consists of a qualitative part and a quantitative part. The qualitative part is a decomposable, or *chordal*, graph; the quantitative part is a set of marginal distributions on the vertex sets of the cliques of this graph. The computational architecture for the algorithm derives from the qualitative part of this decomposable probabilistic network in the following sense: the cliques of the decomposable graph are the autonomous objects in the architecture, and the clique intersections give rise to its communication channels. From a decomposable probabilistic network, a probability of interest is computed by further marginalisation of an appropriate marginal distribution. Processing evidence is performed per clique and by message-passing between neighbouring cliques in the architecture.

The various algorithms proposed for probabilistic inference with a probabilistic network also differ with respect to their computational complexity. We note that probabilistic inference with probabilistic networks without any topological restrictions is known to be NP-hard [Cooper, 1990]. All algorithms proposed therefore have an exponential worst-case computational complexity. However, the algorithms can be shown to behave polynomially under certain restrictions. These restrictions concern the topology of a probabilistic network's digraph and differ among the various algorithms. In general, the sparser the digraph of a probabilistic network, the better most algorithms perform. Experience with constructing probabilistic networks for real-life applications so far has indicated that in fact a probabilistic network's digraph tends to be sparse.

Exercises

Exercise 4.1

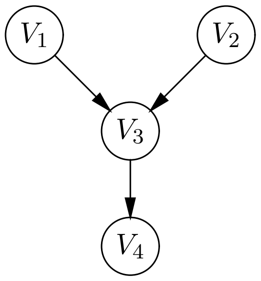
Consider the following acyclic digraph $G = (V(G), A(G))$:



Now suppose that the graph G is a (minimal) directed I-map for the independence relation of the joint probability distribution Pr *on $V(G)$. State which (conditional) probabilities from the distribution* Pr *have to be associated with G to arrive at a probabilistic network comprising G for its qualitative part.*

* Exercise 4.2

Consider the probabilistic network $B = (G, \Gamma)$ where G is the following digraph:



and $\Gamma = \{\gamma_{V_i} \mid V_i \in V(G)\}$ is defined as:

$\gamma_{V_1}(v_1)$	$= 0.25$	$\gamma_{V_2}(v_2)$	$= 0.5$
$\gamma_{V_3}(v_3 \mid v_1 \wedge v_2)$	$= 0.75$	$\gamma_{V_4}(v_4 \mid v_3)$	$= 0.8$
$\gamma_{V_3}(v_3 \mid \neg v_1 \wedge v_2)$	$= 0.4$	$\gamma_{V_4}(v_4 \mid \neg v_3)$	$= 0$
$\gamma_{V_3}(v_3 \mid v_1 \wedge \neg v_2)$	$= 0.25$		
$\gamma_{V_3}(v_3 \mid \neg v_1 \wedge \neg v_2)$	$= 0.2$		

Let Pr *be the joint probability distribution defined by B. Compute the following probabilities by exploiting the property stated in Proposition 4.1.3:*

a. $\Pr(v_1 \wedge v_2 \wedge v_3)$*;*

b. $\Pr(v_2 \wedge v_3)$*;*

c. $\Pr(v_1 \mid v_2 \wedge v_3)$;

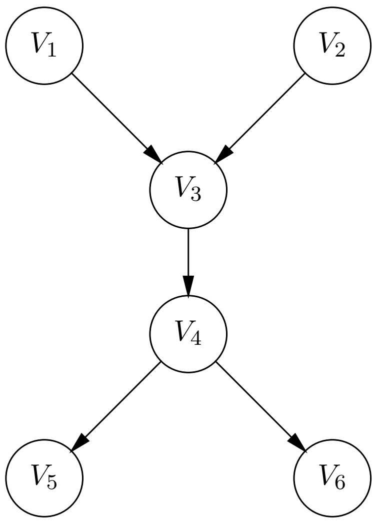
d. $\Pr(v_1 \vee v_2 \vee \neg v_3 \vee v_4)$.

* Exercise 4.3

Let $B = (G, \Gamma)$ be a probabilistic network where $G = (V(G), A(G))$ is a directed tree. Suppose that Pearl's algorithm is employed for probabilistic inference with B. For each vertex $V_i \in V(G)$, let $V_i^- = \sigma^*(V_i)$; let $\lambda_{V_i}(V_i)$ be the compound diagnostic parameter for V_i. Show that if $\tilde{c}_{V_i^-} = \mathsf{True}$, then $\lambda_{V_i}(V_i) = 1$, for all $V_i \in V(G)$.

* Exercise 4.4

Consider the probabilistic network $B = (G, \Gamma)$ where $G = (V(G), A(G))$ is the following digraph:



and $\Gamma = \{\gamma_{V_i} \mid V_i \in V(G)\}$ is defined as

$\gamma_{V_1}(v_1) = 0.5$ $\qquad$ $\gamma_{V_2}(v_2) = 0.4$

$\gamma_{V_3}(v_3 \mid v_1 \wedge v_2) = 0.9$ $\qquad$ $\gamma_{V_4}(v_4 \mid v_3) = 0.5$
$\gamma_{V_3}(v_3 \mid \neg v_1 \wedge v_2) = 0.6$ $\qquad$ $\gamma_{V_4}(v_4 \mid \neg v_3) = 0.65$
$\gamma_{V_3}(v_3 \mid v_1 \wedge \neg v_2) = 0.2$
$\gamma_{V_3}(v_3 \mid \neg v_1 \wedge \neg v_2) = 0.8$

$\gamma_{V_5}(v_5 \mid v_4) = 0.25$ $\qquad$ $\gamma_{V_6}(v_6 \mid v_4) = 0.6$
$\gamma_{V_5}(v_5 \mid \neg v_4) = 0.3$ $\qquad$ $\gamma_{V_6}(v_6 \mid \neg v_4) = 0.1$

Let $\Pr$ be the joint probability distribution defined by the probabilistic network B. Suppose that Pearl's algorithm is employed for probabilistic inference with B.

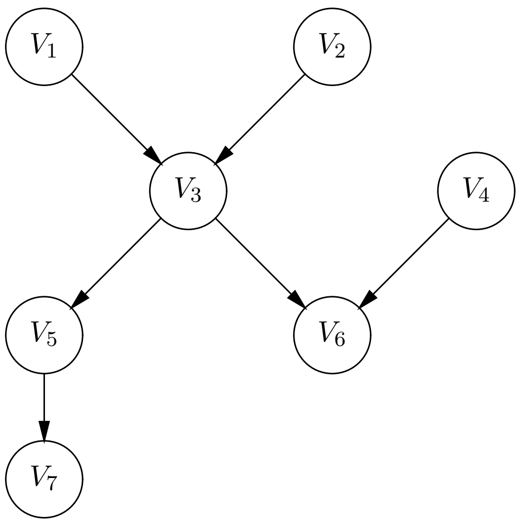
a. *Compute the probabilities $\Pr(v_4)$ and $\Pr(\neg v_4)$ for vertex V_4 from the probabilistic network B.*

b. *Suppose that the evidence $V_6 = true$ is observed and entered into the network. Compute the updated probabilities $\Pr^{v_6}(v_4)$ and $\Pr^{v_6}(\neg v_4)$ for vertex V_4.*

c. *Now, suppose that after processing the evidence $V_6 = true$, the evidence $V_2 = false$ is obtained. For each vertex in the network, examine whether or not this new evidence can influence the probabilities of its values.*

* Exercise 4.5

Consider the probabilistic network $B = (G, \Gamma)$ where $G = (V(G), A(G))$ is the following digraph:



and $\Gamma = \{\gamma_{V_i} \mid V_i \in V(G)\}$ is defined as

$\gamma_{V_1}(v_1) = 0.8$ $\qquad$ $\gamma_{V_2}(v_2) = 0.5$

$\gamma_{V_4}(v_4) = 0.3$

$\gamma_{V_3}(v_3 \mid v_1 \wedge v_2) = 0.2$ $\qquad$ $\gamma_{V_6}(v_6 \mid v_3 \wedge v_4) = 0.3$
$\gamma_{V_3}(v_3 \mid \neg v_1 \wedge v_2) = 0.6$ $\qquad$ $\gamma_{V_6}(v_6 \mid \neg v_3 \wedge v_4) = 0.8$
$\gamma_{V_3}(v_3 \mid v_1 \wedge \neg v_2) = 0.5$ $\qquad$ $\gamma_{V_6}(v_6 \mid v_3 \wedge \neg v_4) = 0.2$
$\gamma_{V_3}(v_3 \mid \neg v_1 \wedge \neg v_2) = 0.1$ $\qquad$ $\gamma_{V_6}(v_6 \mid \neg v_3 \wedge \neg v_4) = 0.1$

$\gamma_{V_5}(v_5 \mid v_3) = 0.6$ $\qquad$ $\gamma_{V_7}(v_7 \mid v_5) = 0.9$
$\gamma_{V_5}(v_5 \mid \neg v_3) = 0.4$ $\qquad$ $\gamma_{V_7}(v_7 \mid \neg v_5) = 0.8$

Let Pr *be the joint probability distribution defined by the probabilistic network B. Suppose that Pearl's algorithm is employed for probabilistic inference with B.*

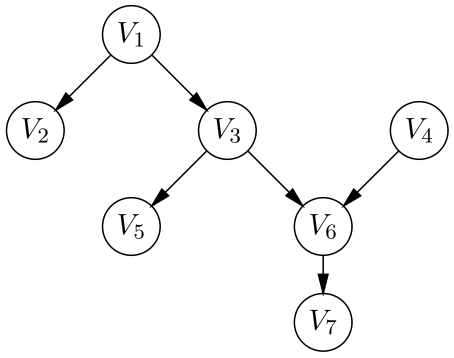
a. *Compute the probabilities* $\Pr(v_5)$ *and* $\Pr(\neg v_5)$ *for vertex V_5 from the probabilistic network B.*

b. *Suppose that the evidence $V_3 = true$ is observed and entered into the probabilistic network B. Compute the updated probabilities* $\Pr^{v_3}(v_5)$ *and* $\Pr^{v_3}(\neg v_5)$ *for vertex V_5 from the network.*

c. *For each vertex in the network, examine whether or not the evidence $V_3 = true$ may influence the probabilities of this vertex' values.*

* Exercise 4.6

Consider the probabilistic network $B = (G, \Gamma)$ where $G = (V(G), A(G))$ is the following digraph:



and $\Gamma = \{\gamma_{V_i} \mid V_i \in V(G)\}$ *is defined as*

$\gamma_{V_1}(v_1) = 0.5$	$\gamma_{V_5}(v_5 \mid v_3) = 0.2$
	$\gamma_{V_5}(v_5 \mid \neg v_3) = 1$
$\gamma_{V_2}(v_2 \mid v_1) = 0.35$	
$\gamma_{V_2}(v_2 \mid \neg v_1) = 0.1$	$\gamma_{V_6}(v_6 \mid v_3 \wedge v_4) = 0.75$
	$\gamma_{V_6}(v_6 \mid \neg v_3 \wedge v_4) = 0.3$
$\gamma_{V_3}(v_3 \mid v_1) = 0.5$	$\gamma_{V_6}(v_6 \mid v_3 \wedge \neg v_4) = 0.25$
$\gamma_{V_3}(v_3 \mid \neg v_1) = 0$	$\gamma_{V_6}(v_6 \mid \neg v_3 \wedge \neg v_4) = 0.2$
$\gamma_{V_4}(v_4) = 0.5$	$\gamma_{V_7}(v_7 \mid v_6) = 0.8$
	$\gamma_{V_7}(v_7 \mid \neg v_6) = 0$

Let Pr *be the joint probability distribution defined by the probabilistic network B. Suppose that Pearl's algorithm is employed for probabilistic inference with B.*

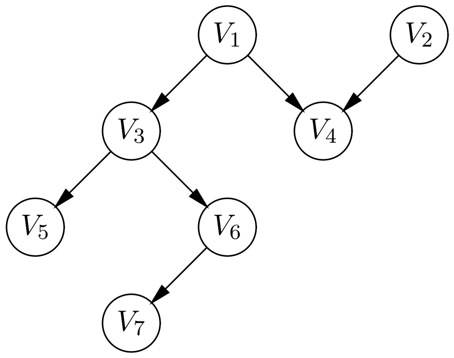
a. *Suppose that the evidence $V_7 = true$ has been entered into the network B. Compute the probabilities* $\Pr^{v_7}(v_1)$ *and* $\Pr^{v_7}(\neg v_1)$ *for vertex V_1 from the probabilistic network B.*

b. *Now suppose that the evidence $V_4 = true$ is entered into the network as well. Does the probability* $\Pr^{v_7,v_4}(v_1)$ *differ from the probability* $\Pr^{v_7}(v_1)$ *?*

c. *Suppose that, for the application for which the probabilistic network B has been developed, the probability* $\Pr^{v_7,v_4}(v_1 \wedge \neg v_2)$ *needs to be computed. Explain how this probability can be computed from the network efficiently.*

* Exercise 4.7

Consider the probabilistic network $B = (G, \Gamma)$ where $G = (V(G), A(G))$ is the following digraph:



and $\Gamma = \{\gamma_{V_i} \mid V_i \in V(G)\}$ is defined as:

$\gamma_{V_1}(v_1) = 0.05$	$\gamma_{V_5}(v_5 \mid v_3) = 0.35$ $\gamma_{V_5}(v_5 \mid \neg v_3) = 0$
$\gamma_{V_2}(v_2) = 0.3$	
$\gamma_{V_3}(v_3 \mid v_1) = 0.75$ $\gamma_{V_3}(v_3 \mid \neg v_1) = 0.4$	$\gamma_{V_6}(v_6 \mid v_3) = 0.8$ $\gamma_{V_6}(v_6 \mid \neg v_3) = 0.6$
$\gamma_{V_4}(v_4 \mid v_1 \wedge v_2) = 0.1$ $\gamma_{V_4}(v_4 \mid \neg v_1 \wedge v_2) = 0.25$ $\gamma_{V_4}(v_4 \mid v_1 \wedge \neg v_2) = 0.8$ $\gamma_{V_4}(v_4 \mid \neg v_1 \wedge \neg v_2) = 0.95$	$\gamma_{V_7}(v_7 \mid v_6) = 0.5$ $\gamma_{V_7}(v_7 \mid \neg v_6) = 0.45$

Let Pr *be the joint probability distribution defined by the probabilistic network B. Suppose that Pearl's algorithm is employed for probabilistic inference with B.*

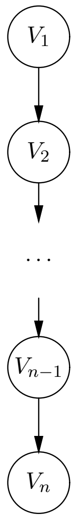
a. *Suppose that the evidence V_1 = true and V_6 = false has been entered into the probabilistic network B. Compute the probabilities* $\Pr^{v_1,\neg v_6}(v_3)$ *and* $\Pr^{v_1,\neg v_6}(\neg v_3)$ *for vertex V_3 from B.*

b. *Suppose that, for the application for which the probabilistic network B has been developed, the probability* $\Pr^{v_1,\neg v_6}(v_2 \wedge v_5)$ *needs to be computed. Explain how this probability can be computed from the network efficiently.*

c. *Now suppose that, for the application for which the probabilistic network has been developed, the probability* $\Pr^{v_1,\neg v_6}(v_2 \vee v_4)$ *needs to be computed. Explain how this probability can be computed from the network efficiently.*

* Exercise 4.8

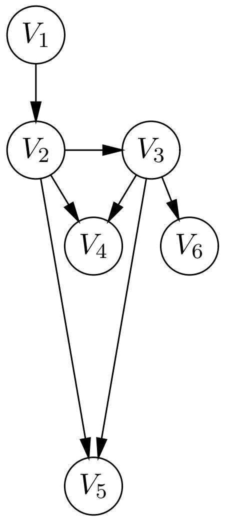
Let $B = (G, \Gamma)$ be a probabilistic network where $G = (V(G), A(G))$ with $V(G) = \{V_1, \ldots, V_n\}$, $n \geq 1$, and $A(G) = \{(V_i, V_{i+1}) \mid i = 1, \ldots, n-1\}$ is the following digraph:



Let Pr *be the joint probability distribution defined by the probabilistic network B. Furthermore, let $B' = (G, \Gamma')$ be the probabilistic network that is obtained from B by substituting for a vertex V_k, $k > 1$, the function value $\gamma_{V_k}(v_k \mid v_{k-1})$ by $\gamma_{V_k}(v_k \mid v_{k-1}) + \epsilon$, for some small $\epsilon > 0$. Let $\Pr'$ be the joint probability distribution defined by B'. Show that $|Pr'(V_i) - Pr(V_i)| \leq \epsilon$, for all $V_i \in V(G)$, $i = 1, \ldots, n$.*

Exercise 4.9

Consider the probabilistic network $B = (G, \Gamma)$ where $G = (V(G), A(G))$ is the following digraph:



and $\Gamma = \{\gamma_{V_i} \mid V_i \in V(G)\}$ *is defined as*

$$\gamma_{V_1}(v_1) = 0.5$$

$$\begin{array}{ll}
\gamma_{V_2}(v_2 \mid v_1) = 0.9 & \gamma_{V_3}(v_3 \mid v_2) = 0.5 \\
\gamma_{V_2}(v_2 \mid \neg v_1) = 0.7 & \gamma_{V_3}(v_3 \mid \neg v_2) = 0.8
\end{array}$$

$$\begin{array}{ll}
\gamma_{V_4}(v_4 \mid v_2 \wedge v_3) = 0.1 & \gamma_{V_5}(v_5 \mid v_2 \wedge v_3) = 0.3 \\
\gamma_{V_4}(v_4 \mid \neg v_2 \wedge v_3) = 0.25 & \gamma_{V_5}(v_5 \mid \neg v_2 \wedge v_3) = 0.7 \\
\gamma_{V_4}(v_4 \mid v_2 \wedge \neg v_3) = 0.6 & \gamma_{V_5}(v_5 \mid v_2 \wedge \neg v_3) = 0.2 \\
\gamma_{V_4}(v_4 \mid \neg v_2 \wedge \neg v_3) = 0.75 & \gamma_{V_5}(v_5 \mid \neg v_2 \wedge \neg v_3) = 0.7
\end{array}$$

$$\begin{array}{l}
\gamma_{V_6}(v_6 \mid v_3) = 0.45 \\
\gamma_{V_6}(v_6 \mid \neg v_3) = 0.65
\end{array}$$

Let Pr *be the joint probability distribution defined by the probabilistic network B.*

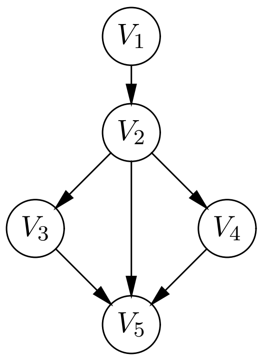
* *a. Compute a loop cutset for the digraph G by employing the heuristic algorithm of Suermondt and Cooper.*

* *b. Compute the probabilities* $\Pr(v_5)$ *and* $\Pr(\neg v_5)$ *for vertex V_5 from the probabilistic network B by Pearl's algorithm with loop cutset conditioning.*

c. Suppose that the evidence $V_3 = false$ is observed and entered into the network. Furthermore, suppose that after entering $V_3 = false$ into the network, the additional evidence $V_1 = true$ is observed. For each vertex in the network, examine whether or not this new evidence can influence the probabilities of its values.

* Exercise 4.10

Consider the probabilistic network $B = (G, \Gamma)$ where $G = (V(G), A(G))$ is the following digraph:



and $\Gamma = \{\gamma_{V_i} \mid V_i \in V(G)\}$ *is defined as*

$$\begin{aligned}
&\gamma_{V_1}(v_1) = 0.9 \\
&\gamma_{V_2}(v_2 \mid v_1) = 0.125 \\
&\gamma_{V_2}(v_2 \mid \neg v_1) = 0.875 \\
&\gamma_{V_3}(v_3 \mid v_2) = 0.7 \\
&\gamma_{V_3}(v_3 \mid \neg v_2) = 0.2 \\
&\gamma_{V_4}(v_4 \mid v_2) = 0.35 \\
&\gamma_{V_4}(v_4 \mid \neg v_2) = 0.1
\end{aligned}
\qquad
\begin{aligned}
&\gamma_{V_5}(v_5 \mid v_2 \wedge v_3 \wedge v_4) = 0.2 \\
&\gamma_{V_5}(v_5 \mid \neg v_2 \wedge v_3 \wedge v_4) = 0.25 \\
&\gamma_{V_5}(v_5 \mid v_2 \wedge \neg v_3 \wedge v_4) = 0.7 \\
&\gamma_{V_5}(v_5 \mid v_2 \wedge v_3 \wedge \neg v_4) = 0.6 \\
&\gamma_{V_5}(v_5 \mid \neg v_2 \wedge \neg v_3 \wedge v_4) = 0.55 \\
&\gamma_{V_5}(v_5 \mid \neg v_2 \wedge v_3 \wedge \neg v_4) = 0.8 \\
&\gamma_{V_5}(v_5 \mid v_2 \wedge \neg v_3 \wedge \neg v_4) = 0.3 \\
&\gamma_{V_5}(v_5 \mid \neg v_2 \wedge \neg v_3 \wedge \neg v_4) = 0.15
\end{aligned}$$

Let Pr *be the joint probability distribution defined by the probabilistic network B.*

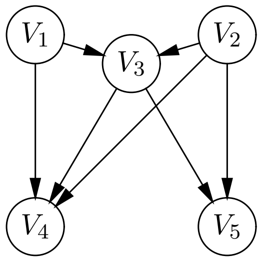
a. *Compute a loop cutset for the digraph G by employing the heuristic algorithm of Suermondt and Cooper.*

b. *Compute the probabilities* $\Pr(v_3)$ *and* $\Pr(\neg v_3)$ *for vertex V_3 from the probabilistic network B by Pearl's algorithm with loop cutset conditioning.*

c. *Suppose that the evidence $V_5 = true$ is entered into the network. Explain how the probabilities* $\Pr^{v_5}(v_3)$ *and* $\Pr^{v_5}(\neg v_3)$ *are computed from the network.*

Exercise 4.11

* a. *Consider a probabilistic network $B = (G, \Gamma)$ where G is the following digraph:*



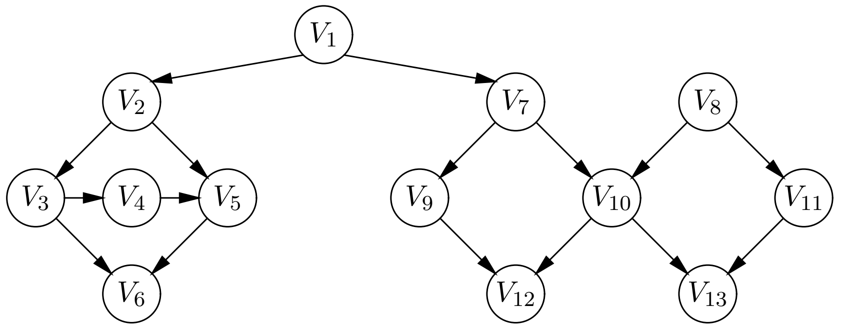
Compute a loop cutset for G by employing the heuristic algorithm of Suermondt and Cooper.

b. *The heuristic algorithm of Suermondt and Cooper aims at computing for a multiply connected digraph an* optimal loop cutset. *Now consider a probabilistic network in which the variables are not necessarily binary but may take a value from an arbitrarily large finite set of values. Illustrate by means of an example that for such a probabilistic network a non-optimal loop cutset may be a better choice for the purpose of loop cutset conditioning than an optimal loop cutset.*

c. *In the heuristic algorithm of Suermondt and Cooper the number of values for the various variables of a probabilistic network are not taken into consideration. Modify the algorithm in such a way that the number of values is taken into consideration to arrive at a 'good' loop cutset.*

Exercise 4.12

* a. *Consider the following digraph G:*



Compute a loop cutset for G by employing the heuristic algorithm of Suermondt and Cooper.

b. *Let $G = (V(G), A(G))$ be an acyclic multiply connected digraph and let $C \subseteq V(G)$ be a set of vertices in G. Furthermore, let G' be the digraph that is obtained from G by deleting for each vertex from C its outgoing arcs, that is, $G' = (V(G), A'(G))$ with $A'(G) = A(G) \setminus \{(V_i, V_j) \mid V_i \in C\}$. Show that the set C is a loop cutset of G if and only if G' is a singly connected digraph.*

c. *Upon employing the heuristic algorithm of Suermondt and Cooper for computing a loop cutset for a multiply connected digraph G, a non-minimal loop cutset C may be yielded. Give a sketch of an algorithm that takes for its input the non-minimal loop cutset C and yields a minimal loop cutset C' of G.*

Exercise 4.13

For probabilistic inference with a probabilistic network, sometimes Pearl's algorithm is supplemented with a method called evidence absorption. *The basic idea of this method is to dynamically modify a probabilistic network after evidence has been entered so as to reflect as many of the new independences occasioned by the evidence as possible. Now consider a probabilistic network $B = (G, \Gamma)$ and suppose that the evidence $V_i = true$ is entered for some vertex $V_i \in A(G)$. Evidence absorption now amounts to modifying the digraph G of B by deleting all outgoing arcs from vertex V_i; with the thus modified digraph G' a new set Γ' of probability assessment functions is constructed such that the new probabilistic network $B' = (G', \Gamma')$ reflects the updated joint probability distribution.*

a. *Show that the digraphs G and G' portray the same set of independences given V_i.*

Hint. *Show that: $\langle X \mid Y \mid Z\rangle^d_G$ if and only if $\langle X \mid Y \mid Z\rangle^d_{G'}$, for all sets of variables $X, Y, Z \subseteq V(G)$ with $V_i \in Y$.*

b. *Show by means of an example that the digraphs G and G' do not represent the same set of independences.*

c. *Explain why repeated application of the method of evidence absorption reduces the average-case computational complexity of reasoning with a probabilistic network.*

Chapter 5

Building a Probabilistic Network

This chapter and the next one differ from previous chapters in the sense that the latter describe more or less finished research, whereas the following chapters discuss topics that are subject of ongoing research. As a result, the chapters tell a number of short stories with lots of open endings. The assumption that we consider binary variables only is now lifted, although employed in some specific situations.

This chapter discusses the construction of a probabilistic network: how do we get the graphical structure (by hand, or automatically?) and where do the probabilities come from (experts, data, . . . ?).

The probabilistic network framework generally is used for applying probability theory for reasoning with uncertainty in knowledge-based systems. For employing the framework for a real-life domain of application, relevant knowledge of the domain at hand is represented in the probabilistic network formalism; the basic algorithms of the framework are taken as building blocks for shaping the system's intelligent problem-solving behaviour. In this chapter we address the task of *building* a probabilistic network for a domain of application; shaping intelligent problem-solving behaviour is the topic of the next chapter.

Building a probabilistic network for a domain of application involves various different tasks. The first of these tasks is to identify the *variables* that are of importance in the domain at hand, along with their possible values. Once the important domain variables have been identified, the qualitative part of the probabilistic network in the making is constructed: the second task in building a probabilistic network therefore is to identify the *independences* among the variables discerned and to express these in an acyclic digraph. After the qualitative part of the network has been established, the *probability assessment functions* are defined: the last task in building a probabilistic network is to estimate the (conditional) probabilities that are required to constitute the network's quantitative part. The three tasks are summarised in Figure 5.1. The various tasks in building a probabilistic network are, in principle, performed one after the other. Building a network, however, often requires a careful *trade-off* between the desire for a large and rich model to obtain accurate results on the one hand, and the costs of construction and maintenance, and the complexity of probabilistic inference on the other hand. In practice, therefore, building a probabilistic network is a cyclic process that iterates over these tasks until a network results that is deemed satisfactory for the domain at hand. We would like to note that, as the probabilistic network

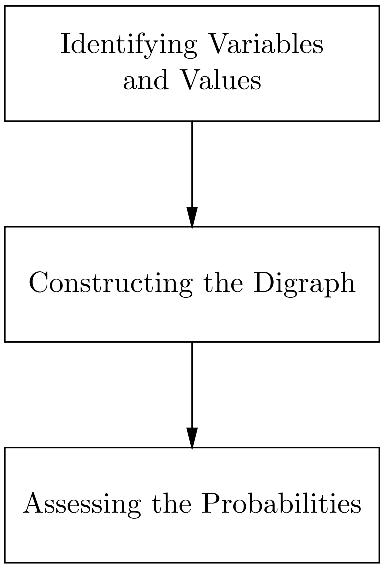


Figure 5.1: The tasks in building a probabilistic network.

framework has not been around for a long time, methodologies for building probabilistic networks do not yet abound.

In the following sections, we address the tasks involved in building a probabilistic network separately. In Section 5.1 we address the identification of the important domain variables and values. In Section 5.2 we discuss the construction of the qualitative part of a probabilistic network in the making. Section 5.3 focuses on the assessment of the probabilities required for the network's quantitative part.

5.1 Identifying Variables and Values

The first task in building a probabilistic network for a domain of application is to identify the variables that are of importance in the domain, along with their possible values. Identifying the important domain variables is typically performed with the help of one or more experts. Since a probabilistic network, as any model, necessarily is a simplification of reality, well-founded decisions have to be taken as to which variables should be included in the network and which may be omitted. We would like to note that this task is not reserved for building probabilistic networks, but instead is quite common in engineering knowledge-based systems. A knowledge engineer can therefore make use of the various elicitation techniques designed for engineering knowledge-based systems in general. Recently, the use of ontologies was suggested to help structure the domain, its variables and the different types of relations that exist between the variables [Helsper & Van der Gaag, 2002]. Such ontologies have proved useful tools upon constructing a probabilistic network's digraph, for distilling the variables and their interrelationships that are important for the problem at hand. We will not further elaborate on this task here and confine ourselves to emphasising that it needs to be performed with care since the domain variables that are modelled in the probabilistic network demarcate the scope of the resulting system.

Once the important domain variables have been identified, each of them has to be expressed as a *statistical variable* to allow for inclusion in the probabilistic network in the making. We recall that a statistical variable is characterised by its values being *mutually exclusive* and *collectively exhaustive*. To allow for inclusion in a probabilistic network, a statistical variable has to take its value from a *finite* set of discrete values. Only if the set of values of a domain variable satisfies these properties, can the variable be included in the network as it is. Otherwise, the domain variable is modelled in terms of one or more statistical variables.

For modelling domain variables for inclusion in a probabilistic network, we distinguish between the following types of variable:

- a *single-valued* domain variable that takes a value from a *finite* set of (discrete) values;
- a *single-valued* domain variable that takes a value from an *infinite* set of values;
- a *multi-valued* domain variable.

We begin by considering a *single-valued* domain variable that takes its value from a *finite* set of discrete values. The values of this variable are mutually exclusive by definition. Moreover, the variable's values are easily rendered collectively exhaustive. A single-valued domain variable can therefore be expressed straightforwardly as a statistical variable in the probabilistic network in the making.

For a *single-valued* domain variable that takes its value from an *infinite* set of values, we equally have that this variable's values are mutually exclusive and are easily rendered collectively exhaustive. However, its set of possible values being infinite prohibits including the domain variable as it is in the probabilistic network in the making. To allow for inclusion in the network, the set of values is *discretised*. To this end, the variable's possible values are clustered in a finite number of mutually exclusive and collectively exhaustive clusters of values; these clusters then are taken as the values of a *new* single-valued variable that is included in the probabilistic network in the making.

Example 5.1.1 Suppose that, in a medical domain of application, a patient's *temperature* has been identified as an important domain variable. This domain variable is single-valued, taking its value from the (in essence) infinite range of values between, say, 35 °C and 41 °C. This range of values can be discretised in, for example, the five intervals [35;36), [36;37), [37;38), [38;40), and [40;41]. Note that these intervals can be looked upon as values that are mutually exclusive and collectively exhaustive. For inclusion in the probabilistic network in the making, the domain variable is now expressed as a statistical variable that takes its value from among the five values mentioned above. □

In general, a domain variable's set of values admits numerous discretisations. The discretisation best chosen is highly dependent on the purpose for which the probabilistic network is being developed and has to be decided upon in consultation with one or more domain experts.

A domain variable taking *multiple* values from a finite set of discrete values cannot be expressed directly as a statistical variable since its values are not mutually exclusive. To allow for inclusion in the probabilistic network in the making, such a domain variable needs to be re-modelled so as to satisfy the properties of a statistical variables. Remodelling may be done by redefining the variable's set of values in such a way that the new values are mutually exclusive and collectively exhaustive.

Example 5.1.2 Suppose that, in a medical domain of application, a patient's *blood count* has been identified as an important domain variable. This domain variable is multi-valued, taking its values from among the set of values {*normal*, *lymphocytosis*, *lymphocytopenia*, *leucocytosis*, *leucocytopenia*}. The values *lymphocytosis* and *lymphocytopenia* refer to the level of lymphocytes in a patient's blood sample and the values *leucocytosis* and *leucocytopenia* refer to the leucocyte level; these values are not mutually exclusive. Domain knowledge now indicates that a patient's *blood count* cannot take any arbitrary combination of the values mentioned above. In fact, only nine combinations of values are meaningful – these are {*normal*}, {*lymphocytosis*}, {*lymphocytopenia*}, {*leucocytosis*}, {*leucocytopenia*}, {*lymphocytosis, leucocytosis*}, {*lymphocytosis, leucocytopenia*}, {*lymphocytopenia, leucocytosis*}, and {*lymphocytopenia, leucocytopenia*}. These combinations are now taken as the mutually exclusive values of a new statistical variable, modelling a patient's blood count, to be included in the probabilistic network in the making. □

Multi-valuedness of a domain variable often arises from the variable being *compound*, that is, implicitly built from several other domain variables. Instead of redefining its set of values, a compound variable may be modelled by decomposing it into the domain variables it is built from and subsequently modelling these separate variables in the probabilistic network in the making.

Example 5.1.3 We consider once more the domain variable expressing a patient's *blood count* as introduced in the previous example. We observe that this domain variable is a compound variable built from two variables. One of these variables expresses the level of lymphocytes in a patient's blood sample, taking its value from the set of values {*normal*, *lymphocytosis*, *lymphocytopenia*}; the other variable captures the level of leucocytes, taking its value from the set of values {*normal*, *leucocytosis*, *leucocytopenia*}. Note that these two variables are statistical variables, that can be included directly in the probabilistic network in the making. The originally multi-valued domain variable is now decomposed into these variables. □

Decomposing a compound variable requires considerable knowledge of the domain at hand and therefore is best performed with the help of a domain expert.

After all important domain variables have been expressed as statistical variables for inclusion in the probabilistic network in the making, the meanings of these statistical variables have to be properly documented. While each statistical variable has a unique meaning in the context of the probabilistic network at hand, its meaning may very well differ from the meanings associated with variables with the same name in other models or in the literature in the domain. The task of documenting the meanings of the statistical variables modelled in the probabilistic network therefore is crucial to avoid any ambiguity in future reference. This task is not typical for building probabilistic networks and has in fact been recognised as vital in knowledge-engineering practice in general.

5.2 Constructing the Digraph

Once the domain variables of importance have been identified and expressed as statistical variables, the construction of the qualitative part of the probabilistic network in the making can commence. In principle, for constructing the network's qualitative part, the independence relation of the joint probability distribution on the variables discerned has to be identified and represented in an acyclic digraph. In practice, however, the digraph typically is conceived without explicitly identifying independences: it is constructed directly, either by hand or by learning from data.

5.2.1 Constructing the Digraph by Hand

For most domains of application, the qualitative part of a probabilistic network in the making is *handcrafted*. For this purpose, one or more domain experts are interviewed. In the interviews, the qualitative relationships between the variables discerned are elicited, using the concept of *causality* as a heuristic guiding principle. Typical questions asked during the interviews are "What could cause this effect ?", "What manifestations could this cause have ?" [Henrion, 1989]. The elicited causal relationships among the variables are expressed in graphical terms by taking the direction of causality for directing the arcs between related variables. The resulting graphical representation can then be taken as a basis for further refinement.

Using the concept of causality as a guiding principle during the interviews with domain experts may meet with some difficulties. For example, not every influential relationship among variables can be interpreted as causal. If a non-causal influential relationship comes to the fore during an interview, a more elaborate analysis of the independences involved is required before it can be expressed in graphical terms. Another problem that may arise from building on causality concerns the creation of cycles in the graphical representation. As the digraph of a probabilistic network has to be acyclic, any cycle that has resulted from the interviews needs to be removed. There are several ways to deal with cycles [Peek & Ottenkamp, 1997], the simplest being the deletion of a single arc from a cycle to be removed. To conclude, causality is not a well-understood concept and therefore may leave room for multiple interpretations.

Despite the fact that building on the concept of causality does not suffice in itself for constructing the digraph of a probabilistic network in the making, it brings the advantage that domain experts are allowed to express their knowledge in either the *causal* or *diagnostic* direction. Since they are allowed to express their knowledge in a form they feel comfortable with, the experts' statements tend to be quite robust. We would like to note that the task of eliciting relationships among variables from domain experts is not reserved for building probabilistic networks: the elicitation task has parallels with engineering knowledge-based systems in general for which several methodologies have been developed [Boose & Gaines, 1988]. Some experiences with the issues discussed above in building a probabilistic network for a real-life application are described in [Van der Gaag & Helsper, 2002].

5.2.2 Learning the Digraph from Data

In various domains of application, data has been collected and maintained over numerous years of every-day problem solving. Such a data collection usually contains highly valuable information about the relationships among the variables discerned, be it implicitly. If a comprehensive data set is available in the domain for which a probabilistic network is probabilistic developed, the construction of the network's qualitative part may be performed *automatically*: the basic idea of *learning* the qualitative part from data is to distill information from the data set and exploit it for constructing a digraph.

The data set

For learning the qualitative part of a probabilistic network, a data set with information from the domain at hand is required.

We formally define the concept of a database.

Definition 5.2.1 *Let V be a set of variables. A* data set *D over V is a multi-set of configurations of V. An element c_V of D is called a* case*.*

To be suitable for learning purposes, a data set has to satisfy various properties. First of all, the data comprised in the data set must have been collected very *carefully*. Biases that are introduced in the data set as a result of the data collection strategies used will have impact on the topology of the resulting digraph, yet may not be desirable for the purpose for which the probabilistic network is being developed; unfortunately, biases are not easily detected in a once constructed network. Also, to be suitable for the purpose of learning a probabilistic network's digraph, the variables and associated values that occur in the data set should *match* the variables and values that are to be modelled in the network, or should at least admit easy translation into these variables and values. Moreover, the data set should comprise enough data to allow for *reliable* identification of probabilistic relationships among the variables discerned.

In addition to the general prerequisites outlined before, a data set should satisfy several properties that are implicitly assumed by most algorithms for learning a probabilistic network's digraph from data. A property that is commonly assumed is that each case in the data set specifies a value for every variable discerned, that is, there are no *missing values*. Unfortunately, for most real-life data sets this property does not hold. To use a data set with missing values for learning purposes, therefore, the missing values have to be *filled in*, for example based upon (roughly) estimated prior probabilities for these values or with the help of domain experts; the EM (Expectation-Maximisation) algorithm is often used for this purpose [Dempster *et al.*, 1977]. Other properties assumed by most learning algorithms concern the real-life process that generated the cases comprised in the data set at hand. It is assumed that this process generates cases *independently*, that is, the values specified for the variables in a case are assumed not to be influenced in any way by the values in previously generated cases. Also, it is assumed that the process is not time-dependent, that is, the data set is assumed not to reflect information that varies over time.

As we will discuss in the sequel, a data set of cases over a set of statistical variables is used for estimating various conditional probabilities. These conditional probabilities are used for analysing the strengths of various different relationships among the variables. The conditional probabilities required for this purpose are obtained from the data set by *counting*. We consider a data set D over the set of variables V, comprising N cases. For each set of variables $X \subseteq V$, we will use $N(c_X)$ to denote the number of cases in D in which the variables from X have adopted the conjunction of values c_X; for the empty set, we take $N(c_\varnothing) = N$. For any two sets of variables $X, Y \subseteq V$, the probability distribution $\Pr(C_X \mid C_Y)$ can now be estimated from the data set by

$$\Pr(c_X \mid c_Y) = \frac{N(c_X \wedge c_Y)}{N(c_Y)}$$

for all configurations c_X and c_Y of X and Y, respectively.

Various algorithms exist for learning probabilistic networks from data. Roughly, two approaches can be distinghuished: one where statistical tests are applied to the data to determine the independences; the other where the space of possible networks is searched for a good scoring network. Research into such *learning algorithms* is very popular now that so many data sets are available. Here, we will focus on a score & search-based method that lies at the basis of most algorithms that are popular today (see for example [Stahlschmidt *et al.*, 2013] for a description of various algorithms).

The quality measure

An algorithm for learning the qualitative part of a probabilistic network from data typically generates various different acyclic digraphs and compares these as to their ability to describe or explain the data at hand. We will focus on the task of comparing digraphs before turning to their generation.

For comparing digraphs as to their ability to describe the data from the database at hand, a *quality measure* is employed. A quality measure is a function that assigns to a digraph a numerical value expressing how well this digraph fits the data; this numerical value is called the *quality* of the digraph given the database. In the literature, various different quality measures have been proposed, originating from different theories. Here, we will discuss only the *MDL quality measure*. This measure originates from coding theory and is built on the minimum description length principle. For an overview of the most popular quality measures in use for learning the qualitative part of a probabilistic network from data, we refer the reader to [Bouckaert, 1995].

We define the MDL quality measure to pertain to an acyclic digraph and a database.

Definition 5.2.2 *Let $V = \{V_1, \ldots, V_n\}$, $n \geq 1$, be a set of statistical variables. Let D be a data set over V and let N be the number of cases in D. Furthermore, let P be a probability distribution over the set of acyclic digraphs with vertex set V. Let $G = (V(G), A(G))$ be an acyclic digraph with $V(G) = V$. Then, the* quality of G given D*, denoted $Q(G, D)$, is defined by*

$$Q(G, D) = \log P(G) - N \cdot H(G, D) - \tfrac{1}{2} K \cdot \log N$$

where

$$H(G, D) = -\sum_{V_i \in V} \sum_{c_{V_i}} \sum_{c_{\rho(V_i)}} \left(\frac{N(c_{V_i} \wedge c_{\rho(V_i)})}{N} \right) \cdot \log \left(\frac{N(c_{V_i} \wedge c_{\rho(V_i)})}{N(c_{\rho(V_i)})} \right)$$

and

$$K = \sum_{V_i \in V} 2^{|\rho(V_i)|}$$

From the previous definition, we have that the quality $Q(G, D)$ of an acyclic digraph G given a data set D involves three terms: $\log P(G)$, $-N \cdot H(G, D)$ and $-\frac{1}{2} K \cdot \log N$. We will take a closer look at these terms separately.

In the first term, $\log P(G)$, of the quality of a digraph G given a data set D, the probability $P(G)$ denotes the prior probability of the digraph G being the qualitative part of a probabilistic network that describes the joint probability distribution that generated the data set D. This term therefore expresses information about the ‘real’ qualitative part *prior* to observation of the database. In a real-life application of a learning algorithm, this term is used for modelling domain knowledge. For example, if a domain expert suggests existence of a specific arc in the qualitative part of the 'real' network, then digraphs that adhere to this suggestion are given a higher prior probability than digraphs that do not. In the sequel, we will assume that no prior domain knowledge is available and, hence, that the probability distribution P is a uniform distribution.

The second term, $-N \cdot H(G, D)$, of the quality of a digraph G given a data set D is proportional to the probability of the data set D being generated by the joint probability distribution represented by a probabilistic network B that involves the digraph G. Now, recall that a probabilistic network not only involves a digraph but also includes a set of probability assessment functions. To enhance the network's 'match' with the data set, all conditional probabilities required are assessed from the data: for each variable $V_i \in V$, we take

$$\gamma_{V_i}(c_{V_i} \mid c_{\rho(V_i)}) = \frac{N(c_{V_i} \wedge c_{\rho(V_i)})}{N(c_{\rho(V_i)})}$$

for all configurations c_{V_i} of V_i and all configurations $c_{\rho(V_i)}$ of the set $\rho(V_i)$ of V_i's predecessors in G. Now consider the probability of the data set D being generated by the joint probability distribution Pr defined by the probabilistic network $B = (G, \Gamma)$ where Γ is the set of probability assessment functions estimated from D. Since we have assumed that all cases in D have been generated independently, we have that the probability $P'(D \mid B)$ of the data set D given the probabilistic network B equals

$$P'(D \mid B) = \prod_{c_V \in D} \Pr(c_V)$$

By exploiting the property stated in Proposition 4.1.3, we find that

$$P'(D \mid B) = \prod_{c_V \in D} \prod_{V_i \in V} \gamma_{V_i}(c_{V_i} \mid c_{\rho(V_i)})$$

where $\bigwedge_{V_i \in V}(c_{V_i} \wedge c_{\rho(V_i)}) \equiv c_V$. By reordering terms, we ultimately find that

$$P'(D \mid B) = \prod_{V_i \in V} \prod_{c_{V_i}} \prod_{c_{\rho(V_i)}} \gamma_{V_i}(c_{V_i} \mid c_{\rho(V_i)})^{N(c_{V_i} \wedge c_{\rho(V_i)})}$$

Substitution of the probability estimates for the function values of the various different assessment functions now yields

$$P'(D \mid B) = \prod_{V_i \in V} \prod_{c_{V_i}} \prod_{c_{\rho(V_i)}} \left(\frac{N(c_{V_i} \wedge c_{\rho(V_i)})}{N(c_{\rho(V_i)})} \right)^{N(c_{V_i} \wedge c_{\rho(V_i)})}$$

Hence,

$$\log P'(D \mid B) = N \cdot \sum_{V_i \in V} \sum_{c_{V_i}} \sum_{c_{\rho(V_i)}} \left(\frac{N(c_{V_i} \wedge c_{\rho(V_i)})}{N} \right) \cdot \log \left(\frac{N(c_{V_i} \wedge c_{\rho(V_i)})}{N(c_{\rho(V_i)})} \right)$$

The result equals

$$\log P'(D \mid B) = -N \cdot H(G, D)$$

where $H(G, D)$ denotes the mutual *entropy* of G and D.

Entropy is an information-theoretic measure of uncertainty. In general, the entropy of a variable is maximal when the uncertainty concerning its value is maximal; the entropy is zero when there is complete knowledge as to the variable's value. Informally speaking, we have, in the context of the MDL quality measure, that the better a digraph fits the database, the lower their mutual entropy. In general, an acyclic digraph with more arcs will have a lower entropy than an acyclic digraph with fewer arcs: a digraph with more arcs expresses fewer independences and therefore is capable of capturing more of the nuances reflected in the cases of the data set. So, for a tightly connected acyclic digraph G the term $-N \cdot H(G, D)$ in this digraph's quality tends to approach zero.

For an acyclic digraph with more arcs more conditional probabilities have to be estimated from the data set to define the probability assessment functions to be associated with the digraph, than for a digraph with fewer arcs. The more probabilities have to be estimated from the data set, the smaller the number of cases in the data set these estimates can be based upon. So, the more probabilities have to be estimated the less reliable the estimates will be. The third term, $-\frac{1}{2}K \cdot \log N$, of the quality $Q(G, D)$ of a digraph G given a database D captures this aspect. In this term, K expresses the number of probabilities that have to be assessed from the data set to define the probability assessment functions for the digraph at hand. So, the more arcs a digraph has, the larger K and the smaller the term $-\frac{1}{2}K \cdot \log N$. The term $-\frac{1}{2}K \cdot \log N$ is often referred to as the *penalty term* as it imposes a penalty on adding arcs to the digraph in the making.

In the sequel, we shall see that an algorithm for learning the digraph of a probabilistic network from data constructs a digraph by successively adding single arcs to an initially arcless graph. For the initial arcless digraph, the entropy term $-N \cdot H(G, D)$ of its quality will be extremely small and the term $-\frac{1}{2}K \cdot \log N$ will be quite close to zero. On successively adding arcs to the digraph, the term $-N \cdot H(G, D)$ will grow rapidly. The term $-\frac{1}{2}K \cdot \log N$ on the other hand will decrease as the digraph becomes more tightly connected. As long as the entropy term increases more rapidly than the penalty term decreases, the quality of the digraph in the making will increase. The increase of the digraph's quality upon successive arc addition continues until the increase of the entropy term is dominated by the decrease of the penalty term.

The search heuristic

We have mentioned before that an algorithm for learning the qualitative part of a probabilistic network from data generates various different acyclic digraphs and compares these digraphs with respect to their quality given the database at hand. The basic purpose of the algorithm is to select from among all possible acyclic digraphs a digraph with *highest quality*. Unfortunately, it is not feasible from a computational point of view to generate all digraphs and compute their qualities. A learning algorithm therefore incorporates a *search heuristic* that searches the set of all possible acyclic digraphs for digraphs that are *likely* to have a high quality; only for these digraphs is the quality given the data set actually computed. Several such search heuristics have been proposed in the literature. Here, we will only discuss the B search heuristic. For an overview of various search heuristics and their properties, we refer the reader once more to [Bouckaert, 1995].

Let V be the set of statistical variables of the probabilistic network in the making. The B search heuristic for selecting a digraph of high quality from among all possible acyclic digraphs with vertex set V commences with investigating the *arcless* digraph

$G_0 = (V, \varnothing)$. To this initially arcless graph, the search heuristic successively adds single arcs to improve the digraph's quality. Now, suppose that after adding some arcs, a digraph $G_k = (V, A(G_k))$ has resulted. For selecting a new arc to add to the digraph, the search heuristic identifies all possible arcs that can be added to the digraph G_k without introducing a cycle. For each identified arc (V_i, V_j), it computes the *gain in quality* that would be yielded by adding this arc to the digraph in the making. So, it computes the difference in quality between the new digraph $G_{k+1} = (V, A(G_k) \cup \{(V_i, V_j)\})$ and the digraph G_k. The search heuristic then selects an arc that yields the highest gain in quality and adds this arc to the digraph in the making. This process of arc addition is repeated until no gain in quality can be achieved anymore. Note that the B search heuristic is a *greedy* search heuristic in the sense that it considers single arcs only.

From the informal outline of the B search heuristic, it will be evident that for selecting a digraph with high quality, the search heuristic has to compare qualities of various different digraphs. We observe, however, that for selecting the next arc to add to the digraph in the making the search heuristic does not actually need to *know* the qualities of the various digraphs considered since an arc is selected on the basis of *differences* in quality between digraphs only. From B being a greedy search heuristic, we further have that these differences are only computed for pairs of digraphs that differ in one arc. The B search heuristic exploits these observations by only *partially* computing the qualities of the digraphs concerned.

For the purpose of partially computing qualities, the B search heuristic makes use of the concept of the quality of a *vertex* in a digraph.

Definition 5.2.3 *Let $V = \{V_1, \ldots, V_n\}$, $n \geq 1$, be a set of statistical variables. Let D be a data set over V and let N be the number of cases in D. Let $G = (V(G), A(G))$ be an acyclic digraph with $V(G) = V$. For each vertex $V_i \in V(G)$, the* quality *of V_i in G given D, denoted by $q(V_i, \rho(V_i), D)$, is defined as*

$$\begin{aligned} q(V_i, \rho(V_i), D) \quad &= \textstyle\sum_{c_{V_i}} \sum_{c_{\rho(V_i)}} N(c_{V_i} \wedge c_{\rho(V_i)}) \cdot \log\left(\frac{N(c_{V_i} \wedge c_{\rho(V_i)})}{N(c_{\rho(V_i)})}\right) + \\ &\qquad - \tfrac{1}{2} \cdot 2^{|\rho(V_i)|} \cdot \log N \end{aligned}$$

The quality of an acyclic digraph given a data set can now be expressed in terms of the qualities of its various vertices; this property is known as the *sum property* of a digraph's quality given a database and is stated more formally in the following lemma.

Lemma 5.2.4 *Let $V = \{V_1, \ldots, V_n\}$, $n \geq 1$, be a set of statistical variables. Let D be a data set over V and let N be the number of cases in D. Let P be a probability distribution over the set of all acyclic digraphs with vertex set V. Let $G = (V(G), A(G))$ be such an acyclic digraph with $V(G) = V$. Furthermore, let $Q(G, D)$ be the quality of G given D and, for each vertex $V_i \in V(G)$, let $q(V_i, \rho(V_i), D)$ be the quality of V_i in G given D. Then,*

$$Q(G, D) = \log P(G) + \sum_{V_i \in V} q(V_i, \rho(V_i), D)$$

Proof. The property stated in the lemma follows directly from Definition 5.2.2 and Definition 5.2.3. □

Note that since we have assumed that the probability distribution P over the set of all acyclic digraphs with a given vertex set is a uniform distribution, the term $\log P(G)$ in a digraph's quality is the same for every digraph G and therefore is a constant with respect to learning the digraph of a probabilistic network from a data set.

Let V once more be the set of statistical variables discerned and let $G_k = (V, A(G_k))$ be a digraph that has resulted at some stage during application of the B search algorithm. Now, consider a new digraph $G_{k+1} = (V, A(G_k) \cup \{(V_i, V_j)\})$ that differs from G_k in the arc (V_i, V_j) only. From Lemma 5.2.4 it is easily seen that the difference in quality between these two digraphs equals the difference in quality of vertex V_j in the two digraphs only; this property is stated in the following corollary.

Corollary 5.2.5 *Let $V = \{V_1, \ldots, V_n\}$, $n \geq 1$, be a set of statistical variables. Let D be a database over V. Let $G = (V(G), A(G))$ and $G' = (V(G'), A(G) \cup \{(V_i, V_j)\})$ be acyclic digraphs with $V(G) = V(G') = V$ Furthermore, let $Q(G, D)$ and $Q(G', D)$ be the quality of G given D and the quality of G' given D, respectively. For each $V_k \in V$, let $q(V_k, \rho_G(V_k), D)$ and $q(V_k, \rho_{G'}(V_k), D)$ be the quality of vertex V_k in G and the quality of V_k in G' given D, respectively. Then,*

$$Q(G', D) - Q(G, D) = q(V_j, \rho_{G'}(V_j), D) - q(V_j, \rho_G(V_j), D)$$

The B search heuristic now exploits the property stated in Corollary 5.2.5 for computing the gain in quality that would result from adding a new arc to the digraph in the making.

The following pseudocode summarises the B search heuristic.

```
procedure construct-digraph($V$,$D$,$G$)
    for each variable $V_i \in V$ do
        $\rho(V_i) := \varnothing$
    od;
    repeat
        for each (ordered) pair $V_i, V_j \in V$ such that addition of the arc $(V_i, V_j)$
            to $G$ does not introduce a (directed) cycle do
            $diff(V_i, V_j) := q(V_j, \rho(V_j) \cup \{(V_i, V_j)\}, D) - q(V_j, \rho(V_j), D)$
        od;
        select the pair $V_i, V_j \in V$ for which $diff(V_i, V_j)$ is maximal;
        if $diff(V_i, V_j) > 0$ then
            $\rho(V_j) := \rho(V_j) \cup \{V_i\}$
        fi
    until $diff(V_i, V_j) \leq 0$
end
```

To conclude, we would like to point out that the learning algorithm as outlined above is not guaranteed to find a minimal I-map for the joint probability distribution underlying the process that generated the database used. It will be evident that one reason for the learning algorithm not finding such a digraph is that it incorporates a search *heuristic* that considers only a limited number of possible acyclic digraphs and therefore incidentally may skip the minimal I-maps. Yet another reason lies in the quality measure used, however. For infinite size databases, the MDL-measure can be shown to prefer minimal I-maps over all other acyclic digraphs. Unfortunately, this property is not retained for finite size databases. So, by employing the MDL-measure for a finite size database, a digraph that is not a minimal I-map can be selected, even if every possible acyclic digraph were considered.

5.3 Assessing Probabilities

Only after the qualitative part of the probabilistic network in the making has been constructed and is considered robust, is its quantitative part specified. Specifying the quantitative part of a probabilistic network amounts to defining the probability assessment functions for the variables modelled in the network. The task of assessing all required probabilities tends to be by far the hardest task in probabilistic network building.

In order to facilitate the assessment task, approaches have been introduced that first attach qualitative signs to the arcs of a network's digraph, resulting in a so-called qualitative probabilistic network [Wellman, 1990]. The benefits are that these qualitative signs are easily elicited from domain expert, for example by using the method described by [Van der Gaag & Helsper, 2004]. In addition, the signs pose constraints on the probability distributions to be assessed, which can be exploited upon further assessment; different combinations of qualitative signs, for example, can be an indication of disjunctive interaction patterns, such as the *noisy-or* described in Section 5.3.2 [Lucas, 2005]. In this section we will not focus on such qualitative approaches to quantification of probabilistic networks; the interested reader is referred to [Renooij, 2001].

5.3.1 Sources of Probabilistic Information

In most problem domains, at least some probabilistic information for probabilistic network quantification is readily available, be it from literature or from domain experts. The most common sources of probabilistic information are [Jensen, 1995]:

- *literature* on the domain of application (textbooks, journals, conferences);
- *(statistical) data*;
- (other) *models* of domain knowledge;
- interviews with domain *experts*.

Literature on the domain of application often provides abundant probabilistic information. Unfortunately, this information is very seldom directly amenable to encoding in the probabilistic network in the making: the information typically is not complete, it concerns variables that are not causally related, and so on. Medical literature, for example, often reports conditional probabilities of the presence of symptoms given a disease, but not always the probabilities of these symptoms occurring in the absence of disease; moreover, conditional probabilities for unobservable intermediate disease states are usually lacking. An additional, commonly found problem that prohibits direct use of probabilistic information from literature is that the characteristics of the population from which the information is derived, is not properly described or deviates seriously from the characteristics of the population for which the probabilistic network is being developed [Korver & Lucas, 1993].

If the literature on the domain at hand does not provide for sufficient and reliable probability assessments for quantification of the network in the making, estimates may be obtained from statistical data or from other models of domain knowledge. Unfortunately, experience shows that even if comprehensive data collections and models are available, they very seldom contribute to the quantification task [Jensen, 1995, Korver & Lucas, 1993]. In a medical data collection, for example, unobservable intermediate pathophysiological states are typically not recorded.

As a consequence, the majority of the probabilities required will have to be assessed by domain experts. The problems encountered when eliciting probabilities from experts are widely known [Tversky *et al.*, 1982]; an expert's assessments may for example reflect various biases and may not be properly calibrated.

5.3.2 Simplifying Probability Assessment

The task of assessing all probabilities required for quantifying a probabilistic network can often be simplified by assuming the presence of one or more *disjunctive interactions* in the network [Pearl, 1988]. A disjunctive interaction pertains to two or more causes and their common effect: it specifies that any of these causes in itself suffices to cause the effect and that the likelihood of this cause causing the effect does not diminish when several other causes are present simultaneously.

More formally, we consider a set of statistical variables V and an acyclic digraph $G = (V(G), A(G))$ with $V(G) = V$ where $(V_1, V_0), \ldots, (V_m, V_0) \in A(G)$, for some vertices $V_0, V_1, \ldots, V_m$, $m \geq 1$. Note that the digraph G expresses that $V_1, \ldots, V_m$ model different causes of a common effect modelled by V_0. The variables $V_1, \ldots, V_m$ are said to exhibit a disjunctive interaction with respect to V_0 if the variables $V_1, \ldots, V_m$ adhere to the properties of accountability and exception independence. The property of *accountability* expresses that at least one of the causes modelled by $V_1, \ldots, V_m$ needs to be present for V_0 to adopt the value *true*, or alternatively, that the variable V_0 takes the value *false* whenever all its causes are absent. The property of accountability is stated in terms of probabilities as

$$\Pr(v_0 \mid \neg v_1 \wedge \cdots \wedge \neg v_m) = 0$$

We now turn to the property of *exception independence.* This property is best explained in terms of a *logical or-gate.* Note that in a logical or-gate, the presence of any one of the causes $V_1, \ldots, V_m$ suffices for V_0 to adopt the value *true.* Now, if the logical relationship between V_0 and its separate causes is perturbed, we say that the logical or-gate is *noisy.* In a noisy or-gate, for each cause V_i a so-called *exception mechanism* may *inhibit* the presence of the cause to result in $V_0 = $ *true*; such an exception mechanism is often termed an *inhibitor.* The exception mechanisms in a noisy or-gate may be looked upon as statistical variables. If the exception mechanism for a cause V_i has the value *true*, then it inhibits the occurrence of the effect $V_0 = $ *true*; if V_i's exception mechanism has the value *false*, then it does not prevent the occurrence of the effect upon presence of the cause modelled by V_i. We therefore can model the exception mechanism of a cause V_i by a statistical variable I_i such that

$$\Pr(v_0 \mid \neg v_1 \wedge \cdots \wedge \neg v_{i-1} \wedge (v_i \wedge i_i) \wedge \neg v_{i+1} \wedge \cdots \wedge \neg v_m) = 0$$

and

$$\Pr(v_0 \mid \neg v_1 \wedge \cdots \wedge \neg v_{i-1} \wedge (v_i \wedge \neg i_i) \wedge \neg v_{i+1} \wedge \cdots \wedge \neg v_m) = 1$$

Exception independence for $V_1, \ldots, V_m$ now is the property that the statistical variables $I_1, \ldots, I_m$ modelling the exception mechanisms for $V_1, \ldots, V_m$, respectively, are mutually independent. Because of the analogy outlined above, a disjunctive interaction is often coined a *noisy or-gate.*

The information that $V_1, \ldots, V_m$ exhibit a disjunctive interaction with respect to V_0 is not directly amenable to encoding in a probabilistic network. The information, however, imposes strong restrictions on the probability assessment function γ_{V_0} for the vertex V_0 in the network. To support this observation, we consider the various values that have to be assessed to define the function γ_{V_0}, separately:

- for the value $\gamma_{V_0}(v_0 \mid \neg v_1 \wedge \cdots \wedge \neg v_m)$, we have

  $$\gamma_{V_0}(v_0 \mid \neg v_1 \wedge \cdots \wedge \neg v_m) = 0$$

  from the property of accountability of a disjunctive interaction;

- for $i = 1, \ldots, m$, let I_i be the inhibitor for cause V_i and let $\Pr(i_i) = q_i$; then, we have that

  $$\gamma_{V_0}(v_0 \mid \neg v_1 \wedge \cdots \wedge \neg v_{i-1} \wedge v_i \wedge \neg v_{i+1} \wedge \cdots \wedge \neg v_m) = 1 - q_i$$

- let c be an arbitrary configuration of the set of causes $\{V_1, \ldots, V_m\}$ and let $T_c = \{i \mid c \wedge v_i \not\equiv \mathsf{False}\}$; then, we have that

  $$\gamma_{V_0}(v_0 \mid c) = 1 - \prod_{i \in T_c} q_i$$

  from the property of exception independence of a disjunctive interaction.

From these observations we have that, once causes $V_1, \ldots, V_m$ are known to exhibit a disjunctive interaction with respect to V_0, it suffices to assess only n probabilities to define the probability assessment function γ_{V_0} for vertex V_0.

Recall that a disjunctive interaction adheres to the properties of accountability and exception independence. We take a closer look at the property of accountability. Recall that this property states that all causes of an effect are known and explicitly modelled in the probabilistic network in the making. We observe that this property is hardly ever satisfied in real-life applications. Since incompleteness is inherent to any model of domain knowledge, a probabilistic network will never be complete: there will always be possible causes of an effect that are not explicitly modelled in the network. Unfortunately, if the property of accountability is not satisfied, it is not possible to exploit the model of the noisy or-gate as described above — not even it the property of exception independence is satisfied.

Consider the variables $V_1, \ldots, V_m$ modelling different causes of a common effect expressed by V_0. Now, assume that the property of exception independence is satisfied and that the property of accountability is not: we have that

$$\Pr(v_0 \mid \neg v_1 \wedge \cdots \wedge \neg v_m) = p$$

where $p > 0$. To be able to nevertheless exploit the model of the noisy or-gate, we can extend the probabilistic network in the making with an extra statistical variable in such a way that the property of accountability is enforced and the property of exception independence is retained. To this end, we introduce a new variable V_{m+1} into the network as modelling an extra cause of the effect V_0 — this variable then is taken to capture all causes of V_0 that are not yet modelled in the original network. For this variable V_{m+1}, we then have

$$\Pr(v_0 \mid \neg v_1 \wedge \cdots \wedge \neg v_m \wedge \neg v_{m+1}) = 0$$

and

$$\Pr(v_0 \mid \neg v_1 \wedge \cdots \wedge \neg v_m \wedge v_{m+1}) = p$$

Furthermore, we take V_{m+1} to be independent of $V_1, \ldots, V_m$. Note that the introduction of the variable V_{m+1} has not altered the meaning of the original causal mechanism. Also note that the property of accountability is now satisfied by $V_1, \ldots, V_{m+1}$ and that the model of the noisy or-gate applies.

Although the introduction of an extra variable as outlined before allows for exploiting the model of the noisy or-gate in the case where the property of accountability is not satisfied, adding an extra variable to the network is not satisfactory from a knowledge representational point of view: the additional variable has no well-defined meaning and therefore in a sense pollutes the probabilistic network. The model of the *leaky* noisy or-gate now allows for capturing all yet unmodelled causes of a common effect *implicitly* [Henrion, 1989]. Consider once more the variables $V_0, V_1, \ldots, V_m$ and assume that the property of exception independence is satisfied and the property of accountability is not; recall that we have that

$$\Pr(v_0 \mid \neg v_1 \wedge \cdots \wedge \neg v_m) = p$$

where $p > 0$. The probability p is called a *leak probability* and is looked upon as expressing the probability that the effect v_0 occurs *spontaneously.* Now, by introducing an extra inhibitor I_0 with probability $\Pr(i_0) = q_0 = 1 - p$ similar observations apply to the leaky noisy or-gate as for the noisy or-gate: the information that the variables $V_0, V_1, \ldots, V_m$ constitute a leak noisy or-gate again imposes strong restrictions on the definition of the probability assessment function γ_{V_0} for the variable V_0:

- for the probability $\gamma_{V_0}(v_0 \mid \neg v_1 \wedge \cdots \wedge \neg v_m)$, we have

  $$\gamma_{V_0}(v_0 \mid \neg v_1 \wedge \cdots \wedge \neg v_m) = p$$

- for $i = 1, \ldots, m$, let I_i be the inhibitor for cause V_i and let $\Pr(i_i) = q_i$; then, we have that

  $$\gamma_{V_0}(v_0 \mid \neg v_1 \wedge \cdots \wedge \neg v_{i-1} \wedge v_i \wedge \neg v_{i+1} \wedge \cdots \wedge \neg v_m) = 1 - q_i$$

- let c be an arbitrary configuration of the set of causes $\{V_1, \ldots, V_m\}$, let $T_c = \{i \mid c \wedge v_i \not\equiv \mathsf{False}\}$, and let $q_0 = 1 - p$; then, we have that

  $$\gamma_{V_0}(v_0 \mid c) = 1 - q_0 \cdot \prod_{i \in T_c} \left(\frac{q_i}{q_0}\right)$$

  from the property of exception independence of a disjunctive interaction.

Note that the values for the probability assessment function γ_{V_0} are defined much in the same way as in the model of the noisy or-gate. Also note that, since in the probabilistic network in the making the effect of a cause cannot be isolated from the spontaneous occurrence of the effect, the probability assessments of the effect given a *single* cause are adjusted to account for this spontaneous leak before they are combined.

5.3.3 Eliciting Probabilities from Experts

The field of *decision analysis* offers various techniques for the elicitation of judgemental probabilities from experts [Von Winterfeldt & Edwards, 1986, Morgan & Henrion, 1990]. We briefly review the two techniques that are most often used for eliciting probabilities for probabilistic networks. The simplest technique is the use of a numerical *probability scale*. A probability scale is a horizontal or vertical line with the endpoints denoting a 0% and a 100% chance, respectively, and a few numerical anchors in between, for example to denote a 50% chance; Figure 5.2 illustrates the basic idea. For each probability required, a domain expert is asked to indicate his or her assessment on a separate scale. In communicating a probability to be assessed to a domain expert, the probability is often transcribed verbally in terms of frequencies. The expert is asked to envisage one hundred cases within a specific context and assess the number of cases that exhibit a certain characteristic. Experience shows that the use of a probability scale along with the frequency method provides experts little to go by and may result in highly inaccurate probability assessments [Van der Gaag *et al.*, 1999].

A more elaborate technique for the elicitation of judgemental probabilities is the use of *reference lotteries*. A domain expert is presented with a choice between two lotteries, one of which pertains to the probability to be assessed and the other one serves as a reference. The reference lottery yields a desired reward with probability p and a less desired outcome with probability $1 - p$. The second lottery yields the same desired reward if a specific case exhibits a certain characteristic and the less desired outcome otherwise.

Example 5.3.1 Suppose that, in a medical domain of application, an expert has to assess the probability of a specific patient with metastatic cancer showing an increased level of serum calcium. The domain expert is presented, for instance, with a choice between a reference lottery and the lottery that yields \$10,000 if the patient upon examination shows an increased level of serum calcium and \$1 if the level of serum calcium is not increased in the patient. Figure 5.3 depicts this choice between lotteries. □

The domain expert is asked to adjust the value of p in the reference lottery until he or she is indifferent between the two lotteries. The resulting value of p then is taken to be the conditional probability that had to be assessed. Experience shows that the use of reference lotteries for eliciting probabilities from domain experts may avert to at least some extent the problems of bias and poor calibration that are typically found in human probability assessment. The use of lotteries, however, tends to be quite time-consuming and, in fact, often turns out to be infeasible for probability elicitation for probabilistic networks as a result of the large size and complexity of a network in the making.

While the use of lotteries for probability elicitation on the one hand tends to be infeasible for quantifying a probabilistic network, the use of a probability scale on the

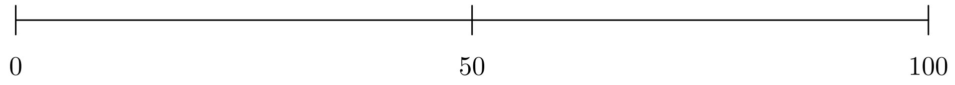


Figure 5.2: A numerical scale for probability elicitation.

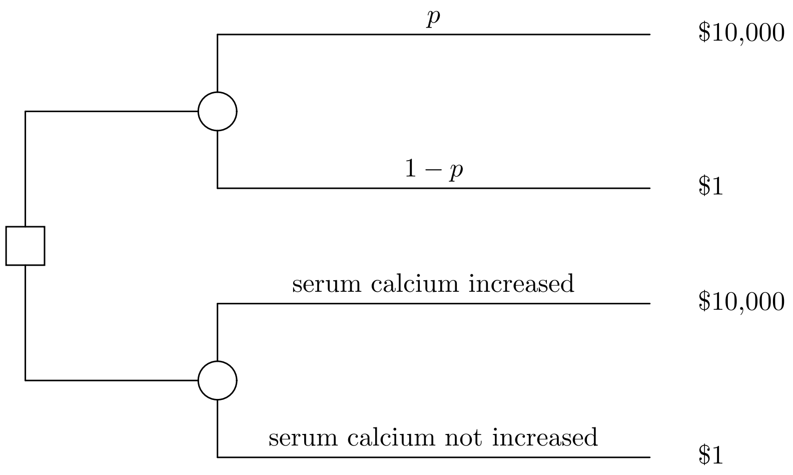


Figure 5.3: An example reference lottery.

other hand tends to yield assessments that are too much inaccurate. The use of a probability scale, nonetheless, serves to yield an assessment, be it an inaccurate one, for every conditional probability required. These assessments may then be used as a starting point for further refinement. More details on probability elicitation methods, including a method designed by Utrecht University researchers, which is based on the use of a verbal-numerical probability scale, can be found in [Renooij, 2001b].

5.3.4 A Procedure for Probability Refinement

The assessments obtained for a probabilistic network are inevitably inaccurate, due to incompleteness of data and partial knowledge of the problem under study. Particularly assessments obtained from experts are known to be highly inaccurate [Tversky *et al.*, 1982]. The inaccuracies in the probability assessments for a probabilistic network influence the reliability of the network's output. *Sensitivity analysis* is a general technique for studying the effects of the uncertainties in the parameters of a mathematical model on this model's outcome [Morgan & Henrion, 1990]. For a probabilistic network, sensitivity analysis provides for example for studying the effects of the uncertainties in the assessments for the network's conditional probabilities on a probability of interest. There are various different types of sensitivity analysis. For a probabilistic network, the simplest type of sensitivity analysis amounts to systematically varying the assessment for one of the network's probabilities while keeping all other assessments fixed. Such an analysis serves to reveal the effect of just the conditional probability whose assessment is being varied, on a probability of interest. A sensitivity analysis in which a single assessment is varied, is termed a *one-way sensitivity analysis.* In a *two-way sensitivity analysis* of a probabilistic network, two probability assessments are varied simultaneously. In addition to the separate effects of variation of the two assessments, a two-way sensitivity analysis reveals the joint effect of their variation on a probability of interest. In essence, it is also possible to systematically vary more than two probability assessments at the same time. The results of such an analysis, however, are often hard to interpret.

Sensitivity analysis of a probabilistic network provides for studying the effects of the uncertainties in the various probability assessments of the network on a probability of interest. Studying these effects can to a large extent serve to support the elicitation

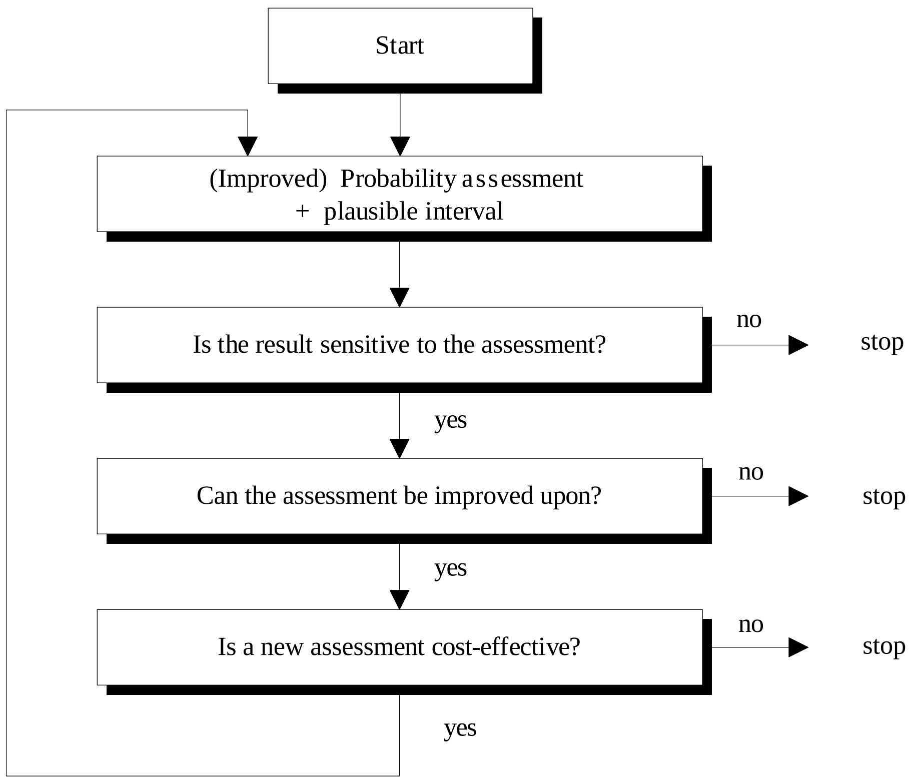

Start
(Improved) Probability assessment + plausible interval
Is the result sensitive to the assessment?
no
stop
yes
Can the assessment be improved upon?
no
stop
yes
Is a new assessment cost-effective?
no
stop
yes

Figure 5.4: A procedure for probability elicitation for probabilistic networks.

of probabilities, as it gives detailed insight into the level of accuracy that is required for the various probabilities of the network and as a result provides for focusing further elicitation efforts. We describe an *elicitation procedure* in which, alternatingly, sensitivity analyses are performed of a probabilistic network in the making and probability assessments are refined; this procedure is summarised in Figure 5.4. We elaborate on the various different steps of the procedure separately.

Step 1 In the first step of the elicitation procedure, *initial assessments* are acquired for all conditional probabilities for a probabilistic network in the making. As we have argued before, for most domains of application, experts will have to provide the majority of these initial assessments. In this phase of the construction of the probabilistic network, we apply an elicitation technique that aims at acquiring assessments *within a short period of time*. To this end, in a limited number of interview sessions, experts are asked to assess all probabilities required off the top of their heads, for example using a numerical, or verbal-numerical, probability scale. In addition, they are asked to indicate *plausible intervals* along with their assessments, that define a range of values in which the 'true' probability lies with reasonable certainty. These intervals should be pessimistic rather than optimistic to ensure that the uncertainties in the various different assessments are not underestimated. We would like to note that, since they are allowed to express the uncertainties in their assessments, experts will tend to be less reluctant to provide numerical statements than when they feel compelled to give exact numbers. Instead of eliciting all probabilities required from domain experts, initial assessments, for at least some of these probabilities, may be obtained from data, if available. If the data at hand is known to be imperfect, incomplete, or biased, then the assessments derived should be supplemented with fairly large plausible intervals to capture the uncertainties involved.

Step 2 The probability assessments obtained in the first step of the elicitation procedure will in general be highly uncertain, that is, these assessments will and should have considerably large plausible intervals. For some of the probabilities, these initial assessments will nonetheless suffice. Other probabilities, however, will require assessments with a higher level of accuracy. The second step of our elicitation procedure is aimed at uncovering the latter probabilities. For this purpose, the probabilistic network in the making is subjected to various *sensitivity analyses.* In these analyses, every single probability assessment for the network is varied, as is every pair of assessments. These analyses require a lot of computation and generate a huge amount of data from which we need to decide which probability assessments require further attention. This computational and informational burden is further discussed in Section 6.1, which also includes the technical details of sensitivity analyses. We would like to note that, if performing all these sensitivity analyses tends to be too much time-consuming to be practicable, the analyses can be restricted to a moderate number of assessments. To this end, experts may be asked to indicate the probability assessments that are the most likely to affect a probability of interest or, alternatively, to indicate a number of conceptually related probability assessments.

Step 3 In the second step of the elicitation procedure, various probability assessments have been identified that induce a considerably large effect on a probability of interest (see Section 6.1 on criteria for determining what 'large effect' means). These assessments are potential candidates for further refinement. In the third step of the procedure, the extent to which these assessments can actually be refined is determined. The extent to which a probability assessment lends itself to further refinement depends on the current uncertainty in the assessment, indicated by the width of its plausible interval, and on the elicitation techniques used to arrive at the assessment. A probability assessment with a rather small plausible interval obtained from applying elaborate elicitation techniques may not lend itself to further refinement. An assessment that is not yet very certain, on the other hand, may be more easily improved upon. Refinement of such an uncertain assessment, however, is only actually possible if reliable sources of probabilistic information remain to be explored; examples of such sources of information include further literature search, the use of a panel of experts, the use of more elaborate elicitation techniques, and the study of data, for instance collected in a prospective study. From among the potential candidates for further refinement, therefore, the assessments are identified that have a considerable plausible interval and for which yet unexplored sources of probabilistic information are available.

Step 4 The probability assessments that have been selected in the third step of the elicitation procedure are assessments that *can* be refined. To actually refine these assessments, an investment of time and money is required. Not for every assessment may this investment be worthwhile, however. In the fourth step of the elicitation procedure, therefore, it is investigated for each of the selected assessments whether refinement is *cost-effective.* The cost-effectiveness of refining a probability assessment is determined by weighing the costs in terms of money and time to be invested with the benefits of higher accuracy. The benefits of having an assessment of higher accuracy in the probabilistic network in the making may be a higher accuracy of a computed probability of interest and an improved performance more in general. For example, for a probabilistic network that is to be used for diagnostic purposes, performance may be measured as the percentage of correctly diagnosed cases. Refining a probability assessment for this network would only be worthwhile if it would increase the number of correct diagnoses (see Chapter 6 on evaluation). Once a probabilistic network in the making exhibits satisfactory overall behaviour, refining assessments may be found to be no longer cost-effective.

Recursive step The probability assessments that have been identified in the fourth step of the elicitation procedure are known to induce a considerable effect on a probability of interest; moreover, any such assessment can be cost-effectively refined. For these assessments, the entire elicitation procedure is recursively repeated. The probabilities concerned are assessed anew in the first step of the next cycle of the procedure, using yet unexplored sources of probabilistic information. Since the plausible intervals of the initial assessments for these probabilities do not underestimate the uncertainties involved, refinement is likely to result in smaller plausible intervals for the new assessments. In the second step, once again several sensitivity analyses are performed. These analyses should not only focus on the new assessments, but also on previously investigated assessments as their effect on a probability of interest upon variation may have changed as a result of the refinement of the network. In addition, analyses may be performed with respect to assessments that have not been investigated before. By recursively refining probability assessments, the performance of the probabilistic network in the making is likely to gradually improve. The elicitation procedure is stopped as soon as the costs of further elicitation outweigh the improvement in the network's performance or higher accuracy can no longer be attained due to lack of knowledge.

Exercises

Exercise 5.1

In constructing the qualitative part of a probabilistic network for a domain of application, a knowledge engineer aims at a digraph that is as sparse as possible. Give at least two reasons for this goal.

Exercise 5.2

In the construction of the qualitative part of a probabilistic network for a domain of application, a cycle may arise. Describe at least two methods for removing the cycle from the network in the making.

Exercise 5.3

Suppose that you are asked to construct a probabilistic network for a major medical center in Utrecht, to support specialists in the diagnosis and prognostication of various types of vascular disease. The network is to be constructed by hand, with the help of a domain expert.

a. *The expert indicates that there are four important types of vascular disease; these types of disease are not mutually exclusive. Now, suppose that for pragmatical reasons, the network should have just one hypothesis variable. Explain how the various types of disease can be modelled in the network's digraph.*

b. *Suppose that you decide to model a single-valued discrete domain variable by means of* two *statistical variables. Explain how the relationship between these two variables can be modelled.*

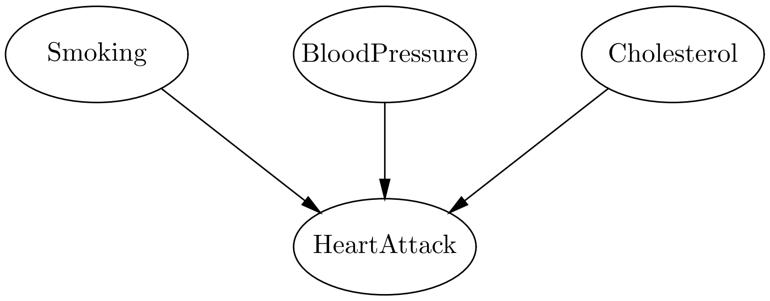
c. *After the digraph of the network has been completed, it becomes clear that it requires simply too many probabilities for its quantification. Describe at least four ways to reduce the number of probabilities required. For each of these, describe in which situations it can lead to a substantial reduction of the number of probabilities required and in which situations it cannot.*

Exercise 5.4

Suppose that for the hospitals in the Netherlands, a probabilistic network is being built for the diagnosis of acute cardiac disorders and that the network is being handcrafted with the help of domain experts.

a. *For eliciting the topology of the digraph of the network in the making, the concept of causality is used as a heuristic guiding rule. Give an example showing that the concept of causality is not always suitable for this purpose.*

Suppose that the digraph of the probabilistic network in the making comprises the following subgraph:



For the variable Smoking *the values* true *and* false *have been identified, denoted as s and $\neg s$, respectively; for the variable* BloodPressure *the values b and $\neg b$ have been identified, for the variable* Cholesterol *the values c and $\neg c$, and for the variable* HeartAttack *the values h and $\neg h$.*

b. *Suppose that from an emergency medical center in New York, a data collection is available with the medical records of 12738 patients with heart disease. Explain why this data collection cannot be used just like that for the assessment of the probabilities required for the variable HeartAttack in the network fragment above.*

c. *Before commencing with the elicitation of the required probabilities, you would like to ask the domain expert whether or not a* disjunctive interaction *may be assumed for the variable* HeartAttack *and its parents. Which questions would you pose?*

* d. *Now suppose that the domain expert indicates that the variables* Smoking, Blood-Pressure *and* Cholesterol *satisfy the property of exception independence with respect to the variable* HeartAttack. *The domain expert further assesses the following probabilities:*

$$Pr(h \mid \neg s \wedge \neg b \wedge \neg c) = 0.05$$
$$Pr(h \mid s \wedge \neg b \wedge \neg c) = 0.6$$
$$Pr(h \mid \neg s \wedge b \wedge \neg c) = 0.8$$
$$Pr(h \mid \neg s \wedge \neg b \wedge c) = 0.9$$

Is it possible to specify a complete probability assessment function for the variable HeartAttack *on the basis of the available information ? If no, then which information is missing; if yes, then give the complete assessment function.*

* e. *In practice, building a probabilistic network for a diagnostic application such as the one described above often turns out to be easier than the construction of a rule-based expert system for the same application. Give at least one explanation for this observation.*

* Exercise 5.5

Suppose that with the help of a domain expert, a digraph of a probabilistic network is constructed consisting of the variables V_1, V_2, V_3, and V_4, and the arcs (V_1, V_2), (V_2, V_3), and (V_2, V_4).

From the literature on the domain under study, the following probabilistic information is available about the four variables:

$$\Pr(v_1 \wedge v_2) = 0.25$$
$$\Pr(\neg v_1 \wedge \neg v_2) = 0.3$$
$$\Pr(v_2 \wedge \neg v_3) = 0.2$$
$$\Pr(v_2 \wedge v_3) = 0.25$$
$$\Pr(\neg v_2 \wedge v_3) = 0.15$$
$$\Pr(v_3 \mid \neg v_1 \wedge v_2 \wedge v_4) = 0.4$$
$$\Pr(v_4) = 0.8$$

No further information had been found. Construct the assessment functions for as many variables as possible from the available information.

Exercise 5.6

Consider the following procedure for eliciting probability assessments from domain experts:

- *the expert is asked to use his own keywords and phrases to indicate his assessments for the various probabilities;*
- *the expert is then asked to rank order the keywords and phrases he used;*
- *the knowledge engineer subsequently associates numerical probabilities with the various keywords and phrases.*

Describe the strengths and weaknesses of this procedure.

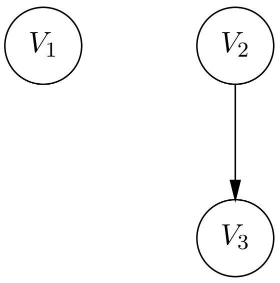
Exercise 5.7

* *a.* *Let $V = \{V_1, V_2, V_3\}$ be a set of binary statistical variables. Let D be a database over V comprising the following cases:*

$v_1 \wedge \neg v_2 \wedge \neg v_3$	$\neg v_1 \wedge v_2 \wedge v_3$
$v_1 \wedge v_2 \wedge v_3$	$\neg v_1 \wedge \neg v_2 \wedge \neg v_3$
$v_1 \wedge \neg v_2 \wedge \neg v_3$	$v_1 \wedge \neg v_2 \wedge \neg v_3$
$v_1 \wedge v_2 \wedge v_3$	$\neg v_1 \wedge \neg v_2 \wedge v_3$
$v_1 \wedge \neg v_2 \wedge \neg v_3$	$\neg v_1 \wedge \neg v_2 \wedge \neg v_3$
$\neg v_1 \wedge \neg v_2 \wedge v_3$	$v_1 \wedge v_2 \wedge v_3$
$\neg v_1 \wedge v_2 \wedge \neg v_3$	$v_1 \wedge v_2 \wedge v_3$
$v_1 \wedge \neg v_2 \wedge \neg v_3$	$v_1 \wedge v_2 \wedge \neg v_3$
$v_1 \wedge \neg v_2 \wedge \neg v_3$	$v_1 \wedge v_2 \wedge v_3$
$v_1 \wedge \neg v_2 \wedge v_3$	$v_1 \wedge v_2 \wedge v_3$

There is no other information available on the variables V_1, V_2 and V_3.

Suppose that the database D is exploited for automated construction of a probabilistic network. For constructing the network, the B search heuristic is used in combination with the MDL quality measure. Now suppose that at some stage, the following digraph is constructed:



$$\begin{tikzcd}
V_1 & V_2 \arrow[d] \\
 & V_3
\end{tikzcd}$$

Compute the difference in quality that is achieved by adding the arc (V_1, V_2) to this digraph.

b. *In practical applications, the B search heuristic is applied not only in combination with the MDL quality measure, but also in combination with other quality measures. An example of such a measure is the* Akaike information criterion*; this criterion is defined as*

$$Q_A(G, D) = \log P(G) - N \cdot H(G, D) - K$$

where G, D, P, N, H and K have the same meaning as in the MDL quality measure. Now, for a given database D, let G_A be the digraph that is yielded by the B search heuristic with the Akaike information criterion and let G_{MDL} be the digraph that is yielded by the heuristic with the MDL quality measure. Describe in which respect the digraphs G_A and G_{MDL} will differ in general.

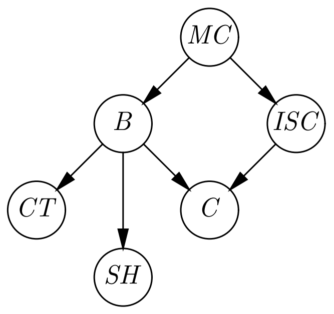
Chapter 6

Bringing Probabilistic Networks into Practice

This chapter tries to bridge the gap between construction and actual use of a probabilistic network application. It fist discusses possible methods for evaluating the behaviour of your network; then some example applications where a probabilistic network is used as a component in a decision support process are discussed.

In various domains of application, ranging from medicine to meteorology, knowledge-based systems are being developed that build upon a probabilistic network for their knowledge representation. The previous chapter has dealt with the construction of such a probabilistic network. In this chapter, we discuss two issues that are concerned with the actual use of a probabilistic network in practice. Before a probabilistic network is introduced into practice, it should be analysed whether the network behaves as expected, and produces acceptable output. In Section 6.1, we will discuss sensitivity analysis as a general method for studying a probabilistic network's robustness. In Section 6.2, we discuss the evaluation of the practical value of a probabilistic network, by providing some measures for establishing a network's quality.

As experience with applying the probabilistic network framework is building, it becomes apparent that, although the framework offers many advantages over earlier approaches to automated reasoning with uncertainty, it lacks with regard to *intelligent control over reasoning.* A probabilistic network provides for computing probabilities only, whereas in practice often decisions, such as what diagnostic tests to perform or what therapy to instill, are to be based on the computed probabilities. The provision of control over reasoning is generally considered one of the main contributions of artificial intelligence research to automated reasoning: knowledge-based systems thank their success to a large extent to their ability to apply specialised knowledge for pruning search spaces and for selectively gathering evidence. Since control over reasoning is a prerequisite for arriving at problem-solving behaviour that is satisfactory both from a computational and a user's point of view, it is not surprising that providing for control is an important issue in present-day probabilistic network research. In Section 6.3, we describe a problem-solving architecture that allows for automated control over reasoning.



$\gamma(mc) =$	0.20	$\gamma(isc \mid mc) =$	0.80
		$\gamma(isc \mid \neg mc) =$	0.20
$\gamma(b \mid mc) =$	0.20		
$\gamma(b \mid \neg mc) =$	0.05	$\gamma(ct \mid b) =$	0.95
		$\gamma(ct \mid \neg b) =$	0.10
$\gamma(c \mid b, isc) =$	0.80		
$\gamma(c \mid \neg b, isc) =$	0.80	$\gamma(sh \mid b) =$	0.80
$\gamma(c \mid b, \neg isc) =$	0.80	$\gamma(sh \mid \neg b) =$	0.60
$\gamma(c \mid \neg b, \neg isc) =$	0.05		

Figure 6.1: An example probabilistic network

6.1 Sensitivity Analysis

The reliability of the output of a probabilistic network can be investigated by studying its robustness. Robustness pertains to the extent to which the network's conditional probabilities influence the output when deviations from the specified assessments are assumed. In a medical application, for example, erroneous diagnoses or non-optimal treatment recommendations may result from building upon inaccurate assessments. For gaining detailed insight in output robustness, a probabilistic network can be subjected to a sensitivity analysis. In the previous chapter, sensitivity analysis was mentioned as part of a probability elicitation procedure. Sensitivity analysis, in addition, can be used as a technique underlying parameter tuning, that is, changing probability assessments so that the network gives the desired output. Perhaps most importantly, sensitivity analysis can give insight into the range of probability assessments, or even evidence profiles, for which the outcome of the network is valid.

A sensitivity analysis of a mathematical model basically amounts to stepwise variation of one or more parameters and computing the output of the model in each step. This is a very demanding process from a computational point of view. Fortunately, the output probability of interest in a probabilistic network relates to the probability assessments, or *parameters*, varied by a simple mathematical function which can be computed quite efficiently from the network under study [Kjærulff & Van der Gaag, 2000, Coupé & Van der Gaag, 2002]. The form of this function heavily depends on how the variables corresponding to the parameters of interest and output of interest are located relative to each other in the network, as well as on the evidence entered. In this syllabus we restrict the discussion to *one-way* and *two-way* sensitivity analyse, where only one or two parameters, respectively, are varied simultaneously. We will consider the general form of the functions, followed by some properties, criteria for selecting interesting parameters, and examples.

As a running example we now consider the network in Figure 6.1, describing a piece of (fictitious and incomplete) medical information. The fragment describes the problems of metastatic cancer (denoted *MC*) for an arbitrary patient, with regard to the development of a brain tumour. Metastatic cancer may be detected by an increased level of serum calcium (*ISC*). The presence of a brain tumour (*B*) may be established from a CT scan (*CT*). Severe headaches (*SH*) are indicative of the presence of a brain tumour. Both a brain tumour and an increased level of serum calcium are likely to ultimately cause a patient to fall into a coma (*C*). The strengths of these dependences are described by the conditional probabilities. The probabilities specified for the variable *ISC*, for example, express that knowing whether or not metastatic cancer is present has a considerable influence on the probability of an increased level of serum calcium in an arbitrary patient. On the other hand, severe headaches are expressed as quite common in both patients with and without a brain tumour. Severe headaches thus have a low predictive value for a brain tumour. From the conditional probabilities specified for vertex C, we see that in the absence of both a brain tumour and an increased level of serum calcium, there is only a very small probability of a patient falling into a coma. The presence of either one of these causes in an arbitrary patient, however, suffices to render the probability of this patient falling into a coma in the near future quite high. Note that the two causes do not contribute to this probability independently: if one of the causes is present, then the presence of the other cause has no further influence on the probability of a patient falling into a coma. The two causes are said to exhibit a (negative) *synergistic influence* on their common effect.

6.1.1 What to Analyse?

A sensitivity analysis with respect to a *prior* probability of interest allows for assessing the quality and robustness of a probabilistic network in its reflecting a prior probability distribution for the domain of application. In the presence of case-specific observations, however, a probabilistic network may show very different sensitivities. To reveal these sensitivities, a sensitivity analysis may be performed with respect to a *posterior*, or conditional, probability. Such an analysis allows for validating the network's behaviour for specific cases or profiles, for example, profiles of typical patient populations in a medical application.

It is clear that a full-blown sensitivity analysis quickly becomes infeasible as it requires analysing the relation between each parameter, or combination of parameters, in the network, and every possible output probability of interest. The number of different output probabilities is exponential in the number of variables in the network. Fortunately, not every theoretically possible output probability will be of interest in the domain of application. In addition, for a given probability of interest, not every parameter has to be varied. In fact, only parameters pertaining to variables in the *sensitivity set* for a variable of interest need to be included in the sensitivity analysis. Given evidence e for a set of variables E, the sensitivity set for a variable of interest A contains all variables whose parameters may affect the posterior probability distribution for A upon variation. More precisely, the sensitivity set is the set of variables V for which *none* of the following holds:

- $V \notin \rho^*(A)$ and $\sigma^*(V) \cap E = \emptyset$;
- $V \in \rho^*(A)$ and $\langle \{V\} \cup \rho(V) \mid E \mid \{A\} \rangle^d$;
- $V \notin \rho^*(A)$, $\langle \{V\} \cup \rho(V) \mid E \mid \{A\} \rangle^d$ and $\sigma^*(V) \cap E \neq \emptyset$;

A sensitivity set clearly depends to a large extent on the case-specific observations that have been entered into the network. In our example probabilistic network, for instance, once the presence or absence of a metastatic cancer has been established in a patient, varying the probability assessments for the variable *ISC* can no longer influence $\Pr(b)$. The sensitivity set can be readily identified by inspecting the network's qualitative part; more details can be found in [Coupé & Van der Gaag, 2002].

Upon varying a parameter, we have to make sure that the probabilities pertaining to the same conditional distribution from which we take the parameter continue to sum to one. Upon varying a single parameter, we adopt the standard assumption of *proportional scaling*: the other parameters are co-varied such that their mutual proportional relationship is kept constant.

6.1.2 One-way Sensitivity Analysis

For a probabilistic network, sensitivity analysis basically amounts to establishing, for each of the network's conditional probabilities, the *sensitivity function* that expresses a probability of interest in terms of the parameter under study.

In the sequel, we denote the probability of interest by $\Pr(A = a \mid e)$, or $\Pr(a \mid e)$ for short, where a is a specific value of the variable A of interest and e denotes the available evidence. The network's parameters are denoted by $x = \gamma(b_i \mid \pi)$, where b_i is a value of a variable B and π is a combination of values for the predecessors of B. We use $f_{\Pr(a|e)}(x)$ to denote the function that expresses the probability $\Pr(a \mid e)$ in terms of the parameter x; we often omit the subscript for the function symbol f, as long as ambiguity cannot occur.

Defining the sensitivity function

Under the assumption of proportional scaling, any sensitivity function is a quotient of two functions that are (multi-)linear in the parameters under study. The numerator of the quotient describes the probability $\Pr(a \wedge e)$ as a function of the parameters and the denominator describes $\Pr(e)$ as a function of the parameters. More formally, in a one-way analysis, the function takes the form

$$f(x) = \frac{c_1 \cdot x + c_2}{c_3 \cdot x + c_4}$$

where the constants c_j, $j = 1, \ldots, 4$, are built from the assessments for the numerical parameters that are not being varied.

If no evidence is entered into a probabilistic network, then the prior probability of interest relates linearly to any network parameter. Moreover, if the parameter under study pertains to a variable that is an ancestor of the variable of interest in the network's qualitative part, and the parameter's variable has no observed descendants, then the sensitivity function reduces to a linear function as well ($c_3 = 0$). For any parameter associated with a variable outside the sensitivity set of the variable of interest, the sensitivity function is constant. We conclude that a sensitivity function is either a *linear* function or a fragment of a *rectangular hyperbola*:

$$f(x) = \frac{r}{x - s} + t, \text{ with } s = -\frac{c_4}{c_3}, \quad t = \frac{c_1}{c_3}, \quad \text{and} \quad r = \frac{c_2}{c_3} + s \cdot t$$

A rectangular hyperbola has two branches and two asymptotes. Figure 6.2 illustrates the locations of the possible hyperbola branches relative to the two asymptotes. For $r < 0$, the branches lie in the second (II) and fourth (IV) quadrants relative to the asymptotes $x = s$ and $f(x) = t$; for $r > 0$, the branches are found in the first (I) and third (III) quadrants. Since any sensitivity function is continuous for $x \in [0, 1]$, a hyperbolic sensitivity function is actually a fragment of one of the four possible hyperbola branches.

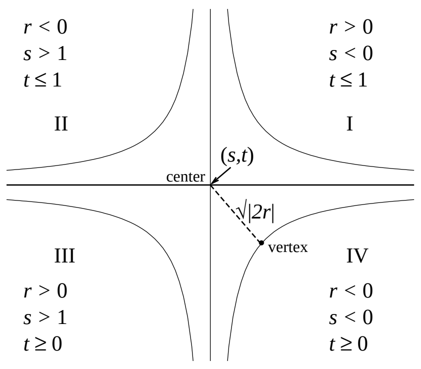


Figure 6.2: Hyperbolas and their constants (the constraints on s and t are specific for sensitivity functions)

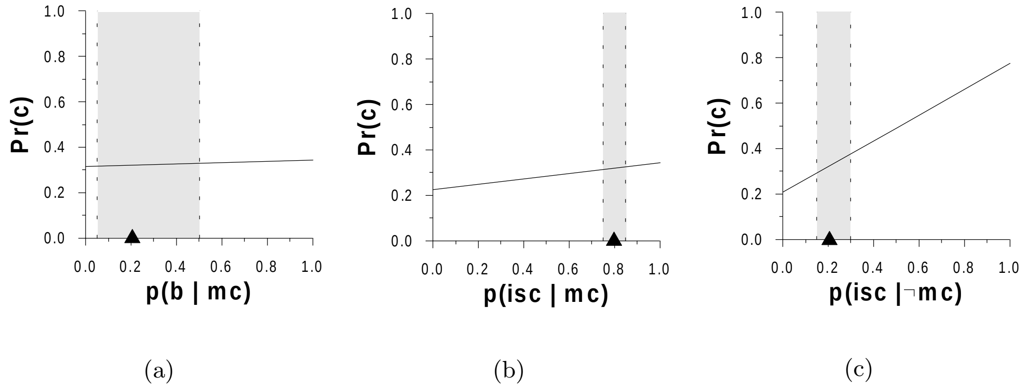
Computing the sensitivity function

There basically exist two methods for computing the constants of a sensitivity function. In the first method, the constants are determined by computing from the network the probability of interest for up to three values for the parameter under study, and subsequently solving the resulting system of linear equations [Coupé & Van der Gaag, 2002]; for the network computations, any of the standard propagation algorithms can be used. If a propagation algorithm is used that builds upon junction trees, then the constants of a sensitivity function can be obtained more efficiently [Kjærulff & Van der Gaag, 2000].

Example 6.1.1 We illustrate performing a *one-way sensitivity analysis* of our example network. We begin by taking the prior $\Pr(c)$ for our probability of interest. We address the one-way analyses with respect to the parameters $\gamma(b \mid mc)$, $\gamma(isc \mid mc)$, and $\gamma(isc \mid \neg mc)$, respectively. The results of these three prior analyses are shown in Figure 6.3. The figure displays, for example, the probability of interest $\Pr(c)$ as a function of the parameter $x = p(isc \mid \neg mc)$, that is,

$$\Pr(c) = 0.57 \cdot x + 0.21$$

The general form of this formula follows, after marginalisation, from the factorisation



(a) (b) (c)

Figure 6.3: A one-way sensitivity analysis of the example probabilistic network.

of the joint distribution of our example network:

$$\begin{aligned}
\Pr(c) &= \sum_{c_{MC}} \sum_{c_B} \sum_{c_{ISC}} \gamma(c \mid c_B, c_{ISC}) \cdot \gamma(c_B \mid c_{MC}) \cdot \gamma(c_{ISC} \mid c_{MC}) \cdot \gamma(c_{MC}) \\
&= \sum_{c_{MC}} \sum_{c_B} \gamma(c \mid c_B, isc) \cdot \gamma(c_B \mid c_{MC}) \cdot \gamma(isc \mid c_{MC}) \cdot \gamma(c_{MC}) \\
&\quad + \sum_{c_{MC}} \sum_{c_B} \gamma(c \mid c_B, \neg isc) \cdot \gamma(c_B \mid c_{MC}) \cdot \gamma(\neg isc \mid c_{MC}) \cdot \gamma(c_{MC}) \\
&= \sum_{c_B} \gamma(c \mid c_B, isc) \cdot \gamma(c_B \mid mc) \cdot \gamma(isc \mid mc) \cdot \gamma(mc) \\
&\quad + \sum_{c_B} \gamma(c \mid c_B, isc) \cdot \gamma(c_B \mid \neg mc) \cdot \gamma(isc \mid \neg mc) \cdot \gamma(\neg mc) \\
&\quad + \sum_{c_B} \gamma(c \mid c_B, \neg isc) \cdot \gamma(c_B \mid mc) \cdot \gamma(\neg isc \mid mc) \cdot \gamma(mc) \\
&\quad + \sum_{c_B} \gamma(c \mid c_B, \neg isc) \cdot \gamma(c_B \mid \neg mc) \cdot \gamma(\neg isc \mid \neg mc) \cdot \gamma(\neg mc) \\
&= \sum_{c_B} \gamma(c \mid c_B, isc) \cdot \gamma(c_B \mid mc) \cdot \gamma(isc \mid mc) \cdot \gamma(mc) \\
&\quad + \sum_{c_B} \gamma(c \mid c_B, isc) \cdot \gamma(c_B \mid \neg mc) \cdot x \cdot \gamma(\neg mc) \\
&\quad + \sum_{c_B} \gamma(c \mid c_B, \neg isc) \cdot \gamma(c_B \mid mc) \cdot \gamma(\neg isc \mid mc) \cdot \gamma(mc) \\
&\quad + \sum_{c_B} \gamma(c \mid c_B, \neg isc) \cdot \gamma(c_B \mid \neg mc) \cdot (1 - x) \cdot \gamma(\neg mc)
\end{aligned}$$

You can now compute the constants directly from this expression and the given assessment functions. Alternatively, you can compute $\Pr(c)$ from the network for two different values of parameter x, e.g. x_1 and x_2 (note that this requires actually changing the assessment function γ_{ISC}!), resulting in two different output probabilities p_1 and p_2; the constants $c_1 \approx 0.57$ and $c_2 \approx 0.21$ then follow by solving the following system of linear equations:

$$\begin{aligned}
p_1 &= c_1 \cdot x_1 + c_2 \\
p_2 &= c_1 \cdot x_2 + c_2
\end{aligned}$$

Note that you choose the values of x_1 and x_2, and subsequently compute the probabilities p_1 and p_2 from the network. The above system therefore contains two expressions with two unknowns, c_1 and c_2, and can therefore be uniquely solved.

Next, we take for the probability of interest the *posterior* probability $\Pr(b \mid sh)$. By doing so, we assess the robustness of the *diagnosis* of a brain tumour for an arbitrary patient with a primary tumour who is suffering from severe headaches. We address the one-way analyses with respect to the parameters $\gamma(mc)$, $\gamma(b \mid \neg mc)$, and $\gamma(sh \mid \neg b)$, respectively. The results of these posterior analyses are shown in Figure 6.4. The figure shows, for example, the probability of interest $\Pr(b \mid sh)$ as a function of the parameter $x = p(b \mid \neg mc)$, that is,

$$\Pr(b \mid sh) = \frac{4 \cdot x + 0.20}{x + 3.80}$$

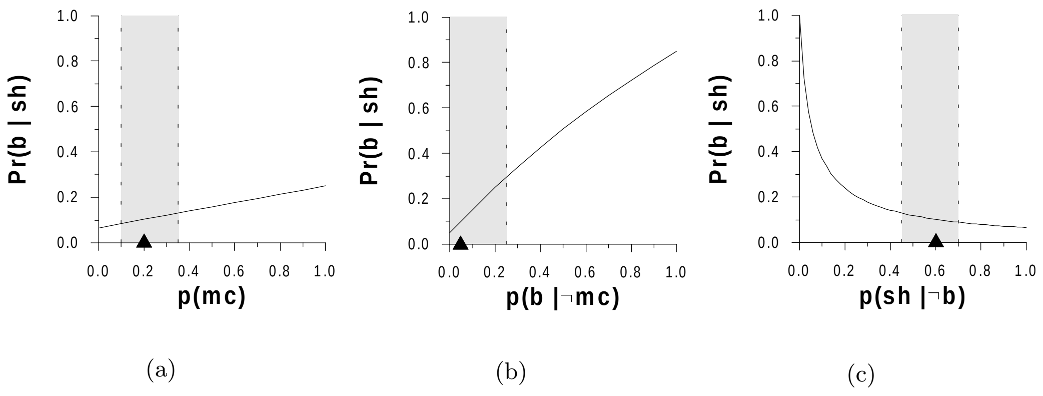


Figure 6.4: A one-way sensitivity analysis of the example probabilistic network.

Note that, in contrast with the prior analyses discussed before, the analyses for the posterior probability of interest reveal a non-linear relationship between the probability assessment that is being varied and the probability of interest. □

Selecting parameters deserving attention

Sensitivity analyses for a large number of parameters and several outputs of interest will typically result in a huge number of sensitivity functions that are to be examined. A number of criteria can be used to decide whether or not the effect of a parameter can be considerable enough to warrant further consideration, either for refining its assessment as discussed in Chapter 5, or for determining the implications of the analysis for the employability of the network in the domain of application. Here we will briefly discuss the following selection criteria: *absolute effect*, *plausible effect*, *sensitivity value*, *vertex proximity*, and *admissible deviation*.

Absolute effect The absolute effect of a parameter on an output probability is simply the absolute difference $|f(0) - f(1)|$.

Example 6.1.2 We continue our example analysis. Figure 6.3(a) shows that varying the assessment for the probability $\gamma(b \mid mc)$ from 0 to 1 has a negligible effect on the probability of interest $\Pr(c)$: the prior probability of a patient falling into a coma within the next three years increases from 0.31 to 0.34, approximately. Figure 6.3(b) shows that varying the initial assessment for the probability $\gamma(isc \mid mc)$ has a somewhat stronger effect on the probability of interest: $\Pr(c)$ now ranges from 0.22 to 0.34. From Figure 6.3(c), to conclude, it is seen that varying the assessment for the probability $\gamma(isc \mid \neg mc)$ has an even stronger effect on $\Pr(c)$: the prior probability of a coma ranges from 0.21 to 0.78. Note that the three analyses reveal the linear relationship between the probability assessment that is being varied and the probability of interest. □

Plausible effect So far we have treated the probability assessments of our probabilistic network as exact point probabilities. As for most applications, however, the initially obtained assessments are quite uncertain. If this uncertainty is captured by supplementing each probability assessment with a plausible interval that defines a range of values in which the 'true' probability lies with reasonable certainty, then we can select parameters of interest based upon their *plausible effect* on the probability of interest. The plausible effect is now defined as the absolute effect within the plausible interval; the plausible effect is bounded by the absolute effect.

Example 6.1.3 In the prior analyses, for the three probability assessments under study in our example network, the plausible intervals are indicated in Figure 6.3 by shading. The figure shows that plausible variation of the parameter $\gamma(isc \mid \neg mc)$ has the strongest effect on the probability of interest $\Pr(c)$. Varying the parameters $\gamma(b \mid mc)$ and $\gamma(isc \mid mc)$, respectively, within their plausible intervals results in a rather small effect on the probability of interest. We recall from Figure 6.3 that the effect on $\Pr(c)$ of varying the assessment for $\gamma(b \mid mc)$ from 0 to 1 is smaller than the effect of varying $\gamma(isc \mid mc)$ from 0 to 1. By taking the plausible intervals into consideration, however, variation of the assessment for $\gamma(b \mid mc)$ has the stronger plausible effect. Especially since the plausible interval for this assessment is quite large, for example further elicitation efforts may better be directed at the probability $\gamma(b \mid mc)$ than at the probability $\gamma(isc \mid mc)$. □

Sensitivity value The *sensitivity value* of a parameter x with respect to a probability of interest is defined as $|\frac{\partial f}{\partial x}(x_0)|$, the absolute value of the first derivative of the sensitivity function at the original value x_0 of the parameter. The sensitivity value thus captures the effect of infinitely small shifts in the parameter on the probability of interest.

Example 6.1.4 For the sensitivity function describing the posterior probability $\Pr(b \mid sh)$ as a function of the parameter $x = \gamma(b \mid \neg mc)$ (see Figure 6.4(b)), we find for example that

$$f'(x) = \frac{0.384}{(0.16 \cdot x + 0.608)^2};$$

the parameter has an original value of 0.05, so the sensitivity value for this parameter is $|f'(0.05)| = 1.01$. Alternatively, if we consider the effect of varying parameter $x = \gamma(sh \mid \neg b)$ on our posterior probability of interest (Figure 6.4(c)), then the sensitivity value equals

$$\left| \frac{-0.059}{(0.92 \cdot 0.60 + 0.064)^2}) \right| = 0.155.$$

□

Vertex proximity The problem of using the sensitivity value as a measure of robustness, is that it often gives insight only in the effect of very small parameter variations. The effects of larger parameter shifts are ofcourse captured by the absolute and plausible effects, but can also be studied by examining the vertex proximity.

The *vertex* of a hyperbola branch is the point $(x_v, f(x_v))$ where $|f'(x_v) = 1|$. The proximity of a parameter's original value x_0 to the x-value of the hyperbola's vertex is an indication of possible sensitivity of the output of interest to variation of the parameter. The vertex-proximity can be easily computed from the constants of the sensitivity function:

$$x_v = \begin{cases} s + \sqrt{|r|}, & \text{if } s < 0 \\ s - \sqrt{|r|}, & \text{if } s > 1 \end{cases}$$

Example 6.1.5 Again consider our example sensitivity functions in Figure 6.4. For the hyperbola branch describing the output probability as a function of parameter $\gamma(b \mid \neg mc)$ with original value 0.05, we have that $x_v = -3.8 + \sqrt{15} = 0.07$; the vertex of the function is therefore very close to the parameter's original value, indicating that non-infinitesimal variation of the parameter could possibly have large effects on the output probability of interest. For parameter $\gamma(sh \mid \neg b)$, on the other hand, the x-value of the vertex is found around 0.19, which can be considered quite distant from the parameter's original value of 0.60. If the original value of the latter parameter had been 0.20, however, then its sensitivity value (0.96) would perhaps deem the parameter irrelevant for further inspection, whereas the vertex-proximity would warn us for possibly significant effects of variation. □

Admissible deviation An *admissible deviation* for a variable of interest and a given parameter, is a pair of real numbers (α, β) that describe the shifts to smaller values and to larger values, respectively, that are allowed in the parameter without inducing a change in the most likely value of the variable of interest. For a parameter with an original value of x_0, the admissible deviation (α, β) thus indicates that the parameter can be safely varied within the interval $[x_0 - \alpha, x_0 + \beta]$. To express that the parameter can be varied as far as the bounds of the probability interval, the symbol ∞ is used.

For establishing an admissible deviation, the intersections of a sensitivity function relating one value of the output variable of interest to a parameter, and those pertaining to the other values of the output variable, are computed. More specifically, the admissible deviation is established by computing the x-coordinates of the points where the sensitivity function $f_{\Pr(a_i|e)}(x)$ for some variable of interest A, intersects with the sensitivity functions $f_{\Pr(a_j|e)}(x)$, $j \neq i$. We note that if the variable of interest has only two values, then the two sensitivity functions always intersect for $f(x) = 0.5$. For so-called *threshold decision making* (see Section 6.3.1), instead of considering a change in most likely value, we can determine when some outcome probability of interest becomes smaller or larger than some pre-specified threshold; this basically amounts to establishing the intersections of a sensitivity function with the constant functions associated with the values of the threshold probabilities.

Example 6.1.6 We illustrate investigating the robustness of decisions by computing admissible deviations for our example probabilistic network. First, consider an arbitrary patient. The effects of varying parameter $x = \gamma(b \mid \neg mc)$ with original value $x_0 = 0.05$ on the probabilities of disease $\Pr(B)$ are shown in Figure 6.5(a) and are described by

$$f_{\Pr(b)}(x) = 0.80 \cdot x + 0.04, \quad \text{and} \quad f_{\Pr(\neg b)}(x) = -0.80 \cdot x + 0.96.$$

The sensitivity functions intersect for $x = 0.575$, resulting in an admissible deviation of $(\infty, 0.525)$. The patient is most likely to not suffer from a brain tumour and this diagnosis is quite robust to variation of the parameter under study. Now consider a patient with severe headaches. The effects of varying parameter $x = \gamma(sh \mid \neg b)$ with

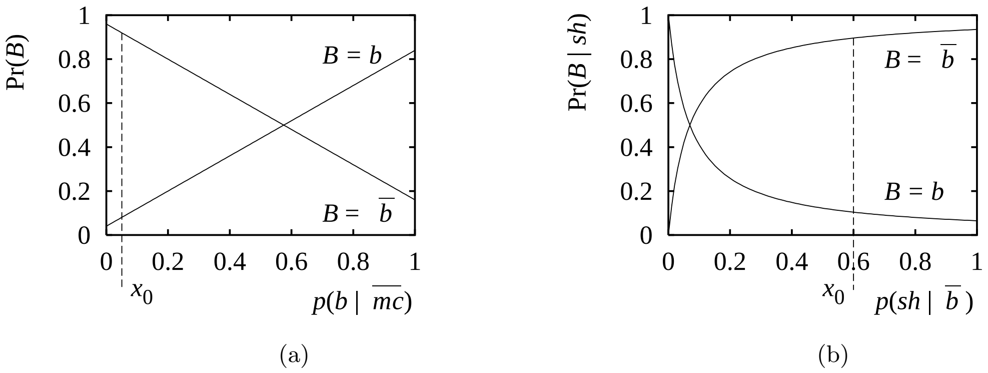


Figure 6.5: The sensitivity functions for the two possible values of the variable of interest B as a function of (**a**) parameter $\gamma(b \mid \neg mc)$, and (**b**) parameter $\gamma(sh \mid \neg b\,)$, given a severe headache.

original value $x_0 = 0.60$ on the posterior probabilities of disease $\Pr(B \mid sh)$ are shown in Figure 6.5(b) and are described by

$$f_{\Pr(b|sh)}(x) = \frac{0.064}{0.92 \cdot x + 0.064}, \quad \text{and} \quad f_{\Pr(\neg b\,|sh)}(x) = \frac{0.92 \cdot x}{0.92 \cdot x + 0.064}.$$

The sensitivity functions intersect for $x = 0.0696$, resulting in an admissible deviation of $(0.53, \infty)$. We conclude that the patient is most likely not to suffer from a brain tumour and this diagnosis again is quite robust to variation of the parameter under study. □

6.1.3 Two-way Sensitivity Analysis

We now address a *two-way sensitivity analysis* of a probabilistic network. In a two-way sensitivity analysis, two probability assessments are varied simultaneously to reveal their joint effect on a probability of interest. Recall that under the assumption of proportional co-variation, any sensitivity function is a quotient of two functions that are (multi-)linear in the parameters under study. In a two-way analysis, the function is bi-linear and takes the general form

$$f(x, y) = \frac{c_1 \cdot x \cdot y + c_2 \cdot x + c_3 \cdot y + c_4}{c_5 \cdot x \cdot y + c_6 \cdot x + c_7 \cdot y + c_8}$$

where the constants c_j, $j = 1, \ldots, 8$, are again built from the assessments for the numerical parameters that are not being varied.

As remarked before, two parameters can have a synergistic effect on the probability of interest; this means that the joint effect of varying the two parameters is different from the sum of their individual effects. Not every pair of varied parameters will have such an interaction effect on a probability of interest, however. For example, any two parameters that pertain to *incompatible* probabilities, in the sense of specifying complementary values for the same variable, will not interact. The function expressing the probability of interest in terms of two such assessments will lack a product term. A two-way sensitivity analysis involving assessments for incompatible probabilities does not reveal any unanticipated effects on a probability of interest beyond the effects shown by one-way sensitivity analyses for the two assessments separately. Any such pair of assessments can therefore be excluded from the analysis.

Two-way sensitivity analysis in probabilistic networks has received far less attention by researchers than one-way analysis. No details are as yet known about the possible shapes of the two-way functions. It is obvious though that if either one of the parameters is fixed to an arbitrary value, then the two-way function degenerates to a one-way function in the other parameter, which is either monotonically increasing or monotonically decreasing; as a result, the minimum and maximum function values are still found for the extreme parameter values 0 and 1. Research, in addition, has not addressed selection criteria. Here we will therefore limit our discussion to the selection criteria that trivially apply.

Selecting parameters deserving attention

For selecting parameters from a two-way analysis that deserve further attention, we can use the *absolute effect*, the *plausible effect*, and the *sensitivity value* as before. In addition, we can use *contour distance* as a selection criterion.

Absolute and plausible effect The absolute effect of parameters x and y on a probability of interest is captured by the largest absolute difference found between the function values of the sensitivity function in the 'corners' $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$ of the domain: $\max\{f(i,j) - f(k,l) \mid i,j,k,l \in \{0,1\}\}$. The plausible effect is defined analogously where the 'corners' of the domain are given by the plausible intervals for the two parameters.

Sensitivity value The sensitivity value of two parameters x and y with respect to a probability of interest can be defined as $|\frac{\partial^2 f}{\partial x \partial y}(x_0, y_0)|$, the absolute value of the mixed derivative of the sensitivity function at the original values x_0 and y_0 of the parameters.

Contour distance Two-way sensitivity functions are most easily interpreted using *contour plots*, in which contour lines connect the combinations of values for the two parameters that result in the same value for the probability of interest. The distance between two contour lines indicates the variation necessary in the two assessments to shift the probability of interest from one contour line to another. If the contour lines are very close to one another, then a small variation in the parameters under study suffices to have a strong effect on the probability of interest; if, in contrast, the contour lines are further apart, then the probability of interest is not very sensitive to variation of the two assessments.

Example 6.1.7 We illustrate performing a two-way sensitivity analysis for our example network. For our probability of interest, we once again take the prior probability $\Pr(c)$. We first address an analysis with respect to the parameters $x = \gamma(b \mid mc)$ and $y = \gamma(isc \mid mc)$. The corresponding sensitivity function equals

$$\Pr(c) = -0.15 \cdot x \cdot y + 0.15 \cdot x + 0.15 \cdot y + 0.194$$

From this function it is readily seen that the two probability assessments under study upon variation have a negative interaction effect on the probability of interest. The function is depicted in Figure 6.6. We observe from the figure that the distances between the contour lines differ, indicating that varying the parameters $\gamma(b \mid mc)$ and $\gamma(isc \mid mc)$ simultaneously has a joint effect on the probability of interest $\Pr(c)$ beyond

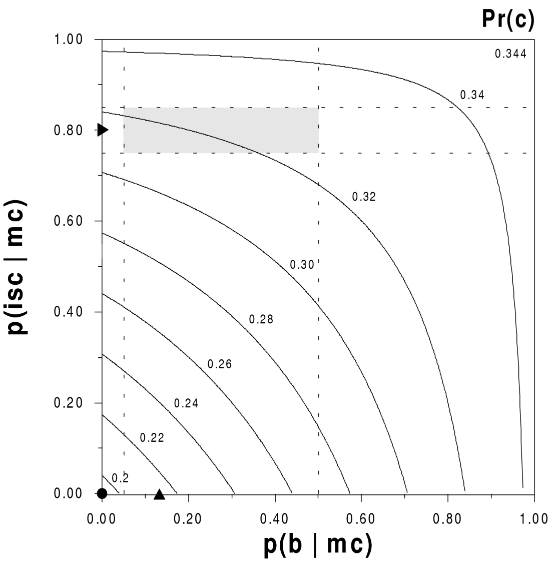


Figure 6.6: A two-way sensitivity analysis of the example probabilistic network.

the effects of their separate variation; this joint effect is due to the synergistic influence of the variables B and ISC on the variable C outlined before. We further observe that the contour lines are closer to one another in the lower left part of the figure than in the upper right part. If the assessments for the parameters $\gamma(b \mid mc)$ and $\gamma(isc \mid mc)$ are both quite small, therefore, their variation will have a stronger effect on the probability of interest than if the initial assessments have a higher value. To variation within the plausible intervals of the assessments $\gamma(b \mid mc) = 0.2$ and $\gamma(isc \mid mc) = 0.8$, as indicated by shading in Figure 6.6, the probability of interest shows a relatively low sensitivity. We further observe that the absolute effect of their joint variation on $\Pr(c)$ is not too strong.

Now we address an analysis pertaining to the assessments for the parameters $\gamma(b \mid mc)$ and $\gamma(isc \mid \neg mc)$; note that the two assessments under study pertain to incompatible probabilities, since they are conditioned on two different values of the same variable. The corresponding sensitivity function equals

$$\Pr(c) = 0.03 \cdot x + 0.57 \cdot y + 0.204$$

This function is depicted in Figure 6.7. We observe from the figure that the contour lines, once again indicating values for the probability of interest $\Pr(c)$, are equidistant. Equidistance of contour lines indicates that simultaneously varying the probability assessments under study has no joint effect on the probability of interest beyond the effects of their separate variation. The two-way analysis therefore does not provide any information in addition to the information yielded by one-way analyses for the separate assessments.

To conclude, we performed a two-way sensitivity analysis of our example network with respect to the posterior probability of interest $\Pr(b \mid sh)$. We first address the analysis of varying the assessments for the parameters $x = \gamma(b \mid \neg mc)$ and $y = \gamma(sh \mid b)$ simultaneously. The corresponding two-way sensitivity function equals

$$\Pr(b \mid sh) = \frac{1.10005 \cdot x \cdot y - 0.00056 \cdot x + 0.0559 \cdot y - 0.00034}{x \cdot y - 0.6268 \cdot x + 0.0835 \cdot y + 0.7811}$$

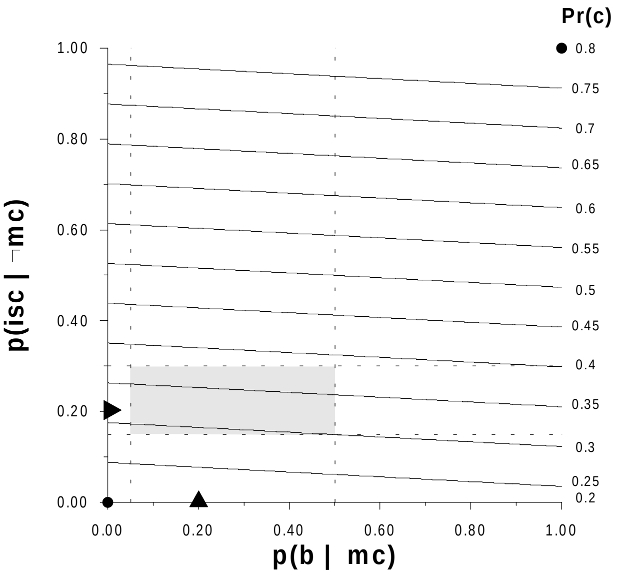


Figure 6.7: A two-way sensitivity analysis of the example probabilistic network.

In this function, the terms $-0.00056 \cdot x$ and $-0.6268 \cdot x$ pertain to the effect of variation of just the probability assessment $p(b \mid mc)$; the terms $0.0559 \cdot y$ and $0.0835 \cdot y$ pertain to the assessment $p(isc \mid mc)$. The terms $1.10005 \cdot x \cdot y$ and $x \cdot y$ with each other capture the interaction effect of the two assessments on the network's probability of interest. These terms provide information that cannot be revealed by one-way analyses with respect to the two assessments separately. The function is depicted in Figure 6.8. Note

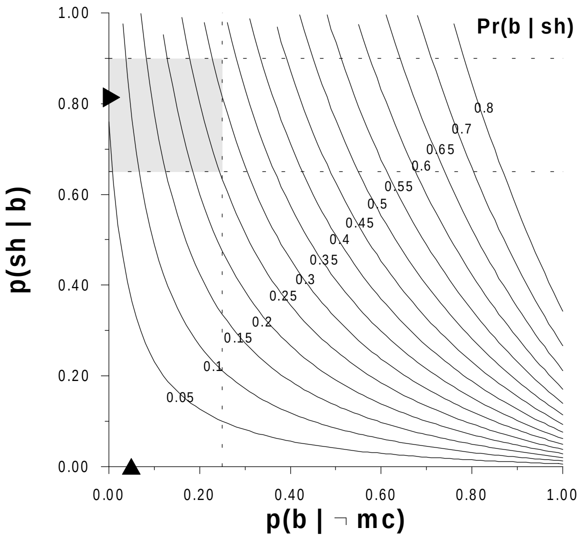


Figure 6.8: A two-say sensitivity analysis of the example probabilistic network.

that the contour lines are closest to one another in the lower right part of the figure, indicating a high sensitivity of the posterior probability of interest to high values for the probability $\gamma(b \mid \neg mc)$ and low values for the probability $\gamma(sh \mid b)$. For variation, within the plausible intervals of the initial assessments $\gamma(b \mid \neg mc) = 0.05$ and $\gamma(sh \mid b) = 0.8$, as indicated by shading in Figure 6.8, however, the probability of interest is relatively stable. $\square$

6.2 Evaluating Probabilistic Networks

To establish its practical value, a real-life probabilistic network is typically subjected to an evaluation study using data from the domain of application. Such a study amounts to entering the data available for each problem case into the network and computing the most likely outcome. This outcome is then compared against a given standard of validity. The results of the study are often summarised in the percentage of correctly computed outcomes. This *percentage correct* is generally taken to convey the practical value of the network. For a medical diagnostic application, for example, a percentage correct of 85% is taken to indicate that the network is likely to establish the correct diagnosis for 85 out of every 100 patients. For many applications, this percentage would convey the information that the network performs quite satisfactorily.

Unfortunately, interpretation of the percentage correct of a probabilistic network is not as straightforward as is often suggested. The percentage should be interpreted with respect to a specific data collection. Now, each data collection is likely to include errors and to reflect the biases exhibited by the experts who collected the data. Moreover, the data will include the effects of random variation, especially in domains of a scientific nature. In the medical domain, for example, there is random variation in patient data, arising from biological differences between patients in the progression of pathological processes and from differences in the physicians' interpretation of symptoms and signs [Fletcher *et al.*, 1996]. When two outcomes are almost equally likely for a patient, chance determines, to at least some extent, which outcome is entered into the patient's medical record as the most likely one. Random variation may thus affect a network's percentage correct, but the extent to which it does so is not expressed by the percentage.

Probabilistic networks in essence do not yield a deterministic outcome. Rather, they produce a posterior probability distribution for their outcome variable. This distribution reflects the network's doubt as to the most likely outcome. Since the percentage correct of a probabilistic network does not take the computed posterior distribution into consideration, it does not reveal the extent of uncertainty in the outcome. To incorporate the network's doubt in the assessment of its practical value, evaluation *scores* from the field of statistical forecasting can be used. The use and interpretation of such a score is illustrated by means of an evaluation study of a real-life probabilistic network for the staging of oesophageal cancer.

6.2.1 The Percentage Correct and its Shortcomings

With the help of two experts in gastrointestinal oncology from the Netherlands Cancer Institute, Antoni van Leeuwenhoekhuis, a probabilistic network for the staging of oesophageal cancer has been constructed. The network captures the state-of-the-art knowledge about oesophageal cancer in the *oesophagus network*; for details on the network and its construction, see [Van der Gaag *et al.*, 2002]. For studying the ability of the oesophagus network to correctly predict the stage of a patient's cancer, the medical records of 156 patients diagnosed with oesophageal cancer are available. For each patient, between 6 and 21 of the 25 symptoms and test results modelled in the network are available. For each patient, also the *stage* of his or her cancer, as established by the attending physician, is recorded. This stage can be either I, IIA, IIB, III, IVA, or IVB, in the order of advanced disease. The tumour's stage is indicative of the effects and complications to be expected from the different available therapeutic alternatives.

In general, to establish the practical value of a probabilistic network, for each case the outcome with highest posterior probability is determined from the network and compared against a given standard of validity. The results of such an evaluation study are summarised in the percentage of cases for which the network yields the correct outcome as the most likely one. To establish the practical value of the oesophagus network, for each patient all diagnostic symptoms and test results available were entered and the most likely stage for the patient's cancer was computed from the probabilistic network. The computed stage was then compared with the stage recorded in the data. The results from this comparison are shown in the matrix of Figure 6.9. The numbers on the diagonal of the matrix are the numbers of patients per stage for whom the network yields the same stage as the one recorded in the data. Taking the stages from the medical records as a standard of validity, we find that the network establishes the correct stage for 133 of the 156 patients, that is, the network has a percentage correct of 85%.

As probabilistic networks in general, the oesophagus network in essence does not produce a deterministic outcome. Rather, it yields a probability distribution for its outcome variable. More specifically, it yields, for each patient, a posterior probability distribution over the six possible stages of his or her cancer. As an example, Figure 6.10 shows the distributions that are computed for three different real patients. Now, for some patients the computed posterior distribution clearly points to a single most likely stage. For other patients, however, the posterior distribution can reveal considerable uncertainty. It is not unlikely, therefore, that incorrect conclusions of a network can be attributed to the effect of random variation, rather than to, for example, a modelling error. In the percentage correct reported for a network, however, the distribution of uncertainty over the various different outcomes is not taken into account. For example, for the patients shown in Figure 6.10, the oesophagus network's outcome is classified simply as correct for the first two patients and as incorrect for patient 3.

		network						
		I	IIA	IIB	III	IVA	IVB	*total*
	I	**2**	0	0	0	0	0	2
	IIA	0	**37**	0	1	0	0	38
data	IIB	0	1	**0**	3	0	0	4
	III	1	10	0	**36**	0	0	47
	IVA	0	0	0	4	**35**	0	39
	IVB	0	0	0	3	0	**23**	26
	total	3	48	0	47	35	23	156

Figure 6.9: The results from the evaluation study, expressed in terms of the numbers of correctly and incorrectly staged patients.

patient 1, stage IVA	
stage I	0
stage IIA	0
stage IIB	0.0159
stage III	0.0882
stage IVA	0.8245
stage IVB	0.0714

(a)

patient 2, stage III	
stage I	0
stage IIA	0
stage IIB	0.0002
stage III	0.3616
stage IVA	0.3498
stage IVB	0.2884

(b)

patient 3, stage III	
stage I	0.0222
stage IIA	0.3753
stage IIB	0.0459
stage III	0.3714
stage IVA	0.0916
stage IVB	0.0936

(c)

Figure 6.10: The posterior probabilities of the six stages for three different patients.

6.2.2 The evaluation score

As mentioned previously, probabilistic networks typically yield a probability distribution for their outcome variable. While for some cases from the domain of application the computed posterior distribution will point to a single most likely outcome, it may reveal considerable uncertainty for other cases. The percentage correct as a summary of evaluation results does not take these uncertainties into account. For assessing the practical value of a network, however, not just the most likely outcome but also the posterior distribution over the various possible outcomes should be taken into consideration.

Probabilistic networks basically are probabilistic *forecasters*, as they repeatedly present predictions for an outcome variable in terms of probabilities. For the oesophagus network, for example, the posterior probability distribution that is computed for a specific patient can be looked upon as a forecast for the stage of this patient's cancer. Establishing the practical value of a probabilistic network thus amounts to assessing its quality as a forecaster. The quality of a probabilistic forecaster is often expressed in terms of its *calibration*, that is, the degree to which its forecasts match the true distribution of outcomes. For the oesophagus network, more specifically, we can say that it is (empirically) *well calibrated* if, among the patients for whom the network predicts a specific stage S with probability x_S, the proportion of patients who in fact have stage S, denoted x'_S, equals x_S. The smaller the difference between x_S and x'_S, that is, the closer the network's distribution matches the true distribution, the better calibrated the network is [Dawid, 1985, DeGroot & Fienberg, 1983]. Building upon this concept of calibration, various scores for expressing the quality of a forecaster have been developed in the field of statistics.

Among the best-known evaluation scores is the *Brier score* [Panofsky & Brier, 1968]. The basic idea of this score is illustrated for our oesophagus network. For each patient i, the network yields a forecast of posterior probabilities p_{ij} over the stages $j = \mathrm{I}, \ldots,$ IVB. The Brier score B_i of this forecast is defined as

$$B_i = \sum_{j=\mathrm{I},\ldots,\mathrm{IVB}} (p_{ij} - s_{ij})^2$$

where $s_{ij} = 1$ if the medical record of patient i states stage j, and $s_{ij} = 0$ otherwise. If the network would yield the correct stage with certainty, that is, if the network would yield a correct deterministic forecast for the patient, then the associated Brier score would be equal to 0. If the network would yield an incorrect deterministic forecast, the score would be 2. For the forecast for a single patient, therefore, the Brier score ranges between 0 and 2, and the better the forecast, the lower the score. The Brier scores for the network's forecasts for the three patients from Figure 6.10 now are:

$$B_1 = 0.04 \qquad B_2 = 0.61 \qquad B_3 = 0.56$$

These scores reveal that the quality of the forecast for the first patient is very good, as expected. The other scores show that the forecasts for the patients 2 and 3 are of less quality. Recall that for patient 3 the forecast is unequivocal as a result of two stages being almost equally likely, among which is the correct stage. For patient 2, there is even more uncertainty in the forecast, as there are three almost equally likely stages. These observations are reflected in the associated Brier scores: the score B_3 for patient 3 indicates higher quality than the score B_2 for the second patient. While in terms of the numbers of correctly and incorrectly staged patients the forecast for patient 2 is correct and the forecast for patient 3 is incorrect, the Brier score results in a more balanced and, hence, a more insightful quality assessment. Figure 6.11 summarises the Brier scores averaged over all, correctly and incorrectly staged, patients.

The quality of the oesophagus network as a forecaster can now be expressed in an overall score that is computed from the scores of the separate forecasts for our collection of patients. For n patients, the overall Brier score B is defined as

$$B = \frac{1}{n} \sum_{i=1,\ldots,n} B_i$$

It is readily seen that the overall Brier score again ranges between 0 and 2, and the better the forecaster, the lower the score. For the oesophagus network, an overall Brier score of 0.29 is found. To interpret this number, we compare the score with the overall scores obtained for two *uninformed* forecasters. The first forecaster gives a uniform probability distribution for each patient; this forecaster has an overall Brier score of 0.83. The second forecaster gives for each patient the prior distribution over the stages recorded in the data; this forecaster has an overall Brier score of 0.76 and is therefore slightly more informed than the uniform forecaster. The much lower Brier score of the oesophagus network now conveys the information that the network is quite informed and indeed builds upon its knowledge of oesophageal cancer to arrive at relatively good forecasts.

		network					
		I	IIA	IIB	III	IVA	IVB
	I	**0.21**	–	–	–	–	–
	IIA	–	**0.28**	–	1.52	–	–
data	IIB	–	1.17	–	0.98	–	–
	III	1.40	0.89	–	**0.26**	–	–
	IVA	–	–	–	0.75	**0.08**	–
	IVB	–	–	–	0.87	–	**0.06**

Figure 6.11: The results from the evaluation study, expressed in terms of average Brier scores.

The percentage correct and the Brier score provide two ways of evaluating the practical value of a probabilistic network. The percentage correct treats outcomes as deterministic, whereas the Brier score takes the actual uncertainties into account.

6.3 A Problem-Solving Architecture

In this section, we describe a problem-solving architecture that allows for automated control over reasoning. The main purpose of exerting control over reasoning is to shape efficient and intelligent problem-solving behaviour. Exerting control involves monitoring and reflecting upon the reasoning process as it develops and taking decisions as to how it should proceed. To this end, strategic knowledge about the domain at hand is employed. As this knowledge may be non-probabilistic in nature, control over probabilistic network reasoning generally cannot be implemented in the framework in itself. The probabilistic network framework can therefore be embedded in a general *problem-solving architecture* [Van der Gaag & Wessels, 1994a].

The probabilistic network problem-solving architecture is composed of two *layers*. The first layer offers the probabilistic network formalism and a variety of associated algorithms. The algorithms in this layer are characterised by their operating on a probabilistic network directly: various algorithms for probabilistic inference are comprised in the layer as are algorithms for example for reading independences from the qualitative part of a network. This layer is called the *probabilistic layer* of the architecture. The second layer of the problem-solving architecture is designed to provide for control over reasoning — it is termed the *control layer*. This layer offers a variety of methods for control for different types of problem solving and provides formalisms for representing the additional knowledge used by these methods. The layer for example can offer methods for selectively gathering evidence for diagnostic applications as well as methods for intelligently pruning and focusing probabilistic network inference. These control methods are the basic building blocks for shaping complex, domain-dependent problem-solving behaviour. The two layers of the architecture are strictly separated and communicate in a highly restricted fashion. The control layer queries the probabilistic layer for information about the represented joint probability distribution and the evidence entered so far, and, based upon this information, takes strategic decisions as to how to proceed. The probabilistic layer computes and returns the information it is asked for by the control layer.

The problem-solving architecture explicitly separates probabilistic reasoning from control over reasoning. Several advantages arise from such an explicit separation. A probabilistic network can be developed and refined, without being hampered by any algorithmic issues. Moreover, the representation of the joint probability distribution on the domain at hand is not obscured by non-probabilistic knowledge. In addition, a probabilistic network can be re-used in different contexts for different purposes; a similar observation holds for the methods of control comprised in the control layer of the architecture. The architecture in addition provides for modelling decision problems. Knowledge about viable decisions and the preferences over their consequences involved in a decision problem are represented in the control layer, along with methods for solving the problem; the probabilistic layer comprises a probabilistic network that is used as a background knowledge base for providing the probabilities required. The idea of a meta-level problem-solving architecture pervades many areas of artificial intelligence research.

6.3.1 Threshold Decision Making

As an example of the type of problems that can be modelled using the problem-solving architecture, the threshold model for decision making is considered.

In the medical domain, probabilistic networks are often used for diagnostic purposes. A diagnostic probabilistic network typically comprises one or more variables modelling the presence of absence of disease, various variables modelling findings and results from diagnostic tests, and a number of intermediate variables modelling unobservable pathophysiological states. In the example network from Section 6.1 (Figure 6.1), for instance, the variable B models the disease of interest, being the presence or absence of a brain tumour; the variable MC models an unobservable state and the remaining variables capture findings and test results. A medical diagnostic probabilistic network is used for computing a most likely diagnosis for a patient given his or her presentation findings and test results.

The most likely diagnosis for a patient, along with its uncertainty, is generally taken by an attending physician to decide upon management of the patient. The physician may decide, for example, to start treatment right away. For the brain tumour example, the physician may decide to perform neurosurgery if a brain tumour is indicated. Alternatively, the physician may defer the decision whether or not to treat the patient until additional diagnostic information has become available, for example from a CT scan. Or, the physician may decide to withhold treatment altogether. To support choosing among these decision alternatives, the threshold model for patient management can be used. The threshold model for patient management is a typical example of something that can be implemented in the control layer of the problem-solving architecture.

The *threshold model for patient management*, or for decision making more in general, builds upon various threshold probabilities of disease [Pauker & Kassirer, 1980]. The *treatment threshold probability* of disease, written $P^*(d)$ for disease d, is the probability at which an attending physician is indifferent between giving treatment and withholding treatment. If, for a specific patient, the probability of disease $\Pr(d)$ exceeds the treatment threshold probability, that is, $\Pr(d) > P^*(d)$, then the physician will decide to treat the patient as if the disease were known to be present with certainty. Alternatively, if $\Pr(d) \leq P^*(d)$, the physician will basically withhold treatment from the patient.

As a consequence of the uncertainty concerning the presence of disease in a patient, additional information from a diagnostic test may affect an attending physician's basic management decision. If the probability of disease exceeds the treatment threshold probability, then interpreting a negative test result may result in an updated probability of disease *below* the threshold probability. Alternatively, if the pretest probability of disease falls below the treatment threshold probability, a positive test result may raise the probability of disease to a value *above* the threshold probability. To reckon with such effects, the threshold model for patient management includes another two threshold probabilities. The *no treatment-test threshold probability* of disease, written $P^-(d)$, is the probability at which the attending physician is indifferent between the decision to withhold treatment and the decision to obtain additional diagnostic information. The *test-treatment threshold probability* of disease, written $P^+(d)$, is the probability at which the physician is indifferent between obtaining additional information and starting treatment right away.

Figure 6.12 summarises the basic idea of the threshold model for patient manage-

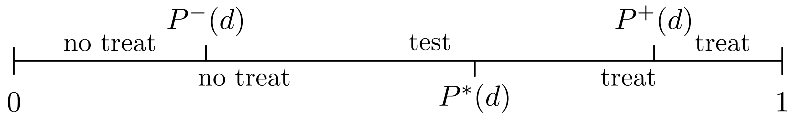


Figure 6.12: The threshold model for patient management, indicating three threshold probabilities and the various decision alternatives at a physician's disposal.

ment. As long as the diagnostic test under consideration has not been performed, a physician has three decision alternatives at his or her disposal. If the probability of disease $\Pr(d)$ for a patient falls below the no treatment-test threshold probability, that is, if $\Pr(d) < P^-(d)$, then the physician will withhold treatment from the patient without gathering additional diagnostic information. If the probability of disease exceeds the test-treatment threshold probability, that is, $\Pr(d) > P^+(d)$, then the physician will start treatment right away. Otherwise, that is, if $P^-(d) \leq \Pr(d) \leq P^+(d)$, the physician will perform the diagnostic test. After testing, there are only two decision alternatives left. If the updated probability of disease for the patient exceeds the treatment threshold probability, the physician will start treatment; otherwise treatment will be withheld from the patient.

The treatment threshold probability of disease $P^*(d)$ used in the threshold model is typically established by a physician after carefully weighing the various *utilities* involved. Utilities are numbers associated with a *utility function* and pertain to the presence or absence of disease on the one hand and giving or withholding treatment on the other hand. A utility function may be based on probabilistic information only and not involve any other information about the domain at hand. Yet, it may also incorporate non-probabilistic issues such as the cost of obtaining. From the *expected utilities* for giving and withholding treatment in view of the uncertainty concerning the presence of disease, the probability of disease at which the physician is indifferent between the two decision alternatives is readily determined; the basic idea is illustrated in Figure 6.13(a). For the brain tumour example, the physician will typically take into consideration the life expectancy for a patient, with and without a brain tumour, and the patient's attitude towards impaired health states; the physician can, for example, set the treatment threshold probability of a brain tumour at 0.15. The two threshold probabilities $P^-(d)$ and $P^+(d)$ for deciding whether or not to perform a diagnostic test are established from the test's characteristics. For the brain tumour example, a possible diagnostic test is the CT scan. Suppose this test is added to the example network as a direct descendant of the variable B, together with the conditional probabilities

$$\gamma_{CT}(ct \mid b) = 0.95, \; \gamma_{CT}(ct \mid \neg b) = 0.10$$

The physician will typically weigh the discomfort of a CT scan for a patient against the additional information yielded by the scan; the physician, for example, may set the no treatment-test threshold probability of a brain tumour at 0.045 and the test-treatment threshold probability at 0.56. The basic idea of establishing these two threshold probabilities is illustrated in Figure 6.13(b).

While the probabilistic network framework offers algorithms for computing the probability of disease and for processing evidence, it does not provide for computing the threshold probabilities and for comparing these to computed probabilities of disease: these tasks involve knowledge that cannot be expressed in the probabilistic network formalism and require computations beyond probabilistic inference. These tasks therefore

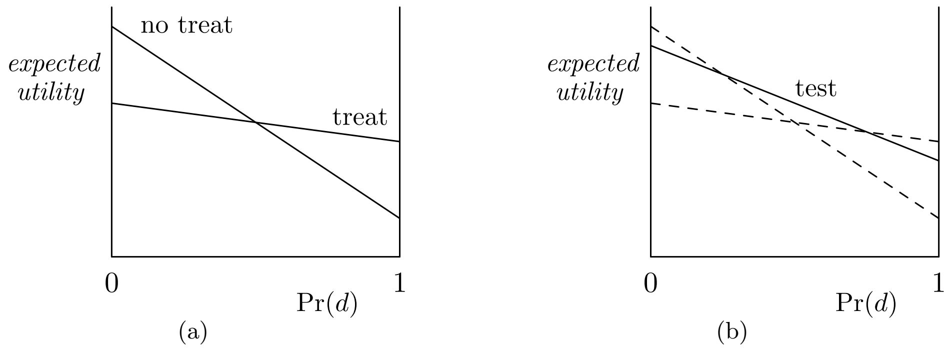


Figure 6.13: The basic idea of establishing (a) the treatment threshold probability of disease, and (b) the no-treatment-test and test-treatment threshold probabilities.

are provided for by the control layer of the problem-solving architecture. The control layer in the problem-solving architecture can thus provide the additional information required to support the decision between treatment, no treatment, or deferring treatment until additional diagnostic information is available. If it is decided that additional diagnostic information is to be obtained, the control layer can also be used for determining what additional information is to be obtained. A simple method for selective gathering of such information is discussed next.

6.3.2 Selective Evidence Gathering

Another example of control over reasoning offered by the control layer of the problem-solving architecture, is a simple method for selective gathering of evidence for diagnostic problem solving with a probabilistic network.

In diagnostic problem solving, the objective is to identify a most likely explanation for a problem under consideration — this explanation then is the diagnosis of the problem. Establishing a diagnosis is achieved by gathering information about the manifestations of the problem at hand by applying tests to the problem. In most domains, it is not necessary to collect evidence on all possible manifestations before an accurate diagnosis is reached: information from only a few tests generally suffices. Moreover, it often is not desirable to apply all tests available as testing may be costly or damaging. In diagnostic problem solving, therefore, tests are not applied as a matter of course but instead are selected carefully. *Selective evidence gathering*, or *test planning*, now amounts to selecting the most useful tests to apply to a problem under consideration.

In essence, selective evidence gathering is concerned with three tasks. The first of these is to select the test that is expected to yield the most useful information in the context of the evidence that is already available. When a test has been selected, the user, for example a physician, is requested to apply the test and to enter the evidence yielded. The second task of selective evidence gathering is to process this evidence. The third task is to decide whether enough evidence has been obtained as yet to confirm a diagnosis to sufficient extent. If still further information is required, the three tasks are executed recursively. We now take a closer look at these tasks in view of diagnostic problem solving with a probabilistic network. While the probabilistic network framework offers algorithms for computing probabilities and for processing evidence, thus providing for the second task of selective evidence gathering, it does not provide for valuing and selecting tests nor for deciding when to stop gathering information: these tasks involve knowledge that cannot be expressed in the probabilistic network formalism and require computations beyond probabilistic inference. These tasks therefore are provided for by the control layer of our problem-solving architecture.

In diagnostic problem solving, the variables discerned in the domain at hand play different roles; for example, some variables represent test outcomes, others represent unobservable, intermediate process states. For distinguishing between these different roles, the following types of vertex in the digraph of a probabilistic network are discerned [Henrion, 1989]: a *hypothesis vertex* represents one or more (mutually exclusive) hypotheses or disorders; an *evidence vertex* represents a variable whose value can be obtained by testing; all other vertices are *intermediate vertices*. The set of all vertices of the digraph G of a probabilistic network can thus be partitioned into three mutually disjoint sets of vertices: $H(G) = \{H_1, \ldots, H_n\}$, $n \geq 1$, is the set of hypothesis vertices; the set of evidence vertices is denoted as $E(G) = \{E_1, \ldots, E_m\}$, $m \geq 1$; $I(G)$ is the set of intermediate vertices. The roles of the various vertices are modelled in the control layer of the problem-solving architecture and are not known to the probabilistic layer. In addition to knowledge concerning the roles of the vertices discerned, the control layer also specifies the additional knowledge required for assessing for each test the (expected) usefulness of information yielded by testing, and the extra knowledge involved in deciding when to stop gathering information.

For selective evidence gathering in diagnostic problem solving with a probabilistic network, generally two simplifying assumptions are made. First, a *myopic* approach to evidence gathering is taken, that is, evidence vertices to acquire information on are selected one by one. It is conceivable that in practical applications a non-myopic approach in which vertices are selected groupwise outperforms any method based on a myopic approach. Naively adopting a non-myopic approach, however, poses unsurmountable problems concerning computational complexity. The second simplifying assumption generally made is that the probabilistic network at hand comprises *one* hypothesis vertex only, that is, it is assumed that all hypotheses discerned in the domain are mutually exclusive. Note that this assumption prohibits reasoning about multiple interacting disorders. Relaxing this assumption and straightforwardly applying selective evidence gathering in view of a set of hypothesis vertices also causes serious computational problems, since then all possible combinations of values for all hypothesis vertices have to be considered. In the remainder of this section, we will equally take up the two assumptions mentioned above, although research results have indicated that the simplifying assumptions may be eased to some extent [Heckerman *et al.*, 1993, Van der Gaag & Wessels, 1994b, Sent, 2005].

Selective evidence gathering for diagnostic problem solving with a probabilistic network may be envisioned as outlined below in pseudo-code. The evidence-gathering procedure takes the digraph G of a probabilistic network and the set E of all yet uninstantiated evidence vertices for its input and yields a diagnosis D for its output.

```
procedure evidence-gathering($G$,$E$,$D$)
  enough := $false$;
  while $E \neq \varnothing$ and not enough do
        dependent-vertices($H$,$E$,$E'$);
        if $E' \neq \varnothing$ then
              select-vertex($E'$,$E_j$);
              enter-evidence($E_j$);
```

```
            $E := E \setminus \{E_j\}$;
            enough := verify-enough()
        else enough := $true$
  od;
  diagnosis(D)
end
```

In principle, for selecting from a probabilistic network an appropriate vertex to acquire information on, each yet uninstantiated evidence vertex has to be examined as to the (expected) usefulness of information yielded. To this end, several probabilities are computed from the probabilistic network. These probabilities may reveal that for some of the uninstantiated evidence vertices, entering information has no influence whatsoever on the probabilities of the values of the hypothesis vertex and therefore is utterly useless in view of establishing a diagnosis. This property holds for all vertices that are independent of the hypothesis vertex given the evidence obtained so far. Now recall that the probabilistic network formalism allows for identifying independences from the digraph of a network without having to resort to probabilistic computations. In the main evidence-gathering procedure, this property is exploited to save on the number of probabilities that has to be computed from the network. The dependent-vertices procedure is called upon to determine from a set E of uninstantiated evidence vertices the subset E' of those vertices that are not d-separated from the hypothesis vertex H given the evidence entered so far. For selecting an appropriate evidence vertex to acquire information on now only the vertices comprised in this set E' are examined. Here, we will not further elaborate on the dependent-vertices procedure; for the computational issues involved, the reader is referred to [Geiger *et al.*, 1990]. We would like to note that although the dependent-vertices procedure is called from the control layer of the problem-solving architecture, it itself is comprised in the probabilistic layer.

Once the set E' of relevant uninstantiated evidence vertices has been determined, the select-vertex procedure selects from this set the evidence vertex to best acquire information on. Recently, the interest for selective evidence gathering — or *test selection* as it is most often referred to nowadays — has revived, resulting in a number of new measures and corresponding algorithms to select those vertices for which to acquire evidence (see e.g. [Sent, 2005]). Here, we will focus only on one of the earlier approaches where a utility function is used for discriminating between the various evidence vertices. This utility function assigns to each value of every evidence vertex a numerical quantity expressing the desirability, or utility, of obtaining this value. The utility functions in use for selective evidence gathering differ considerably in the way they value information [Ben-Bassat, 1978, Glasziou & Hilden, 1989]. For selecting an appropriate evidence vertex, the select-vertex procedure may employ any such utility function. As an example, we consider here a very simple utility function tailored to binary variables [Van der Gaag & Wessels, 1994a]: the *linear-value utility function* u is defined by

$$u(E_i) = |\Pr(h \mid c) - \Pr(h \mid c \wedge E_i)|$$

for each evidence vertex $E_i \in E'$, where H is the hypothesis vertex and c denotes the conjunction of all evidence obtained so far. Note that for an uninstantiated evidence vertex E_i, the difference between $\Pr(h \mid c)$ and $\Pr(h \mid c \wedge e_i)$ indicates the confidence gained in the hypothesis h if the evidence $E_i = true$ is observed; an analogous observation holds for the evidence $E_i = false$. The usefulness of acquiring information on an evidence vertex, however, does not depend on a single value as it is uncertain which test result will be yielded for the problem at hand. For examining an evidence vertex, therefore, the utilities of its separate values are weighted with the probabilities that these values will be found. The result models the *expected utility* of acquiring information on the vertex. When employing the linear-value utility function u, the expected utility $\hat{u}$ for an evidence vertex $E_i \in E'$ is computed from

$$\hat{u}(E_i) = \sum_{c_{E_i}} \Pr(c_{E_i} \mid c) \cdot u(c_{E_i})$$

To select the evidence vertex to best acquire information on, the select-vertex procedure computes the expected utilities for all relevant uninstantiated evidence vertices and then selects the vertex with highest expected utility.

Once an appropriate evidence vertex has been selected, the enter-evidence procedure prompts the user for a value for this vertex. The value obtained from the user is entered and subsequently processed in the probabilistic network at hand by the basic algorithms for probabilistic inference offered by the probabilistic layer of the problem-solving architecture. Note that the while-loop of the evidence-gathering procedure yields a sequence of prompts to the user concerning various evidence vertices.

The last task of selective evidence gathering is to investigate whether enough evidence has been collected to justify a decision to stop further gathering of information. For this task, a *stopping criterion* is employed. Such a stopping criterion may be based on several different principles. The principle of *sufficiency of confirmation* is to stop evidence gathering as soon as a diagnosis has been confirmed to sufficient extent by the available information: the probability of a (tentative) diagnosis is compared with a pre-set threshold value, and if this probability has surpassed the threshold value and is expected not to drop considerably, evidence gathering is stopped. Note that such a stopping criterion is based on probabilistic information only. Another principle a stopping criterion may be based upon is the principle of *sufficiency of information*. This principle is to stop evidence gathering if the expected utilities of all remaining evidence vertices have dropped below a pre-set threshold value; pursuing evidence gathering then is expected not to further contribute to establishing a diagnosis. A stopping criterion based on this principle may involve both probabilistic and non-probabilistic information from the domain at hand. In the evidence-gathering procedure the two principles are combined. The principle of sufficiency of information is seen in the condition of the main while-loop of the procedure: evidence gathering is stopped if all remaining uninstantiated evidence vertices are independent of the hypothesis vertex given the evidence obtained so far. The verify-enough function completes the stopping criterion by implementing a test on sufficiency of confirmation.

Exercises

* Exercise 6.1

Consider the digraph of a probabilistic network with eight vertices E_1, E_2, E_3, E_4, E_5, H_1, H_2, *and* H_3, *and nine arcs* (E_1, E_2), (E_1, H_1), (E_1, H_2), (E_3, E_5), (H_1, E_2), (H_1, E_3), (H_2, E_3), (H_2, E_4), *and* (H_3, E_4).

a. *Establish the sensitivity set for the variable of interest* H_1*; also try to explain in* *an informal, intuitive way why certain variables are included in the sensitivity set and why other variables are not;*

b. *same question for H_1 given E_1;*

c. *same for H_1 given E_1 and E_3;*

d. *same for H_2 given E_1, E_2, E_3, E_4 and E_5;*

Now suppose that an observation is entered for the variable E_2; the variable of interest is H_2 and we perform a sensitivity analysis. Assume that all variables are binary, that if some parameter probability $\Pr(v_i \mid c_{\rho(V_i)})$ is varied then its complement $\Pr(\neg v_i \mid c_{\rho(V_i)})$ is excluded from the analysis, and that we're interested only in the output probability $\Pr(h_2)$.

e. *How many calls to a standard propagation algorithm for probabilistic inference are required for a sensitivity analysis of the network if the analysis is conducted straightforwardly, using 10 steps.?*

f. *How many of such network propagations are required if the functional form of the sensitivity function is exploited?*

Exercise 6.2

Consider the small Brain-tumour network from Figure 6.1.

* a. *Suppose that only the variable B has been observed. Which output probabilities are not influenced by variation of the parameters for B?*

* b. *Establish the minimal set of variables that need be observed to render variable CT given C insensitive to variation of the parameter probabilities of ISC.*

* c. *To which parameter probabilities does the output probability $\Pr(c \mid sh)$ relate linearly? Which parameter probabilities may, upon variation, have a non-linear effect on this output probability?*

d. *Analytically express the probability $\Pr(b)$ as a function of the parameter $p(b \mid mc)$.*

e. *Analytically express the probability $\Pr(c \mid b \wedge mc)$ as a function of the parameter $p(isc \mid mc)$.*

Chapter 7

Conclusions

Probabilistic reasoning with the probabilistic network framework is an exciting research area. First and foremost, the probabilistic network framework may be looked upon as a mathematically sound computational framework for probabilistic inference. From this point of view, we have addressed the algorithms offered by the framework. We have argued that although these algorithms have an exponential worst-case time complexity, they tend to behave polynomially for most real-life probabilistic networks. However, as applications of the framework grow larger, the probabilistic networks involved increase in size accordingly. Networks comprising hundreds or even thousands of vertices are no exception. For probabilistic networks of this size, the basic algorithms for probabilistic inference inevitably slow down problem solving despite their polynomial behaviour. Modern probabilistic network research therefore aims at developing more efficient algorithms. Efficiency is sought after in many different ways: existing algorithms are further optimised, new algorithms are designed for example based on simulation techniques, yet other optimisations derive from knowledge-based pruning and focusing.

The probabilistic network framework may also be looked upon as a framework for building knowledge-based systems. In fact, experience with developing applications of the framework is progressing rapidly. From this point of view, we have addressed the issue of building a probabilistic network for a domain of application. In many respects, building a probabilistic network resembles engineering a knowledge-based system more in general. Available knowledge-engineering methodologies, however, will have to be supplemented with methodologies tailored to probabilistic network building. Such methodologies are now emerging. As experience with applying the probabilistic network framework is building, the need for methods for knowledge-based control over reasoning is beginning to manifest itself. We have briefly addressed this issue and outlined shaping diagnostic problem solving with a probabilistic network. Research on the provision of control over probabilistic network reasoning has scarcely begun and different control methods as demanded by various types of problem solving have yet to be designed.

Bibliography

[Andreassen *et al.*, 1987] S. Andreassen, M. Woldbye, B. Falck, S.K. Andersen. MUNIN - A causal probabilistic network for interpretation of electromyographic findings. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 1987, pp. 366 – 372.

[Andreassen *et al.*, 1991] S. Andreassen, R. Hovorka, J. Benn, K.G. Olesen, E.R. Carson. A model-based approach to insulin adjustment. In: M. Stefanelli, A. Hasman, M. Fieschi, J. Talmon. *Proceedings of the Third Conference on Artificial Intelligence in Medicine.* Lecture Notes in Medical Informatics 44, Springer Verlag, Berlin, 1991, pp. 239 – 248.

[Baioletti *et al.*, 2011] M. Baioletti, G. Busanello, B. Vantaggi. Acyclic directed graphs representing independence models. *International Journal of Approximate Reasoning*, 52(1), 2011, pp. 2–18.

[Bellazzi *et al.*, 1991] R. Bellazzi, C. Berzuini, S. Quaglini, D.J. Spiegelhalter, M. Leaning. Cytotoxic chemotherapy monitoring using stochastic simulation on graphical models. In: M. Stefanelli, A. Hasman, M. Fieschi, J. Talmon. *Proceedings of the Third Conference on Artificial Intelligence in Medicine.* Lecture Notes in Medical Informatics 44, Springer Verlag, Berlin, 1991, pp. 227 – 238.

[Ben-Bassat, 1978] M. Ben-Bassat. Myopic policies in sequential classification. *IEEE Transactions on Computers*, vol. C-27, 1978, pp. 170 – 174.

[Blanco *et al.*, 2005] R. Blanco, I. Inza, M. Merino, J. Quiroga, P. Larranñaga. Feature selection in Bayesian classifiers for the prognosis of survival of cirrhotic patients treated with TIPS. *Journal of Biomedical Informatics*, vol. 38. 2005, pp. 376 – 388.

[Boose & Gaines, 1988] J. Boose, B. Gaines. *Knowledge Acquisition Tools for Expert Systems*, Academic Press, London, 1988.

[Bouckaert, 1995] R.R. Bouckaert. *Bayesian Belief Networks: from Construction to Inference*, Ph.D. thesis, Utrecht University, 1995.

[Bruza & Van der Gaag, 1994] P.D. Bruza, L.C. van der Gaag. Index expression belief networks for information disclosure. *International Journal of Expert Systems: Research and Applications*, vol. 7, 1994, pp. 107 – 138.

[Buchanan & Shortliffe, 1984] B.G. Buchanan, E.H. Shortliffe. *Rule-based Expert Systems. The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading, Massachusetts, 1984.

[Cheeseman, 1988] P. Cheeseman. An inquiry into computer understanding. *Computational Intelligence*, vol. 4, 1988, pp. 58 – 66.

[Cooper, 1990] G.F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, vol. 42, 1990, pp. 393 – 405.

[Coupé & Van der Gaag, 1998] V.M.H. Coupé and L.C. van der Gaag. Practicable sensitivity analysis of Bayesian belief networks. *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*, Union of Czech Mathematicians and Physicists, 1998, pp. 81 – 86.

[Coupé & Van der Gaag, 2002] V.M.H. Coupé and L.C. van der Gaag. Properties of sensitivity analysis of Bayesian belief networks. *Annals of Mathematics and Artificial Intelligence*, vol. 36, 2002, pp. 323 – 356.

[Cousins *et al.*, 1993] S.B. Cousins, W. Chen, M.E. Frisse. A tutorial to stochastic simulation algorithms for inference in belief networks. *Artificial Intelligence in Medicine*, vol. 5, 1993, pp. 315 – 340.

[Cox, 1979] R.T. Cox. Of inference and inquiry — an essay in inductive logic. *The Maximum Entropy Formalism*, MIT Press, Cambridge, Massachusetts, 1970.

[Dagum & Luby, 1993] P. Dagum, M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, vol. 60, 1993, pp. 141 – 153.

[Dawid, 1985] A.P. Dawid. Calibration-based empirical probability. *Annals of Statistics*, vol. 13, 1985, pp. 1251 – 1274.

[De Dombal *et al.*, 1972] F.T. de Dombal, D.J. Leaper, J.R. Staniland, A.P. McCann, J.C. Horrocks. Computer-aided diagnosis of acute abdominal pain. *British Medical Journal*, vol. 2, 1972, pp. 9 – 13.

[De Dombal *et al.*, 1974] F.T. de Dombal, D.J. Leaper, J.C. Horrocks, J.R. Staniland, A.P. McCann. Human and computer-aided diagnosis of abdominal pain: further report with emphasis on the performance of clinicians. *British Medical Journal*, vol. 4, 1974, pp. 376 – 380.

[DeGroot & Fienberg, 1983] M.H. DeGroot and S.E. Fienberg. The comparison and evaluation of forecasters. *The Statistician*, vol. 32, 1983, pp. 12 – 22.

[Dempster *et al.*, 1977] A.P. Dempster, N.M. Laird, D.B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B* 39(1): 1-38, 1977.

[Druzdzel & Van der Gaag, 1995] M.J. Druzdzel, L.C. van der Gaag. Elicitation of probabilities for belief networks: combining qualitative and quantitative information. *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, 1995, pp. 141 – 148.

[Druzdzel & Van der Gaag, 2000] M.J. Druzdzel, L.C. van der Gaag. Building probabilistic networks: 'Where do the numbers come from?' Guest editors' introduction. *IEEE Transactions on Knowledge and Data Engineering*, 12(4):481-486, 2000.

[Finetti, 1970] B. de Finetti. *Theory of Probability*, Wiley, New York, 1970.

[Fletcher *et al.*, 1996] R.H. Fletcher, S.W. Fletcher, and E.H. Wagner. *Clinical Epidemiology. The Essentials*, 3rd ed, Williams & Wilkins, Baltimore, 1996.

[Friedman *et al.*, 1997] N. Friedman, D. Geiger, M. Goldszmidt, G. Provan, P. Langley, P. Smyth. Bayesian Network Classifiers, *Machine Learning*, 1997, pp. 131–163.

[Geiger & Pearl, 1988] D. Geiger, J. Pearl. On the logic of causal models, *Proceedings of the Fourth Workshop on Uncertainty in Artificial Intelligence*, 1988, pp. 136 – 147.

[Geiger *et al.*, 1990] D.E. Geiger, T. Verma, J. Pearl. *d*-separation: from theorems to algorithms. In: M. Henrion, R.D. Shachter, L.N. Kanal, J.F. Lemmer. *Uncertainty in Artificial Intelligence 5*, Elsevier Science, Amsterdam, 1990, pp. 139 – 148.

[Glasziou & Hilden, 1989] P. Glasziou, J. Hilden. Test selection measures. *Medical Decision Making*, vol. 9, 1989, pp. 133 – 141.

[Gorry & Barnett, 1968] G.A. Gorry, G.O. Barnett. Experience with a model of sequential diagnosis. *Computers and Biomedical Research*, vol. 1, 1968, pp. 490 – 507.

[Guida & Tasso, 1989] G. Guida, C. Tasso. *Topics in Expert System Design*, North-Holland, Amsterdam, 1989.

[Heckerman *et al.*, 1992] D.E. Heckerman, E.J. Horvitz, B.N. Nathwani. Toward normative expert systems. Part 1: The Pathfinder project. *Methods of Information in Medicine*, vol. 31, 1992, pp. 90 – 105.

[Heckerman *et al.*, 1993] D.E. Heckerman, E.J. Horvitz, B. Middleton. An approximate nonmyopic computation for value of information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, 1993, pp. 292 – 298.

[Helsper & Van der Gaag, 2002] E.M. Helsper, L.C. van der Gaag. Building Bayesian networks through ontologies. In: F. van Harmelen (ed.). *Proceedings of the 15th European Conference on Artificial Intelligence*. IOS Press, Amsterdam, 2002, pp. 680 – 684.

[Henrion, 1989] M. Henrion. Some practical issues in constructing belief networks. In: L.N. Kanal, T.S. Levitt, J.F. Lemmer (eds). *Uncertainty in Artificial Intelligence 3*, Elsevier Science, North-Holland, 1989.

[Jackson, 1990] P. Jackson. *Introduction to Expert Systems*. Addison-Wesley, Wokingham, 1990.

[Jensen, 1995] A.L. Jensen. Quantification experience of a DSS for mildew management in winter wheat. In: M.J. Druzdzel, L.C. van der Gaag, M. Henrion, F.V. Jensen (eds). *IJCAI-95 Workshop on Building Probabilistic Networks: Where Do the Numbers Come From ?*, 1995, pp. 23 – 31.

[Jensen, 1996] F.V. Jensen. *An Introduction to Bayesian Networks.* UCL Press, London, 1996.

[Jensen *et al.*, 1990] F.V. Jensen, J. Nielsen, H.I. Christensen. *Use of Causal Probabilistic Networks as High Level Models in Computer Vision.* Technical Report R-90-39, University of Aalborg, Denmark, 1990.

[Karp *et al.*, 1989] R. Karp, M. Luby, N. Madras. Monte-Carlo approximation algorithms for enumeration problems. *Journal of Algorithms*, vol. 10, 1989, pp. 429 – 448.

[Kjærulff & Van der Gaag, 2000] U. Kjærulff and L.C. van der Gaag. Making sensitivity analysis computationally efficient. *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Francisco, 2000, pp. 317 – 325.

[Korver & Lucas, 1993] M. Korver, P.J.F. Lucas. Converting a rule-based expert system into a belief network. *Medical Informatics*, vol. 18, 1993, pp. 219 – 241.

[Lauritzen & Spiegelhalter, 1988] S.L. Lauritzen, D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, vol. 50, 1988, pp. 157 – 224.

[Lucas, 2005] P.J.F. Lucas. Bayesian network modelling through qualitative patterns. *Artificial Intelligence,* vol 163, 2005, pp. 233 – 263.

[Lucas & Van der Gaag, 1991] P.J.F. Lucas, L.C. van der Gaag. *Principles of Expert Systems.* Addison-Wesley, Wokingham, 1991.

[Morgan & Henrion, 1990] M.G. Morgan and M. Henrion. *Uncertainty, a Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis.* Cambridge University Press, Cambridge, 1990.

[Neapolitan, 2003] R.E. Neapolitan. *Learning Bayesian Networks*, Prentice Hall, 2003.

[Panofsky & Brier, 1968] H.A. Panofsky and G.W. Brier. *Some Applications of Statistics to Meteorology.* The Pennsylvania State University, University Park, Pennsylvania, 1968.

[Pauker & Kassirer, 1980] S.G. Pauker and J.P. Kassirer. The threshold approach to clinical decision making. *New England Journal of Medicine*, vol. 302, 1980, pp. 1109 – 1117.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems. Networks of Plausible Inference.* Morgan Kaufmann, Palo Alto, 1988.

[Pearl *et al.*, 1990] J. Pearl, D. Geiger, and T. Verma. The logic of influence diagrams, in: R.M. Oliver and J.Q. Smith (eds). *Influence Diagrams, Belief Nets and Decision Analysis*, John Wiley & Sons, 1990, pp. 67 – 86.

[Pearl & Paz, 1985] J. Pearl, A. Paz. GRAPHOIDS: a graph-based logic for reasoning about relevance relations. In: B. Du Boulay, D. Hogg, L. Steels (eds). *Advances in Artificial Intelligence 2*, 1985, North-Holland.

[Pearl & Verma, 1987] J. Pearl, T.S. Verma. The logic of representing dependencies by directed acyclic graphs, *Proceedings of the Sixth National Conference on Artificial Intelligence*, 1987, pp. 374 – 379.

[Peek & Ottenkamp, 1997] N.B. Peek, J. Ottenkamp. Developing a decision-theoretic network for a congenital heart disease, in: E. Keravnou, C. Garbay, R. Baud J. Wyatt (eds). *Proceedings of the 6th Conference on Artificial Intelligence in Medicine Europe.* Springer-Verlag, Berlin, 1997, pp. 157 – 168.

[Pourret, Naim & Marcot, 2008] O. Pourret, P. Naim, B. Marcot (editors). *Bayesian Networks. A Practical Guide to Applications.*Wiley, England, 2008.

[Renooij, 2001] S. Renooij. *Qualitative Approaches to Quantifying Probabilistic Networks.* Ph.D. Thesis, Institute of Information and Computing Sciences, Utrecht University, The Netherlands.

[Renooij, 2001b] S. Renooij. Probability elicitation for belief networks: Issues to consider. *Knowledge Engineering Review*, vol. 16, 2001, pp. 255-269.

[Sent, 2005] D. Sent. *Test-selection Strategies for Probabilistic Networks.* Ph.D. Thesis, Department of Information and Computing Sciences, Utrecht University, The Netherlands.

[Shafer, 1976] G. Shafer. *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, 1976.

[Shafer & Pearl, 1990] G. Shafer, J. Pearl. *Readings in Uncertain Reasoning.* Morgan Kaufmann, Palo Alto, 1990.

[Shortliffe & Buchanan, 1984] E.H. Shortliffe, B.G. Buchanan. A model of inexact reasoning in medicine. In: B.G. Buchanan, E.H. Shortliffe. *Rule-based Expert Systems. The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading, Massachusetts, 1984, pp. 233 – 262.

[Shwe *et al.*, 1991] M.A. Shwe, B. Middleton, D.E. Heckerman, M. Henrion, E.J. Horvitz, H.P. Lehmann, G.F. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base I: the probabilistic model and inference algorithms. *Methods of Information in Medicine*, vol. 30, 1991, pp. 241 – 255.

[Sommerville, 1992] I. Sommerville. *Software Engineering*, Addison-Welsy, Wokingham, 1992.

[Stahlschmidt *et al.*, 2013] S. Stahlschmidt, H. Tausendteufel, W.K. Härdle. Bayesian networks for sex-related homicides: structure learning and prediction. *Journal of Applied Statistics* 40(6), 2013, pp. 1155–1171.

[Studený, 1989] M. Studený. Multiinformation and the problem of characterization of conditional-independence relations. *Problems Control Information Theory*, vol. 18, 1989, pp. 3 – 16.

[Studený, 1992] M. Studený. Conditional independence relations have no finite complete characterization, in: S. Kubik, J.A. Visek (eds). *Information Theory, Statistical Decision Functions and Random Processes*, Kluwer, Dordrecht, 1992, pp. 377 – 396.

[Studený, 1998] M. Studený. Bayesian networks from the point of view of chain graphs, in: G. F. Cooper, S. Moral (eds). *Uncertainty in Artificial Intelligence. Proceedings of the 14th Conference*, Morgan Kaufmann, San Francisco 1998, pp. 496–503.

[Suermondt & Cooper, 1990] H.J. Suermondt, G.F. Cooper. Probabilistic inference in multiply connected belief networks using loop cutsets. *International Journal of Approximate Reasoning*, vol. 4, 1990, pp. 283 – 306.

[Suermondt & Cooper, 1991a] H.J. Suermondt, G.F. Cooper. Initialization for the method of conditioning in Bayesian belief networks. *Artificial Intelligence*, vol. 50, 1991, pp. 83 – 94.

[Suermondt & Cooper, 1991b] H.J. Suermondt, G.F. Cooper. A combination of exact algorithms for inference on Bayesian belief networks. *International Journal of Approximate Reasoning*, vol. 5, 1991, pp. 521 – 542.

[Taroni *et al.*, 2006] F. Taroni, C. Aitken, P. Garbolino, A. Biedermann. *Bayesian Networks and Probabilistic Inference in Forensic Science*, Wiley & Sons, Chichester, 2006.

[Tversky *et al.*, 1982] D. Kahneman, P. Slovic, and A. Tversky. *Judgment under Uncertainty: Heuristics and Biases*, Cambridge University Press, Cambridge, 1982.

[Van der Gaag, 1994] L.C. van der Gaag. A pragmatic view of the certainty factor model. *The International Journal of Expert Systems: Research and Applications*, vol. 7, 1994, pp. 289 – 300.

[Van der Gaag & Helsper, 2002] L.C. van der Gaag, E.M. Helsper. Experiences with modelling issues in building probabilistic networks. In: A. Go'mez-Pe'rez and V.R. Benjamins (eds.). *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web. Proceedings of EKAW 2002.* LNAI vol. 2473, Springer-Verlag, 2002, pp. 21 – 26.

[Van der Gaag & Helsper, 2004] L.C. van der Gaag, E.M. Helsper. Defining classes of influences for the acquisition of probability constraints for Bayesian networks. In: R. Lo'pez de Ma'ntaras and L. Saitta (eds.). *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004).* IOS Press, Amsterdam, 2004, pp. 1101 - 1102.

[Van der Gaag & Meyer, 1996] L.C. van der Gaag, J.-J.Ch. Meyer. Characterising normal forms for informational independence. *Proceedings of IPMU'96: Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 1996, pp. 973 – 978.

[Van der Gaag & Renooij, 2001] L.C. van der Gaag, S. Renooij. Analysing sensitivity data from probabilistic networks. *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, 2001, pp. 530 – 537.

[Van der Gaag *et al.*, 1999] L.C. van der Gaag, S. Renooij, C.L.M. Witteman, B. Aleman, B.G. Taal. How to elicit many probabilities. *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, 1999, pp. 647 – 654.

[Van der Gaag *et al.*, 2002] L.C. van der Gaag, S. Renooij, C.L.M. Witteman, B.M.P. Aleman, and B.G. Taal. Probabilities for a probabilistic network: A case-study in oesophageal cancer. *Artificial Intelligence in Medicine*, vol. 25, 2002, pp. 123 – 148.

[Van der Gaag & Wessels, 1994a] L.C. van der Gaag, M.L. Wessels. Selective evidence gathering for diagnostic belief networks. *AISB Quarterly*, vol. 86, 1994, pp. 23 – 34.

[Van der Gaag & Wessels, 1994b] L.C. van der Gaag, M.L. Wessels. Multiple-disorder diagnosis with belief networks. *Proceedings of the Fifth Workshop on Principles of Diagnosis — DX'94*, 1994, pp. 343 – 351.

[Van der Gaag & Meyer, 1998] L.C. van der Gaag, J.-J.Ch. Meyer. Informational independence: models and normal forms. *International Journal of Intelligent Systems*, 13, 1998, pp. 83 – 109.

[Waal & Van der Gaag, 2005] P.R. de Waal, L.C. van der Gaag. Stable Independence in Perfect Maps. *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, AUAI Press, 2005.

[Warner *et al.*, 1961] H.R. Warner, A.F. Toronto, L.G. Veasy, R. Stephenson. A mathematical approach to medical diagnosis: application to congenital heart disease. *Journal of the American Medical Association*, vol. 177, 1961, pp. 177 – 183.

[Wellman, 1990] M.P. Wellman. Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, vol. 44, 1990, pp. 257 – 303.

[Von Winterfeldt & Edwards, 1986] D. von Winterfeldt, W. Edwards. *Decision Analysis and Behavioral Research.* Cambridge University Press, New York, 1986.

Chapter 8

Solutions, Answers and Hints

This chapter presents solutions, answers or hints for selected exercises.

Exercise 2.1

d. We prove the conditioning property stated in Proposition 2.2.8.

Let V be a set of statistical variables and let Pr be a joint probability distribution on V. We have to show that

$$\Pr(C_X) = \sum_{c_Y} \Pr(C_X \mid c_Y) \cdot \Pr(c_Y)$$

for all sets of variables $X, Y \subseteq V$. From the definition of conditional probability, we have that

$$\begin{aligned}\sum_{c_Y} \Pr(C_X \mid c_Y) \cdot \Pr(c_Y) &= \sum_{c_Y} \frac{\Pr(C_X \wedge c_Y)}{\Pr(c_Y)} \cdot \Pr(c_Y) = \\ &= \sum_{c_Y} \Pr(C_X \wedge c_Y)\end{aligned}$$

By employing the marginalisation property stated in Proposition 2.2.7, we further find that

$$\begin{aligned}\sum_{c_Y} \Pr(C_X \mid c_Y) \cdot \Pr(c_Y) &= \sum_{c_Y} \Pr(C_X \wedge c_Y) = \\ &= \Pr(C_X)\end{aligned}$$

which concludes our proof.

Exercise 2.4

To prove the property stated in the exercise, we show that

$$\Pr(C_X \mid C_Y \wedge C_Z) = \Pr(C_X \mid C_Z) \Leftrightarrow \Pr(C_X \wedge C_Y \mid C_Z) = \Pr(C_X \mid C_Z) \cdot \Pr(C_Y \mid C_Z)$$

From the definition of conditional probability, we have that

$$\Pr(C_X \mid C_Y \wedge C_Z) = \Pr(C_X \mid C_Z) \quad \Leftrightarrow \frac{\Pr(C_X \wedge C_Y \wedge C_Z)}{\Pr(C_Y \wedge C_Z)} = \frac{\Pr(C_X \wedge C_Z)}{\Pr(C_Z)}$$

By division of both the left hand side and the right hand side of the latter expression by $\Pr(C_Z)$ and subsequent rearrangement of terms, we find that

$$\Pr(C_X \mid C_Y \wedge C_Z) = \Pr(C_X \mid C_Z) \Leftrightarrow$$

$$\Leftrightarrow \frac{\Pr(C_X \wedge C_Y \wedge C_Z)}{\Pr(C_Z)} = \frac{\Pr(C_X \wedge C_Z)}{\Pr(C_Z)} \cdot \frac{\Pr(C_Y \wedge C_Z)}{\Pr(C_Z)} \Leftrightarrow$$

$$\Leftrightarrow \Pr(C_X \wedge C_Y \mid C_Z) = \Pr(C_X \mid C_Z) \cdot \Pr(C_Y \mid C_Z)$$

which concludes our proof.

Exercise 3.1

For $I_{\Pr}$ you need to use definitions based on Pr.

a. See course transparencies.

b. A sketch of the proof is on the course transparencies (the full proof is quite complex).

c. We show that the independence relation $I_{\Pr}$ satisfies the property

$$I_{\Pr}(X, Z, Y \cup W) \rightarrow I_{\Pr}(X, Z \cup W, Y)$$

for all mutually disjoint sets of variables $X, Y, Z, W \subseteq V$, that is, we prove the third property stated in Theorem 3.1.2.

We assume that $I_{\Pr}(X, Z, Y \cup W)$. From this observation, we have

$$\Pr(C_X \mid C_Z \wedge C_{Y \cup W}) = \Pr(C_X \mid C_Z)$$

by definition, that is, we have that

$$\frac{\Pr(C_X \wedge C_Y \wedge C_Z \wedge C_W)}{\Pr(C_Y \wedge C_Z \wedge C_W)} = \frac{\Pr(C_X \wedge C_Z)}{\Pr(C_Z)}$$

From our assumption $I_{\Pr}(X, Z, Y \cup W)$, we further have $I_{\Pr}(X, Z, W)$ by the second property stated in the exercise. By definition, we therefore have that

$$\Pr(C_X \mid C_Z \wedge C_W) = \Pr(C_X \mid C_Z)$$

that is,

$$\frac{\Pr(C_X \wedge C_Z \wedge C_W)}{\Pr(C_Z \wedge C_W)} = \frac{\Pr(C_X \wedge C_Z)}{\Pr(C_Z)}$$

Now consider the conditional probability $\Pr(C_X \mid C_{Z \cup W} \wedge C_Y)$. By definition, we find that

$$\Pr(C_X \mid C_{Z \cup W} \wedge C_Y) = \frac{\Pr(C_X \wedge C_Y \wedge C_Z \wedge C_W)}{\Pr(C_Y \wedge C_Z \wedge C_W)}$$

From the previous observations, we further find that

$$\begin{aligned}\Pr(C_X \mid C_{Z \cup W} \wedge C_Y) &= \frac{\Pr(C_X \wedge C_Z)}{\Pr(C_Z)} = \\ &= \frac{\Pr(C_X \wedge C_Z \wedge C_W)}{\Pr(C_Z \wedge C_W)} = \\ &= \Pr(C_X \mid C_{Z \cup W})\end{aligned}$$

From $\Pr(C_X \mid C_{Z \cup W} \wedge C_Y) = \Pr(C_X \mid C_{Z \cup W})$, we have by definition that $I_{\Pr}(X, Z \cup W, Y)$. We conclude that $I_{\Pr}(X, Z, Y \cup W) \rightarrow I_{\Pr}(X, Z \cup W, Y)$.

Exercise 3.2

We begin our proof by observing that, since I is a semi-graphoid independence relation, it obeys the first four axioms stated in Definition 3.1.3. Now, we assume that $I(X, Z, Y \cup W)$ and $I(Y, Z, W)$. We have that

$$I(X, Z, Y \cup W) \rightarrow I(X, Z \cup W, Y) \rightarrow I(Y, Z \cup W, X)$$

by the weak union and symmetry axioms; in conjunction with our assumption $I(Y, Z, W)$, we find

$$I(Y, Z \cup W, X) \wedge I(Y, Z, W) \rightarrow I(Y, Z, W \cup X) \rightarrow I(X \cup W, Z, Y)$$

by the contraction and symmetry axioms.

Exercise 3.3

For I you need to use the four properties on which it is defined (Definition 3.1.3).

Exercise 3.5

This exercise requires you to draw 16 D-maps and 4 I-maps.

Exercise 3.7

a. The property holds (why?).

b. The property doesn't hold (why?).

c. The property $\langle \{V_2\} \mid \{V_1\} \mid \{V_3\} \rangle_G^d$ holds in the digraph G since all chains in G from V_2 to V_3 are blocked by the set of vertices $\{V_1\}$. For example, the chain V_2, V_1, V_3 from V_2 to V_3 is blocked by $\{V_1\}$ since $V_1 \in \{V_1\}$; the chain V_2, V_5, V_3 from V_2 to V_3 is blocked by $\{V_1\}$ since $\{V_5, V_6\} \cap \{V_1\} = \varnothing$.

d. The property holds (why?).

e. The property $\langle \{V_2\} \mid \{V_3, V_4\} \mid \{V_6\} \rangle_G^d$ does *not* hold in the digraph G since not every chain in G from V_2 to V_6 is blocked by the set of vertices $\{V_3, V_4\}$. For example, the chain V_2, V_5, V_6 from V_2 to V_6 is not blocked by $\{V_3, V_4\}$.

f. The property doesn't hold (why?).

Exercise 3.8

This exercise requires you to draw some of the 15 D-maps and 63 I-maps (24 with 6 arrows, 28 with 5, 10 with 4 and 1 with 3 arrows).

Exercise 3.9

A similar proposition for *un*directed D-maps and I-maps is proven in Lemma 3.2.6, as well as on the course transparencies.

Exercise 3.10

Hint: it is easiest to try and use a graph as small as possible, with only three vertices.

Exercise 3.13

Not necessarily (give an example!)

Exercise 3.14

Hint: it is easiest to try and use a graph as small as possible; use three vertices for a) and c), four vertices for b) and d).

Exercise 4.2

Your computations should give $\Pr(v_1 \wedge v_2 \wedge v_3) = 0.09375$; $\Pr(v_2 \wedge v_3) = 0.24375$; $\Pr(v_1 \mid v_2 \wedge v_3) = 0.38462$; $\Pr(v_1 \vee v_2 \vee \neg v_3 \vee v_4) = 0.985$.

Exercise 4.3

Be inspired by Lemma 4.2.8 or use the definition of $\lambda_{V_i}(V_i)$.

Exercise 4.4

NB Contrary to the examples in Chapter 4, it is sufficient to illustrate the working of Pearl's algorithm by computing only those parameters required to establish the requested probabilities, rather than performing each and every computation done by the algorithm. See for example the answer to Exercise 4.5a.

a. Your computations should give: $\Pr(v_4) = 0.56$, $\Pr(\neg v_4) = 0.44$.

b. Your computations should give: $\Pr(v_4 \mid v_6) = 0.88$, $\Pr(\neg v_4 \mid v_6) = 0.12$.

c. Entering the evidence $V_2 =$ *false* cannot influence the probabilities of the values of the vertex V_6 since this vertex is already instantiated. The evidence, however, may influence the probabilities of the values of each of the other vertices: neither of V_1, V_3, V_4, and V_5 is d-separated from V_2 by the set $\{V_6\}$, the set of vertices for which evidence has been entered.

Exercise 4.5

a. For computing the probabilities $\Pr(v_5)$ and $\Pr(\neg v_5)$ of its values vertex V_5 applies the *data fusion lemma*:

$$\begin{aligned}\Pr(v_5) &= \alpha \cdot \pi_{V_5}(v_5) \cdot \lambda_{V_5}(v_5)\\ \Pr(\neg v_5) &= \alpha \cdot \pi_{V_5}(\neg v_5) \cdot \lambda_{V_5}(\neg v_5)\end{aligned}$$

Since no evidence has been entered in the network as yet, we have from Lemma 4.2.8 for the compound diagnostic parameter λ_{V_5} of V_5 that $\lambda_{V_5}(v_5) = 1$ and $\lambda_{V_5}(\neg v_5) = 1$.

For the value $\pi_{V_5}(v_5)$ of its compound causal parameter, vertex V_5 computes

$$\pi_{V_5}(v_5) = \gamma_{V_5}(v_5 \mid v_3) \cdot \pi^{V_3}_{V_5}(v_3) + \gamma_{V_5}(v_5 \mid \neg v_3) \cdot \pi^{V_3}_{V_5}(\neg v_3)$$

where

$$\begin{aligned}\pi^{V_3}_{V_5}(v_3) &= \gamma_{V_3}(v_3 \mid v_1 \wedge v_2) \cdot \gamma_{V_1}(v_1) \cdot \gamma_{V_2}(v_2) + \\ &\quad + \gamma_{V_3}(v_3 \mid \neg v_1 \wedge v_2) \cdot \gamma_{V_1}(\neg v_1) \cdot \gamma_{V_2}(v_2) + \\ &\quad + \gamma_{V_3}(v_3 \mid v_1 \wedge \neg v_2) \cdot \gamma_{V_1}(v_1) \cdot \gamma_{V_2}(\neg v_2) + \\ &\quad + \gamma_{V_3}(v_3 \mid \neg v_1 \wedge \neg v_2) \cdot \gamma_{V_1}(\neg v_1) \cdot \gamma_{V_2}(\neg v_2) = \\ &= 0.2 \cdot 0.8 \cdot 0.5 + 0.6 \cdot 0.2 \cdot 0.5 + 0.5 \cdot 0.8 \cdot 0.5 + 0.1 \cdot 0.2 \cdot 0.5 = \\ &= 0.35\end{aligned}$$

and

$$\begin{aligned}\pi^{V_3}_{V_5}(\neg v_3) &= \gamma_{V_3}(\neg v_3 \mid v_1 \wedge v_2) \cdot \gamma_{V_1}(v_1) \cdot \gamma_{V_2}(v_2) + \\ &\quad + \gamma_{V_3}(\neg v_3 \mid \neg v_1 \wedge v_2) \cdot \gamma_{V_1}(\neg v_1) \cdot \gamma_{V_2}(v_2) + \\ &\quad + \gamma_{V_3}(\neg v_3 \mid v_1 \wedge \neg v_2) \cdot \gamma_{V_1}(v_1) \cdot \gamma_{V_2}(\neg v_2) + \\ &\quad + \gamma_{V_3}(\neg v_3 \mid \neg v_1 \wedge \neg v_2) \cdot \gamma_{V_1}(\neg v_1) \cdot \gamma_{V_2}(\neg v_2) = \\ &= 0.8 \cdot 0.8 \cdot 0.5 + 0.4 \cdot 0.2 \cdot 0.5 + 0.5 \cdot 0.8 \cdot 0.5 + 0.9 \cdot 0.2 \cdot 0.5 = \\ &= 0.65\end{aligned}$$

So,

$$\pi_{V_5}(v_5) \;=\; 0.6 \cdot 0.35 + 0.4 \cdot 0.65 \;=\; 0.47$$

Analogously, vertex V_5 computes

$$\pi_{V_5}(\neg v_5) \;=\; 0.53$$

By substituting the values $\lambda_{V_5}(v_5)$, $\lambda_{V_5}(\neg v_5)$, $\pi_{V_5}(v_5)$, and $\pi_{V_5}(\neg v_5)$ in the *data fusion lemma* and subsequently eliminating the normalisation constant α we find

$$\begin{aligned}\Pr(v_5) &= 0.47 \\ \Pr(\neg v_5) &= 0.53\end{aligned}$$

b. Your computions should result in $\Pr(v_5 \mid v_3) = 0.6$, $\Pr(\neg v_5 \mid v_3) = 0.4$.

c. Entering the evidence $V_3 = $ *true* cannot influence the probabilities of the values of the vertex V_4. By exploiting the d-separation criterion it is easily seen that the vertex V_4 is independent of the vertex V_3 given the empty set $\varnothing$, that is, the set of vertices for which evidence has been entered: the only chain in the digraph G between the vertices V_3 and V_4 is blocked by $\varnothing$. Since no other vertex in G is d-separated from vertex V_3 by the empty set, the evidence for V_3 may influence the probabilities of the values of all other vertices in the network.

Exercise 4.6

NB See hints for Exercises 4.4 and 4.5a: it is sufficient to compute only those parameters required to establish the requested probabilities.

a. $\Pr(v_1 \mid v_7) = 0.6$, $\Pr(\neg v_1 \mid v_7) = 0.4$.

b. Yes, possibly (why?).

c. This probability of a conjunction cannot be *directly* computed using Pearl's algorithm (why?). Explain how you would compute it (consider Pearl's algorithm as a black box).

Exercise 4.7

NB See hints for Exercises 4.4 and 4.5a: it is sufficient to compute only those parameters required to establish the requested probabilities.

a. $\Pr(v_3 \mid v_1 \wedge \neg v_6) = 0.6$, $\Pr(\neg v_3 \mid v_1 \wedge \neg v_6) = 0.4$.

b. This probability of a conjunction cannot be *directly* computed using Pearl's algorithm (why?). Explain how you would compute it (consider Pearl's algorithm as a black box).

c. See b.

Exercise 4.8

This can be proven using induction.

Exercise 4.9

a. You should find either $\{V_2\}$ or $\{V_3\}$.

b. $\Pr(v_5) = 0.34$, $\Pr(\neg v_5) = 0.66$. *NB See hints for Exercises 4.4 and 4.5a: it is sufficient to compute only those parameters required to establish the requested probabilities.*

Exercise 4.10

a. $\{V_2\}$.

b. $\Pr(v_3) = 0.3$, $\Pr(\neg v_3) = 0.7$. *NB See hints for Exercises 4.4 and 4.5a: it is sufficient to compute only those parameters required to establish the requested probabilities.*

c. These probabilities cannot be *directly* computed using Pearl's (datafusion) algorithm (why?). Explain how you would compute them (consider Pearl's algorithm as a black box).

Exercise 4.11

a. You should find $\{V_1, V_2\}$ or $\{V_2, V_3\}$.

Exercise 4.12

a. A possible lcs is $\{V_2, V_3, V_7, V_8\}$; you won't find $\{V_{10}\}$ (why?).

Exercise 5.4

d. The available information suffices for specifying a complete probability assessment function for the variable *HeartAttack*. Since the property of exception independence is satisfied and sufficient probabilities have been assessed, the model of the *leaky noisy or-gate* can be exploited. The complete probability assessment function is defined by

$$\begin{array}{lcl}
\gamma_H(h \mid s \wedge b \wedge c) & = & 1 - 0.95 \cdot \frac{0.4}{0.95} \cdot \frac{0.2}{0.95} \cdot \frac{0.1}{0.95} = 0.991 \\
\gamma_H(h \mid \neg s \wedge b \wedge c) & = & 1 - 0.95 \cdot \frac{0.2}{0.95} \cdot \frac{0.1}{0.95} = 0.980 \\
\gamma_H(h \mid s \wedge \neg b \wedge c) & = & 1 - 0.95 \cdot \frac{0.4}{0.95} \cdot \frac{0.1}{0.95} = 0.958 \\
\gamma_H(h \mid s \wedge b \wedge \neg c) & = & 1 - 0.95 \cdot \frac{0.4}{0.95} \cdot \frac{0.2}{0.95} = 0.916 \\
\gamma_H(h \mid \neg s \wedge \neg b \wedge c) & = & 0.9 \\
\gamma_H(h \mid \neg s \wedge b \wedge \neg c) & = & 0.8 \\
\gamma_H(h \mid s \wedge \neg b \wedge \neg c) & = & 0.6 \\
\gamma_H(h \mid \neg s \wedge \neg b \wedge \neg c) & = & 0.05
\end{array}$$

and the complementary function values.

e. Rule-based expert systems have been discussed at length in the Expert Systems course.

Exercise 5.5

Full assessment functions for all variables, except for variable V_4 can be established. Note the incoherence in the information!

Exercise 5.7

a. The result of your computation should be -0.55 (what will happen as a result?).

Exercise 6.1

a. $\{E_1, H_1\}$

b. $\{H_1\}$

c. $\{E_3, H_1, H_2\}$

d. $\{E_2, E_3, E_4, H_1, H_2, H_3\}$

e. 200 propagations are required if we assume 10-step variation per function.

f. 25 propagations are required if we assume 3 propagations only per non-linear function.

Exercise 6.2

a. $\Pr(ct \mid B)$, $\Pr(sh \mid B)$, and their complements.

b. $\{B\}$ (in addition to the already observed variable C)

c. Linear: parameters of ISC and C; non-linear: parameters of MC, B and SH. Variation of parameters of CT has no effect.