Chapter 4
Probabilistic Networks

In this chapter we introduce probabilistic networks for belief update and decision making under uncertainty.

Many real-life situations can be modeled as a domain of entities represented as random variables in a probabilistic network. A probabilistic network is a clever graphical representation of dependence and independence relations between random variables. A domain of random variables can, for instance, form the basis of a decision support system to help decision makers identify the most beneficial decision in a given situation.

A probabilistic network represents and processes probabilistic knowledge. The representational components of a probabilistic network are a qualitative and a quantitative component. The qualitative component encodes a set of (conditional) dependence and independence statements among a set of random variables, informational precedence, and preference relations. The statements of (conditional) dependence and independence, information precedence, and preference relations are visually encoded using a graphical language. The quantitative component, on the other hand, specifies the strengths of dependence relations using probability theory and preference relations using utility theory.

The graphical representation of a probabilistic network describes knowledge of a problem domain in a precise manner. The graphical representation is intuitive and easy to comprehend, making it an ideal tool for communication of domain knowledge between experts, users, and systems. For these reasons, the formalism of probabilistic networks is becoming an increasingly popular knowledge representation for belief update and decision making under uncertainty.

Since a probabilistic network consists of two components, it is customary to consider its construction as a two-phase process: the construction of the qualitative component and subsequently the construction of the quantitative component. The qualitative component defines the structure of the quantitative component. As the first step, the qualitative structure of the model is constructed using a graphical language. This step consists of identifying variables and relations between variables. As the second step, the parameters of the quantitative part as defined by the qualitative part are assessed.

U.B. Kjærulff and A.L. Madsen, *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis*, ISS 22, DOI 10.1007/978-1-4614-5104-4_4,

In this book, we consider the subclass of probabilistic networks known as Bayesian networks and influence diagrams. Bayesian networks and influence diagrams are ideal knowledge representations for use in many situations involving belief update and decision making under uncertainty. These models are often characterized as normative expert systems as they provide model-based domain descriptions, where the model is reflecting properties of the problem domain (rather than the domain expert) and probability calculus is used as the calculus for uncertainty.

A Bayesian network model representation of a problem domain can be used as the basis for performing inference and analysis about the domain. Decision options and utilities associated with these options can be incorporated explicitly into the model, in which case the model becomes an influence diagram, capable of computing expected utilities of all decision options given the information known at the time of decision. Bayesian networks and influence diagrams are applicable for a large range of domain areas with inherent uncertainty.

Section 4.1 considers Bayesian networks as probabilistic models for belief update. We consider Bayesian network models containing discrete variables only and models containing a mixture of continuous and discrete variables. Section 4.2 considers influence diagrams as probabilistic networks for decision making under uncertainty. The influence diagram is a Bayesian network augmented with decision variables, informational precedence relations, and preference relations. We consider influence diagram models containing discrete variables only and models containing a mixture of continuous and discrete variables. In Sect. 4.3 object-oriented probabilistic networks are considered. An object-oriented probabilistic network is a flexible framework for building hierarchical knowledge representations using the notions of classes and instances. In Sect. 4.4 dynamic probabilistic networks are considered. A dynamic probabilistic network is a method for representing dynamic systems that are changing over time.

4.1 Belief Update

A probabilistic interaction model between a set of random variables may be represented as a joint probability distribution. Considering the case where random variables are discrete, it is obvious that the size of the joint probability distribution will grow exponentially with the number of variables as the joint distribution must contain one probability for each configuration of the random variables. Therefore, we need a more compact representation for reasoning about the state of large and complex systems involving a large number of variables.

To facilitate an efficient representation of a large and complex domain with many random variables, the framework of Bayesian networks uses a graphical representation to encode dependence and independence relations among the random variables. The dependence and independence relations induce a compact representation of the joint probability distribution.

4.1.1 *Discrete Bayesian Networks*

A (discrete) Bayesian network, $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P})$, over variables, $\mathcal{X}$, consists of a DAG $\mathcal{G} = (V, E)$ and a set of conditional probability distributions $\mathcal{P}$. Each node v in G corresponds one-to-one with a discrete random variable $X_v \in \mathcal{X}$ with a finite set of mutually exclusive states. The directed links $E \subseteq V \times V$ of $\mathcal{G}$ specify assumptions of conditional dependence and independence between random variables according to the d-separation criterion (see Proposition 2.4 on page 33).

There is a conditional probability distribution, $P(X_v \mid X_{\text{pa}(v)}) \in \mathcal{P}$, for each variable $X_v \in \mathcal{X}$. The set of variables represented by the parents, pa(v), of $v \in V$ in $\mathcal{G} = (V, E)$ is sometimes called the *conditioning variables* of X_v — the *conditioned* variable.

Definition 4.1 (Jensen 2001). A (discrete) Bayesian network $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P})$ consists of:

- A DAG $\mathcal{G} = (V, E)$ with nodes $V = \{v_1, \ldots, v_n\}$ and directed links E
- A set of discrete random variables, $\mathcal{X}$, represented by the nodes of $\mathcal{G}$
- A set of conditional probability distributions, $\mathcal{P}$, containing one distribution, $P(X_v \mid X_{\text{pa}(v)})$, for each random variable $X_v \in \mathcal{X}$

A Bayesian network encodes a joint probability distribution over a set of random variables, $\mathcal{X}$, of a problem domain. The set of conditional probability distributions, $\mathcal{P}$, specifies a multiplicative factorization of the joint probability distribution over $\mathcal{X}$ as represented by the chain rule of Bayesian networks (see Sect. 3.7 on page 62):

$$P(\mathcal{X}) = \prod_{v \in V} P(X_v \mid X_{\text{pa}(v)}). \tag{4.1}$$

Even though the joint probability distribution specified by a Bayesian network is defined in terms of conditional independence, a Bayesian network is most often constructed using the notion of cause–effect relations, see Sect. 2.4. In practice, cause–effect relations between entities of a problem domain can be represented in a Bayesian network using a graph of nodes representing random variables and links representing cause–effect relations between the entities. Usually, the construction of a Bayesian network (or any probabilistic network for that matter) proceeds according to an iterative procedure where the set of nodes and their states and the set of links are updated iteratively as the model becomes more and more refined. In Chaps. 6 and 7, we consider in detail the art of building efficient probabilistic network representations of a problem domain.

To solve a Bayesian network $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P})$ is to compute all posterior marginals given a set of evidence ε, that is, $P(X \mid \varepsilon)$ for all $X \in \mathcal{X}$. If the evidence set is empty, that is, $\varepsilon = \emptyset$, then the task is to compute all prior marginals, that is, $P(X)$ for all $X \in \mathcal{X}$.

Example 4.1 (Apple Jack (Madsen, Nielsen & Jensen 1998)). Let us consider the small orchard belonging to Jack Fletcher (also called Apple Jack). One day, Apple

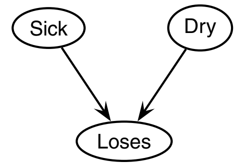


Fig. 4.1 The Apple Jack network

Jack discovers that his finest apple tree is *losing its leaves*. Apple Jack wants to know why this is happening. He knows that if the tree is *dry* (for instance, caused by a drought), there is no mystery as it is common for trees to lose their leaves during a drought. On the other hand, the loss of leaves can be *an indication of a disease*.

The qualitative knowledge about the cause–effect relations of this situation can be modeled by the DAG $\mathcal{G}$ shown in Fig. 4.1. The graph consists of three nodes: Sick, Dry, and Loses that represent variables of the same names. Each variable may be in one of two states: no and yes, that is, dom(Dry) = dom(Loses) = dom(Sick) = (no, yes). The variable Sick tells us that the apple tree is sick by being in state yes. Otherwise, it will be in state no. The variables Dry and Loses tell us whether or not the tree is dry and whether or not the tree is losing its leaves, respectively.

The graph, $\mathcal{G}$, shown in Fig. 4.1 models the cause–effect relations between variables Sick and Loses as well as between Dry and Loses. This is represented by the two (causal) links (Sick, Loses) and (Dry, Loses). In this way, Sick and Dry are common causes of the effect Loses.

Let us return to the discussion of causality considered previously in Sect. 2.4. When there is a causal dependence relation going from a variable A to another variable B, we expect that when A is in a certain state, this has an impact on the state of B. One should be careful when modeling causal dependence relations in a Bayesian network. Sometimes it is not quite obvious in which direction a link should point. In the Apple Jack example, we say that there is a causal impact from Sick on Loses, because when a tree is sick, this might cause the tree to lose its leaves. Why can we not say that when the tree loses its leaves, it might be sick and turn the link in the other direction? The reason is that it is the sickness that causes the tree to lose its leaves and not the lost leaves that causes the sickness.

Figure 4.1 shows the graphical representation of the Bayesian network model. This is referred to as the qualitative representation. To have a complete Bayesian network, we need to specify the quantitative representation. Recall that each variable has two states, no and yes.

The quantitative representation of a Bayesian network is the set of conditional probability distributions, $\mathcal{P}$, defined by the structure of $\mathcal{G}$. Table 4.1 shows the conditional probability distribution of Loses given Sick and Dry, that is, $P(\text{Loses} \mid \text{Dry}, \text{Sick})$. For variables Sick and Dry, we assume that $P(S) = (0.9, 0.1)$ and $P(D) = (0.9, 0.1)$ (we use D as short for Dry, S as short for Sick, and L as short for Loses).

Table 4.1 The conditional probability distribution $P(L \mid D, S)$

		L	
D	S	no	yes
no	no	0.98	0.02
no	yes	0.1	0.9
yes	no	0.15	0.85
yes	yes	0.05	0.95

Note that all distributions specify the probability of a variable being in a specific state depending on the configuration of its parent variables, but since Sick and Dry do not have any parent variables, their distributions are marginal distributions.

The model may be used to compute all prior marginals and the posterior distribution of each nonevidence variable given evidence in the form of observations on a subset of the variables in the model. The priors for D and S equal the specified marginal distributions, that is, $P(D) = P(S) = (0.9, 0.1)$, while the prior distribution for L is computed through combination of the distributions specified for the three variables, followed by marginalization, where variables D and S are marginalized out. This yields $P(L) = (0.82, 0.18)$ (see Example 3.10 on page 50 for details on combination and marginalization). Following a similar procedure, the posteriors of D and S given $L =$ yes can be computed to be $P(D \mid L = \text{yes}) = (0.53, 0.47)$ and $P(S \mid L = \text{yes}) = (0.51, 0.49)$. Thus, according to the model, the tree being sick is the most likely cause of the loss of leaves. □

The specification of a conditional probability distribution $P(X_v \mid X_{\text{pa}(v)})$ can be a labor-intensive knowledge acquisition task as the number of parameters grows exponentially with the size of $\text{dom}(X_{\text{fa}(v)})$, where $\text{fa}(v) = \text{pa}(v) \cup \{v\}$. Different techniques can be used to simplify the knowledge acquisition task, assumptions can be made, or the parameters can be estimated from data.

The complexity of a Bayesian network is defined in terms of the family $\text{fa}(v)$ with the largest state space size $\|X_{\text{fa}(v)}\| \triangleq |\text{dom}(X_{\text{fa}(v)})|$. As the state space size of a family of variables grows exponentially with the size of the family, we seek to reduce the size of the parent sets to a minimum. Another useful measure of the complexity of a Bayesian network is the number of cycles and the length of cycles in its graph.

Definition 4.2. A Bayesian network $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P})$ is minimal if and only if for every variable $X_v \in \mathcal{X}$ and for every parent $Y \in X_{\text{pa}(v)}$, X_v is not independent of Y given $X_{\text{pa}(v)} \setminus \{Y\}$.

Definition 4.2 says that the parent set $X_{\text{pa}(v)}$ of X_v should be limited to the set of variables with a direct impact on X_v.

Example 4.2 (Chest Clinic (Lauritzen & Spiegelhalter 1988)). A physician at a chest clinic wants to diagnose her patients with respect to three diseases based on observations of symptoms and possible causes of the diseases. The fictitious qualitative medical knowledge is the following.

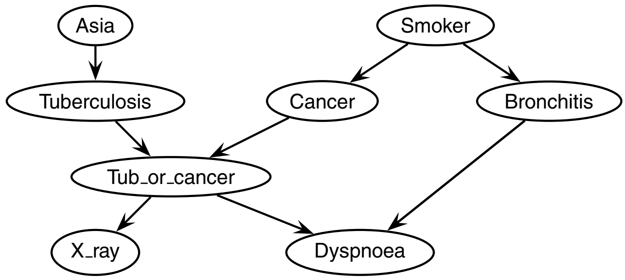


Fig. 4.2 A graph specifying the independence and dependence relations of the Asia example

The physician is trying to diagnose a patient who may be suffering from one or more of *tuberculosis*, *lung cancer*, or *bronchitis*. *Shortness of breath* (dyspnoea) may be due to tuberculosis, lung cancer, bronchitis, none of them, or more than one of them. *A recent visit to Asia* increases the chances of tuberculosis, while *smoking* is known to be a risk factor for both lung cancer and bronchitis. *The results of a single chest X-ray* do not discriminate between lung cancer and tuberculosis, as neither does the presence nor absence of dyspnoea.

From the description of the situation, it is clear that there are three possible diseases to consider (lung cancer, tuberculosis, and bronchitis). The three diseases produce three variables Tuberculosis (T), Cancer (L), and Bronchitis (B) with states no and yes. These variables are the targets of the reasoning and may, for this reason, be referred to as *hypothesis variables*.The diseases may be manifested in two symptoms (results of the X-ray and shortness of breath). The two symptoms produce two variables X_ray (X), and Dyspnoea (D) with states no and yes. In addition, there are two causes or risk factors (smoking and a visit to Asia) to consider. The two risk factors produce variables Asia (A) and Smoker (S) with states no and yes.

An acyclic, directed graph, $\mathcal{G}$, encoding the above medical qualitative knowledge is shown in Fig. 4.2, where the variable Tub_or_cancer (E) is a mediating variable (modeling trick, see Sect. 6.2.2 on page 152) specifying whether or not the patient has tuberculosis or lung cancer (or both).

Using the structure of $\mathcal{G}$, we may perform an analysis of dependence and independence properties between variables in order to ensure that the qualitative structure encodes the domain knowledge correctly. This analysis would be based on an application of the d-separation criterion.

Figure 4.2 only presents the qualitative structure $\mathcal{G}$ (and the variables) of $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P})$. In order to have a fully specified Bayesian network, it is necessary to specify the quantitative part, $\mathcal{P}$, too.

The quantitative domain knowledge is specified in the following set of (conditional) probability distributions $P(A) = (0.99, 0.01)$, $P(S) = (0.5, 0.5)$, and the remaining conditional probability distributions, except $P(E \mid L, T)$, are shown in Tables 4.2 and 4.3.

Table 4.2 The conditional probability distributions $P(L \mid S)$, $P(B \mid S)$, $P(T \mid A)$, and $P(X \mid E)$

$P(L \mid S)$	$S = \text{no}$	$S = \text{yes}$
$L = \text{no}$	0.99	0.9
$L = \text{yes}$	0.01	0.1

$P(B \mid S)$	$S = \text{no}$	$S = \text{yes}$
$B = \text{no}$	0.7	0.4
$B = \text{yes}$	0.3	0.6

$P(T \mid A)$	$A = \text{no}$	$A = \text{yes}$
$T = \text{no}$	0.99	0.95
$T = \text{yes}$	0.01	0.05

$P(X \mid E)$	$E = \text{no}$	$E = \text{yes}$
$X = \text{no}$	0.95	0.02
$X = \text{yes}$	0.05	0.98

Table 4.3 The conditional probability distribution $P(D \mid B, E)$

	$B = \text{no}$		$B = \text{yes}$	
	$E = \text{no}$	$E = \text{yes}$	$E = \text{no}$	$E = \text{yes}$
$D = \text{no}$	0.9	0.3	0.2	0.1
$D = \text{yes}$	0.3	0.7	0.8	0.9

Table 4.4 Posterior distributions of the disease variables given various evidence scenarios

ε	$P(B = \text{yes} \mid \varepsilon)$	$P(L = \text{yes} \mid \varepsilon)$	$P(T = \text{yes} \mid \varepsilon)$
$\emptyset$	0.45	0.055	0.01
$\{S = \text{yes}\}$	0.6	0.1	0.01
$\{S = \text{yes}, D = \text{yes}\}$	0.88	0.15	0.015
$\{S = \text{yes}, D = \text{yes}, X = \text{yes}\}$	0.71	0.72	0.08

The conditional probability table of the random variable E can be generated from a mathematical expression. From our domain knowledge of the diagnosis problem, we know that E represents the disjunction of L and T. That is, E represents whether or not the patient has tuberculosis or lung cancer. From this, we can express E as $E = T \vee L$. This produces the conditional probability $P(E = \text{yes} \mid L = l, T = t) = 1$ whenever l or t is yes and 0 otherwise.

We will, in a later section, consider in more detail how to build mathematical expressions for the generation of conditional probability distributions (see Sect. 6.5.3 on page 180).

Using the Bayesian network model just developed, we may compute the posterior probability of the three diseases given various subsets of evidence on the causes and symptoms as shown in Table 4.4. □

4.1.2 *Conditional Linear Gaussian Bayesian Networks*

Up until now, we have considered Bayesian networks over discrete random variables only. However, there are many reasons for extending our considerations to include continuous variables. In this section we will consider Bayesian networks consisting of both continuous and discrete variables. For reasons to become clear later, we restrict our attention to the case of conditional linear Gaussian (also known as normal) distributions and the case of conditional linear Gaussian Bayesian networks. We refer to a conditional linear Gaussian Bayesian network as a CLG Bayesian network.

A CLG Bayesian network $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P}, \mathcal{F})$ consists of an acyclic, directed graph $\mathcal{G} = (V, E)$, a set of conditional probability distributions $\mathcal{P}$, and a set of density functions $\mathcal{F}$. There will be one conditional probability distribution for each discrete random variable X of $\mathcal{X}$ and one density function for each continuous random variable Y of $\mathcal{X}$.

A CLG Bayesian network specifies a distribution over a mixture of discrete and continuous variables (Lauritzen 1992b, Lauritzen & Jensen 2001). The variables, $\mathcal{X}$, are partitioned into the set of continuous variables, $\mathcal{X}_\Gamma$, and the set of discrete variables, $\mathcal{X}_\Delta$. Each node of $\mathcal{G}$ represents either a discrete random variable with a finite set of mutually exclusive and exhaustive states or a continuous random variable with a conditional linear Gaussian distribution conditional on the configuration of its discrete parent variables. This implies an important constraint on the structure of $\mathcal{G}$, namely, that a discrete random variable X_v may only have discrete parents, that is, $X_{\text{pa}(v)} \subseteq \mathcal{X}_\Delta$ for any $X_v \in \mathcal{X}_\Delta$.

Any Gaussian distribution function can be specified by its mean and variance parameter. As mentioned above, we consider the case where a continuous random variable can have a single Gaussian distribution function for each configuration of its discrete parent variables. If a continuous variable has one or more continuous variables as parents, the mean may depend linearly on the state of the continuous parent variables. Continuous parent variables of discrete variables are disallowed.

A random variable, X, has a continuous distribution if there exists a nonnegative function p, defined on the real line, such that for any interval J,

$$P(X \in J) = \int_J p(x)dx,$$

where the function p is the probability density function of X (DeGroot 1986). The probability density function of a *Gaussian* (or *normal*)- *distributed variable*, X, with a mean value, μ, and a positive variance, σ^2, is (i.e., $X \sim \mathbb{N}(\mu, \sigma^2)$ or $\mathcal{L}(X) = \mathbb{N}(\mu, \sigma^2)$)

$$p(x; \mu, \sigma^2) = \mathbb{N}(\mu, \sigma^2) = \frac{1}{\sqrt{(2\pi\sigma^2)}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right],$$

where $x \in \mathbb{R}$.[1]

A continuous random variable, X, has a *conditional linear Gaussian distribution* (or CLG distribution), conditional on the configuration of the parent variables ($Z \subseteq \mathcal{X}_\Gamma, I \subseteq \mathcal{X}_\Delta$) if

$$\mathcal{L}(X \mid Z = z, I = i) = \mathbb{N}(A(i) + B(i)^T z, C(i)), \tag{4.2}$$

[1] $\mathcal{L}(X)$ should be read as "the law of X."

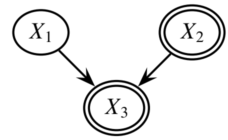


Fig. 4.3 CLG Bayesian network with X_1 discrete and X_2 and X_3 continuous

where A is a table of mean values (one value for each configuration i of the discrete parent variables I), B is a table of regression coefficient vectors (one vector for each configuration i of I with one regression coefficient for each continuous parent variable), and C is a table of variances (one for each configuration i of I). Notice that the mean value $A(i) + B(i)^T z$ of X depends linearly on the values of the continuous parent variables Z, while the variance is independent of Z. We allow for the situation where the variance is zero such that deterministic relations between continuous variables can be represented.

The quantitative part of a CLG Bayesian network consists of a conditional probability distribution for each $X \in \mathcal{X}_\Delta$ and a conditional Gaussian distribution for each $X \in \mathcal{X}_\Gamma$. For each $X \in \mathcal{X}_\Gamma$ with discrete parents, I, and continuous parents, Z, we need to specify a one-dimensional Gaussian probability distribution for each configuration i of I as shown in (4.2).

Definition 4.3. A CLG Bayesian network $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P}, \mathcal{F})$ consists of:

- A DAG $\mathcal{G} = (V, E)$ with nodes V and directed links E
- A set of random variables, $\mathcal{X}$, represented by the nodes of $\mathcal{G}$
- A set of conditional probability distributions, $\mathcal{P}$, containing one distribution, $P(X_v \mid X_{\text{pa}(v)})$, for each discrete random variable X_v
- A set of conditional linear Gaussian probability density functions, $\mathcal{F}$, containing one density function, $p(Y_v \mid X_{\text{pa}(v)})$, for each continuous random variable Y_v

The joint distribution over all the variables in a CLG Bayesian network has the form $P(\mathcal{X}_\Delta = i) * \mathbb{N}_{|\mathcal{X}_\Gamma|}(\mu(i), \sigma^2(i))$, where $\mathbb{N}_k(\mu, \sigma^2)$ denotes a k-dimensional Gaussian distribution. The chain rule of CLG Bayesian networks is

$$P(\mathcal{X}_\Delta = i) * \mathbb{N}_{|\mathcal{X}_\Gamma|}(\mu(i), \sigma^2(i)) = \prod_{v \in V_\Delta} P(i_v \mid i_{\text{pa}(v)}) * \prod_{w \in V_\Gamma} p(y_w \mid X_{\text{pa}(w)}),$$

for each configuration i of $\mathcal{X}_\Delta$.

Recall from Table 2.2 that in the graphical representation of a CLG Bayesian network, continuous variables are represented by double ovals.

Example 4.3 (CLG Bayesian Network). Figure 4.3 shows an example of the qualitative specification of a CLG Bayesian network, $\mathcal{N}$, with three variables, that is, $\mathcal{X} = \{X_1, X_2, X_3\}$, where $\mathcal{X}_\Delta = \{X_1\}$ and $\mathcal{X}_\Gamma = \{X_2, X_3\}$. Hence, $\mathcal{N}$ consists of a continuous random variable X_3 having one discrete random variable X_1 (binary with states false and true) and one continuous random variable X_2 as parents.

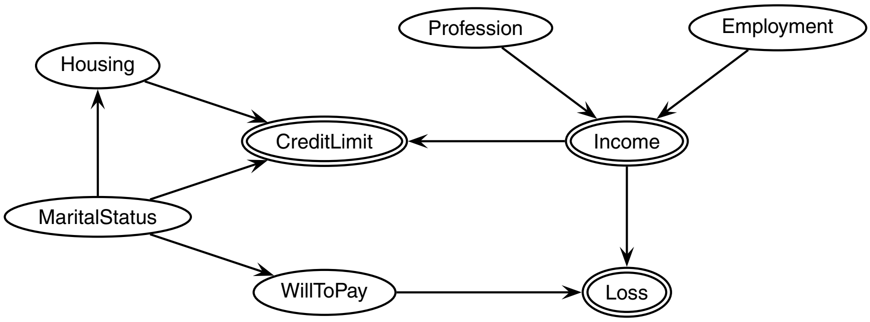


Fig. 4.4 CLG Bayesian network for credit account management

To complete the model, we need to specify the relevant conditional probability distribution and density functions. The quantitative specification could, for instance, consist of the following conditional linear Gaussian distribution functions for X_3:

$$\mathcal{L}(X_3 \,|\, \mathsf{false}, x_2) = \mathbb{N}(-5 + (-2 * x_2), 1.1)$$
$$\mathcal{L}(X_3 \,|\, \mathsf{true}, x_2) = \mathbb{N}(5 + (2 * x_2), 1.2).$$

The quantitative specification is completed by letting X_2 have a standard normal distribution (i.e., $X_2 \sim \mathbb{N}(0, 1)$) and $P(X_1) = (0.75, 0.25)$.

The qualitative and quantitative specifications complete the specification of $\mathcal{N}$. The joint distribution induced by $\mathcal{N}$ is

$$P(X_1 = \mathsf{false}) * p(X_2, X_3) = 0.75 * \mathbb{N}\left(\begin{pmatrix} 0 \\ -5 \end{pmatrix}, \begin{pmatrix} 1 & 10 \\ 10 & 5.1 \end{pmatrix}\right),$$

$$P(X_1 = \mathsf{true}) * p(X_2, X_3) = 0.25 * \mathbb{N}\left(\begin{pmatrix} 0 \\ 5 \end{pmatrix}, \begin{pmatrix} 1 & 10 \\ 10 & 5.2 \end{pmatrix}\right).$$

□

Determining the joint distribution induced by $\mathcal{N}$ requires a series of nontrivial computations. We refer the reader to the next chapter for a brief treatment of inference in CLG Bayesian networks. A detailed treatment of these computations is beyond the scope of this book.

Example 4.4 (Adapted from Lauritzen (1992a)). Consider a banker monitoring her clients in order to limit future loss from each client account. The task of the banker is to identify clients who may have problems repaying their loans by predicting potential future loss originating from each individual customer based on demographic information and credit limit.

Figure 4.4 shows a simple CLG Bayesian network model for this scenario. Loss is a linear function of variables Income (I) given variable WillToPay (W).

CreditLimit (C) is a linear function of Income given Housing (H) and Marital Status (M). In addition MaritalStatus is also a causal factor of Housing and WillToPay, while Profession and Employment are causal factors of Income.

With the model, the banker may enter observations on each client and compute an expected loss for that client. The model may be extended to include various risk indicators and controls in order to facilitate a scenario-based analysis on each client. □

The reason for restricting our attention to the case of conditional linear Gaussian distributions is that only for this case is exact probabilistic inference feasible by local computations. For most other cases, it is necessary to resort to approximate algorithms.

4.2 Decision Making Under Uncertainty

The framework of influence diagrams (Howard & Matheson 1981) is an effective modeling framework for representation and analysis of (Bayesian) decision making under uncertainty. Influence diagrams provide a natural representation for capturing the semantics of decision making with a minimum of clutter and confusion for the decision maker (Shachter & Peot 1992). Solving a decision problem amounts to (1) determining an (optimal) strategy that maximizes the expected utility for the decision maker and (2) computing the expected utility of adhering to this strategy.

An influence diagram is a type of causal model that differs from a Bayesian network. A Bayesian network is a probabilistic network for belief update, whereas an influence diagram is a probabilistic network for reasoning about decision making under uncertainty. An influence diagram is a graphical representation of a decision problem involving a sequence of interleaved decisions and observations. Similar to Bayesian networks, an influence diagram is a compact and intuitive probabilistic knowledge representation (a probabilistic network). It consists of a graphical representation describing dependence relations between entities of a problem domain, points in time where decisions are to be made, and a precedence ordering specifying the order on decisions and observations. It also consists of a quantification of the strengths of the dependence relations and the preferences of the decision maker. As such, an influence diagram can be considered as a Bayesian network augmented with decision variables, utility functions specifying the preferences of the decision maker, and a precedence ordering.

As decision makers we are interested in making the best possible decisions given our model of the problem domain. Therefore, we associate utilities with state configurations of the network. These utilities are represented by *utility functions* (also known as *value functions*). Each utility function associates a utility value with each configuration of its domain variables. The objective of decision analysis is to identify the decision options that produce the highest expected utility.

By making decisions, we influence the probabilities of the configurations of the network. To identify the decision option with the highest expected utility, we compute the expected utility of each decision alternative. If A is a decision variable with options $a_1, \dots, a_m$, H is a hypothesis with states $h_1, \dots, h_n$, and ε is a set of observations in the form of evidence, then we can compute the utility of each outcome of the hypothesis and the expected utility of each action. The utility of an outcome (a_i, h_j) is $U(a_i, h_j)$ where $U(\cdot)$ is our utility function. The expected utility of performing action a_i is

$$\mathrm{EU}(a_i) = \sum_j U(a_i, h_j) P(h_j \mid \varepsilon),$$

where $P(\cdot)$ represents our belief in H given ε. The utility function $U(\cdot)$ encodes the preferences of the decision maker on a numerical scale.

We shall choose the alternative with the highest expected utility; this is known as the (maximum) expected utility principle. Choosing the action, which maximizes the expected utility, amounts to selecting an option a^* such that

$$a^* = \arg\max_{a \in A} \mathrm{EU}(a).$$

There is an important difference between observations and actions. An observation of an event is passive in the sense that we assume that an observation does not affect the state of the world, whereas the decision on an action is active in the sense that an action enforces a certain event. The event enforced by a decision may or may not be included in the model depending on whether or not the event is relevant for the reasoning. If the event enforced by an action A is represented in our model, then A is referred to as an intervening action, otherwise it is referred to as a nonintervening action.

4.2.1 Discrete Influence Diagrams

An (discrete) influence diagram $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P}, \mathcal{U})$ is a four-tuple consisting of a set, $\mathcal{X}$, of discrete random variables and discrete decision variables, an acyclic, directed graph $\mathcal{G}$, a set of conditional probability distributions $\mathcal{P}$, and a set of utility functions $\mathcal{U}$. The acyclic, directed graph, $\mathcal{G} = (V, E)$, contains nodes representing random variables, decision variables, and utility functions (also known as value or utility nodes).

Each decision variable, D, represents a specific point in time under the model of the problem domain where the decision maker has to make a decision. The decision options or alternatives are the states $(d_1, \dots, d_n)$ of D where $n = \|D\|$. The usefulness of each decision option is measured by the local utility functions associated with D or one of its descendants in $\mathcal{G}$. Each local utility function $u(X_{\mathrm{pa}(v)}) \in \mathcal{U}$, where $v \in V_U$ is a utility node, represents an additive contribution to the total utility function $u(\mathcal{X})$ in $\mathcal{N}$. Thus, the total utility function is the sum of all the utility functions in the influence diagram, that is, $u(\mathcal{X}) = \sum_{v \in V_U} u(X_{\text{pa}(v)})$.

Definition 4.4. A (discrete) influence diagram $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P}, \mathcal{U})$ consists of:

- A DAG $\mathcal{G} = (V, E)$ with nodes, V, and directed links, E, encoding dependence relations and information precedence including a total order on decisions
- A set of discrete random variables, $\mathcal{X}_C$, and discrete decision variables, $\mathcal{X}_D$, such that $\mathcal{X} = \mathcal{X}_C \cup \mathcal{X}_D$ represented by nodes of $\mathcal{G}$
- A set of conditional probability distributions, $\mathcal{P}$, containing one distribution, $P(X_v \,|\, X_{\text{pa}(v)})$, for each discrete random variable X_v
- A set of utility functions, $\mathcal{U}$, containing one utility function, $u(X_{\text{pa}(v)})$, for each node v in the subset $V_U \subset V$ of utility nodes

An influence diagram supports the representation and solution of sequential decision problems with multiple local utility functions under the *no-forgetting assumption* (Howard & Matheson 1981), that is, perfect recall is assumed of all observations and decisions made in the past.

An influence diagram, $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P}, \mathcal{U})$, should be constructed such that one can determine exactly which variables are known prior to making each decision. If the state of a variable $X_v \in \mathcal{X}_C$ will be known at the time of making a decision $D_w \in \mathcal{X}_D$, this will (probably) have an impact on the choice of alternative at D. An observation on X_v made prior to decision D_w is represented in $\mathcal{N}$ by making v a parent of w in $\mathcal{G}$. If v is a parent of w in $\mathcal{G} = (V, E)$ (i.e., $(v, w) \in E$, implying $X_v \in X_{\text{pa}(w)}$), then it is assumed that X_v is observed prior to making the decision represented by D_w. The link (v, w) is then referred to as an *informational link*.

In an (perfect recall) influence diagram, there must also be a total order on the decision variables $\mathcal{X}_D = \{D_1, \ldots, D_n\} \subseteq \mathcal{X}$. This is referred to as the *regularity* constraint. That is, there can be only one sequence in which the decisions are made. We add informational links to specify a total order $(D_1, \ldots, D_n)$ on $\mathcal{X}_D = \{D_1, \ldots, D_n\}$. There need only be a directed path from one decision variable to the next one in the decision sequence in order to enforce a total order on the decisions.

In short, a link, (w, v), into a node representing a random variable, X_v, denotes a possible probabilistic dependence relation of X_v on Y_w, while a link from a node representing a variable, X, into a node representing a decision variable, D, denotes that the state of X is known when decision D is to be made. A link, (w, v), into a node representing a local utility function, u, denotes functional dependence of u on $X_v \in \mathcal{X}$.

The chain rule of influence diagrams is

$$\text{EU}(\mathcal{X}) = \prod_{X_v \in \mathcal{X}_C} P(X_v \,|\, X_{\text{pa}(v)}) \sum_{w \in V_U} u(X_{\text{pa}(w)}).$$

An influence diagram is a compact representation of a joint expected utility function.

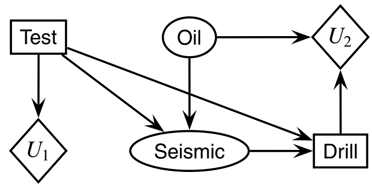


Fig. 4.5 The oil wildcatter network

Table 4.5 The conditional probability distribution $P(\text{Seismic} \mid \text{Oil}, \text{Test} = \text{yes})$

	Seismic		
Oil	diffuse	open	closed
dry	0.6	0.3	0.1
wet	0.3	0.4	0.3
soaking	0.1	0.4	0.5

In the graphical representation of an influence diagram, utility functions are represented by rhombuses (diamond-shaped nodes), whereas decision variables are represented as rectangles, see Table 2.2.

Example 4.5 (Oil Wildcatter (Raiffa 1968)). Consider the fictitious example of an oil wildcatter about to decide whether or not to drill for oil at a specific site.

The situation of the oil wildcatter is the following. The oil wildcatter must decide either to *drill* or *not to drill*. He is uncertain whether the *hole* will be dry, wet, or soaking with oil. The wildcatter could *take seismic soundings* that will help determine the *geological structure* of the site. The soundings will give a closed reflection pattern (indication of much oil), an open pattern (indication of some oil), or a diffuse pattern (almost no hope of oil).

The qualitative domain knowledge extracted from the above description can be formulated as the DAG shown in Fig. 4.5. The state spaces of the variables are as follows $\text{dom}(\text{Drill}) = (\text{no}, \text{yes})$, $\text{dom}(\text{Oil}) = (\text{dry}, \text{wet}, \text{soaking})$, $\text{dom}(\text{Seismic}) = (\text{closed}, \text{open}, \text{diffuse})$, and $\text{dom}(\text{Test}) = (\text{no}, \text{yes})$.

Figure 4.5 shows how the qualitative knowledge of the example can be compactly specified in the structure of an influence diagram $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P}, \mathcal{U})$.

The quantitative probabilistic knowledge as defined by the structure of $\mathcal{G}$ consists of $P(\text{Oil})$ and $P(\text{Seismic} \mid \text{Oil}, \text{Test})$, while the quantitative utility knowledge consists of $U_1(\text{Test})$ and $U_2(\text{Drill}, \text{Oil})$.

The cost of testing is $10k$, whereas the cost of drilling is $70k$. The utility of drilling is $0k$, $120k$, and $270k$ for a dry, wet, and soaking hole, respectively. Hence, $U_1(\text{Test}) = (0, -10)$ and $U_2(\text{Drill} = \text{yes}, \text{Oil}) = (-70, 50, 200)$. The test result Seismic depends on the amount of oil represented by variable Oil as specified in Table 4.5. The prior belief of the oil wildcatter on the amount of oil at the site is $P(\text{Oil}) = (0.5, 0.3, 0.2)$.

This produces a completely specified influence diagram representation of the oil wildcatter decision problem. The decision strategy of the oil wildcatter will be considered in Example 4.7 on the following page. □

As a consequence of the total order on decisions and the set of informational links, the set of discrete random variables and decision variables is subjected to a partial ordering. The random variables are partitioned into disjoint *information sets* $\mathcal{I}_0, \ldots, \mathcal{I}_n$ (i.e., $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset$ for $i \neq j$) relative to the decision variables specifying the precedence order. The information set $\mathcal{I}_i$ is the set of variables observed after decision D_i and before decision D_{i+1}. The partition induces a partial ordering, $\prec$, on the variables $\mathcal{X}$. The set of variables observed between decisions D_i and D_{i+1} precedes D_{i+1} and succeeds D_i in the ordering

$$\mathcal{I}_0 \prec D_1 \prec \mathcal{I}_1 \prec \cdots \prec D_n \prec \mathcal{I}_n,$$

where $\mathcal{I}_0$ is the set of discrete random variables observed before the first decision, $\mathcal{I}_i$ is the set of discrete random variables observed after making decision D_i and before making decision D_{i+1}, for all $i = 1, \ldots, n-1$, and $\mathcal{I}_n$ is the set of discrete random variables never observed or observed after the last decision D_n has been made. If the influence diagram is not constructed or used according to this constraint, the computed expected utilities will (of course) not be correct.

Example 4.6 (Partial Order of Information Set). The total order on decisions and the informational links of Example 4.5 on the preceding page induce the following partial order:

$$\{\} \prec \mathsf{Test} \prec \{\mathsf{Seismic}\} \prec \mathsf{Drill} \prec \{\mathsf{Oil}\}.$$

This partial order turns out to be a total order. In general, this is not the case. The total order specifies the flow of information in the decision problem. No observations are made prior to the decision on whether or not to Test. After testing and before deciding on whether or not to Drill, the oil wildcatter will make an observation on Seismic, that is, the test result is available before the Drill decision. After drilling Oil is observed. □

To solve an influence diagram $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P}, \mathcal{U})$ with decision variables, $\mathcal{X}_D$, is to identify an optimal strategy, $\hat{\Delta}$, over $\mathcal{X}_D$ maximizing the expected utility for the decision maker and to compute the *(maximum) expected utility* $\mathrm{EU}(\hat{\Delta})$ of $\hat{\Delta}$. A *strategy*, Δ, is an ordered set of decision policies $\Delta = (\delta_1, \ldots, \delta_n)$ including one decision policy for each decision $D \in \mathcal{X}_D$. An *optimal strategy* $\hat{\Delta} = (\hat{\delta}_1, \ldots, \hat{\delta}_n)$, maximizes the expected utility over all possible strategies, that is, it satisfies

$$\mathrm{EU}(\hat{\Delta}) \geq \mathrm{EU}(\Delta),$$

for all strategies Δ.

The *decision history* of D_i, denoted $\mathcal{H}(D_i)$, is the set of previous decisions and their parent variables

$$\mathcal{H}(D_i) = \bigcup_{j=1}^{i-1} (\{D_j\} \cup X_{\mathrm{pa}(v_j)}) = \{D_1, \ldots, D_{i-1}\} \cup \bigcup_{j=0}^{i-2} \mathcal{I}_j,$$

where v_j denotes the node that represents D_j.

The *decision past* of D_j, denoted $\mathcal{I}(D_i)$, is the set of its parent variables and the decision history $\mathcal{H}(D_i)$

$$\begin{aligned}\mathcal{I}(D_i) &= X_{\mathrm{pa}(v_i)} \cup \mathcal{H}(D_i) \\ &= X_{\mathrm{pa}(v_i)} \cup \bigcup_{j=1}^{i-1} (\{D_j\} \cup X_{\mathrm{pa}(v_j)}) \\ &= \{D_1, \ldots, D_{i-1}\} \cup \bigcup_{j=1}^{i-1} \mathcal{I}_j.\end{aligned}$$

Hence, $\mathcal{I}(D_i) \setminus \mathcal{H}(D_i) = \mathcal{I}_{i-1}$ are the variables observed between D_{i-1} and D_i.

The *decision future* of D_i, denoted $\mathcal{F}(D_i)$ is the set of its descendant variables

$$\begin{aligned}\mathcal{F}(D_i) &= \mathcal{I}_i \cup \left(\bigcup_{j=i+1}^{n} (\{D_j\} \cup X_{\mathrm{pa}(v_j)}) \right) \\ &= \{D_{i+1}, \ldots, D_n\} \cup \bigcup_{j=i}^{n} \mathcal{I}_j.\end{aligned}$$

A *policy* δ_i is a mapping from the information set $\mathcal{I}(D_i)$ of D_i to the state space $\mathrm{dom}(D_i)$ of D_i such that $\delta_i : \mathcal{I}(D_i) \to \mathrm{dom}(D_i)$. A policy for decision D specifies the (optimal) action for the decision maker for all possible observations made prior to making decision D.

It is only necessary to consider δ_i as a function from relevant observations on $\mathcal{I}(D_i)$ to $\mathrm{dom}(D_i)$, that is, observations with an unblocked path to a utility descendant of D_i. Relevance of an observation with respect to a decision is defined in Sect. 4.2.3 on page 93.

Example 4.7 (Oil Wildcatter Strategy). After solving the influence diagram, we obtain an optimal strategy $\hat{\Delta} = \{\hat{\delta}_{\mathsf{Test}}, \hat{\delta}_{\mathsf{Drill}}\}$. Hence, the optimal strategy $\hat{\Delta}$ (we show how to identify the optimal strategy for this example in Example 5.11 on page 129) consists of a policy $\hat{\delta}_{\mathsf{Test}}$ for Test and a policy $\hat{\delta}_{\mathsf{Drill}}$ for Drill given Test and Seismic

$$\hat{\delta}_{\mathsf{Test}} = \mathsf{yes}$$

$$\hat{\delta}_{\mathsf{Drill}}(\mathsf{Seismic}, \mathsf{Test}) = \begin{cases} \mathsf{yes} & \mathsf{Seismic} = \mathsf{closed}, \mathsf{Test} = \mathsf{no} \\ \mathsf{yes} & \mathsf{Seismic} = \mathsf{open}, \mathsf{Test} = \mathsf{no} \\ \mathsf{yes} & \mathsf{Seismic} = \mathsf{diffuse}, \mathsf{Test} = \mathsf{no} \\ \mathsf{yes} & \mathsf{Seismic} = \mathsf{closed}, \mathsf{Test} = \mathsf{yes} \\ \mathsf{yes} & \mathsf{Seismic} = \mathsf{open}, \mathsf{Test} = \mathsf{yes} \\ \mathsf{no} & \mathsf{Seismic} = \mathsf{diffuse}, \mathsf{Test} = \mathsf{yes} \end{cases}$$

The policy for Test says that we should always test, while the policy for Drill says that we should drill except when the test produces a diffuse pattern indicating almost no hope of oil. □

An intervening decision D of an influence diagram is a decision that may impact the state or value of another variable X represented in the model. In order for D to potentially impact the value of X, X must be a descendant of D in G. This can be realized by considering the d-separation criterion (consider the information blocking properties of the converging connection) and the set of evidence available when making the decision D. Consider, for instance, the influence diagram shown in Fig. 4.5. The decision Test is an intervening decision as it impacts the value of Seismic. It cannot, however, impact the value of Oil as Oil is a non-descendant of Test, and we have no *down-stream* evidence when making the decision on Test. Since decision D may only have a potential impact on its descendants, the usefulness of D can only be measured by the utility descendants of D.

A total ordering on the decision variables is usually assumed. This assumption can, however, be relaxed. Nielsen & Jensen (1999) describe when decision problems with only a partial ordering on the decision variables are *well defined*. In addition, the limited memory influence diagram (Lauritzen & Nilsson 2001), see Sect. 4.2.3, and the unconstrained influence diagram (Vomlelová & Jensen 2002) support the use of unordered decision variables.

Example 4.8 (Apple Jack). We consider once again the problems of Apple Jack from Example 4.1 on page 71. A Bayesian network for reasoning about the causes of the apple tree losing its leaves was shown in Fig. 4.1 on page 72.

We continue the example by assuming that Apple Jack wants to decide whether or not to invest resources in giving the tree some treatment against a possible disease. Since this involves a decision through time, we have to extend the Bayesian network to capture the impact of the treatment on the development of the disease. We first add three variables similar to those already in the network. The new variables Sick*, Dry*, and Loses* correspond to the original variables, except that they represent the situation at the time of harvest, that is, after the treatment decision. These variables have been added in Fig. 4.6.

The additional variables have the same states as the original variables: Sick*, Dry*, and Loses* all have states no and yes. In the extended model, we expect a causal influence from the original Sick variable on the Sick* variable and from the original Dry variable on the Dry* variable. The reason is the following. If, for

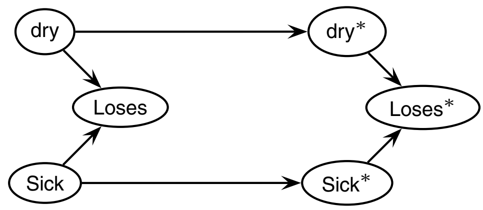


Fig. 4.6 We model the system at two different points in time (before and after a decision) by replicating the structure

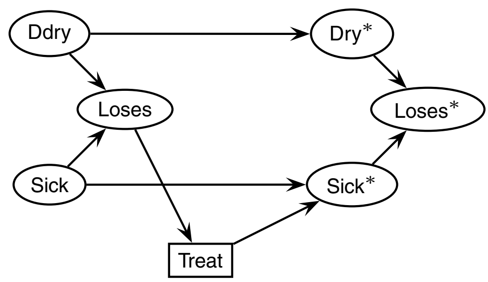


Fig. 4.7 Addition of a decision variable for treatment to the Bayesian network in Fig. 4.6

example, we expect the tree to be sick now, then this is also likely to be the case in the future and especially at the time of harvest. Of course, the strength of the influence depends on how far out in the future we look. Perhaps one could also have a causal influence from Loses on Loses*, but we have chosen not to model such a possible dependence relation in this model.

Apple Jack may try to heal the tree with a treatment to get rid of the possible disease. If he expects that the loss of leaves is caused by drought, he might save his money and just wait for rain. The action of giving the tree a treatment is now added as a decision variable to the Bayesian network, which will then no longer be a Bayesian network. Instead, it becomes the influence diagram shown in Fig. 4.7.

The treat decision variable has the states no and yes. There is a causal link (Treat, Sick*) from the decision Treat to Sick* as we expect the treatment to have a causal impact on the future health of the tree. There is an informational link from Loses to Treat as we expect Apple Jack to observe whether or not the apple tree is losing its leaves prior to making the decision on treatment.

We need to specify the utility functions enabling us to compute the expected utility of the decision options. This is done by adding utility functions to the influence diagram. Each utility function will represent a term of an additively decomposing utility function, and each term will contribute to the total utility. The utility functions are shown in Fig. 4.8.

The utility function C specifies the cost of the treatment, while utility function H specifies the reward of the harvest. The latter depends on the state of Sick*, indicating that the production of apples depends on the health of the tree.

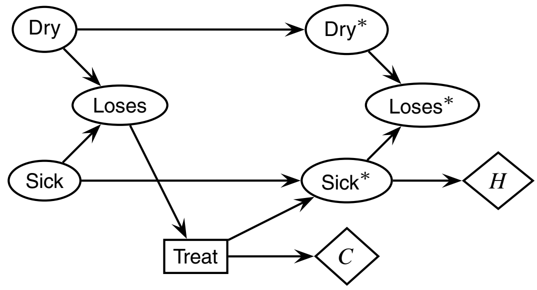


Fig. 4.8 A complete qualitative representation of the influence diagram used for decision making in Apple Jack's orchard

Table 4.6 The conditional probability distribution $P(\text{Sick}^* \mid \text{Treat}, \text{Sick})$

		Sick*	
Treat	Sick	no	yes
no	no	0.98	0.02
no	yes	0.01	0.99
yes	no	0.99	0.01
yes	yes	0.8	0.2

Table 4.7 The conditional probability distribution $P(\text{Dry}^* \mid \text{Dry})$

	Dry*	
Dry	no	yes
no	0.95	0.05
yes	0.4	0.6

Table 4.8 The conditional probability distribution $P(\text{Loses}^* \mid \text{Dry}^*, \text{Sick}^*)$

		Loses*	
Dry*	Sick*	no	yes
no	no	0.98	0.02
no	yes	0.1	0.9
yes	no	0.15	0.85
yes	yes	0.05	0.95

Figure 4.8 shows the complete qualitative representation of the influence diagram $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P}, \mathcal{U})$. To complete the quantitative representation as well, we need to specify the conditional probability distributions, $\mathcal{P}$, and utility functions, $\mathcal{U}$, of $\mathcal{N}$. Recall that a decision variable does not have any distribution. The appropriate probability distributions are specified in Tables 4.6–4.8.

If we have a healthy tree (Sick* is in state no), then Apple Jack will get an income of € 200, while if the tree is sick (Sick* is in state yes), his income is only € 30, that is, $H(\text{Sick}^*) = (200, 30)$. To treat the tree, he has to spend € 80, that is, $C(\text{Treat}) = (0, -80)$.

Since Dry* and Loses* are not relevant for the decision on whether or not to treat and since we do not care about their distribution, we remove them from our model producing the final model shown in Fig. 4.9. Variables Dry* and Loses* are in fact

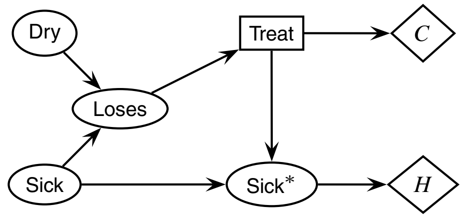


Fig. 4.9 A simplified influence diagram for the decision problem of Apple Jack

barren variables, see Sect. 3.3.4 on page 53. In an influence diagram, a variable is a barren variable when none of its descendants are utility nodes, and none of its descendants are ever observed.

The purpose of our influence diagram is to be able to determine the optimal strategy for Apple Jack. After solving $\mathcal{N}$, we obtain the following policy ($\delta_{\mathsf{Treat}} : \mathsf{Loses} \rightarrow \text{dom}(\mathsf{Treat})$) for Treat:

$$\delta_{\mathsf{Treat}}(\mathsf{Loses}) = \begin{cases} \mathsf{no} & \mathsf{Loses} = \mathsf{no} \\ \mathsf{yes} & \mathsf{Loses} = \mathsf{yes} \end{cases}$$

Hence, we should only treat the tree when it loses its leaves. In Sect. 5.2, we describe how to solve an influence diagram. □

Notice that since a policy is a mapping from all possible observations to decision options, it is sufficient to solve an influence diagram once. Hence, the computed strategy can be used by the decision maker each time she or he is faced with the decision problem.

Implications of Perfect Recall

As mentioned above, when using influence diagrams to represent decision problems, we assume perfect recall. This assumption states that at the time of any decision, the decision maker remembers all past decisions and all previously known information (as enforced by the informational links). This implies that a decision variable and all of its parent variables are informational parents of all subsequent decision variables. Due to this assumption, it is not necessary to include no-forgetting links in the DAG of the influence diagram as they—if missing—will implicitly be assumed present.

Example 4.9 (Jensen, Jensen & Dittmer (1994)). Let $\mathcal{N}$ be the influence diagram in Fig. 4.10 on the facing page. This influence diagram represents a decision problem involving four decisions D_1, D_2, D_3, and D_4 in that order.

From the structure of $\mathcal{N}$, the following partial ordering on the random and decision variables can be read:

$$\{B\} \prec D_1 \prec \{E, F\} \prec D_2 \prec \{\} \prec D_3 \prec \{G\} \prec D_4 \prec \{A, C, D, H, I, J, K, L\}.$$

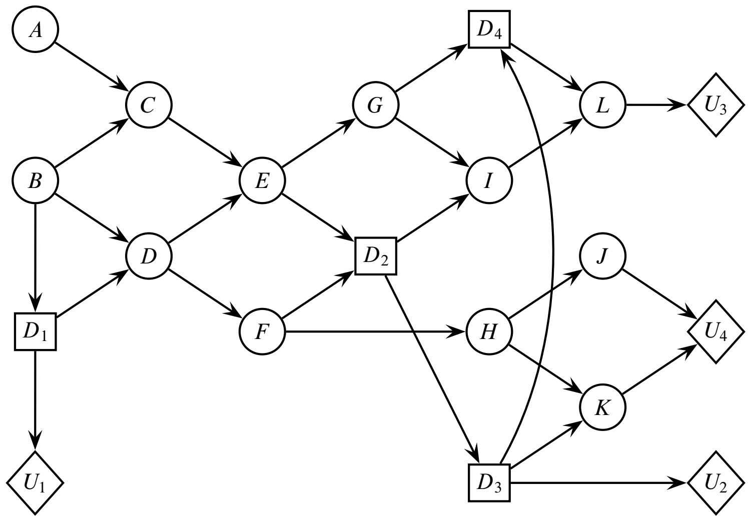


Fig. 4.10 An influence diagram representing the sequence of decisions D_1, D_2, D_3, D_4

This partial ordering specifies the flow of information in the decision problem represented by $\mathcal{N}$. Thus, the initial (relevant) information available to the decision maker is an observation of B. After making a decision on D_1, the decision maker observes E and F. After the observations of E and F, a decision on D_2 is made, and so on.

Notice that no-forgetting links have been left out, for example, there are no links from B to D_2, D_3, or D_4. These links are included in Fig. 4.11. The difference in complexity of reading the graph is apparent.

As this example shows, a rather informative analysis can be performed by reading only the structure of the graph of $\mathcal{N}$. □

4.2.2 Conditional LQG Influence Diagrams

Conditional linear–quadratic Gaussian influence diagrams combine conditional linear Gaussian Bayesian networks, discrete influence diagrams, and quadratic utility functions into a single framework supporting decision making under uncertainty with both continuous and discrete variables (Madsen & Jensen 2005).

Definition 4.5. A CLQG influence diagram $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P}, \mathcal{F}, \mathcal{U})$ consists of:

- A DAG $\mathcal{G} = (V, E)$ with nodes, V, and directed links, E, encoding dependence relations and information precedence including a total order on decisions

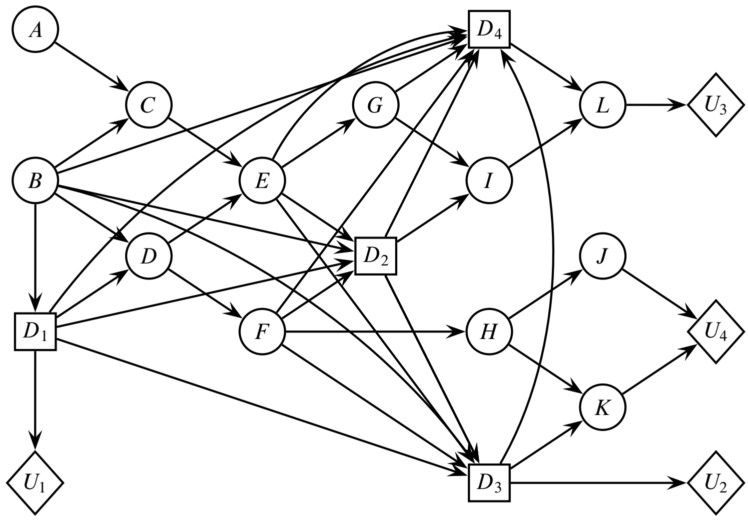


Fig. 4.11 The influence diagram of Fig. 4.10 with no-forgetting links

- A set of random variables, $\mathcal{X}_C$, and decision variables, $\mathcal{X}_D$, such that $\mathcal{X} = \mathcal{X}_C \cup \mathcal{X}_D$ represented by nodes of $\mathcal{G}$
- A set of conditional probability distributions, $\mathcal{P}$, containing one distribution, $P(X_v \mid X_{\text{pa}(v)})$, for each discrete random variable X_v
- A set of conditional linear Gaussian probability density functions, $\mathcal{F}$, containing one density function, $p(Y_w \mid X_{\text{pa}(w)})$, for each continuous random variable Y_w
- A set of linear–quadratic utility functions, $\mathcal{U}$, containing one utility function, $u(X_{\text{pa}(v)})$, for each node v in the subset $V_U \subset V$ of utility nodes

We refer to a conditional linear–quadratic Gaussian influence diagram as a CLQG influence diagram. The chain rule of CLQG influence diagrams is

$$\begin{aligned}\text{EU}(\mathcal{X}_\Delta = i, \mathcal{X}_\Gamma) &= P(\mathcal{X}_\Delta = i) * \mathbb{N}_{|\mathcal{X}_\Gamma|}\big(\mu(i), \sigma^2(i)\big) * \sum_{z \in V_U} u(X_{\text{pa}(z)}) \\ &= \prod_{v \in V_\Delta} P(i_v \mid i_{\text{pa}(v)}) * \prod_{w \in V_\Gamma} p(y_w \mid X_{\text{pa}(w)}) * \\ &\quad \sum_{z \in V_U} u(X_{\text{pa}(z)})\end{aligned}$$

for each configuration i of $\mathcal{X}_\Delta$.

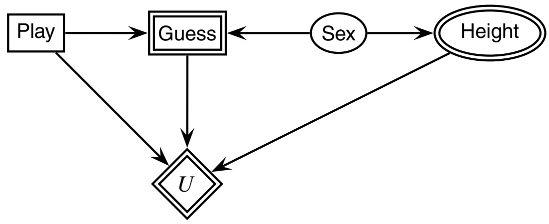


Fig. 4.12 A CLQG influence diagram for a simple guessing game

Recall that in the graphical representation of a CLQG influence diagram, continuous utility functions are represented by double rhombuses and continuous decision variables as double rectangles, see Table 2.2 on page 23 for an overview of vertex symbols.

A CLQG influence diagram is a compact representation of a joint expected utility function over continuous and discrete variables, where continuous variables are assumed to follow a linear Gaussian distribution conditional on a subset of discrete variables, while utility functions are assumed to be linear–quadratic in the continuous variables (and constant in the discrete). This may seem a severe assumption which could be limiting to the usefulness of the CLQG influence diagram. The assumption seems to indicate that all local utility functions specified in a CLQG influence diagram should be linear–quadratic in the continuous variables. This is not the case, however, as the following examples show. We will consider the assumption in more detail in Sect. 5.2 on solving decision models.

Example 4.10 (Guessing Game (Madsen & Jensen 2005)). Figure 4.12 illustrates a CLQG influence diagram, $\mathcal{N}$, representation of a simple guessing game with two decisions.

The first decision, represented by the discrete decision variable Play with states reward and Play, is to either accept an immediate reward or to play a game where you will receive a payoff determined by how good you are at guessing the height of a person, represented by the continuous random variable Height, based on knowledge about the sex of the person, represented by the discrete random variable Sex with states female and male. The second decision, represented by the real-valued decision variable Guess, is your guess on the height of the person given knowledge about the sex of the person.

The payoff is a constant (higher than the reward) minus the distance of your guess from the true height of the person measured as height minus guess squared.

To quantify $\mathcal{N}$, we need to specify a prior probability distribution for Sex, a conditional Gaussian distribution for Height and a utility function over Play, Guess, and Height. Assume the prior distribution on Sex is $P(\text{Sex}) = (0.5, 0.5)$, whereas the distribution for Height is

$$\mathcal{L}(\text{Height} \mid \text{female}) = \mathbb{N}(170, 400)$$
$$\mathcal{L}(\text{Height} \mid \text{male}) = \mathbb{N}(180, 100).$$

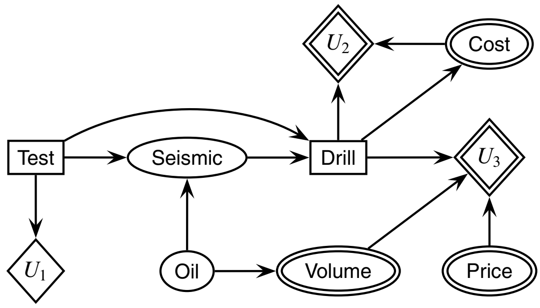


Fig. 4.13 A revised version of the oil wildcatter problem

We assume the average height of a female to be 170 cm with a standard deviation of 20 cm and average height of a male to be 180 cm with a standard deviation of 10 cm. The utility function over Play, Guess, and Height is

$$u(\mathsf{play}, d_2, h) = 150 - (h - d_2)^2$$
$$u(\mathsf{reward}, d_2, h) = 100.$$

We assume the immediate reward is 100. After solving $\mathcal{N}$, we obtain an optimal strategy $\Delta = \{\delta_{\mathsf{Play}}, \delta_{\mathsf{Guess}}\}$

$$\delta_{\mathsf{Play}} = \mathsf{play}$$
$$\delta_{\mathsf{Guess}}(\mathsf{play}, \mathsf{female}) = 170$$
$$\delta_{\mathsf{Guess}}(\mathsf{play}, \mathsf{male}) = 180.$$

The optimal strategy is to guess that the height of a female person is 170 cm and the height of a male person is 180 cm.

In this example the policy for Guess reduces to a constant for each configuration of its parent variables. In the general case, the policy for a continuous decision variable is a multilinear function in its continuous parent variables given the discrete parent variables. □

As another example of a CLQG influence diagram, consider a revised extension of the oil wildcatter problem of Raiffa (1968) (Example 4.5 on page 82). The revised Oil Wildcatter problem, which is further revised here, is due to Cobb & Shenoy (2004).

Example 4.11 (Oil Wildcatter (Madsen & Jensen 2005)). The network of the revised version of the Oil Wildcatter problem is shown in Fig. 4.13. First, the decision maker makes a decision on whether or not to perform a test Test of the geological structure of the site under consideration. When performed, this test will produce a test result, Seismic depending on the amount of oil Oil. Next, a decision Drill on whether or not to drill is made. There is a cost Cost associated with drilling, while the revenue is a function of oil volume Volume and oil price Price.

We assume the continuous random variables (i.e., cost of drilling, oil price, and oil volume) to follow (conditional) Gaussian distributions. The utility function can be stated in thousands of euros as $U_1(\text{Test} = \text{yes}) = -10$, $U_2(\text{Cost} = c, \text{Drill} = \text{yes}) = -c$, $U_3(\text{Volume} = v, \text{Price} = p, \text{Drill} = \text{yes}) = v * p$, and zero for the no drill and no test situations.

If the hole is dry, then no oil is extracted: $\mathcal{L}(\text{Volume} \mid \text{Oil} = \text{dry}) = \mathbb{N}(0, 0)$. If the hole is wet, then some oil is extracted: $\mathcal{L}(\text{Volume} \mid \text{Oil} = \text{wet}) = \mathbb{N}(6, 1)$. If the hole is soaking with oil, then a lot of oil is extracted: $\mathcal{L}(\text{Volume} \mid \text{Oil} = \text{soaking}) = \mathbb{N}(13.5, 4)$. The unit is a thousand barrels. The cost of drilling follows a Gaussian distribution $\mathcal{L}(\text{Cost} \mid \text{Drill} = \text{yes}) = \mathbb{N}(70, 100)$. We assume that the price of oil Price also follows a Gaussian distribution $\mathcal{L}(\text{Price}) = \mathbb{N}(20, 4)$.

Notice that the continuous utility functions U_2 and U_3 are not linear–quadratic in their continuous domain variables. □

4.2.3 *Limited Memory Influence Diagrams*

The framework of influence diagrams offers compact and intuitive models for reasoning about decision making under uncertainty. Two of the fundamental assumptions of the influence diagram representation are the no-forgetting assumption implying perfect recall of the past and the assumption of a total order on the decisions. The limited memory influence diagram framework (LIMID) (Lauritzen & Nilsson 2001) relaxes both of these fundamental assumptions.

Relaxing the no-forgetting and the total order (on decisions) assumptions largely increases the class of multistage decision problems that can be modeled. LIMIDs allow us to model more types of decision problems than the ordinary influence diagrams.

The graphical difference between the LIMID representation and the ordinary influence diagram representation is that the latter representation (as presented in this book) assumes some informational links to be implicitly present in the graph. This assumption is not made in the LIMID representation. For this reason, it is necessary to explicitly represent all information available to the decision maker at each decision.

The definition of a limited memory influence diagram is as follows.

Definition 4.6. A LIMID $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P}, \mathcal{U})$ consists of:

- A DAG $\mathcal{G} = (V, E)$ with nodes V and directed links E encoding dependence relations and information precedence.
- A set of random variables, $\mathcal{X}_C$, and discrete decision variables, $\mathcal{X}_D$, such that $\mathcal{X} = \mathcal{X}_C \cup \mathcal{X}_D$ represented by nodes of $\mathcal{G}$.
- A set of conditional probability distributions, $\mathcal{P}$, containing one distribution, $P(X_v \mid X_{\text{pa}(v)})$, for each discrete random variable X_v.
- A set of utility functions, $\mathcal{U}$, containing one utility function, $u(X_{\text{pa}(v)})$, for each node v in the subset $V_U \subset V$ of utility nodes.

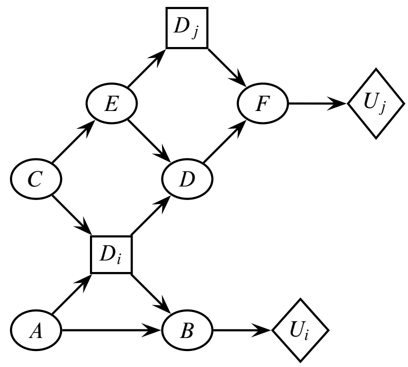


Fig. 4.14 A LIMID representation of a decision scenario with two unordered decisions

Using the LIMID representation, it is possible to model multistage decision problems with unordered sequences of decisions and decision problems in which perfect recall cannot be assumed or may not be appropriate. This makes the LIMID framework a good candidate for modeling large and complex domains using an appropriate assumption of forgetfulness of the decision maker. Notice that all decision problems that can be represented as an ordinary influence diagram can also be represented as a LIMID.

Example 4.12 (LIMID). Figure 4.14 shows an example of a LIMID representation $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P}, \mathcal{U})$ of a decision scenario with two unordered decisions. Prior to decision D_i, observations on the values of A and C are made, while prior to decision D_j, an observation on the value of E is made. Notice that the observations on A and C made prior to decision D_i are not available at decision D_j and vice versa for the observation on E. □

Example 4.13 (Breeding Pigs (Lauritzen & Nilsson 2001)). A farmer is growing pigs for a period of four months and subsequently selling them. During this period, the pigs may or may not develop a certain disease. If a pig has the disease at the time, it must be sold for slaughtering; its expected market price is €40. If it is disease free, its expected market price as a breeding animal is €135.

Once a month, a veterinarian inspects each pig and makes a test for presence of the disease. If a pig is ill, the test will indicate this with probability 0.80, and if the pig is healthy, the test will indicate this with probability 0.90. At each monthly visit, the doctor may or may not treat a pig for the disease by injecting a certain drug. The cost of an injection is €13.

A pig has the disease in the first month with probability 0.10. A healthy pig develops the disease in the following month with probability 0.20 without injection, whereas a healthy and treated pig develops the disease with probability 0.10, so the injection has some preventive effect. An untreated pig that is unhealthy will remain so in the following month with probability 0.90, whereas the similar probability is 0.50 for an unhealthy pig that is treated. Thus, spontaneous cure is possible, but treatment is beneficial on average.

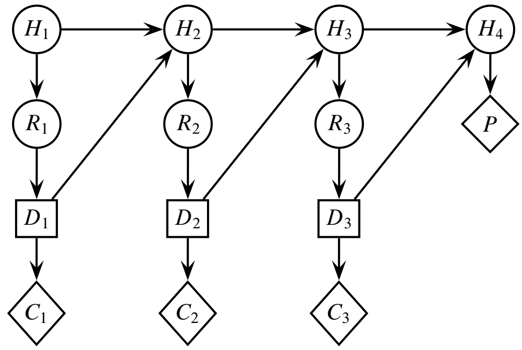


Fig. 4.15 Three test-and-treat cycles are performed prior to selling a pig

The qualitative structure of the LIMID representation of this decision problem is shown in Fig. 4.15. Notice that we make the assumption that the test result R_i is only available for decision D_i. This implies that the test result is not taken into account for future decisions as it is either forgotten or ignored. □

The above example could be modeled as a standard influence diagram (assuming perfect recall), but if more test-and-treat cycles must be performed, the state space size of the past renders decision making intractable. Therefore, it is appropriate to make the decision on whether or not to treat based on the current test result (and not considering past test results and possible treatments)—in this case, individual records for the pigs need not be kept. In short, the example illustrates a situation where instead of keeping track of all past observations and decisions, some of these are deliberately ignored (in order to maintain tractability of the task of computing policies).

4.3 Object-Oriented Probabilistic Networks

As large and complex systems are often composed of collections of identical or similar components, models of such systems will naturally contain repetitive patterns. A complex system will typically be composed of a large number of similar or even identical components. This composition of the system should be reflected in models of the system to support model construction, maintenance, and reconfiguration. For instance, a diagnosis model for diagnosing car start problems could reflect the natural decomposition of a car into its engine, electrical system, fuel system, etc.

To support this approach to model development, the framework of object-oriented probabilistic networks has been developed, see, for example, (Koller & Pfeffer 1997, Laskey & Mahoney 1997, Neil, Fenton & Nielsen 2000). Object-orientation may be defined in the following way

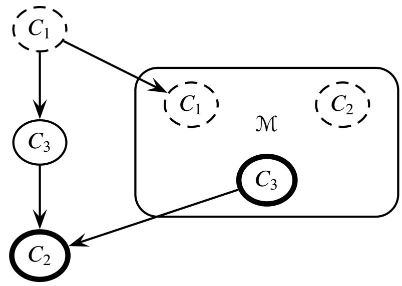


Fig. 4.16 $\mathcal{M}$ is an instance of a network class $C_{\mathcal{M}}$ within another network class $C_{\mathcal{N}}$

$$\text{object-orientation} = \text{objects} + \text{inheritance},$$

where objects are instances of classes and inheritance defines a relationship between classes. Thus, we need to introduce the notion of objects and classes. In this section, we introduce the notion of *object-oriented probabilistic networks (OOPNs).*

The basic OOPN mechanisms described below support a type of object-oriented specification of probabilistic networks, which makes it simple to reuse models, to encapsulate submodels (providing a means for hierarchical model specification), and to perform model construction in a top-down fashion, a bottom-up fashion, or a mixture of the two (allowing repeated changes of level of abstraction).

An object-oriented modeling paradigm provides support for working with different levels of abstraction in constructing network models. Repeated changes of focus are partly due to the fact that humans naturally think about systems in terms of hierarchies of abstractions and partly due to lack of ability to mentally capture all details of a complex system simultaneously. Specifying a model in a hierarchical fashion often makes the model less cluttered and thus provides a better means of communicating ideas among knowledge engineers, domain experts, and users.

In the OOPN paradigm we present, an *instance* or *object* has a set of variables and related functions (i.e., probability distributions, probability densities, utility functions, and precedence constraints). This implies that in addition to the usual types of nodes, the graph of an OOPN model may contain nodes representing instances of other networks encapsulated in the model. A node that does not represent an instance of a network class is said to represent a *basic* variable.

An instance represents an instantiation of a network class within another network class. A network class is a blueprint for an instance. As such, a *network class* is a named and self-contained description of a probabilistic network, characterized by its name, interface, and hidden part. As instances can be nested, an object-oriented network can be viewed as a hierarchical description of a problem domain. In this way, an instance $\mathcal{M}$ is the instantiation (or realization) of a network class $C_{\mathcal{M}}$ within another network class $C_{\mathcal{N}}$, see Fig. 4.16.

An instance connects to other variables via some of its (basic) variables. These variables are known as its *interface* variables. As we wish to support information hiding, the interface variables usually only constitute a subset of the variables in the network class.

Let us be more precise. A network class C is a DAG over three pairwise disjoint sets of nodes $\mathcal{I}(C)$, $\mathcal{H}(C)$, and $\mathcal{O}(C)$, where $\mathcal{I}(C)$ are the input nodes, $\mathcal{H}(C)$ are the hidden nodes, and $\mathcal{O}(C)$ are the output nodes of C. The set $\mathcal{I}(C) \cup \mathcal{O}(C)$ is the interface of C. Interface nodes may represent either decision or random variables, whereas hidden nodes may be instances of network classes, decision variables, random variables, and utility functions.

Definition 4.7. An OOPN network class $C = (\mathcal{N}, \mathcal{I}, \mathcal{O})$ consists of:

- A probabilistic network $\mathcal{N}$ over variables $\mathcal{X}$ with DAG $\mathcal{G}$
- A set of basic variables $\mathcal{I} \subseteq \mathcal{X}$ specified as input variables and a set of basic variables $\mathcal{O} \subseteq \mathcal{X}$ specified as output variables such that $\mathcal{I} \cap \mathcal{O} = \emptyset$ and $\mathcal{H} = \mathcal{X} \setminus (\mathcal{I} \cup \mathcal{O})$

In the graphical representation of an OOPN instances are represented as rectangles with arc-shaped corners, whereas input variables are represented as dashed ovals, and output variables are represented as bold ovals. If the interface variables of a network instance are not shown, then the instance is collapsed. Otherwise, it is expanded.

Since an OOPN implements information hiding through encapsulation, we need to be clear on scope rules. First, we define the notations of simple and qualified names. If X is a variable of a network instance $\mathcal{N}$, then X is the *simple name* of the variable, whereas $\mathcal{N}.X$ is the *qualified name* (also known as the long name) of the variable. The scope $\mathbb{S}(X)$ of a variable X (i.e., a basic variable or an instance) is defined as the part of a model in which the declaration of X can be referred to by its simple name.

The (internal) scope $\mathbb{S}(C)$ of a network class C is the set of variables and instances which can be referred to by their simple names inside C. For instance, the internal scope of the network $C_{\mathcal{N}}$ in Fig. 4.16 on the facing page is $\mathbb{S}(C_{\mathcal{N}}) = \{C_1, C_3, C_2, \mathcal{M}\}$. The scope of an instance $\mathcal{M}$ of a network class $C_{\mathcal{M}}$, that is, $\text{class}(\mathcal{M}) = C_{\mathcal{M}}$, is defined in a similar manner.

The interface variables $\mathcal{I}(C) \cup \mathcal{O}(C)$ of C are used to enlarge the visibility of basic variables in the instantiations of C. The visibility of a variable X can be enlarged by specifying it as either an input or an output variable of its class.

An input variable X of an instance $\mathcal{M}$ is a placeholder for a variable (the parent of X) in the encapsulating class of $\mathcal{M}$. Therefore, an input variable has at most one parent. An output variable X of an instance $\mathcal{M}$, on the other hand, enlarges the visibility of X to include the encapsulating network class of $\mathcal{M}$.

Notice that the scope of a variable is distinct from visibility of the variable. In Fig. 4.16, the scope of output variable C_3 is $\mathcal{M}$, whereas its visibility is enlarged to include $\mathcal{N}$ by defining it as an output variable of $\mathcal{M}$.

An input variable I of an instance $\mathcal{M}$ of network class C is *bound* if it has a parent X in the network class encapsulating $\mathcal{M}$. Each input random variable I of a class C is assigned a default prior probability distribution $P(I)$, which becomes the probability distribution of the variable I in all instances of C where I is an unbound input variable. A link into a node representing an input variable may be referred to as a *binding link*.

Let $\mathcal{M}$ be an instance of network class C. Each input variable $I \in \mathcal{I}(C)$ has no parent in C, no children outside C, and the corresponding variable of M has at most one parent in the encapsulating class of $\mathcal{M}$. Each output variable $O \in \mathcal{O}(C)$ may only have parents in $\mathcal{I}(C) \cup \mathcal{H}(C)$. The children and parents of $H \in \mathcal{H}(C)$ are subsets of the variables of C.

Example 4.14 (Object-Oriented Probabilistic Network). Figure 4.16 shows an instance $\mathcal{M}$ of a network class $C_{\mathcal{M}}$ instantiated within another network class $C_{\mathcal{N}}$. Network class $C_{\mathcal{N}}$ has input variable C_1, hidden variables C_3 and $\mathcal{M}$, and output variable C_2. The network class $C_{\mathcal{M}}$ has input variables C_1 and C_2, output variable C_3, and unknown hidden variables. The input variable C_1 of instance $\mathcal{M}$ is bound to C_1 of C_N, whereas C_2 is unbound.

Since $C_1 \in \mathcal{I}(C_{\mathcal{N}})$ is bound to $C_1 \in \mathcal{I}(\mathcal{M})$, the visibility of $C_1 \in \mathcal{I}(C_{\mathcal{N}})$ is extended to include the internal scope of $\mathcal{M}$. Hence, when we refer to $C_1 \in \mathcal{I}(C_{\mathcal{M}})$ inside $C_{\mathcal{M}}$, we are in fact referring to $C_1 \in \mathcal{I}(C_{\mathcal{N}})$ as $C_1 \in \mathcal{I}(C_{\mathcal{M}})$ in instance M is a placeholder for $C_1 \in \mathcal{I}(C_{\mathcal{N}})$ (i.e., you may think of $C_1 \in \mathcal{I}(C_{\mathcal{M}})$ as the formal parameter of $C_{\mathcal{M}}$ and $C_1 \in \mathcal{I}(C_{\mathcal{N}})$ as the actual parameter of $\mathcal{M}$). □

Since an input variable $I \in \mathcal{I}(\mathcal{M})$ of an instance $\mathcal{M}$ is a placeholder for a variable Y in the internal scope of the encapsulating instance of $\mathcal{M}$, type checking becomes important when the variable Y is bound to I. The variable I enlarges the visibility of Y to include the internal scope of $\mathcal{M}$, and it should therefore be equivalent to Y. We define two variables Y and X to be equivalent as follows:

Definition 4.8. Two variables X and Y are *equivalent* if and only if they are of the same kind, category, and subtype with the same state labels in the case of discrete variables.

This approach to type checking is referred as *strong type checking*.

If a model contains a lot of repetitive structure, its construction may be tiresome, and the resulting model may even be rather cluttered. Both issues are solved when using object-oriented models. Another key feature of object-oriented models is modularity. Modularity allows knowledge engineers to work on different parts of the model independently once an appropriate interface has been defined. The following example will illustrate this point.

Example 4.15 (Apple Jack's Garden). Let us assume that Apple Jack from Example 4.1 on page 71 has a garden of three apple trees (including his finest apple tree). He may want to reason about the sickness of each tree given observations on whether or not some of the trees in the garden are losing their leaves.

Figure 4.17 shows the apple tree network class. The prior of each tree being sick will be the same, while the dryness of a tree is caused by a drought. The drought is an input variable of the apple tree network class. If there is a drought, this will impact the dryness of all trees. The prior on drought is $P(\text{Drought}) = (0.9, 0.1)$, while the conditional distribution of Dry conditional on Drought is shown in Table 4.9.

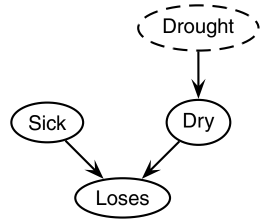


Fig. 4.17 The apple tree network class

Table 4.9 The conditional probability distribution $P(\mathsf{Drought} \mid \mathsf{Dry})$

	Dry	
Drought	no	yes
no	0.85	0.15
yes	0.35	0.65

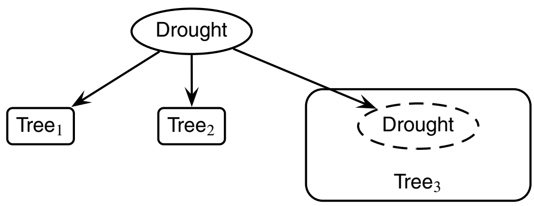


Fig. 4.18 The apple garden network consisting of three instantiations of the apple tree network

Figure 4.18 shows the network class of the apple garden. The input variable Drought of each of the instances of the apple tree network class is bound to the Drought variable in the apple garden network class. This enlarges the visibility of the Drought variable (in the apple garden network class) to the internal scope defined by each instance.

The two instances Tree_1 and Tree_2 are collapsed (i.e., not showing the interface variables), while the instance Tree_3 is expanded (i.e., not collapsed) illustrating the interface of the network class.

The Drought variable could be an input variable of the apple garden network class as well as it is determined by other complex factors. For the sake of simplicity of the example, we have made it a hidden variable of the apple garden network class. □

As mentioned above, a default prior distribution $P(X)$ is assigned to each input variable $X \in \mathcal{I}(C)$ of the class $C = (\mathcal{N}, \mathcal{O}, \mathcal{I})$. Assigning a default potential to each input variable X implies that any network class is a valid probabilistic network model.

4.3.1 Chain Rule

It should be clear from the above discussion that each OOPN encodes either a probability distribution or an expected utility function. For simplicity, we will discuss only the chain rule for object-oriented (discrete) Bayesian networks. The chain rule of an object-oriented Bayesian network reflects the hierarchical structure of the model.

An instance $\mathcal{M}$ of network class C encapsulates a conditional probability distribution over its random variables given its unbound input nodes. For further simplicity, let $C = (\mathcal{N}, \mathcal{I}, \mathcal{O})$ be a network class over basic discrete random variables only (i.e., no instances, no decisions, and no utilities) with $\mathcal{N} = (\mathcal{X}, \mathcal{G}, \mathcal{P})$ where $X \in \mathcal{X}$ is the only input variable, that is, $X \in \mathcal{I}$ and $|\mathcal{I}| = 1$. Since X has a default prior distribution, $\mathcal{N}$ is a valid model representing the joint probability distribution

$$P(\mathcal{X}) = P(X) \prod_{Y_v \neq X} P(Y_v \,|\, X_{\mathrm{pa}(v)}).$$

In general, an instance $\mathcal{M}$ is a representation of the conditional probability distribution $P(\mathcal{O}|\mathcal{I}')$ where $\mathcal{I}' \subseteq \mathcal{I}$ is the subset of bound input variables of $\mathcal{M}$

$$P(\mathcal{O}|\mathcal{I}') = \prod_{X \in \mathcal{I} \setminus \mathcal{I}'} P(X) \prod_{Y_v \notin \mathcal{I}} P(Y_v \,|\, X_{\mathrm{pa}(v)}).$$

4.3.2 Unfolded OOPNs

An object-oriented network $\mathcal{N}$ has an equivalent *flat* or *unfolded* network model representation $\mathcal{M}$. The unfolded network model of an object-oriented network $\mathcal{N}$ is obtained by recursively unfolding the instance nodes of $\mathcal{N}$. The unfolded network representation of a network class is important as it is the structure used for inference.

The joint distribution of an object-oriented Bayesian network model is equivalent to the joint distribution of its unfolded network model

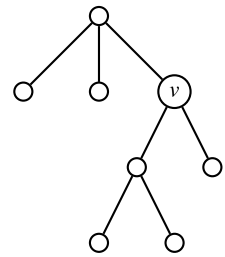
$$P(\mathcal{X}) = \prod_{X_v \in \mathcal{X}_{\mathcal{M}}} P(X_v \,|\, X_{\mathrm{pa}(v)}),$$

where $\mathcal{M} = (\mathcal{X}, \mathcal{G}, \mathcal{P})$ is the unfolded network.

4.3.3 Instance Trees

An object-oriented model is a hierarchical model representation. The *instance tree* T of an object-oriented model $\mathcal{N}$ is a tree over the set of instances of classes in $\mathcal{N}$. Two

Fig. 4.19 An instance tree



nodes v_i and v_j in T (with v_i closer to the root of T than v_j) are connected by an undirected link if and only if the instance represented by v_i contains the instance represented by v_j. The root of an instance tree is the top-level network class not instantiated in any other network class within the model. Notice that an instance tree is unique.

In addition to the notion of default potentials, there is the notion of the *default instance*. Let C be a network class with instance tree T. Each non-root node v of T represents an instance of a class C_v, whereas the root node r of T represents an instance of the unique class C_r, which has not been instantiated in any class. This instance is referred to as the default instance of C_r.

Example 4.16 (Instance Tree). Figure 4.19 shows the instance tree of a network class $\mathcal{N}$ where the root is the default instance of $\mathcal{N}$.

Each node v of T represents an instance $\mathcal{M}$, and the children of v in T represent instances in $\mathcal{M}$. □

4.3.4 Inheritance

Another important concept of the OOPN framework is inheritance. For simplicity, we define inheritance as the ability of an instance to take its interface definition from another instance. Let C_1 be a network class with input variables $I(C_1)$ and output variables $O(C_1)$, that is, $C_1 = (\mathcal{N}_1, \mathcal{I}_1, \mathcal{O}_1)$. A network class $C_2 = (\mathcal{N}_2, \mathcal{I}_2, \mathcal{O}_2)$ may be specified as a subclass of C_1 if and only if $\mathcal{I}_1 \subseteq \mathcal{I}_2$ and $\mathcal{O}_1 \subseteq \mathcal{O}_2$. Hence, subclasses may enlarge the interface.

Inheritance is not to the knowledge of the authors implemented in any widely available software supporting OOPN.

4.4 Dynamic Models

The graph of a probabilistic network is restricted to be a finite acyclic directed graph, see Sect. 2.1. This seems to imply that probabilistic networks as such do not support models with feedback loops or models of dynamic systems changing over time. This is not the case. A common approach to representing and solving dynamic models or models with feedback loops is to unroll the dynamic model for the desired number of time steps and treat the resulting network as a static network. Similarly, a feedback loop can be unrolled and represented using a desired number of time steps. The unrolled static network is then solved using a standard algorithm applying evidence at the appropriate time steps.

As an example of a dynamic model, consider the problem of monitoring the state of a dynamic process over a specific period of time. Assume the network of Fig. 4.20 is an appropriate model of the causal relations between variables representing the system at any point in time. The structure of this network is static in the sense that it represents the state of the system at a certain point in time. In the process of monitoring the state of the system over a specific period of time, we will make observations on a subset of the variables in the network and make inference about the remaining unobserved variables. In addition to reasoning about the current state of the system, we may want to reason about the state of the system at previous and future points in time. For this usage, the network in Fig. 4.20 is inadequate. Furthermore, the state of the system at the current point in time will impact the state of the system in the future and be impacted by the state of the system in the past.

What is needed is a time-sliced model covering the period of time over which the system should be monitored. Figure 4.21 indicates a time-sliced model constructed based on the static network shown in Fig. 4.20. Each time-slice consists of the structure shown in Fig. 4.20, while the development of the system is specified by links between variables of different time-slices.

The *temporal links* of a time-slice t_i are the set of links from variables of time-slice t_{i-1} into variables of time-slice t_i. The temporal links of time-slice t_i define the conditional distribution of the variables of time-slice t_i given the variables of time-

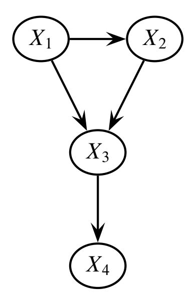


Fig. 4.20 The structure of a static network model

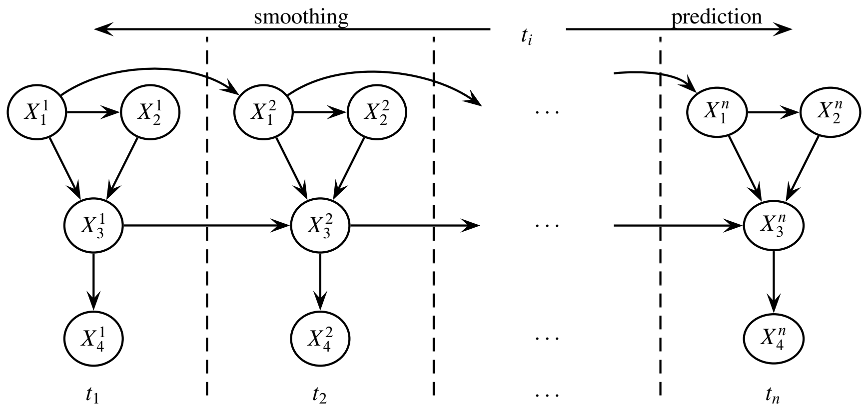


Fig. 4.21 The structure of a dynamic model with n time-slices

slice t_{i-1}. The temporal links connect variables of adjacent time-slices. For instance, the temporal links of time-slice t_2 in Fig. 4.21 is the set $\{(X_1^1, X_1^2), (X_3^1, X_3^2)\}$.

The *interface* of a time-slice is the set of variables with parents in the previous time-slice. For instance, the interface of time-slice t_2 in Fig. 4.21 is the set $\{X_1^2, X_3^2\}$.

Three additional concepts are often used in relation to dynamic models. Let i be the current time step, then *smoothing* is the process of querying about the state of the system at a previous time step $j < i$ given evidence about the system at time i, *filtering* is the process of querying about the state of the system at the current time step, and *prediction* is the process of querying about the state of the system at a future time step $j > i$.

A dynamic Bayesian network is *stationary* when the transition probability distributions are invariant between time steps. A dynamic Bayesian network is first-order Markovian when the variables at time step $i + 1$ are d-separated from the variables at time step $i - 1$ given the variables at time step i. When a system is stationary and Markovian, the state of the system at time $i + 1$ only depends on its state at time i, and the probabilistic dependence relations are the same for all i. The Markovian property implies that arcs between time-slices only go from one time-slice to the subsequent time-slice.

A dynamic Bayesian network is referred to as either a *dynamic Bayesian network* (DBN) or a *time-sliced Bayesian network* (TBN). See Kjærulff (1995) for more details on dynamic Bayesian networks.

Example 4.17 (Apple Jack's Finest Tree). Consider the Apple Jack network in Fig. 4.1 of Example 4.1 on page 71. The network is used for reasoning about the cause of Apple Jack's finest apple tree losing its leaves. The network is static and models the dependence relations between two diseases and a symptom at four specific points in time where Apple Jack is observing his tree.

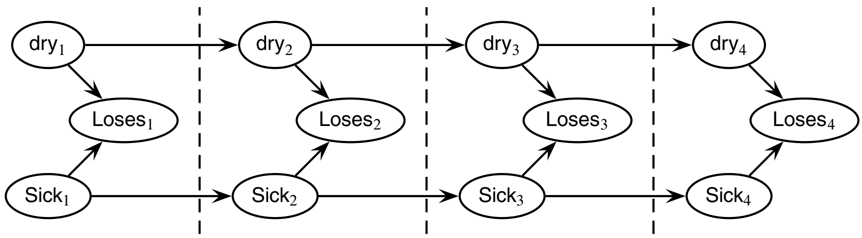


Fig. 4.22 A model with four time-slices

Consider the case where Apple Jack is monitoring the development of the disease over a period of time by observing the tree each day in the morning. In this case, the level of dryness of the tree on a specific day will depend on the level of dryness on the previous day and impact the level of dryness on the next day, similarly for the level of sickness. The levels of dryness and sickness on the next day are independent of the levels of dryness and sickness on the previous day given the levels of dryness and sickness on the current day. This can be captured by a dynamic model.

Figure 4.22 shows a dynamic model with four time-slices. Each time step models the state of the apple tree at a specific point in time (the dashed lines illustrate the separation of the model into time-slices). The conditional probability distributions $P(\mathsf{Dry}_i \,|\, \mathsf{Dry}_{i-1})$ and $P(\mathsf{Sick}_i \,|\, \mathsf{Sick}_{i-1})$ are the transition probability distributions. The interface between time-slices $i-1$ and i consists of Dry_i and Sick_i.

Assume that it is the second day when Apple Jack is observing his tree. The observations on Loses of the first and second day are entered as evidence on the corresponding variables. Filtering is the task of computing the probability of the tree being sick on the second day, smoothing is the task of computing the probability of sickness on the first day, and prediction is the task of computing the probability of the tree being sick on the third or fourth day. □

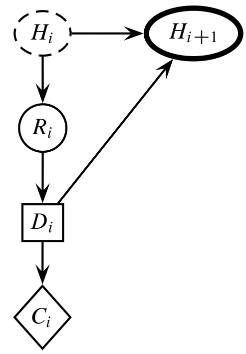
Dynamic models are not restricted to be Bayesian networks. Influence diagrams and LIMIDs can also be represented as dynamic models.

4.4.1 Time-Sliced Networks Represented as OOPNs

Time-sliced networks are often represented using object-oriented networks as the following example illustrates.

Example 4.18 (Breeding Pigs). Example 4.13 shows a LIMID representation of a decision problem related to breeding pigs, see Fig. 4.15 on page 95. The decision problem is in fact modeled as a time-sliced model where the structure of each time-slice representing a test-and-treat cycle is shown in Fig. 4.23.

Fig. 4.23 The test-and-treat cycle of the breeding pigs network in Fig. 4.15



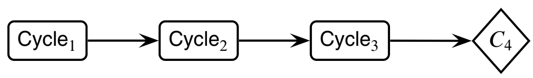


Fig. 4.24 The breeding pigs network as a time-sliced OOPN

Three instances of the network class in Fig. 4.23 are constructed to create the network in Fig. 4.24. The use of object-oriented modeling has simplified the network construction.

The network in Fig. 4.24 is equivalent to the network in Fig. 4.15 on page 95. □

Kjærulff (1995) has described a computational system for dynamic time-sliced Bayesian networks. The system implemented is referred to as dHugin. Boyen & Koller (1998) have described an approximate inference algorithm for solving dynamic Bayesian networks with bounds on the approximation error.

4.5 Summary

In this chapter we have introduced probabilistic networks for belief update and decision making under uncertainty. A probabilistic network represents and processes probabilistic knowledge. The qualitative component of a probabilistic network encodes a set of (conditional) dependence and independence statements among a set of random variables, informational precedence, and preference relations. The quantitative component specifies the strengths of dependence relations using probability theory and preference relations using utility theory.

We have introduced discrete Bayesian network models and CLG Bayesian network models for belief update. A discrete Bayesian network supports the use of discrete random variables, whereas a CLG Bayesian network supports the use of a mixture of continuous and discrete random variables. The continuous variables are constrained to be conditional linear Gaussian variables. This chapter contains a number of examples that illustrate the use of Bayesian networks for belief update.

Discrete influence diagrams, CLQG influence diagrams, and limited memory influence diagrams were introduced as models for belief update and decision making under uncertainty. An influence diagram is a Bayesian network augmented with decision variables, informational precedence relations, and preference relations. A discrete influence diagram supports the use of discrete random and decision variables with an additively decomposing utility function. A CLQG influence diagram supports the use of a mixture of continuous and discrete variables. The continuous random variables are constrained to be conditional linear Gaussian variables, while the utility function is constrained to be linear–quadratic. A limited memory influence diagram is an extension of the discrete influence diagram where the assumptions of no-forgetting and regularity (i.e., a total order on the decisions) are relaxed. This allows us to model a large set of decision problems that cannot be modeled using the traditional influence diagram representation. This chapter contains a number of examples that illustrate the use of influence diagrams for decision making under uncertainty.

Finally, we have introduced OOPNs. The basic OOPN mechanisms introduced support a type of object-oriented specification of probabilistic networks, which makes it simple to reuse models, to encapsulate submodels, and to perform model construction at different levels of abstraction. This chapter contains a number of examples that illustrate the use of the basic OOPN mechanisms in the model development process. OOPNs are well suited for constructing time-sliced networks. Time-sliced networks are used to represent dynamic models.

In Chap. 5, we discuss techniques for solving probabilistic networks.

Exercises

Exercise 4.1. Peter and Eric are chefs at Restaurant Bayes. Peter works 6 days a week, while Eric works one day a week. In 90% of the cases, Peter's food is high quality, while Eric's food is high quality in 50% of the cases. One evening Restaurant Bayes serves an awful meal.

Is it fair to conclude that Eric prepared the food that evening?

Exercise 4.2. One in a thousand people has a prevalence for a particular heart disease. There is a test to detect this disease. The test is 100% accurate for people who have the disease and is 95% accurate for those who do not (this means that 5% of people who do not have the disease will be wrongly diagnosed as having it).

(1) If a randomly selected person tests positive, what is the probability that the person actually has the heart disease?

Exercise 4.3. Assume a math class is offered once every semester, while an AI class is offered twice. The number of students taking a class depends on the subject. On average, 120 students take AI ($\sigma^2 = 500$), while 180 students take math ($\sigma^2 = 1,000$). Assume that on average 25% pass the AI exam ($\sigma^2 = 400$) while 50% pass the math exam ($\sigma^2 = 500$).

(a) What is the average number of students passing either a math or AI exam?
(b) What is the average number of students passing a math exam?
(c) What is the average number of students taking a math class when 80 students pass the exam?

Exercise 4.4. Frank goes to the doctor because he believes that he has got the flu. At this particular time of the year, the doctor estimates that one out of 1,000 people suffers from the flu. The first thing the doctor checks is whether Frank appears to have the standard symptoms of the flu; if Frank suffers from the flu, then he will exhibit these symptoms with probability 0.9, but if he does not have the flu, he may still have these symptoms with probability 0.05. After checking whether or not Frank has the symptoms, the doctor can decide to have a test performed which may reveal more information about whether or not Frank suffers from the flu; the cost of performing the test is € 40. The test can either give a positive or a negative result, and the frequency of false-positives and false-negatives is 0.05 and 0.1, respectively. After observing the test result (if any), the doctor can decide to administer a drug that with probability 0.6 may shorten the sickness period if Frank suffers from the flu (if he has not got the flu, then the drug has no effect). The cost of administering the drug is € 100, and if the sickness period is shortened, the doctor estimates that this is worth € 1,000.

(a) Construct an influence diagram for the doctor from the description above.
(b) Specify the probability distributions and the utility functions for the influence diagram.

Exercise 4.5. Assume that Frank is thinking about buying a used car for € 20,000, and the market price for similar cars with no defects is € 23,000. The car may, however, have defects which can be repaired at the cost of € 5,000; the probability that the car has defects is 0.3. Frank has the option of asking a mechanic to perform (exactly) one out of two different tests on the car. Test_1 has three possible outcomes, namely, no defects, defects, and inconclusive. For Test_2 there are only two possible outcomes (no defects and defects). If Frank chooses to have a test performed on the car, the mechanic will report the result back to Frank who then decides whether or not to buy the car; the cost of Test_1 is € 300, and the cost of Test_2 is € 1,000.

(a) Construct an influence diagram for Frank's decision problem.
(b) Calculate the expected utility and the optimal strategy for the influence diagram; calculate the required probabilities from the joint probability table (over the variables Test_1, Test_2, and StateOfCar) specified below.

		Test_1		
		no defects	defects	inconclusive
Test_2	no defects	(0.448, 0.00375)	(0.028, 0.05625)	(0.084, 0.015)
	defects	(0.112, 0.01125)	(0.007, 0.16875)	(0.021, 0.045)

Exercise 4.6. An environmental agency visits a site where a chemical production facility has previously been situated. Based on the agency's knowledge about the

facility, they estimate that there is a 0.6 risk that chemicals from the facility have contaminated the soil. If the soil is contaminated (and nothing is done about it), all people in the surrounding area will have to undergo a medical examination due to the possible exposure; there are $1,000$ people in the area, and the cost of examining/treating one person is \$100. To avoid exposure, the agency can decide to remove the top layer of the soil which, in case the ground is contaminated, will completely remove the risk of exposure; the cost of removing the soil is \$30,000. Before making the decision of whether or not to remove the top layer of soil, the agency can perform a test which will give a positive result (with probability 0.9) if the ground is contaminated; if the ground is not contaminated, the test will give a positive result with probability 0.01. The cost of performing the test is \$1,000.

(a) Construct an influence diagram for the environmental agency from the description above.
(b) Specify the probability distributions and the utility functions for the influence diagram.

Exercise 4.7. A company has observed that one of their software systems is unstable, and they have identified a component which they suspect is the cause of the instability. The company estimates that the prior probability for the component being faulty is 0.01, and if the component is faulty, then it causes the system to become unstable with probability 0.99; if the component is not faulty, then the system may still be unstable (due to some other unspecified element) with probability 0.001.

To try to solve the problem, the company must first decide whether to *patch* the component at a cost € 10,000 : if the component is faulty, then the patch will solve the fault with probability 0.95 (there may be several things wrong, not all of which may be covered by the patch), but if the component is not faulty, then the patch will have no effect. The company also knows that in the near future the vendor of the component will make another patch available at the cost of € 20,000; the two patches focus on different parts of the component. This new patch will solve the problem with probability 0.99, and (as for the first patch) if the component is not faulty, then the patch will have no effect. Thus, after deciding on the first patch, the company observes whether or not the patch solved the problem (i.e., is the system still unstable?) and it then has to decide on the second patch. The company estimates that (after the final decision has been made) the value of having a fully functioning component is worth € 100,000.

(a) Construct an influence diagram for the company from the description above.
(b) Specify the probability distributions and the utility functions for the influence diagram.

Exercise 4.8. Consider a stud farm with ten horses where Cecily has unknown mare and sire, John has mare Irene and sire Henry, Henry has mare Dorothy and sire Fred, Irene has mare Gwenn and sire Eric, Gwenn has mare Ann and unknown sire, Eric has mare Cecily and sire Brian, Fred has mare Ann and unknown sire, Brian

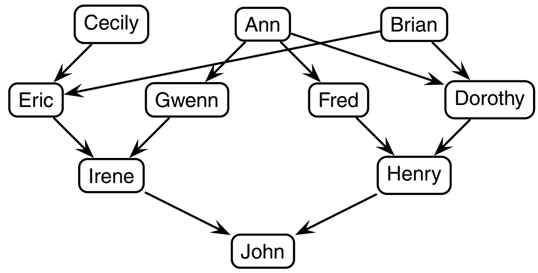


Fig. 4.25 The stud farm pedigree

has unknown mare and sire, Dorothy has mare Ann and sire Brian, and Ann has unknown mare and sire, see Fig. 4.25.

A sick horse has genotype aa, a carrier of the disease has genotype aA, and a noncarrier has genotype AA. $P(aa, aA, AA) = (0.04, 0.32, 0.64)$.

(a) Construct an object-oriented network representation of the stud farm problem.

(b) What is the probability of each horse being sick/a carrier/a noncarrier once we learn that John is sick?