rather are summarized as "noise parameters." If *P(Fever|Cold)* = 0.4, *P(Fever|Flu)* = 0.8, and *P(Fever|Malaria)* = 0.9, then the noise parameters are 0.6, 0.2, and 0.1, respectively. If no parent node is true, then the output node is false with 100% certainty. If exactly one parent is true, then the output is false with probability equal to the noise parameter for that node. In general, the probability that the output node is *False* is just the product of the noise parameters for all the input nodes that are true. For this example, we have the following:

Cold	*Flu*	*Malaria*	*P(Fever)*	*P(¬Fever)*
F	F	F	0.0	1.0
F	F	T	0.9	0.1
F	T	F	0.8	0.2
F	T	T	0.98	$0.02 = 0.2 \times 0.1$
T	F	F	0.4	0.6
T	F	T	0.94	$0.06 = 0.6 \times 0.1$
T	T	F	0.88	$0.12 = 0.6 \times 0.2$
T	T	T	0.988	$0.012 = 0.6 \times 0.2 \times 0.1$

In general, noisy logical relationships in which a variable depends on k parents can be described using $O(k)$ parameters instead of $O(2^k)$ for the full conditional probability table. This makes assessment and learning much easier. For example, the CPSC network (Pradhan *et al.*, 1994) uses noisy-OR and noisy-MAX, and requires "only" 8,254 values instead of 133,931,430 for a network with full CPTs.

Conditional independence relations in belief networks

The preceding analysis shows that a belief network expresses the conditional independence of a node and its predecessors, given its parents, and uses this independence to design a construction method for networks. If we want to design inference algorithms, however, we will need to know whether more general conditional independences hold. If we are given a network, is it possible to "read off" whether a set of nodes X is independent of another set Y, given a set of evidence nodes E? The answer is yes, and the method is provided by the notion of **direction-dependent separation** or **d-separation**.

D-SEPARATION

First, we will say what d-separation is good for. *If every undirected path*[2] *from a node in X to a node in Y is d-separated by E, then X and Y are conditionally independent given E.* The definition of d-separation is somewhat complicated. We will need to appeal to it several times in constructing our inference algorithms. Once this is done, however, the process of constructing and using belief networks does not involve any uses of d-separation.

BLOCKED

A set of nodes E d-separates two sets of nodes X and Y if every undirected path from a node in X to a node in Y is **blocked** given E. A path is blocked given a set of nodes E if there is a node Z on the path for which one of three conditions holds:

1. Z is in E and Z has one arrow on the path leading in and one arrow out.

[2] An undirected path is a path through the network that ignores the direction of the arrows.

only in - out or both out.

2. Z is in E and Z has both path arrows leading out.
3. Neither Z nor any descendant of Z is in E, and both path arrows lead in to Z. not both in

Figure 15.4 shows these three cases. The proof that d-separated nodes are conditionally independent is also complicated. We will use Figure 15.5 to give examples of the three cases:

1. Whether there is *Gas* in the car and whether the car *Radio* plays are independent given evidence about whether the *SparkPlugs* fire.
2. *Gas* and *Radio* are independent if it is known if the *Battery* works.
3. *Gas* and *Radio* are independent given no evidence at all. But they are dependent given evidence about whether the car *Starts*. For example, if the car does not start, then the radio playing is increased evidence that we are out of gas. *Gas* and *Radio* are also dependent given evidence about whether the car *Moves*, because that is enabled by the car starting.

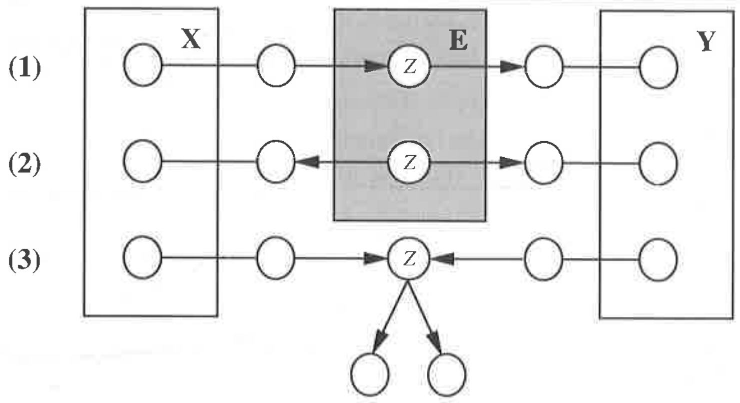

Figure 15.4 Three ways in which a path from X to Y can be blocked, given the evidence E. If every path from X to Y is blocked, then we say that E d-separates X and Y.

15.3 INFERENCE IN BELIEF NETWORKS

The basic task for any probabilistic inference system is to compute the posterior probability distribution for a set of **query variables**, given exact values for some **evidence variables**. That is, the system computes **P**(*Query*|*Evidence*). In the alarm example, *Burglary* is an obvious query variable, and *JohnCalls* and *MaryCalls* could serve as evidence variables. Of course, belief networks are flexible enough so that any node can serve as either a query or an evidence variable. There is nothing to stop us from asking **P**(*Alarm*|*JohnCalls*, *Earthquake*), although it would be somewhat unusual. In general, *an agent gets values for evidence variables from its percepts (or from other reasoning), and asks about the possible values of other variables so that it can decide*

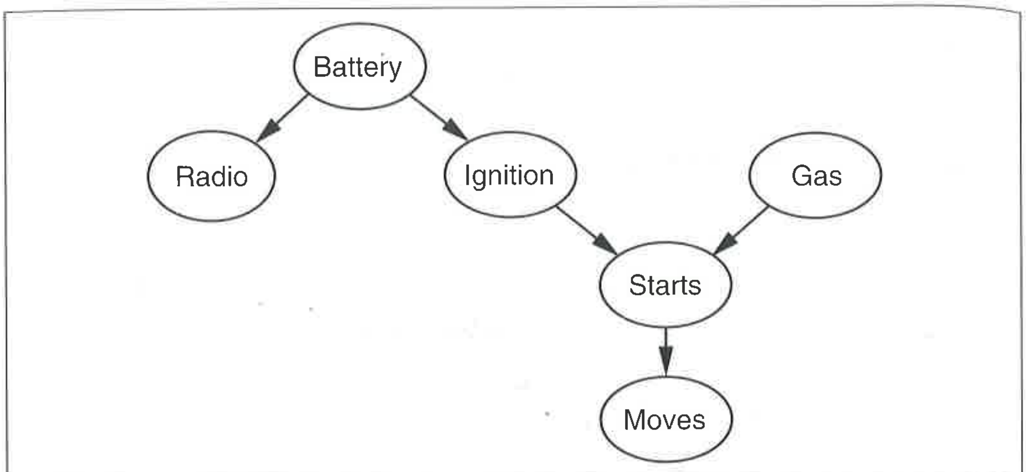

Figure 15.5 A belief network describing some features of a car's electrical system and engine.

what action to take. The two functions we need are BELIEF-NET-TELL, for adding evidence to the network, and BELIEF-NET-ASK, for computing the posterior probability distribution for a given query variable.

The nature of probabilistic inferences

Before plunging into the details of the inference algorithms, it is worthwhile to examine the kinds of things such algorithms can achieve. We will see that a single mechanism can account for a very wide variety of plausible inferences under uncertainty.

Consider the problem of computing $P(Burglary|JohnCalls)$, the probability that there is a burglary given that John calls. This task is quite tricky for humans, and therefore for many reasoning systems that attempt to encode human judgment. The difficulty is not the complexity of the problem, but keeping the reasoning straight. An incorrect but all-too-common line of reasoning starts by observing that when the alarm goes off, *JohnCalls* will be true 90% of the time. The alarm is fairly accurate at reflecting burglaries, so $P(Burglary|JohnCalls)$ should also be about 0.9, or maybe 0.8 at worst. The problem is that this line of reasoning ignores the prior probability of John calling. Over the course of 1000 days, we expect one burglary, for which John is very likely to call. However, John also calls with probability 0.05 when there actually is no alarm—about 50 times over 1000 days. Thus, we expect to receive about 50 false alarms from John for every 1 burglary, so $P(Burglary|JohnCalls)$ is about 0.02. In fact, if we carry out the exact computation, we find that the true value is 0.016. It is less than our 0.02 estimate because the alarm is not perfect.

Now suppose that as soon as we get off the phone with John, Mary calls. We are now interested in incrementally updating our network to give $P(Burglary|JohnCalls \land MaryCalls)$. Again, humans often overestimate this value; the correct answer is only 0.29. We can also determine that $P(Alarm|JohnCalls \land MaryCalls)$ is 0.76 and $P(Earthquake|JohnCalls \land MaryCalls)$ is 0.18.