The basic task for any probabilistic inference system is to compute the posterior probability distribution for a set of query variables, given some observed event—that is, some assignment of values to a set of evidence variables. To simplify the presentation, we will consider only one query variable at a time; the algorithms can easily be extended to queries with multiple variables. We will use the notation from Chapter 13: \( X \) denotes the query variable; \( E \) denotes the set of evidence variables \( E_1, \ldots, E_m \), and \( e \) is a particular observed event; \( Y \) will denote the nonevidence, nonquery variables \( Y_1, \ldots, Y \) (called the hidden variables). Thus, the complete set of variables is \( X = X_1 \cup E \cup Y \). A typical query asks for the posterior probability distribution \( P(X|e) \).
In the burglary network, we might observe the event in which \( \text{JohnCalls} = \text{true} \) and \( \text{MaryCalls} = \text{true} \). We could then ask for, say, the probability that a burglary has occurred:

\[
P(\text{Burglary} \mid \text{JohnCalls} = \text{true}, \text{MaryCalls} = \text{true}) = 0.284, 0.716.
\]

In this section we discuss exact algorithms for computing posterior probabilities and will consider the complexity of this task. It turns out that the general case is intractable, so Section 14.5 covers methods for approximate inference.

### 14.4.1 Inference by enumeration

Chapter 13 explained that any conditional probability can be computed by summing terms from the full joint distribution. More specifically, a query \( P(X \mid e) \) can be answered using Equation (13.9), which we repeat here for convenience:

\[
P(X \mid e) = P(X, e) = P(X, e, y).
\]

Now, a Bayesian network gives a complete representation of the full joint distribution. More specifically, Equation (14.2) on page 513 shows that the terms \( P(x, e, y) \) in the joint distribution can be written as products of conditional probabilities from the network. Therefore, a query can be answered using a Bayesian network by computing sums of products of conditional probabilities from the network.

Consider the query \( P(\text{Burglary} \mid \text{JohnCalls} = \text{true}, \text{MaryCalls} = \text{true}) \). The hidden variables for this query are \( \text{Earthquake} \) and \( \text{Alarm} \). From Equation (13.9), using initial letters for the variables to shorten the expressions, we have

\[
P(B \mid j, m) = \alpha P(B, j \in n) = \alpha \sum_a \sum_e P(B, j, m, e, a).
\]

The semantics of Bayesian networks (Equation (14.2)) then gives us an expression in terms of CPT entries. For simplicity, we do this just for \( \text{Burglary} = \text{true} \):

\[
P(\text{Burglary} \mid \text{JohnCalls} = \text{true}, \text{MaryCalls} = \text{true}) = \alpha \sum_a \sum_e P(b) P(e) P(a \mid b, e) P(j \mid a) P(m \mid a).
\]

To compute this expression, we have to add four terms, each computed by multiplying five numbers. In the worst case, where we have to sum out almost all the variables, the complexity of the algorithm for a network with \( \alpha \) Boolean variables is \( O(2^\alpha) \).

An improvement can be obtained from the following simple observations: the \( P(b) \) term is a constant and can be moved outside the \( \sum \) over \( a \) and \( e \), and the \( P(e) \) term can be moved outside the \( \sum \) over \( a \). Hence, we have

\[
P(b \mid j, m) = P(b) \sum_a P(a \mid b, e) P(j \mid a) P(m \mid a).
\]

This expression can be evaluated by looping through the variables in order, multiplying CPT entries as we go. For each \( \sum \), we also need to loop over the variable's possible

\footnote{An expression such as \( \sum_a P(a) \) means to sum \( P(A = a) \) for all possible values of \( a \). When \( a \) is Boolean, there is an ambiguity in that \( P(e) \) is used to mean both \( P(E = \text{true}) \) and \( P(E = \text{false}) \), but it should be clear from context which is intended; in particular, in the context of a \( \sum \) the latter is intended.}