variable. After its removal, there may be some more leaf nodes, and these too may be irrelevant. Continuing this process, we eventually find that *every variable that is not an ancestor of a query variable or evidence variable is irrelevant to the query.* A variable elimination algorithm can therefore remove all these variables before evaluating the query.

When applied to the insurance network shown in Figure 13.9, variable elimination shows considerable improvement over the naive enumeration algorithm. Using reverse topological order for the variables, exact inference using elimination is about 1,000 times faster than the enumeration algorithm.

13.3.3 The complexity of exact inference

The complexity of exact inference in Bayes nets depends strongly on the structure of the network. The burglary network of Figure 13.2 belongs to the family of networks in which there is at most one undirected path (i.e., ignoring the direction of the arrows) between any two nodes in the network. These are called **singly connected** networks or **polytrees**, and they have a particularly nice property: *The time and space complexity of exact inference in polytrees is linear in the size of the network.* Here, the size is defined as the number of CPT entries; if the number of parents of each node is bounded by a constant, then the complexity will also be linear in the number of nodes. These results hold for any ordering consistent with the topological ordering of the network (Exercise 13.VEEX).

Singly connected

Polytree

For **multiply connected** networks, such as the insurance network in Figure 13.9, variable elimination can have exponential time and space complexity in the worst case, even when the number of parents per node is bounded. This is not surprising when one considers that *because it includes inference in propositional logic as a special case, inference in Bayes nets is NP-hard.* To prove this, we need to work out how to encode a propositional satisfiability problem as a Bayes net, such that running inference on this net tells us whether or not the original propositional sentences are satisfiable. (In the language of complexity theory, we **reduce** satisfiability problems to Bayes net inference problems.) This turns out to be quite straightforward. Figure 13.14 shows how to encode a particular 3-SAT problem. The propositional variables become the root variables of the network, each with prior probability 0.5. The next layer of nodes corresponds to the clauses, with each clause variable C_j connected to the appropriate variables as parents. The conditional distribution for a clause variable is a deterministic disjunction, with negation as needed, so that each clause variable is true if and only if the assignment to its parents satisfies that clause. Finally, S is the conjunction of the clause variables.

Multiply connected

Reduction

To determine if the original sentence is satisfiable, we simply evaluate $P(S = true)$. If the sentence is *satisfiable*, then there is some possible assignment to the logical variables that makes S true; in the Bayes net, this means that there is a possible world with nonzero probability in which the root variables have that assignment, the clause variables have value *true*, and S has value *true*. Therefore, $P(S = true) > 0$ for a satisfiable sentence. Conversely, $P(S = true) = 0$ for an unsatisfiable sentence: all worlds with $S = true$ have probability 0. Hence, we can use Bayes net inference to solve 3-SAT problems; from this, we conclude that Bayes net inference is NP-hard.

We can, in fact, do more than this. Notice that the probability of each satisfying assignment is 2^{-n} for a problem with n variables. Hence, the *number* of satisfying assignments is $P(S = true)/(2^{-n})$. Because computing the *number* of satisfying assignments for a 3-SAT