UNIT - V

Uncertain knowledge and Learning
Uncertainty: Acting under Uncertainty, Basic Probability Notation, Inference Using Full Joint Distributions, Independence, Bayes' Rule and Its Use,
Probabilistic Reasoning: Representing Knowledge in an Uncertain Domain, The Semantics of Bayesian Networks, Efficient Representation of Conditional Distributions, Approximate Inference in Bayesian Networks, Relational and First-Order Probability, Other Approaches to Uncertain Reasoning; Dempster- Shafer theory.
Learning: Forms of Learning, Supervised Learning, Learning Decision Trees. Knowledge in Learning: Logical Formulation of Learning, Knowledge in Learning, Explanation-Based Learning, Learning Using Relevance Information, Inductive Logic Programming.

Uncertain knowledge: When the available knowledge has multiple causes leading to multiple effects or incomplete knowledge of causality in the domain. Uncertain knowledge representation: The representation which provides a restricted model of the real system, or has limited expressiveness.

ACTING UNDER UNCERTAINTY

Agents may need to handle uncertainty, whether due to partial observability, nondeterminism, or a combination of the two. An agent may never know for certain what state it's in or where it will end up after a sequence of actions.

We have seen problem-solving agents (Chapter 4) and logical agents (Chapters 7 and 11) designed to handle uncertainty by keeping track of a belief state—a representation of the set of all possible world states that it might be in—and generating a contingency plan that handles every possible eventuality that its sensors may report during execution. Despite its many virtues, however, this approach has significant drawbacks when taken literally as a recipe for creating agent programs:

• When interpreting partial sensor information, a logical agent must consider every logically possible explanation for the observations, no matter how unlikely. This leads to impossible large and complex belief-state representations.
• A correct contingent plan that handles every eventuality can grow arbitrarily large and must consider arbitrarily unlikely contingencies.
• Sometimes there is no plan that is guaranteed to achieve the goal—yet the agent must act. It must have some way to compare the merits of plans that are not guaranteed.

Suppose, for example, that an automated taxi!automated has the goal of delivering a passenger to the airport on time. The agent forms a plan, A90, that involves leaving home 90 minutes before the flight departs and driving at a reasonable speed. Even though the airport is only about 5 miles away, a logical taxi agent will not be able to conclude with certainty that "Plan A90 will get us to the airport in time." Instead, it reaches the weaker conclusion "Plan A90 will get us to the airport in time, as long as the car doesn't break down or run out of gas, and I don't get into an accident, and there are no accidents on the bridge, and the plane doesn't leave early, and no meteorite hits the car, and . . . ." None of these conditions can be deduced for sure, so the plan's success cannot be inferred. This is the qualification problem for which we so far have seen no real solution.

Nonetheless, in some sense A90 is in fact the right thing to do. What do we mean by this? As we discussed in Chapter 2, we mean that out of all the plans that could be executed, A90 is expected to maximize the agent's performance measure (where the expectation is relative to the agent's knowledge about the environment). The performance measure includes getting to the airport in time for the flight, avoiding a long, unproductive wait at the airport, and avoiding speeding tickets along the way. The agent's knowledge cannot guarantee any of these outcomes for A90, but it can provide some degree of belief that they will be achieved. Other plans, such as A180, might increase the agent's belief that it will get to the airport on time, but also increase the likelihood of a long wait. The right thing to do—the rational decision—therefore depends on both the relative importance of various goals and the likelihood that, and degree to which, they will be achieved. The remainder of this section hone these ideas, in preparation for the development of the general theories of uncertain reasoning and rational decisions that we present in this and subsequent chapters.

Probability provides a way of summarizing the uncertainty that comes from our laziness and ignorance, thereby solving the qualification problem. Utility theory says that every state has a degree of usefulness, or utility, to an agent and that the agent will prefer states with higher utility.

Preferences, as expressed by utilities, are combined with probabilities in the general theory of rational decisions called decision theory:

Decision theory = probability theory + utility theory .

The fundamental idea of decision theory is that an agent is rational if and only if it chooses the action that yields the highest expected utility, averaged over all the possible outcomes of the action. This is called the principle of maximum expected utility (MEU). Note that "expected" might seem like a vague, hypothetical term, but as it is used here it has a precise meaning: it means the "average," or "statistical mean" of the outcomes, weighted by the probability of the outcome. We saw this principle in action in Chapter 5 when we touched briefly on optimal decisions in backgammon; it is in fact a completely general principle.

Core vs. Probabilistic AI

- Knowledge Reasoning : work with facts/assertions; develop rules of logical inference Planning: work with applicability/effects of actions; develop searches for actions which achieve goals/avert disasters.
- Expert Systems: develop by hand a set of rules for examining inputs, updating internal states and generating outputs
- Learning approach: use probabilistic models to tune performance based on many data examples.
- Probabilistic AI: emphasis on noisy measurements, approximation in hard cases, learning, algorithmic issues.

o logical assertions ⇒ probability distributions
o logical inference ⇒ conditional probability distributions
o logical operators ⇒ probabilistic generative models

BASIC PROBABILITY NOTATION

Probability notation is an efficient way of writing the probability of events happening or not happening. To do this we use set notation, which is used when working with Venn diagrams. Events are usually notated using capital letters, as well as the use of some greek letters.

Any notation for describing degrees of belief must be able to deal with two main issues:
1) The nature of the sentences to which degrees of belief are assigned
2) The dependence of the degree of belief on the agent's experience.

Propositions: Probability theory typically uses a language that is slightly more expressive than propositional logic. The basic element of the language is the random variable, which can be thought of as referring to a "part" of the world whose "status" is initially unknown.

➢ For example, Cavity might refer to whether my lower left wisdom tooth has a cavity. Random variables play a role similar to that of CSP variables in constraint satisfaction problems and that of proposition symbols in propositional logic. We will always capitalize the names of random variables. For example: $P(a) = 1 - P(\neg a))$.

➢ Each random variable has a domain of values that it can take on. For example, the domain of Cavity might be (true,fail).

➢ For example, Cavity = true might represent the proposition that I do in fact have a cavity in my lower left wisdom tooth. As with CSP variables, random variables are typically divided into three kinds, depending on the type of the
domain:

❖ Boolean random variables, such as Cavity, have the domain (true, false). We will often abbreviate a proposition such as Cavity = true simply by the lowercase name cavity. Similarly, Cavity = false would be abbreviated by 1 cavity.

❖ Discrete random variables, which include Boolean random variables as a special case, take on values from a countable domain. For example, the domain of Weather might be (sunny, rainy, cloudy, snow). The values in the domain must be mutually exclusive and exhaustive. Where no confusion arises, we: will use, for example, snow as an abbreviation for Weather =snow.

❖ Continuous random variables take on values from the: real numbers. The domain can be either the entire real line or some subset such as the interval [0, 1]. For example, the proposition X = 4.02 asserts that the random variable .X has the exact value 4.02.

Elementary propositions, such as Cavity = true and Toothache =false, can be combined to form complex propositions using all the standard logical connectives. For example, Cavity = true A Toothache =false is a proposition to which one may ascribe a degree of belief. As explained in the previous paragraph, this proposition may also be written as cavity ∧ toothache.

Atomic events: The notion of an atomic event is useful in understanding the foundations of probability theory. An atomic event is a complete specification of the state of the world about which the agent is uncertain. It can be thought of as an assignment of particular values to all the variables of which the world is composed. For example, if my world consists of only the Boolean variablesCavity and Toothache, then there are just four distinct atomic events; the proposition

Cavity =false ∧ Toothache = true is one such event. Atomic events have some important properties:

➢ They're Mutually Exclusive-at most one can actually be the case. For Example,cavity a toothache and cavity ∧ -toothache cannot both be the case.

➢ The set of all possible atomic events is exhaustive-at least one must be the case. That is, the disjunction of all atomic events is logically equivalent to true.

➢ Any particular atomic event entails the truth or falsehood of every proposition, whether simple or complex. This can be seen by using the standard semantics for logical connectives. For example, the atomic event cavity ∧ ⌐ toothache entails the truth of cavity and the falsehood of cavity =>toothache.

➢ Any proposition is logically equivalent to the disjunction of all atomic events that entail the truth of the proposition. For example, the proposition cavity is equivalent to disjunction of the atomic events cavity ∧ toothache and cavity ∧⌐toothache.

Prior Probability: The unconditional or prior probability associated with a proposition ‘a’ is the degree of belief accorded to it in the absence of any other information; it is written as P (a). For example, if the prior probability that I have a cavity is 0.1, then

P (Cavity = true) = 0.1 or P (cavity) = 0.1

It is important to remember that P (a) can be used only when there is no other information. As soon as some new information is known, we must reason with the conditional probability of a given that new information.

Now if we talk about the probabilities of all the possible values of a random variable. In that case, we will use an expression such as P (Weather), which denotes a vector of values for the probabilities of each individual state of the weather. So, Instead of writing the four equations

P (Weather = sunny) = 0.7 P (Weather = rain) = 0.2
P (Weather = cloudy) = 0.08 P (Weather = snow) = 0.02.

We may simply write P (Weather) = (0.7, 0.2, 0.08, 0.02).

This statement defines a prior probability distribution for the random variable Weather, We will also use expressions such as P(Weather, Cavity) to denote the probabilities of all combinations of the values of a set of random variable^ In that case, P(Weather, Cavity) can be represented by a 4 x 2 table of probabilities. This is called the joint probability distribution of Weather and Cavity.

➢ A joint probability distribution that covers this complete set is called the full joint probability distribution. For example, if the world consists of just the variables Cavity, Toothache, and Weather, then the full joint distribution is given by

P (Cavity, Toothache, Weather)
This joint distribution can be represented as a 2 x 2 x 4 table with 16 entries. So, any probabilistic query can be answered from the full joint distribution. But, for continuous variables, it is not possible to write out the entire distribution as a table, because there are infinitely many values. Instead, one usually defines the probability that a random variable takes on some value x as a parameterized function of x.

For example, let the random variable X denote tomorrow's maximum temperature in Berkeley. Then the sentence P(X = x) = U [18, 26] (x) X is distributed uniformly between 18 and 26 degrees Celsius. Probability distributions for continuous variables are called probability density functions. Density functions differ in meaning from discrete distributions.

Example, using the temperature distribution given earlier, we find that
P (X = 20.5) = U [18, 26] (2 0.5) ==0 .125/C.

The technical meaning is that the probability that the temperature is in a small region around 20.5 degrees is equal, in the limit, to 0.125 divided by the width of the region in degrees Celsius:

$$\lim_{dx \to 0} P\ (20.5 \leq X \leq 20.5 + dx)/dx = 0.125/C.$$

Conditional probability: Once the agent has obtained some evidence concerning the previously unknown random variables making up the domain, prior probabilities are no longer applicable. Instead, we use conditional or posterior probabilities.

The notation used is P (a / b), where a and b are any proposition. This is read as "the probability of a, given that all we know is b.

For example,
P (cavity / toothache) = 0.8
Indicates that if a patient is observed to have a toothache and no other information is yet available, then the probability of the patient having a cavity will be 0.8. A prior probability, such as P (cavity), can be thought of as a special case of the conditional probability P (cavity /), where the probability is conditioned on no evidence. Conditional probabilities can be defined in terms of unconditional probabilities. The defining equation is which holds whenever P (b) > 0. This equation can also be written as

$$P(a|b) = \frac{P(a \wedge b)}{P(b)}$$

Which holds whenever P (b) > 0. This equation can also be written as
P (a ^ b) = P (a / b) P (b)
Which is called the product rule. It comes from the fact that, for a and b to be true, we need b to be true, and we also need a to be true given b. We can also have it the other way;
P (a ^ b) = P (b / a) P (a)
We can also use the P notation for conditional distributions. P(X / Y) gives the values of P(X = xi / Y = yj) for each possible i, j. As an example consider applying the product rule to each case where the propositions a and b assert particular values of X and Y respectively. We obtain the following equations:

$$P(X=x_1 \wedge Y=y_1) = P(X=x_1|Y=y_1)P(Y=y_1)$$
$$P(X=x_1\ AY=y_2) = P(X=x_1|Y=y_2)P(Y=y_2)$$

We can combine all these into the single equation

P(X, Y) = P(X / Y) P(Y)

It is wrong, because to view conditional probabilities as if they were logical implications with uncertainty added. For example, the sentence P (a / b) = 0.8 cannot be interpreted to mean "whenever b holds, conclude that P (a) is 0.8." Such an interpretation would be wrong on two counts:
➢ First, P(a) always denotes the prior probability of a, not the posterior probability given some evidence;
➢ Second, the statement P (a / b) = 0.8 is immediately relevant just when b is the only available evidence. When additional information c is available, the degree of belief in a is P (a / b ^ c), which might have little relation to P (a /b).
➢ For example, c might tell us directly whether a is true or false. If we examine a patient who complains of toothache, and discover a cavity, then we have additional evidence of cavities, and we conclude (trivially) that P (cavity / toothache ^ cavity) =1.0.

The Axioms Of Probability : So far, we have defined a syntax for propositions and for prior and conditional probability statements about those propositions. Now we must provide some sort of semantics for probability statements. We begin with the basic axioms that serve to define the probability scale and its endpoints:
1. All probabilities are between 0 and 1. For any propositions,

$$0 \leq P(a) \leq 1$$

2. Necessarily true (i.e., valid) propositions have probability I, and necessarily false (i.e., unsatisfiable) propositions have probability0.

$$P(true) = 1 \qquad P(false) = 0\ .$$

Next, we need an axiom that connects the probabilities of logically related propositions. The simplest way to do this is to define the probability of a disjunction as follows:
3. The probability of a disjunction is given by

$$P(a \vee b) = P(a) + P(b) - P(a \wedge b)$$

This rule is easily remembered by noting that the cases where a holds, together with the cases where b holds, certainly cover all the cases where a V b holds; but summing the two sets of cases counts their intersection twice, so we need to subtract Y (a ∧ b).

These three axioms are often called Kolmogorov's axioms.

Using the axioms of probability: We can derive a variety of useful facts from the basic axioms. For example, the familiar rule for negation follows by substituting ~a for b in axiom 3, giving us:

$$\begin{aligned}
P(a \vee \neg a) &= P(a) + P(\neg a) - P(a \wedge \neg a) && \text{(by axiom 3 with } b = \neg a\text{)}\\
P(true) &= P(a) + P(\neg a) - P(false) && \text{(by logical equivalence)}\\
1 &= P(a) + P(\neg a) && \text{(by axiom 2)}\\
P(\neg a) &= 1 - P(a) && \text{(by algebra).}
\end{aligned}$$

The third line of this derivation is itself a useful fact and can be extended from the Boolean case to the general discrete case. Let the discrete variable D have the domain (dl, . . . , d,). Then it is easy to show that

$$\sum_{i=1}^{n} P(D = d_i) = 1 .$$

The probability of a proposition is equal to the sum of the probabilities of the atomic events in which it holds; that is,

$$P(a) = \sum_{e_i \in e(a)} P(e_i) .$$

Why the axioms of probability are reasonable: The axioms of probability can be seen as restricting the set of probabilistic beliefs that an agent can hold. Where a logical agent cannot simultaneously believe A, B, and ~ (A ^ B) for example. In the logical case, the semantic definition of conjunction means that at least one of the three beliefs just mentioned must be false in the world, so it is unreasonable for an agent to believe all three. With probabilities, on the other hand, statements refer not to the world directly, but to the agent's own state of knowledge. Why, then, can an agent not hold the following set of beliefs, which clearly violates axiom 3?

$$\begin{array}{ll}
P(a) = 0.4 & P(a \wedge b) = 0.0\\
P(b) = 0.3 & P(a \vee b) = 0.8
\end{array}$$

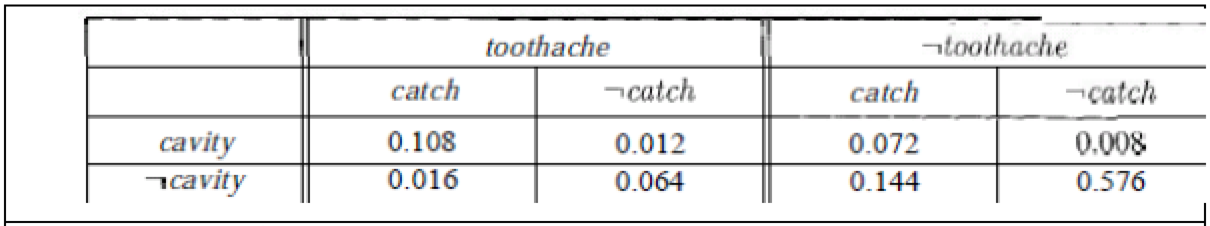
Finetti proved something much stronger: If Agent I expresses a set of degrees of belief that violate the axioms of probability theory then there is a combination of bets by Agent 2 that guarantees that Agent I will lose money every time.

We will not provide the proof of de Finetti's theorem, but we will show an example. Suppose that Agent 1 has the set of degrees of belief from Equation given above. Figure 3.1 shows that if Agent 2 chooses to bet \$4 on a, \$3 on b, and \$2 on ~ (a V b), then Agent 1 always loses money, regardless of the outcomes for a and b.

Agent 1		Agent 2		Outcome for Agent 1			
Proposition	Belief	Bet	Stakes	$a \wedge b$	$a \wedge \neg b$	$\neg a \wedge b$	$\neg a \wedge \neg b$
a	0.4	a	4 to 6	–6	–6	4	4
b	0.3	b	3 to 7	–7	3	–7	3
$a \vee b$	*0.8*	$\neg(a \vee b)$	2 to 8	2	2	2	–8
				–11	–1	–1	–1

INFERENCE USING FULL JOINT DISTRIBUTIONS

Here we will use the full joint distribution as the "knowledge base" from which answers to all questions may be derived. Along the way we will also introduce several useful techniques for manipulating equations involving probabilities. We begin with a very simple example: a domain consisting of just the three Boolean variablesToothache, Cavity, and Catch. The full joint distribution is a 2 x 2 x 2 table as shown In Figure 4.1.



	toothache		*¬toothache*	
	catch	*¬catch*	*catch*	*¬catch*
cavity	0.108	0.012	0.072	0.008
¬cavity	0.016	0.064	0.144	0.576

Figure 4.1 *A full joint distribution for the Toothache, Cavity, and Catch world.*

Now identify those atomic events in which the proposition is true and add up their probabilities. For example, there are six atomic events in which cavity V toothacheholds:

$$P(cavity \vee toothache) = 0.108 + 0.012 + 0.072 + 0.008 + 0.016 + 0.064 = 0.28$$

One common task is to extract the distribution over some subset of variables or a single variable. For example, adding the entries in the first row gives the unconditional or marginal probability of cavity:

$$P(cavity) = 0.108 + 0.012 + 0.072 + 0.008 = 0.2$$

This process is called marginalization, or summing out-because the variables other than Cavity are summed out. We can write the following general marginalization rule for any sets of variables Y and Z:

$$\mathbf{P}(\mathbf{Y}) = \sum_{\mathbf{z}} \mathbf{P}(\mathbf{Y}, \mathbf{z})$$

That is, a distribution over Y can be obtained by summing out all the other variables from any joint distribution containing Y. A variant of this rule involves conditional probabilities instead of joint probabilities, using product rule:

$$\mathbf{P}(\mathbf{Y}) = \sum_{\mathbf{z}} \mathbf{P}(\mathbf{Y}|\mathbf{z})P(\mathbf{z})$$

This rule is called conditioning. Marginalization and conditioning will turn out to be useful rules for all kinds of derivations involving probability expressions.
For example, we can compute the probability of a cavity, given evidence of a toothache, as follows:

$$\begin{aligned} P(cavity|toothache) &= \frac{P(cavity\ A\ toothache)}{P(toothache)} \\ &= \frac{0.108 + 0.012}{0.108 + 0.012 + 0.016 + 0.064} = 0.6 \end{aligned}$$

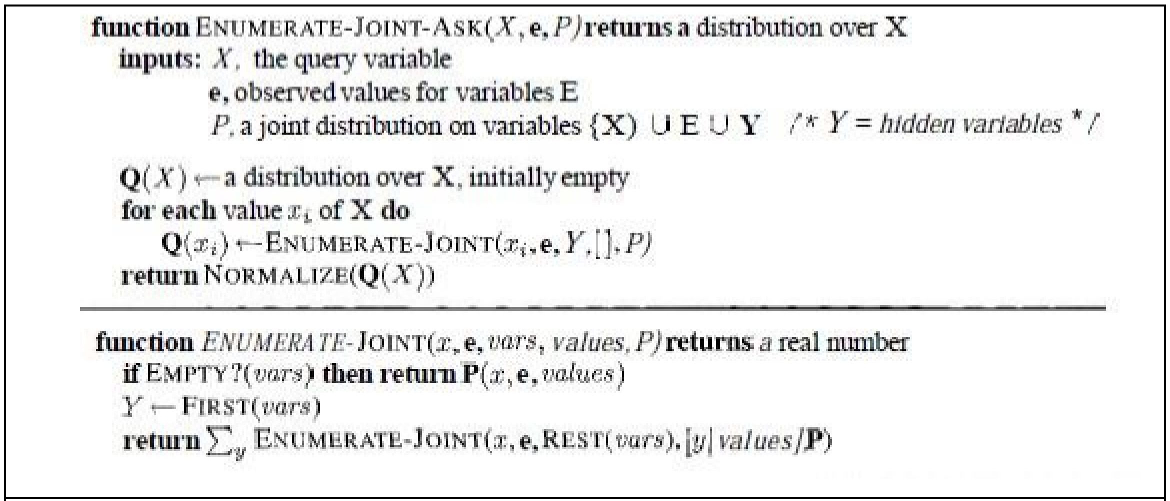
Just to check, we can also compute the probability that there is no cavity, given a toothache:

$$\begin{aligned} P(\neg cavity|toothache) &= \frac{P(\neg cavity \wedge toothache)}{P(toothache)} \\ &= \frac{0.016 + 0.064}{0.108 + 0.012 + 0.016 + 0.064} = 0.4 \end{aligned}$$

In these two calculations the term 1/P (toothache) remains constant, no matter which value of Cavity we calculate. In fact, it can be viewed as a normalization constant for the distribution P(Cavity/ toothache), ensuring that it adds up to1.
We will use a to denote such constants. With this notation, we can write the two preceding equations in one:

$$\begin{aligned} \mathbf{P}(Cavity(toothache) &= a\mathbf{P}(Cavity, toothache) \\ &= a[\mathbf{P}(Cavity, toothache, catch) + \mathbf{P}(Cavity, toothache, \neg catch)] \\ &= \alpha [\langle 0.108, 0.016\rangle + \langle 0.012, 0.064\rangle] = \alpha\ \langle 0.12, 0.08\rangle = \langle 0.6, 0.4\rangle\ . \end{aligned}$$

Algorithm for probabilistic inference:



```
function ENUMERATE-JOINT-ASK($X$, e, $P$) returns a distribution over X
   inputs: $X$, the query variable
           e, observed values for variables E
           $P$, a joint distribution on variables {X} ∪ E ∪ Y   /* $Y$ = hidden variables */

   Q($X$) ← a distribution over X, initially empty
   for each value $x_i$ of X do
       Q($x_i$) ← ENUMERATE-JOINT($x_i$, e, $Y$, [], $P$)
   return NORMALIZE(Q($X$))
```

```
function ENUMERATE-JOINT($x$, e, $vars$, $values$, $P$) returns a real number
   if EMPTY?($vars$) then return P($x$, e, $values$)
   $Y$ ← FIRST($vars$)
   return $\sum_y$ ENUMERATE-JOINT($x$, e, REST($vars$), [$y$| $values$]P)
```

Figure 4.2 An algorithm for probabilistic inference by enumeration of the entries in a full joint: distribution.

Given the full joint distribution to work with, ENUMERATE-JOINT-ASK is a complete algorithm for answering probabilistic queries for discrete variables. It does not scale well, however: For a domain described by n Boolean variables, it requires an input table of size 0(2 pow n) and takes 0(2 pow n) time to process the table. So, the full joint distribution in tabular form is not a practical tool for building reasoning systems.

INDEPENDENCE

Let us expand the full joint distribution in Figure 13.3 by adding a fourth variable, Weather .The full joint distribution then becomes P(Toothache, Catch, Cavity, Weather), which has 32 entries (because Weather has four values). It contains four "editions" of the table shown in Figure 4.1, one for each kind of weather. Here we may ask what relationship these editions have to each other and to the original three-variable table. For example, how are P(toothache, catch, cavity, Weather = cloudy) and P(toothache, catch, cavity) related?

To answer this question is to use the product rule:
P(toothache, catch, cavity, Weather = cloudy) = P (Weather = cloudy / toothache, catch, cavity) P (toothache, catch, cavity). One should not imagine that one's dental problems influence the weather. Therefore, the following assertion seems reasonable:
P(Weather= cloudy / toothache, catch, cavity) = P (Weather= cloudy) ----------- (1)
From this, we can deduce;
P(toothache, catch, cavity, weather = cloudy) = P(Weather = cloudy)P(toothache, catch, cavity).

Similar equation exists for every entry in P(Toothache, Catch, Cavity, Weather). In fact, we can write the general equation;
P(Toothache, Catch, Cavity, Weather) = P(Toothache, Catch, Cavity)P(Weather) .
Thus, the 32-element table for four variables can be constructed from one 8-element table and one four-element table. This decomposition is illustrated schematically in Figure 5.1(a). The property we used in writing Equation is called “independence”.
Independence between propositions a and b can be written as
P (a / b)= P (a) or P(b / a)= P (b) or P (a Ab) = P (a) P (b).
Independence between variables X and Y can be written as follows (again, these are all equivalent):
P (X / Y)= P (X) or P (Y / X)=P(Y) or P (X ,Y)= P (X) P (Y).

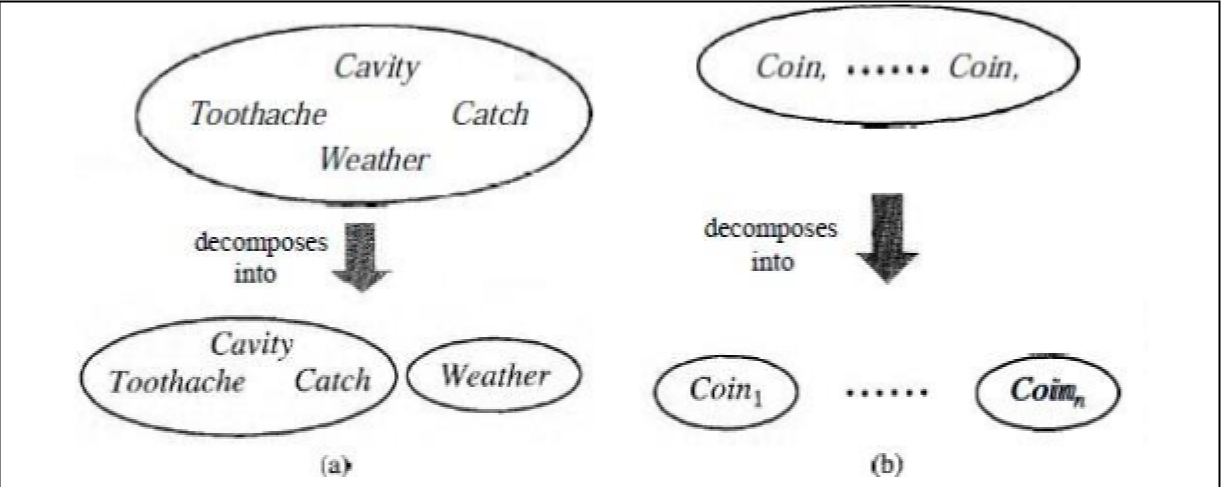


Figure 5.1 Two examples of factoring a large joint distribution into smaller distributions, using absolute independence. (a) Weather and dental problems are independent. (b) Coinflips are independent.

BAYES' RULE AND ITS USE

We defined the product rule and pointed out that it can be written in two forms because of the commutativity of conjunction:

$$P(a \wedge b) = P(a|b)P(b)$$
$$P(a \wedge b) = P(b|a)P(a)$$

Equating the two right-hand sides and dividing by P(a), we get

$$P(b|a) = \frac{P(a|b)P(b)}{P(a)}$$

This equation is known as Bayes' rule (also Bayes' law or Bayes' theorem) .This simple equation underlies all modern AI systems for probabilistic inference. The more general case of multi valued variables can be written in the P notation as;

$$\mathbf{P}(Y|X) = \frac{\mathbf{P}(X|Y)\mathbf{P}(Y)}{\mathbf{P}(X)}$$

Where again this is to be taken as representing a set of equations, each dealing with specific values of The variables. We will also have occasion to use a more general version conditionalized on some background evidence

$$\mathbf{P}(Y|X, \mathbf{e}) = \frac{\mathbf{P}(X|Y, \mathbf{e})\mathbf{P}(Y|\mathbf{e})}{\mathbf{P}(X|\mathbf{e})}$$

Applying Bayes' rule: The simple case: It requires three terms-a conditional probability and two unconditional probabilities-just to compute conditional probability. Bayes' rule is useful in practice because there are many cases where we do have good probability estimates for these three numbers and need to compute the fourth. In a task such as medical diagnosis, we often have conditional probabilities on causal relationships and want to derive a diagnosis. A doctor knows that the disease meningitis causes the patient to have a stiff neck, say, 50% of the time. The doctor also knows some unconditional facts: the prior probability that a patient has meningitis is 1150,000, and the prior probability that any patient has a stiff neck is 1120. Letting s be the proposition that the patient has a stiff neck and m be the proposition that the patient has meningitis, we have;

$$P(s|m) = 0.5$$
$$P(m) = 1/50000$$
$$P(s) = 1/20$$
$$P(m|s) = \frac{P(s|m)P(m)}{P(s)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002\ .$$

That is, we expect only 1 in 5000 patients with a stiff neck to have meningitis. Notice that, even though a stiff neck is quite strongly indicated by meningitis (with probability 0.5), the probability of meningitis in the patient remains small. This is because the prior probability on stiff necks is much higher than that on meningitis. The same process can be applied when using Bayes' rule. We have

$$\mathbf{P}(M|s) = a\langle P(s|m)P(m), P(s|\neg m)P(\neg m)\rangle$$

Thus, in order to use this approach we need to estimate P(s /~ m) instead of P(s) The general form of Bayes' rule with normalization is P(Y / X) = a P(X /Y) P(Y), where a is the normalization constant needed to make the entries in P(Y / X) sum to 1.

Using Bayes' rule: Combining evidence: We have seen that Bayes' rule can be useful for answering probabilistic queries conditioned on one piece of evidence-for example, the stiff neck. In particular, we have argued that probabilistic information is often available in the form P(effect / cause). What happens when we have two or more pieces of evidence? For example, what can a dentist conclude if her nasty steel probe catches in the aching tooth of a patient? If we know the full joint distribution, one can read off the answer:

$$\mathbf{P}(Cavity\ (toothache A\ catch) = a\langle 0.108, 0.016\rangle \approx \langle 0.871, 0.129\rangle$$

We know, however, that such an approach will not scale up to larger numbers of variables. We can try using Bayes' rule to reformulate the problem:

$$\mathbf{P}(Cavity|toothache\ A\ catch) = \alpha\mathbf{P}(toothache \wedge catch|Cavity)\mathbf{P}(Cavity)$$

For this reformulation to work, we need to know the conditional probabilities of the conjunction toothache A catch for each value of Cavity. That might be feasible for just two evidence variables, but again it will not scale up. If there are n possible evidence variables (X rays, diet, oral hygiene, etc.), then there are 2n possible combinations, so f observed values for which we would need to know conditional probabilities. We might as well go back to using the full joint distribution.

Rather than taking this route, we need to find some additional assertions about the domain that will enable us to simplify the expressions. The notion of independence provides a clue, but needs refining. It would be nice if Toothache and Catch were independent, but they are not: if the probe catches in the tooth, it probably has a cavity and that probably causes a toothache. These variables are independent.
Mathematically, this property is written as;

$$\mathbf{P}(toothache\ A\ catch|Cavity) = \mathbf{P}(toothache|Cavity)\mathbf{P}(catch|Cavity)$$

This equation expresses the conditional independence of toothache and catch given Cavity. We can plug it into above equation to obtain the probability of a cavity:

$$\mathbf{P}(Cavity|toothache\ A\ catch) = a\mathbf{P}(toothache|Cavity)\mathbf{P}(catch|Cavity)\mathbf{P}(Cavity).$$

The general definition of conditional independence of two variables X and Y, given a third variable Z is

$$\mathbf{P}(X, Y|Z) = \mathbf{P}(X|Z)\mathbf{P}(Y|Z)$$

In the dentist domain, for example, it seems reasonable to assert conditional independence of the variables Toothache and Catch, given Cavity:

$$\mathbf{P}(Toothache, Catch|Cavity) = \mathbf{P}(Toothache|Cavity)\mathbf{P}(Catch|Cavity)\ .$$

Which asserts independence only for specific values of Toothache and Catch? As with absolute independence in Equation

$$\mathbf{P}(X|Y,Z) = \mathbf{P}(X|Z) \quad \text{and} \quad \mathbf{P}(Y|X,Z) = \mathbf{P}(Y|Z)$$

It turns out that the same is true for conditional independence assertions. For example, given the assertion in Equation, We can derive decomposition as follows:

$$\mathbf{P}(Toothache, Catch, Cavity)$$

$$= \mathbf{P}(Toothache, Catch|Cavity)\mathbf{P}(Cavity) \quad \text{(product rule)}$$

$$= \mathbf{P}(Toothache|Cavity)\mathbf{P}(Catch|Cavity)\mathbf{P}(Cavity)$$

In this way, the original large table is decomposed into three smaller tables.

PROBABILISTIC REASONING:

Probabilistic reasoning Causes of uncertainty: Following are some leading causes of uncertainty to occur in the real world.

- Information occurred from unreliable sources.
- Experimental Errors
- Equipment fault
- Temperature variation
- Climate change.

Probabilistic reasoning is a way of knowledge representation where we apply the concept of probability to indicate the uncertainty in knowledge. In probabilistic reasoning, we combine probability theory with logic to handle the uncertainty.

We use probability in probabilistic reasoning because it provides a way to handle the uncertainty that is the result of someone's laziness and ignorance.

In the real world, there are lots of scenarios, where the certainty of something is not confirmed, such as "It will rain today," "behavior of someone for some situations," "A match between two teams or two players." These are probable sentences for which we can assume that it will happen but are not sure about it, so here we use probabilistic reasoning.

Probability: Probability can be defined as a chance that an uncertain event will occur. It is the numerical measure of the likelihood that an event will occur. The value of probability always remains between 0 and 1 that represent ideal uncertainties.

$0 \leq P(A) \leq 1$, where P(A) is the probability of an event A.
$P(A) = 0$, indicates total uncertainty in an event A.
$P(A) =1$, indicates total certainty in an event A.

We can find the probability of an uncertain event by using the below formula.
$P(\neg A)$ = probability of a not happening event.
$P(\neg A) + P(A) = 1$.
Event: Each possible outcome of a variable is called an event.
Sample space: The collection of all possible events is called sample space.
Random variables: Random variables are used to represent the events and objects in the real world.
Prior probability: The prior probability of an event is probability computed before observing new information.
Posterior Probability: The probability that is calculated after all evidence or information has taken into account. It is a combination of prior probability and new information.
Conditional probability: Conditional probability is a probability of occurring an event when another event has already happened.

Let's suppose, we want to calculate the event A when event B has already occurred, "the probability of A under the conditions of B", it can be written as:

Where P(A^B)= Joint probability of a and B
P(B)= Marginal probability of B.

If the probability of A is given and we need to find the probability of B, then it will be given as:

It can be explained by using the below Venn diagram, where B is occurred event, so sample space will be reduced to set B, and now we can only calculate event A when event B is already occurred by dividing the probability of P(A^B) by P(B).

Example:
In a class, there are 70% of the students who like English and 40% of the students who like English and mathematics, and then what is the percentage of students who like English and also like mathematics?

Solution:
Let, A is an event that a student likes Mathematics
B is an event where a student likes English.
Hence, 57% of the students who like English also like Mathematics.

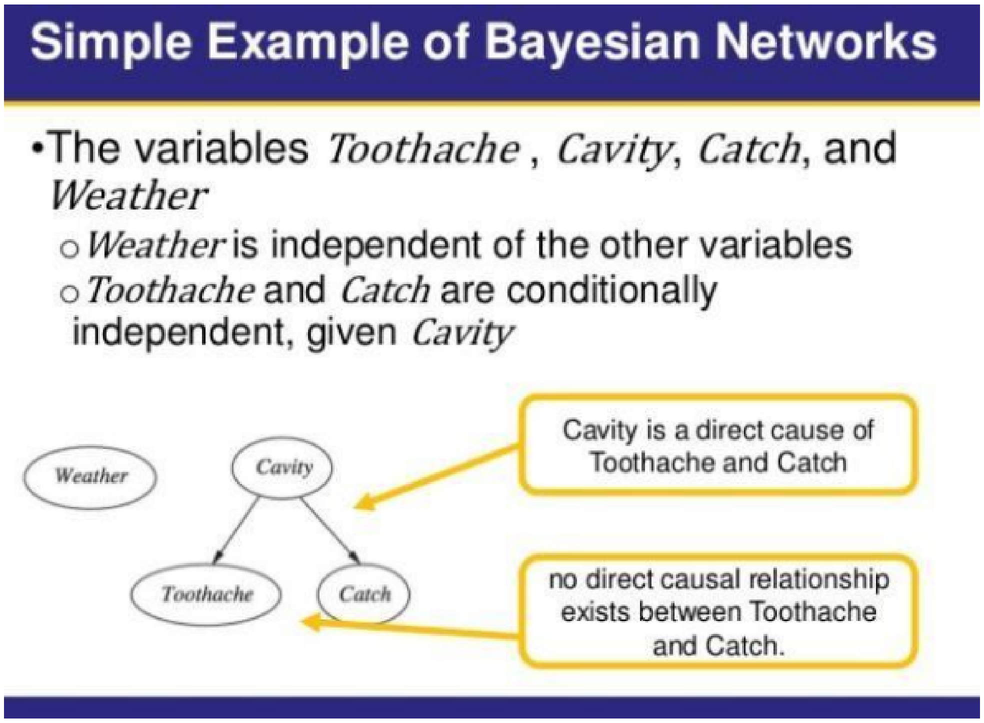
Why Reason Probabilistically?

In many problem domains it isn't possible to create complete, consistent models of the world. Therefore agents (and people) must act in uncertain worlds (which the real world is). Want an agent to make rational decisions even when there is not enough information to prove that an action will work.

Some of the reasons for reasoning under uncertainty:
- True uncertainty. E.g., flipping a coin.
- Theoretical ignorance. There is no complete theory which is known about the problem domain. E.g., medical diagnosis.
- Laziness. The space of relevant factors is very large, and would require too much work to list the complete set of antecedents and consequences. Furthermore, it would be too hard to use the enormous rules that resulted.
- Practical ignorance. Uncertain about a particular individual in the domain because all of the information necessary for that individual has not been collected.
- Probability theory will serve as the formal language for representing and reasoning with uncertain knowledge.

REPRESENTING KNOWLEDGE IN AN UNCERTAIN DOMAIN

A directed graph in which each node is annotated with quantitative probability information Definition
1. Each node corresponds to a random variable, which may be discrete or continuous .
2. A set of directed links or arrows connects pairs of nodes. (If there is an arrow from node X to node Y , X is said to be a parent of Y).
3. The graph has no directed cycle.
4. Each node Xi has a conditional probability distribution P(Xi|Parents(Xi)) that quantifies the effect of the parents on the node.



Simple Example of Bayesian Networks

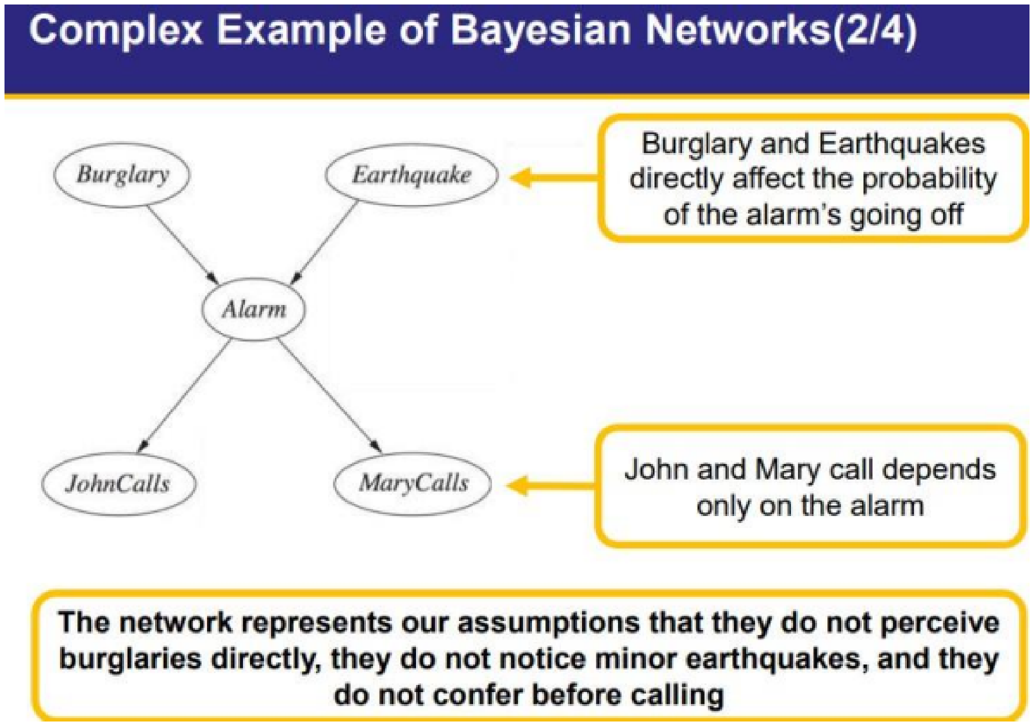
- The variables *Toothache* , *Cavity*, *Catch*, and *Weather*
  - *Weather* is independent of the other variables
  - *Toothache* and *Catch* are conditionally independent, given *Cavity*

Weather
Cavity
Toothache
Catch
Cavity is a direct cause of Toothache and Catch
no direct causal relationship exists between Toothache and Catch.

Complex Example of Bayesian Networks

The variables Burglary, Earthquake, Alarm, MaryCalls and JohnCalls

- New burglar alarm installed at home
- Fairly reliable at detecting a burglary
- Responds on occasion to minor earthquakes
- Two neighbors, John and Mary
- They call you at work when they hear the alarm
- John nearly always calls when he hears the alarm
- But sometimes confuses the telephone ringing
- Mary likes rather loud music and misses the alarm



Complex Example of Bayesian Networks(2/4)

Burglary
Earthquake
Alarm
JohnCalls
MaryCalls
Burglary and Earthquakes directly affect the probability of the alarm's going off
John and Mary call depends only on the alarm

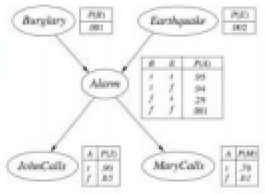
The network represents our assumptions that they do not perceive burglaries directly, they do not notice minor earthquakes, and they do not confer before calling

THE SEMANTICS OF BAYESIAN NETWORKS

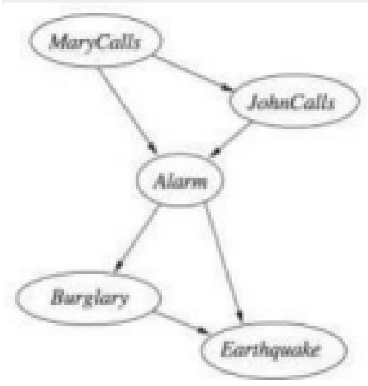
The two ways to understand the meaning of Bayesian Networks oTo see the network as a representation of the joint probability distribution To be helpful in understanding how to construct networks, oTo view it as an encoding of a collection of conditional independence statements To be helpful in designing inference procedures.

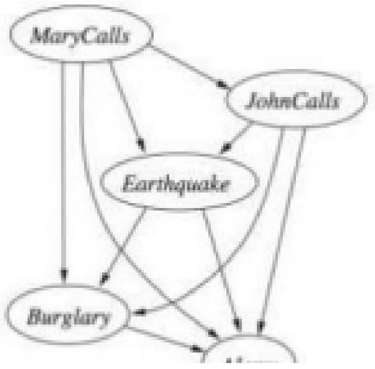
Representing the full joint distribution

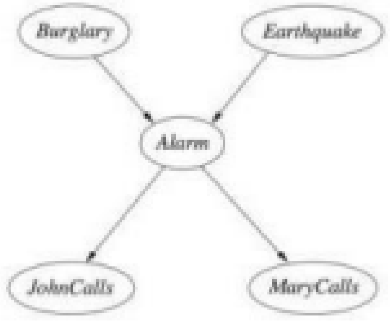
$$\begin{aligned}&P(j,m,a,\neg b,\neg e)\\&= P(j|parents(j))P(m|parents(m)P(a|parents(a))\\&\quad P(\neg b\ |parents(\neg b))P(\neg e\ |parents(\neg e))\\&= P(j|a)P(m|a)P(a|\neg b\wedge\neg e)P(\neg b)P(\neg e)\\&= 0.90\times0.70\times0.001\times0.999\times0.998=0.000628\end{aligned}$$



1. A method for constructing Bayesian networks (1)
   - How to construct a GOOD Bayesian network
   - Full Joint DistributionPx1,.....,xn=i=1nPxixi−1,.....,x1)Px1,.....,xn=i=1nPxiParent(xi))
   - Correct representation only if each node is conditionally independent of its other predecessors in the node ordering, given its parents The parents of node Xi should contain all those nodes in X1,..,Xi−1 that directly influences Xi.
2. A method for constructing Bayesian networks(2)
   - Suppose we have completed the network in Figure except for choices of parents for MaryCalls
   - MaryCallsis certainly influenced by whether there is a Burglary or an Earthquake, but not directly influenced.
   - Also, given the state of the alarm, whether John calls has no influence on Mary's calling P(MaryCalls | JohnCalls, Alarm, Earthquake, Burglary) = P(MaryCalls |Alarm)
3. Compactness and node ordering •Bayesian network can often be far more compact than the full joint distribution
   - It may not be worth the additional complexity in the network for the small gain in accuracy.
   - The correct procedure for adding a node is to first add the root cause first and then give the variables that they affect.
4. Compactness and node ordering
   - We will get a compact Bayesian network only if we choose the node ordering well.
   - What happens if we happen to choose the wrong order? MaryCalls→JohnCalls → Alarm→Burglary→EarthquakeMaryCalls→JohnCalls→Earthquake→Burglary→AlarmBurglary→Earthquake →Alarm→MaryCalls→JohnCalls







THE SEMANTICS OF BAYESIAN NETWORKS:

The syntax of a Bayes net consists of a directed acyclic graph with some local probability information attached to each node. The semantics defines how the syntax corresponds to a joint distribution over the variables of the network.

The two ways to understand the meaning of Bayesian Networks

- To see the network as a representation of the joint probability distribution
- To be helpful in understanding how to construct networks
- To view it as an encoding of a collection of conditional independence statements
- To be helpful in designing inference procedures

EFFICIENT REPRESENTATION OF CONDITIONAL DISTRIBUTIONS:

A conditional distribution is a distribution of values for one variable that exists when you specify the values of other variables. This type of distribution allows you to assess the dispersal of your variable of interest under specific conditions, hence the name.

- CPT cannot handle large number or continuous value variables.
- Relationships between parents and children are usually describable by some proper canonical distribution.
- Use the deterministic nodes to demonstrate relationships.
- Values are specified by some function.
- nondeterminism(no uncertainty) Ex. X = f(parents(X))
- Can Be logical NorthAmerica ↔ Canada ∨ US ∨ Mexico
- OrnumercialWater level = inflow + precipitation – outflow –evaporation
- Uncertain relationships can be characterized by noisy logical relationships.
- Ex. noisy-OR relation.
- Logical OR with probability oEx. Cold ∨ Flu ∨ Malaria → Fever In the real world, catching a cold sometimes does not induce fever.
- There is some probability of catching a cold and having a fever. Noisy-OR oAll possible causes are listed. (the missing can be covered by leak node) oCompute probability from the inhibition probability

$$P(x_i \mid parents(X_i)) = 1 - \prod_{\{j:X_j = true\}} q_j \,,$$

APPROXIMATE INFERENCE IN BAYESIAN NETWORKS:

The methodology of BN has two main components: inference and learning. Inference aims to estimate the posterior distribution of the state variables based on evidence. Usually this evidence is the observation y of nodes Y, thus the inference is to calculate the posterior probability distribution P (X | Y = y) .

- Difficult to calculate multiply connected networks
- It is essential to consider approximate inference methods
- Monte Carlo algorithms
- Randomized sampling algorithms
- Two families of algorithms: direct sampling and Markov chain sampling
- Apply to the computation

RELATIONAL AND FIRST-ORDER PROBABILITY MODELS:

A probabilistic relational model (PRM) or a relational probability model is a model in which the probabilities are specified on the relations, independently of the actual individuals. Different individuals share the probability parameters. A parametrized random variable is either a logical atom or a term.First-order probabilistic models allow us to model situations in which a random variable in the first-order model may have a large and varying number of parent variables in the ground ("unrolled") model.

- Bayesian networks are essentially propositional logic.
- The set of random variables is fixed and finite.
- However, if the number becomes large, intractable.
- Need another method to represent the model The set of first-order models is infinite.
- Use database semantics instead called "Relational Probability models".
- Make unique name assumptions and assume domain closure.
- Like first-order logic oConstantoFunctionoPredicatesymbols

OTHER APPROACHES TO UNCERTAIN REASONING:

- Rule-based methods for uncertain reasoning
- Emerged from logical inference
- Require 3 desirable properties
- Locality : If A B, we can conclude B given evidence A without worrying about any other rules. But in probabilistic systems, we need to consider all evidence.
- Detachment : If we can derive B, we can use it without caring how it was derived.
- Truth-functionality : truth value of complex sentences can be computed from the truth of the components
- Probability combination does not work this way

DEMPSTER SHAFER THEORY

The Dempster-Shafer Theory was given by Arthur P. Dempster in 1967 and his student Glenn Shafer in 1976. This theory was released because of the following reason:-

- Bayesian theory is only concerned about single evidence.
- Bayesian probability cannot describe ignorance.

DST is an evidence theory, it combines all possible outcomes of the problem. Hence it is used to solve problems where there may be a chance that a piece of different evidence will lead to some different result.
The uncertainty in this model is given by:-

1. Consider all possible outcomes.
2. Belief will lead to belief in some possibility by bringing out some evidence. (What is this supposed to mean?)
3. Plausibility will make evidence compatible with possible outcomes.

Example: Let us consider a room where four people are present, A, B, C, and D. Suddenly the lights go out and when the lights come back, B has been stabbed in the back by a knife, leading to his death. No one came into the room and no one left the room. We know that B has not committed suicide. Now we have to find out who the murderer is.
To solve these there are the following possibilities:

- Either {A} or {C} or {D} has killed him.
- Either {A, C} or {C, D} or {A, D} have killed him.
- Or the three of them have killed him i.e; {A, C, D}

- None of them have killed him {o} (let’s say).

There will be possible evidence by which we can find the murderer by the measure of plausibility.
Using the above example we can say:
Set of possible conclusion (P): {p1, p2….pn}
where P is a set of possible conclusions and cannot be exhaustive, i.e. at least one (p) I must be true.
(p)I must be mutually exclusive.
The Power Set will contain 2n elements where n is the number of elements in the possible set.
For eg:-
If P = { a, b, c}, then Power set is given as
{o, {a}, {b}, {c}, {a, d}, {d ,c}, {a, c}, {a, c ,d }}= 23 elements.
Mass function m(K): It is an interpretation of m({K or B}) i.e; it means there is evidence for {K or B} which cannot be divided among more specific beliefs for K and B.
Belief in K: The belief in element K of Power Set is the sum of masses of the element which are subsets of K. This can be explained through an example
Let's say K = {a, d, c}
Bel(K) = m(a) + m(d) + m(c) + m(a, d) + m(a, c) + m(d, c) + m(a, d, c)
Plausibility in K: It is the sum of masses of the set that intersects with K.
i.e; Pl(K) = m(a) + m(d) + m(c) + m(a, d) + m(d, c) + m(a, c) + m(a, d, c)

Characteristics of Dempster Shafer Theory:

- It will ignore the part such that the probability of all events aggregate to 1. (What is this supposed to mean?)
- Ignorance is reduced in this theory by adding more and more evidence.
- Combination rule is used to combine various types of possibilities.

Advantages:

- As we add more information, the uncertainty interval reduces.
- DST has a much lower level of ignorance.
- Diagnose hierarchies can be represented using this.
- Person dealing with such problems is free to think about evidence.

Disadvantages:

- In this, computation effort is high, as we have to deal with 2n sets

LEARNING: An agent is learning if it improves its performance on future tasks after making observations about the world. Learning can range from the trivial, as exhibited by jotting down a phone number, to the profound, as exhibited by Albert Einstein, who inferred a new theory of the universe. Here we will concentrate on one class of learning problem, which seems restricted but actually has vast applicability: from a collection of input–output pairs, learn a function that predicts the output for new inputs.

Why would we want an agent to learn? If the design of the agent can be improved, why wouldn’t the designers just program in that improvement to begin with? There are three main reasons.

- First, the designers cannot anticipate all possible situations that the agent might find itself in. For example, a robot designed to navigate mazes must learn the layout of each new maze it encounters.
- Second, the designers cannot anticipate all changes over time; a program designed to predict tomorrow’s stock market prices must learn to adapt when conditions change from boom to bust.
- Third, sometimes human programmers have no idea how to program a solution themselves. For example, most people are good at recognizing the faces of family members, but even the best programmers are unable to program a computer to accomplish that task, except by using learning algorithms.

FORMS OF LEARNING

Any component of an agent can be improved by learning from data. The improvements, and the techniques used to make them, depend on four major factors:

- Which component is to be improved.
- What prior knowledge the agent already has.
- What representation is used for the data and the component.
- What feedback is available to learn from.

Components to be learned
The components of these agents include:

1. A direct mapping from conditions on the current state to actions.
2. A means to infer relevant properties of the world from the percept sequence.
3. Information about the way the world evolves and about the results of possible actions the agent can take.
4. Utility information indicating the desirability of world states.
5. Action-value information indicating the desirability of actions.
6. Goals that describe classes of states whose achievement maximizes the agent's utility.

Each of these components can be learned. Consider, for example, an agent training to become a taxi driver.

1. Every time the instructor shouts "Brake!" the agent might learn a condition–action rule for when to brake;
2. The agent also learns every time the instructor does not shout. By seeing many camera images that it is told contain buses, it can learn to recognize them.
3. By trying actions and observing the results—for example, braking hard on a wet road—it can learn the effects of its actions.
4. Then, when it receives no tip from passengers who have been thoroughly shaken up during the trip, it can learn a useful component of its overall utility function.

Representation and prior knowledge

We have seen several examples of representations for agent components: propositional and first-order logical sentences for the components in a logical agent; Bayesian networks for the inferential components of a decision-theoretic agent, and so on. Effective learning algorithms have been devised for all of these representations. Here (and most of current machine learning research) covers inputs that form a factored representation—a vector of attribute values—and outputs that can be either a continuous numerical value or a discrete value.

There is another way to look at the various types of learning. We say that learning a (possibly incorrect) general function or rule from specific input–output pairs is called inductive learning. We will see that we can also do analytical or deductive learning: going from a known general rule to a new rule that is logically entailed, but is useful because it allows more efficient processing.

Feedback to learn from

There are three types of feedback that determine the three main types of learning: In unsupervised learning the agent learns patterns in the input even though no explicit feedback is supplied. The most common unsupervised learning task is clustering: detecting potentially useful clusters of input examples. For example, a taxi agent might gradually develop a concept of "good traffic days" and "bad traffic days" without ever being given labeled examples of each by a teacher.
In reinforcement learning the agent learns from a series of reinforcements—rewards or punishments. For example, the lack of a tip at the end of the journey gives the taxi agent an indication that it did something wrong. The two points for a win at the end of a chess game tells the agent it did something right. It is up to the agent to decide which of the actions prior to the reinforcement were most responsible for it.

In supervised learning the agent observes some example input–output pairs and learns a function that maps from input to output.

- In component 1 above, the inputs are percepts and the output are provided by a teacher who says “Brake!” or “Turn left.”
- In component 2, the inputs are camera images and the outputs again come from a teacher who says “that’s a bus.”
- In 3, the theory of braking is a function from states and braking actions to stopping distance in feet. In this case the output value is available directly from the agent’s percepts (after the fact); the environment is the teacher.

In practice, these distinctions are not always so crisp. In semi-supervised learning we are given a few labeled examples and must make what we can of a large collection of unlabeled examples. Even the labels themselves may not be the oracular truths that we hope for. Imagine that you are trying to build a system to guess a person’s age from a photo. You gather some labeled examples by snapping pictures of people and asking their age. That’s supervised learning. But in reality some of the people lied about their age. It’s not just that there is random noise in the data; rather the inaccuracies are systematic, and to uncover them is an unsupervised learning problem involving images, self-reported ages, and true (unknown) ages. Thus, both noise and lack of labels create a continuum between supervised and unsupervised learning.

SUPERVISED LEARNING

Supervised learning is the machine learning task of inferring a function from labeled training Data. Moreover, The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and the desired output value (also called the supervisory signal). Training set A training set of data used in various areas of information science to discover potentially predictive relationships. Training sets used in artificial intelligence, machine learning, genetic programming, intelligent systems, and statistics.

In all these fields, a training set has much the same role and is often used in conjunction with a test set. Testing set: A test set is a set of data used in various areas of information science to assess the strength and utility of a predictive relationship. Moreover, Test sets are used in artificial intelligence, machine learning, genetic programming, and statistics. In all these fields, a test set has much the same role.

Accuracy of classifier: Supervised learning In the fields of science, engineering, industry, and statistics. The accuracy of a measurement system is the degree of closeness of measurements of a quantity to that quantity’s actual (true) value.

Sensitivity analysis: Supervised learning Similarly, Local Sensitivity as correlation coefficients and partial derivatives can only be used, if the correlation between input and output is linear.

Regression: Supervised learning In statistics, regression analysis is a statistical process for estimating the relationships among variables. Moreover, It includes many techniques for modeling and analyzing several variables. When they focus on the relationship between a dependent variable and one or more independent variables. More specifically, regression analysis helps one understand how the typical value of the dependent variable (or ‘criterion variable’) changes when any one of the independent variables varies. Moreover, While the other independent variables were fixed.

LEARNING DECISION TREES

Decision tree induction is one of the simplest and yet most successful forms of machine learning. We first describe the representation of the hypothesis space and then show how to learn a good hypothesis. A decision tree represents a function that takes as input a vector of attribute values and returns a “decision”—a single output value. The input and output values can be discrete or continuous. For now we will concentrate on problems where the inputs have discrete values and the output has exactly two possible values; this is Boolean classification, where each example input will be classified as true (a positive example) or false (a negative example).

A decision tree reaches its decision by performing a sequence of tests. Each internal node in the tree corresponds to a test of the value of one of the input attributes, Ai, and the branches from the node are labeled with the possible values of the attribute, Ai =vik. Each leaf node in the tree specifies a value to be returned by the function. The decision tree representation is natural for humans; indeed, many "How To" manuals (e.g., for car repair) are written entirely as a single decision tree stretching over hundreds of pages. As an example, we will build a decision tree to decide whether to wait for a table at a restaurant. The aim here is to learn a definition for the goal predicate WillWait . First we list the attributes that we will consider as part of the input:

1. Alternate: whether there is a suitable alternative restaurant nearby.
2. Bar : whether the restaurant has a comfortable bar area to wait in.
3. Fri/Sat: true on Fridays and Saturdays.
4. Hungry: whether we are hungry.
5. Patrons: how many people are in the restaurant (values are None, Some, and Full).
6. Price: the restaurant's price range ($, $$, $$$).
7. Raining: whether it is raining outside.
8. Reservation: whether we made a reservation.
9. Type: the kind of restaurant (French, Italian, Thai, or burger).
10. WaitEstimate: the wait estimated by the host (0–10 minutes, 10–30, 30–60, or >60).

Note that every variable has a small set of possible values; the value of WaitEstimate, for example, is not an integer, rather it is one of the four discrete values 0–10, 10–30, 30–60, or >60. The decision tree usually used by one of us (SR) for this domain is shown in Figure 18.2. Notice that the tree ignores the Price and Type attributes. Examples are processed by the tree starting at the root and following the appropriate branch until a leaf is reached. For instance, an example with Patrons =Full and WaitEstimate =0–10 will be classified as positive (i.e., yes, we will wait for a table).

Logical Representation of a Path
r[Patrons(r,full) □ Wait_Estimate(r, 10-30) □ Hungry(r, yes)] □ Will_Wait(r)

Expressiveness of Decision Trees

- Any Boolean function can be written as a decision tree
- E.g., for Boolean functions, truth table row → path to leaf:
- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless f nondeterministic in x) but it probably won't generalize to new examples
- Prefer to find more compact decision treesLimitations
- Can only describe one object at a time.
- Some functions require an exponentially large decision tree.
- E.g. Parity function, majority function
- Decision trees are good for some kinds of functions, and bad for others.
- There is no one efficient representation for all kinds of functions.

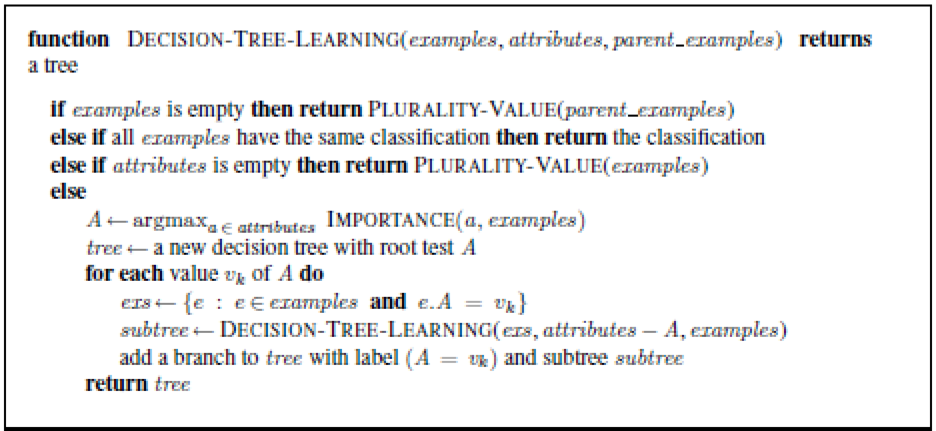
Principle Behind the Decision-Tree-Learning Algorithm
- Uses a general principle of inductive learning often calledOckham's razor: "The most likely hypothesis is the simplest one that is consistent with all observations."
- Decision trees can express any function of the inputattributes.

Decision Tree Learning Algorithm:

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree Choosing an attribute tests:
- Idea: a good attribute splits the examples into subsets that are (ideally)"all positive" or "all negative"
- Patrons? is a better choice Attribute-based representations
- Examples described by attribute values (Boolean, discrete,continuous)
- E.g., situations where I will/won't wait for a table:
- Classification of examples is positive (T) or negative (F) Using information theory
- To implement Choose-Attribute in the DTL algorithm
- Information Content(Entropy): I(P(v1), ... , P(vn)) = Σi=1 -P(vi) log2P(vi)

For a training set containing p positive examples and n negative examples:
A chosen attribute A divides the training set E into subsets E1, ... ,Ev according to their values for A, where A has v distinct values. Information Gain (IG) or reduction in entropy from the attribute test: remainder (A),

- Choose the attribute with the largestIG
- For the training set, p = n = 6, I(6/12, 6/12) = 1 bit
- Patrons has the highest IG of all attributes and so is chosen by the DTL algorithm as the root



```
function DECISION-TREE-LEARNING(examples, attributes, parent_examples) returns
a tree

  if examples is empty then return PLURALITY-VALUE(parent_examples)
  else if all examples have the same classification then return the classification
  else if attributes is empty then return PLURALITY-VALUE(examples)
  else
      A ← argmax_{a ∈ attributes} IMPORTANCE(a, examples)
      tree ← a new decision tree with root test A
      for each value v_k of A do
          exs ← {e : e ∈ examples and e.A = v_k}
          subtree ← DECISION-TREE-LEARNING(exs, attributes − A, examples)
          add a branch to tree with label (A = v_k) and subtree subtree
      return tree
```

Figure 18.5 The decision-tree learning algorithm. The function IMPORTANCE is described in Section 18.3.4. The function PLURALITY-VALUE selects the most common output value among a set of examples, breaking ties randomly.

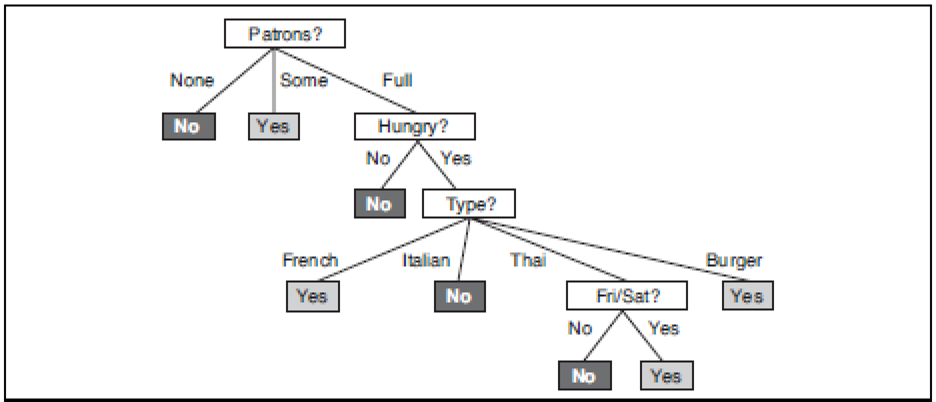


Figure 18.6 The decision tree induced from the 12-example training set.

Assessing the performance of the learning algorithm:

- A learning algorithm is good if it produces hypotheses that do a good job of predicting the classifications of unseen examples
- Test the algorithm's prediction performance on a set of new examples, called a testset.

- Patrons has the highest IG of all attributes and so is chosen by the DTL algorithm as the root
- Choose the attribute with the largest IG
- For the training set, p = n = 6, I(6/12, 6/12) = 1 bit
- Assessing the performance of the learning algorithm:
- A learning algorithm is good if it produces hypotheses that do a good job of predicating the classifications of unseen examples

KNOWLEDGE IN LEARNING: In all of the approaches to learning described in the previous chapter, the idea is to construct a function that has the input–output behavior observed in the data. In each case, the learning methods can be understood as searching a hypothesis space to find a suitable function, starting from only a very basic assumption about the form of the function, such as "second-degree polynomial" or "decision tree" and perhaps a preference for simpler hypotheses. Doing this amounts to saying that before you can learn something new, you must first forget (almost) everything you know.
Here, we study learning methods that can take advantage of prior knowledge about the world. In most cases, the prior knowledge is represented as general first-order logical theories; thus for the first time we bring together the work on knowledge representation and learning.

A LOGICAL FORMULATION OF LEARNING

Current-best-hypothesis search
The idea behind current-best-hypothesis search is to maintain a single hypothesis, and to adjust it as new examples arrive in order to maintain consistency. The extension of the hypothesis must be increased to include new examples. This is called generalization.

```
function CURRENT-BEST-LEARNING(examples, h) returns a hypothesis or fail
if examples is empty then
    return h
e←FIRST(examples)
if e is consistent with h then
    return CURRENT-BEST-LEARNING(REST(examples), h)
else if e is a false positive for h then
    for each hin specializations of h consistent with examples seen so far do
        h←CURRENT-BEST-LEARNING(REST(examples), h)
        if h = fail then return h
else if e is a false negative for h then
    for each hin generalizations of h consistent with examples seen so far do
        h←CURRENT-BEST-LEARNING(REST(examples), h)
        if h = fail then return h
return fail
```

The extension of the hypothesis must be decreased to exclude the example. This is called specialization. Least-commitment search Backtracking arises because the current-best-hypothesis approach has to choose a particular hypothesis as its best guess even though it does not have enough data yet to be sure of the choice. What we can do instead is to keep around all and only those hypotheses that are consistent with all the data so far. Each new example will either have no effect or will get rid of some of the hypotheses. One important property of this approach is that it is incremental: one never has to go back and reexamine the old examples.

Boundary Set :
We also have an ordering on the hypothesis space, namely, generalization/specialization. This is a partial ordering, which means that each boundary will not be a point but rather a set of hypotheses called a boundary set. The great thing is that we can represent the entire G-SET version space using just two boundary sets: a most general boundary (the G-set) and a most S-SET specific boundary (the S-set). Everything in between is guaranteed to be consistent with the examples.

The members Si and Gi of the S- and G-sets.
For each one, the new example may be a false positive or a false negative.

1. False positive for Si: This means Si is too general, but there are no consistent specializations of Si (by definition), so we throw it out of the S-set.
2. False negative for Si: This means Si is too specific, so we replace it by all its immediate generalizations, provided they are more specific than some members of G.
3. False positive for Gi: This means Gi is too general, so we replace it by all its immediate specializations, provided they are more general than some members of S.
4. False negative for Gi: This means Gi is too specific, but there are no consistent generalizations of Gi (by definition) so we throw it out of the G-set

EXPLANATION-BASED LEARNING

Explanation-based learning is a method for extracting general rules from individual observations. Memoization The technique of memoization has long been used in computer science to speed up programs by saving the results of computation. The basic idea of memo functions is to accumulate a database of input output pairs; when the function is called, it first checks the database to see whether it can avoid solving the problem from scratch. Explanation-based learning takes this a good deal further, by creating general rules that cover an entire class of cases.

Basic EBL process works as follows:

1. Given an example, construct a proof that the goal predicate applies to the example using the available background knowledge
2. In parallel, construct a generalized proof tree for the variablized goal using the same inference steps as in the original proof.
3. Construct a new rule whose left-hand side consists of the leaves of the proof tree and whose right hand side is the variablized goal (after applying the necessary bindings from the generalized proof).
4. Drop any conditions from the left-hand side that are true regardless of the values of the variables in the goal.

Three factors involved in the analysis of efficiency gains from EBL:

1. Adding large numbers of rules can slow down the reasoning process, because the inference mechanism must still check those rules even in cases where they do not yield a solution. In other words, it increases the branching factor in the search space.
2. To compensate for the slowdown in reasoning, the derived rules must offer significant increases in speed for the cases that they do cover. These increases come about mainly because the derived rules avoid dead ends that would otherwise be taken, but also because they shorten the proof itself.
3. Derived rules should be as general as possible, so that they apply to the largest possible set of cases.

LEARNING USING RELEVANCE INFORMATION

The learning algorithm is based on a straightforward attempt to find the simplest determination consistent with the observations. A determination P ' Q says that if any examples match on P, then they must also match on Q. A determination is therefore consistent with a set of examples if every pair that matches on the predicates on the left-hand side also matches on the goal predicate.
An algorithm for finding a minimal consistent determination function MINIMAL-CONSISTENT-DET(E,A) returns a set of attributes inputs:

```
E, a set of examples
A, a set of attributes, of size n
for i = 0 to n do
for each subset Ai of A of size i do
if CONSISTENT-DET?(Ai,E) then return Ai
```

```
function CONSISTENT-DET?(A,E) returns a truth value
inputs: A, a set of attributes
E, a set of examples
local variables: H, a hash table
for each example e in E do
if some example in H has the same values as e for the attributes A
but a different classification then return false
store the class of e in H, indexed by the values for attributes A of the example e
return true
```

Given an algorithm for learning determinations, a learning agent has a way to construct a minimal hypothesis within which to learn the target predicate. For example, we can combine MINIMAL- CONSISTENT-DET with the DECISION-TREE-LEARNING algorithm.

This yields a relevance-based decision-tree learning algorithm RBDTL that first identifies a minimal set of relevant attributes and then passes this set to the decision tree algorithm for learning.

INDUCTIVE LOGIC PROGRAMMING

Inductive logic programming (ILP) combines inductive methods with the power of first-order representations, concentrating in particular on the representation of hypotheses as logic programs. It has gained popularity for three reasons.

1. ILP offers a rigorous approach to the general knowledge-based inductive learning problem.
2. It offers complete algorithms for inducing general, first-order theories from examples, which can therefore learn successfully in domains where attribute-based algorithms are hard to apply.
3. Inductive logic programming produces hypotheses that are (relatively) easy for humans to read

The object of an inductive learning program is to come up with a set of sentences for the Hypothesis such that the entailment constraint is satisfied. Suppose, for the moment, that the agent has no background knowledge: Background is empty. Then one possible solution we would need to make pairs of people into objects. Top-down inductive learning methods The first approach to ILP works by starting with a very general rule and gradually specializing it so that it fits the data. This is essentially what happens in decision-tree learning, where a decision tree is gradually grown until it is consistent with the observations. To do ILP we use first-order literals instead of attributes, and the hypothesis is a set of clauses instead of a decision tree.

Three kinds of literals

1. Literals using predicates
2. Equality and inequality literals
3. Arithmetic comparisons

Inductive learning with inverse deduction

The second major approach to ILP involves inverting the normal deductive proof process. Inverse resolution is based INVERSE on the observation.
Recall that an ordinary resolution step takes two clauses C1 and C2 and resolves them to produce the resolvent C.
An inverse resolution step takes a resolvent C and produces two clauses C1 and C2, such that C is the result of resolving C1 and C2.
Alternatively, it may take a resolvent C and clause C1 and produce a clause C2 such that C is the result of resolving C1 and C2.
A number of approaches to taming the search implemented in ILP systems

1. Redundant choices can be eliminated
2. The proof strategy can be restricted
3. The representation language can be restricted
4. Inference can be done with model checking rather than theorem proving
5. Inference can be done with ground prepositional clauses rather than in first-order logic.