11

Probabilistic Reasoning

The knowledge representation languages and reasoning algorithms discussed in previous chapters allowed an agent to reason with various forms of incomplete knowledge. Defaults gave agents the ability to make assumptions about completeness of their knowledge and to withdraw these assumptions if they contradicted new information. Incompleteness of the agent's knowledge was manifested by multiple answer sets, some of which contained proposition p whereas others contained its negation. This was the case for, say, the program consisting of the awareness axiom,

$$p(a) \ or \ \neg p(a)$$

for $p(a)$. Another form of incompleteness, exhibited by program

$$q(a). \quad q(b). \quad p(b).$$

corresponded to the case when the agent associated with the program was simply not aware of $p(a)$. Technically, the program has one answer set containing neither $p(a)$ nor $\neg p(a)$. In both these cases, propositions could only have three truth values: *true*, *false*, and *unknown*. In this chapter we discuss an augmentation of our knowledge representation languages with constructs for representing a finer gradation of the strength of an agent's beliefs. The resulting language preserves all the power of ASP, including the ability to represent defaults, nonmonotonically update the knowledge base with new information, define rules recursively, and so on, while allowing an agent to do sophisticated probabilistic reasoning.

11.1 Classical Probabilistic Models

Mathematicians and computer scientists have developed a number of mathematical models capturing various aspects of quantifying the beliefs of a rational agent. The oldest and most developed collection of such models

can be found in probability theory – a branch of mathematics with a several hundred year history of development and applications. Even though probability theory is a well-developed branch of mathematics, the methodology of its use for knowledge representation and even the intuitive meaning of basic notions of the theory are still matters of contention. In this book we view probabilistic reasoning as *commonsense reasoning about the degree of an agent's beliefs in the likelihood of different events*. In what follows we clarify the meaning of this statement. Let us start with a simple example.

Example 11.1.1. *(Lost in the Jungle)*

Imagine yourself lost in a dense jungle. A group of natives has found you and offered to help you survive, provided you can pass their test. They tell you they have an Urn of Decision from which you must choose a stone at random. (The urn is sufficiently wide for you to easily get access to every stone, but you are blindfolded so you cannot cheat.) You are told that the urn contains nine white stones and one black stone. Now you must choose a color. If the stone you draw matches the color you chose, the tribe will help you; otherwise, you can take your chances alone in the jungle. (The reasoning of the tribe is that they do not wish to help the exceptionally stupid or the exceptionally unlucky.)

It does not take knowledge of probability theory to realize that you will have a much better chance of obtaining their help if you choose white for the color of the stone. All that is required is common sense. Your line of reasoning may be as follows: "Suppose I choose white. What would be my chances of getting help? They are the same as the chances of drawing a white stone from the urn. There are nine white stones out of a possible ten. Therefore, my chances of picking a white stone and obtaining help are $\frac{9}{10}$." The number $\frac{9}{10}$ can be viewed as the degree of belief that help will be obtained if you select white. To double-check yourself (since your life may depend on your choice), you might consider what would happen if you choose black. In this case, to get help you need to draw the black stone. The chances of that are one out of ten ($\frac{1}{10}$). Clearly, you should choose white.

To mathematically study this type of reasoning, one may use the classical notion of a **probabilistic model**. Such models consist of

- a finite set Ω whose elements are referred to as **possible worlds**. Intuitively possible worlds correspond to possible outcomes of random experiments we attempt to perform (such as drawing a stone from the urn);

- a function μ from possible worlds of Ω to the set of real numbers such that

$$\text{for all } W \in \Omega \; \mu(W) \geq 0 \text{ and}$$

$$\sum_{W \in \Omega} \mu(W) = 1.$$

Function μ is referred to as a **probabilistic measure**. Intuitively $\mu(W)$ quantifies the agent's degree of belief in the likelihood of the outcomes of random experiments represented by W.

A function

$$P : 2^{\Omega} \rightarrow [0, 1]$$

such that

$$P(E) = \sum_{W \in E} \mu(W)$$

for any $E \subseteq \Omega$ is called a **probability function** induced by μ; it characterizes the degree of belief in the likelihood of the actual outcome of the agent's experiment belonging to E.

Of course, this simple definition only works for situations in which there is only a finite number of possible worlds. Modern probability theory generalizes this notion to the infinite case and discovers many nontrivial properties of probability and other similar measures. But even the simple definition described earlier allows us to obtain nontrivial properties of probability and to successfully use the resulting calculus to reason about the likelihood of a particular outcome of random events.

In logic-based probability theory, possible worlds are often identified with logical interpretations, and a set E of possible worlds is often represented by a formula F such that $W \in E$ iff W is a model of F. In this case the probability function may be defined on propositions

$$P(F) =_{def} P(\{W : W \in \Omega \text{ and } W \text{ is a model of } F\}).$$

Now let us go back to our traveler saying, "Suppose I choose white," and performing the mental experiment of randomly drawing a stone from the urn. To understand a mathematical model that predicts his chances of getting help, we need to come up with a collection Ω of possible worlds that correspond to possible outcomes of this random experiment. There are, of course, many ways of doing this task. We start with a fairly detailed representation that includes individual stones. To be able to refer to stones, we

enumerate them by integers from 1 to 10 starting with the black stone. The possible world describing the effect of the traveler drawing stone number 1 from the urn looks like this:

$$W_1 = \{select\_color = white, draw = 1, \neg help\}.$$

Drawing the second stone results in possible world

$$W_2 = \{select\_color = white, draw = 2, help\}$$

etc. We will have 10 possible worlds, 9 of which contain $help$. To define a probabilistic measure μ on these possible worlds, we use the so-called **Principle of Indifference** – a commonsense rule that states that *possible outcomes of a random experiment are assumed to be equally probable if we have no reason to prefer one of them to any other*. This rule suggests that $\mu(W) = 0.1$ for any possible world $W \in \Omega$. According to our definition of probability function P, the probability that the outcome of the experiment contains $help$ is 0.9. A similar argument for the case in which the traveler selects black gives 0.1. The probabilistic model allowed us to produce the expected result.

The reader, of course, noticed that the most difficult part of this argument was setting up a probabilistic model of our domain – especially the selection of possible worlds. This observation raises a typical computer science question: "*How can possible worlds of a probabilistic model be found and represented?*" In what follows we discuss one of the possible ways of doing this: We simply encode the knowledge presented to us in a story by a program in knowledge representation language **P-log** – an extension of ASP and/or CR-Prolog that allows us to combine logical and probabilistic knowledge. Answer sets of a program of the new language are identified with possible worlds of the domain.

11.2 The Jungle Story in P-log

We now show how to encode the jungle story in P-log. (We call the resulting program Π_{jungle}.) The syntax of P-log is similar to that of ASP but unlike standard ASP, the signature of a P-log program is sorted – each function symbol comes with sorts for its parameters and its range. Π_{jungle} has two sorts, $stones$ and $colors$, consisting of objects of our domain. The sorts are defined by the following P-log statements:

$$stones = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}.$$

$$colors = \{black, white\}.$$

For the sake of making it easy to talk about the stones, we assume that the first one is black and the others are white. (It really does not matter which one is black.) This is exactly what we did in setting up our first model. To record this we introduce a function $color$ mapping stones into colors:

$$color : stones \rightarrow colors.$$

(The usual mathematical notation is used for the function declarations.) The values of the function are given by the following P-log rules.

$$color(1) = black.$$
$$color(X) = white \leftarrow X \neq 1.$$

Note that the only difference between rules of P-log and ASP is the form of the atoms.

To proceed, let us call the random process of selecting one of these stones from the urn $draw$. In P-log this process corresponds to the following declaration:

$$draw : stones.$$
$$random(draw).$$

The first statement says that $draw$ is a zero-arity function that takes its values from the sort $stones$. The second statement, referred to as a **random selection rule**, states that *normally*, in a (mental or physical) experiment conducted by the reasoner, the values for $draw$ are selected at random. Random processes of P-log are often referred to as **random attributes**. Finally we need to declare two zero-arity functions

$$select\_color : colors$$

and

$$help : boolean$$

where $boolean$ is a predefined sort with the usual values *true* and *false*. The tribal laws are then represented by these rules:

$$\begin{aligned} help \quad &\leftarrow draw = X, \\ &\quad\ color(X) = C, \\ &\quad\ select\_color = C. \\ \neg help &\leftarrow draw = X, \\ &\quad\ color(X) = C, \\ &\quad\ select\_color \neq C. \end{aligned}$$

Here $help$ and $\neg help$ are used as shorthand for $help = true$ and $help = false$, respectively. Also, $select\_color \neq C$ is understood as shorthand for $\neg(select\_color = C)$.

This concludes the construction of our first P-log program, Π_{jungle}. It represents the traveler's knowledge about his situation. Recall that informally the traveler starts by asking himself the following question: *"Suppose I choose white. What would be my chances of getting help?"* To answer this question the reasoner first expands the program Π_{jungle} by the statement

$$select\_color = white.$$

The resulting program, $\Pi_{jungle}(white)$, contains one random attribute, $draw$.

Each possible outcome of random selection for $draw$ defines one possible world. The precise definition of possible worlds of the program is given in the next section, but intuitively one should be able to compute them. If the result of our random selection were 1, then this world would be

$$W_1 = \{draw = 1, select\_color = white, \neg help\}.^1$$

(Since $color(1) = black$ and $select\_color = white$ are facts of the program, the result follows immediately from the definition of $help$.) If the result of our random selection were 2, then the world determined by this selection would be

$$W_2 = \{draw = 2, select\_color = white, help\}.$$

Similarly for stones from 3 to 10. Possible worlds defined by our program are exactly those introduced in our previous solution. The new representation, however, is more elaboration tolerant and is closer to the informal description of the story. The semantics of P-log uses the indifference principle to automatically compute the probabilistic measure of every possible world and hence the probabilities of the corresponding events. Since in this case all worlds are equally plausible, the ratio of possible worlds in which arbitrary statement F is true to the number of all possible worlds gives the probability of F. Hence the probability of $help$ defined by the program $\Pi_{jungle}(white)$ is $\frac{9}{10}$.

Now the reasoner considers a program $\Pi_{jungle}(black)$ obtained from Π_{jungle} by adding the statement

$$select\_color = black.$$

A similar argument shows that the probability of getting help in this situation is $\frac{1}{10}$. Clearly, the first choice is preferable.

[1] Actually this possible world also contains other atoms that do not depend on the value of $draw$ such as sorts, statements of the form $color(1) = black$, $color(2) = white$, etc. However, they are the same in all the possible worlds, and we do not show them in our representation.

11.3 Syntax and Semantics of P-log

In this section we use the jungle program to informally explain the syntax and semantics of P-log. Since P-log is a sorted language, its programs contain definitions of sorts and declarations of functions. The atoms of P-log are properly typed expressions of the form $a(\bar{t}) = y$, where y is a constant from the range of a function symbol a or a variable. A term of the form $a(\bar{t})$ is called an **attribute**. The negation of an atom, $\neg(a(\bar{t}) = y)$, is written as $a(\bar{t}) \neq y$. As usual, atoms and their negations are referred to as literals. In addition to declarations and the regular ASP rules formed from P-log literals, the P-log programmer may declare some of the attributes to be random.

It is not difficult to show that program Π_{jungle}, as well as any other P-log program Π, can be translated into a regular ASP program, say $\tau(\Pi)$, with the same logical meaning. In fact the logical semantics of Π are defined via the semantics of $\tau(\Pi)$.

Definition 11.3.1. *($\tau(\Pi)$)*
For every attribute $a(\bar{t})$ with $range(a) = \{y_1, \ldots, y_n\}$, the mapping τ

- *represents the sort information by a corresponding set of atoms; e.g., $s = \{1, 2\}$ is turned into facts $s(1)$ and $s(2)$;*
- *replaces every occurrence of an atom*

  $$a(\bar{t}) = y$$

  by

  $$a(\bar{t}, y),$$

  and expands the program by rules of the form

  $$\neg a(\bar{t}, Y_2) \leftarrow a(\bar{t}, Y_1), Y_1 \neq Y_2;$$

- *replaces every occurrence of $a(\bar{t}, true)$ and $a(\bar{t}, false)$ by $a(\bar{t})$ and $\neg a(\bar{t})$, respectively, and removes double negation $\neg\neg$, which might have been introduced by this operation;*
- *replaces every rule of the form*

  $$random(a(\bar{t})) \leftarrow body$$

  by

  $$a(\bar{t}) = y_1 \text{ or } \ldots \text{ or } a(\bar{t}) = y_n \leftarrow body,\ not\ intervene(a(\bar{t})) \tag{11.1}$$

  where intervene *is a new predicate symbol;*

The translation is justified by the intuitive reading of $random(a(\bar{t}))$ *that says that,* under normal circumstances, during the construction of a possible set of beliefs, the reasoner associated with the program must randomly select exactly one value of $a(\bar{t})$ from a's range. *Actually, P-log allows more-general random selection rules that have the form:*

$$random(a(\bar{t}) : \{X : p(X)\}) \leftarrow body.$$

The set $\{X : p(X)\}$ *is often referred to as the* **dynamic range** *of* $a(\bar{t})$. *The rule limits the selection of the value of* $a(\bar{t})$ *to elements of* a*'s range that satisfy property* p. *For each such rule,* τ *creates an additional ASP rule:*

$$\leftarrow a(\bar{t}, Y),\ not\ p(Y),\ not\ intervene(a(\bar{t})). \qquad (11.2)$$

- *grounds the resulting program by replacing variables with elements of the corresponding sorts.*

This completes the construction of $\tau(\Pi)$.

Last, we note that P-log's random selection rules may be preceded by terms used as rule names. In fact, because of technical reasons, some P-log solvers require such names. They are not, however, used in this book.

Definition 11.3.2. *Collections of atoms from answer sets of* $\tau(\Pi)$ *are called* **possible worlds** *of* Π. *(We use the new term simply to stay close to the traditional terminology of probability theory.)*

Declarations and rules of the form described so far form the logical part of a P-log program Π. This part defines sets of beliefs of a rational reasoner associated with Π or, in the language of probability, possible worlds of Π.

In addition, program Π may contain so-called *pr*-atoms[2] describing the likelihood of particular random events, as well as constructs describing observations and actions of the reasoner associated with the program. The syntax and semantics of these constructs are defined later.

Meanwhile we proceed with defining the probabilistic semantics of a P-log program. We first define a probabilistic measure on its possible worlds. This measure, a real number from the interval $[0, 1]$, represents the degree of a reasoner's belief that a possible world W matches a true state of the

[2] Here *pr* stands for *probabilistic*.

world. Zero means that the agent believes that the possible world does not correspond to the true state; one corresponds to the certainty that it does. *The probability of a set of possible worlds is the sum of the probabilistic measures of its elements.* This definition is easily expanded to define the probability of propositions.[3]

Definition 11.3.3. *The* **probability of a proposition** *is the sum of the probabilistic measures of possible worlds in which this proposition is true.*

Our next task is to explain the definition of a probabilistic measure of possible worlds of a P-log program. The jungle example shows how this can be done using the indifference principle for a program with one random selection rule. The following example demonstrates how the indifference principle is used to define a probabilistic measure for programs with multiple random selection rules.

Example 11.3.1. *(Dice)*

Mike and John each own a die. Each die is rolled once. We would like to estimate the chance that the sum of the rolls is high (i.e., greater than 6).

To do that we construct a P-log program Π_{dice} encoding our knowledge of the domain. (The program contains some knowledge that is not necessary for making the desired prediction, but is useful for further modifications of the example.) Π_{dice} has a signature Σ containing the sort $dice$ consisting of the names of the two dice, d_1 and d_2; a (random) attribute $roll$ mapping each die to the value it indicates when thrown, which is an integer from 1 to 6; an attribute $owner$ mapping each die to a person; relation $high$, which holds when the sum of the rolls of the two dice is greater than 6. The corresponding declarations look like this:

$$
\begin{array}{l}
die = \{d_1, d_2\}. \\
score = \{1, 2, 3, 4, 5, 6\}. \\
person = \{mike, john\}. \\
roll : die \rightarrow score. \\
random(roll(D)). \\
owner : die \rightarrow person. \\
high : boolean.
\end{array}
$$

[3] Here by "proposition" we mean ground atoms or formulas obtained from these atoms using connectives $\neg$, $\wedge$, and or .

The regular part of the program consists of the following rules:

$$
\begin{aligned}
&owner(d_1) = mike.\\
&owner(d_2) = john.\\
&high \leftarrow roll(d_1) = Y_1,\\
&\qquad\quad roll(d_2) = Y_2,\\
&\qquad\quad (Y_1 + Y_2) > 6.\\
&\neg high \leftarrow roll(d_1) = Y_1,\\
&\qquad\quad\ \ roll(d_2) = Y_2,\\
&\qquad\quad\ \ (Y_1 + Y_2) \leq 6.
\end{aligned}
$$

The translation, $\tau(\Pi_{dice})$, of our program into ASP looks like this:

$$
\begin{aligned}
&die(d_1).\\
&die(d_2).\\
&score(1..6).\\
&person(mike).\\
&person(john).\\
&roll(D,1)\ or\ \ldots\ or\ roll(D,6) \leftarrow\ not\ intervene(roll(D)).\\
&\neg roll(D,Y_2) \leftarrow roll(D,Y_1),\\
&\qquad\qquad\qquad Y_1 \neq Y_2.\\
&owner(d_1, mike).\\
&owner(d_2, john).\\
&\neg owner(D,P_2) \leftarrow owner(D,P_1),\\
&\qquad\qquad\qquad\quad P_1 \neq P_2.\\
&high \leftarrow roll(d_1,Y_1),\\
&\qquad\quad roll(d_2,Y_2),\\
&\qquad\quad (Y_1 + Y_2) > 6.\\
&\neg high \leftarrow roll(d_1,Y_1),\\
&\qquad\qquad roll(d_2,Y_2),\\
&\qquad\qquad (Y_1 + Y_2) \leq 6.
\end{aligned}
$$

Notice that, in accordance with the definition of τ, function declarations are simply omitted.

By computing answer sets of $\tau(\Pi_{dice})$ we obtain 36 possible worlds of our program – with each world corresponding to a possible selection of values for random attributes $roll(d_1)$ and $roll(d_2)$; i.e.,

$$
\begin{aligned}
W_1 &= \{roll(d_1) = 1, roll(d_2) = 1, high = false, \ldots\},\\
&\vdots\\
W_{36} &= \{roll(d_1) = 6, roll(d_2) = 6, high = true, \ldots\}.
\end{aligned}
$$

(The possible worlds only differ by atoms formed by $roll(d_1)$, $roll(d_2)$, and $high$; other atoms, such as $die(d_1)$ and $owner(d_1, mike)$, are the same in all possible worlds and are not listed.) The selection for d_1 has six possible outcomes that, by the principle of indifference, are equally likely. The same goes for d_2. *The mechanisms controlling the way the agent selects the values of* $roll(d_1)$ *and* $roll(d_2)$ *during the construction of his beliefs are independent from each other.* This independence justifies the definition of the probabilistic measure of a possible world containing $roll(d_1) = i$ and $roll(d_2) = j$ as the product of the agent's degrees of belief in $roll(d_1) = i$ and $roll(d_2) = j$.[4] Hence the measure of a possible world containing $roll(d_1) = i$ and $roll(d_2) = j$ for every possible i and j is $\frac{1}{6} \times \frac{1}{6} = \frac{1}{36}$. The probability $P_{\Pi_{dice}}(high)$ is the sum of the measures of the possible worlds that satisfy $high$. Since $high$ holds in 21 worlds, the probability $P_{\Pi_{dice}}(high)$ of $high$ being true is $\frac{7}{12}$. If the reasoner associated with Π_{dice} had to bet on the outcome of the game, betting on $high$ would be better.

Note that the jungle example did not require the use of the product rule because it contained only one random selection rule. Since the dice example has two such rules, the product rule is necessary.

Suppose now that we learned from a reliable source that, although the die owned by John is fair, the die owned by Mike is biased. The source conducted multiple statistical experiments that allowed him to conclude that, on average, Mike's die will roll a 6 in one out of four rolls. To incorporate this type of knowledge in our program, we need some device allowing us to express the numeric probabilities of possible values of random attributes. This is expressed in P-log through **causal probability statements**, or pr**-atoms**. A pr-atom takes the form

$$pr_r(a(\bar{t}) = y|_c\ B) = v \tag{11.3}$$

where $a(\bar{t})$ is a random attribute, B is a conjunction of literals, r is the name of the random selection rule used to generate the values of $a(\bar{t})$, $v \in [0, 1]$, and y is a possible value of $a(\bar{t})$. If the rule generating the values of $a(\bar{t})$ is uniquely determined by the program, then its name, r, can be omitted. The

[4] In probability theory two events A and B are called independent if the occurrence of one does not affect the probability of another. Mathematically, this intuition is captured by the following definition: Events A and B are independent (with respect to probability function P) if $P(A \wedge B) = P(A) \times P(B)$. For example, the event d_1 *shows a* 5 is independent of d_2 *shows a* 5, whereas the event *the sum of the scores on both dice shows a* 5 is dependent on the event d_1 *shows a* 5.

"causal stroke" '$|_c$' and the "rule body" B may also be omitted in case B is empty.

The statement has a rather sophisticated informal reading. First it says that *if the value of $a(\bar{t})$ is generated by rule r, and B holds, then the probability of the selection of y for the value of $a(\bar{t})$ is v*. In addition, it indicates the potential *existence of a direct causal relationship between B and the possible value of $a(\bar{t})$*. Sometimes relations expressed by causal probability statements of P-log are referred to as causal probabilities. In this book we only briefly discuss causal probability and its properties. For an in-depth study of this relation, defined in terms of acyclic directed graphs and probability tables, one can consult Judea Pearl's book on causality (2000). This remarkable book contains a wealth of information about causality, probability, and their applications to various types of causal and probabilistic reasoning.

Example 11.3.2. *(Biased Dice)*
Information about the fairness of Mike's die after the bias was discovered is expressed by pr-atom

$$pr(roll(D) = 6 \mid_c owner(D) = mike) = \frac{1}{4}.$$

Let us now demonstrate how these statements can be used to define the probabilistic measures of possible worlds of the resulting program Π_{biased}. Clearly programs Π_{dice} and Π_{biased} have the same possible worlds. First consider a possible world

$$W = \{roll(d_1) = 6, roll(d_2) = 1, high = true, \dots\}$$

Die d_1 belongs to Mike, and hence, by the earlier causal probability statement, the likelihood of this die showing 6 as the result of the corresponding selection is $\frac{1}{4}$. By the principle of indifference the likelihood of d_2 showing 1 is $\frac{1}{6}$. As explained before, the measure of W is the product of our degrees of belief in $roll(d_1) = 6$ and $roll(d_2) = 1$. Hence the measure of W_1 is $\frac{1}{24}$. Similarly for other worlds in which $roll(d_1) = 6$. What about the measure of each world in which Mike's die shows 5 or less? Since we have no information about the likelihood of these five possible outcomes of Mike's die, by the principle of indifference, we have that our degree of belief in each such outcome is $\frac{(1-\frac{1}{4})}{5} = \frac{3}{20}$. So the measure of each such world is $\frac{3}{20} \times \frac{1}{6} = \frac{1}{40}$. Now we know the probabilistic measure of every possible world of Π_{biased}. To compute the probability of the sum of the points on both dice being greater than 6, we need to sum up the measures of the worlds satisfying $high$. It is easy to check that $P_{\Pi_{biased}}(high) = \frac{5}{8}$.

To better understand the intuition behind our definition of a probabilistic measure, it may be useful to consider an intelligent agent associated with the program in the process of constructing possible worlds. Suppose the agent has already constructed a part V of a (not yet completely constructed) possible world W, and suppose that V satisfies the precondition of some random selection rule r defining the value of some random attribute $a(\bar{t})$. The agent can continue construction by considering a mental experiment associated with r – a random selection of the possible value of $a(\bar{t})$. If y is a possible outcome of this experiment, then the agent may continue construction by adding the atom $a(\bar{t}) = y$ to V. To define the probabilistic measure μ of the possible world W under construction, we need to know the likelihood of y being the outcome of r, which we call the **causal probability of atom $a(t) = y$ in W.** The agent can obtain this information from a pr-atom $pr(a(\bar{t}) = y \mid_c B) = v$ of our program with B satisfied by W or compute it by using the principle of indifference. In the former case the corresponding causal probability is v. In the latter case the agent needs to consider the collection R of possible outcomes of experiment r. For example if $y \in R$, there is no pr-atom assigning probability to outcomes of R, and $|R| = n$, then the causal probability of $a(\bar{t}) = y$ in W will be $\frac{1}{n}$.

In general, to compute the probabilistic measure of W, one needs to consider values $y_1, \ldots, y_n$ of random attributes $a_1(\bar{t}_1), \ldots, a_n(\bar{t}_n)$ that determine a possible world W. The **unnormalized probabilistic measure**, $\hat{\mu}$ of W, is defined as the product of causal probabilities of atoms $a_1(\bar{t}_1), \ldots, a_n(\bar{t}_n)$ in W. The probabilistic measure $\mu(W)$ of a possible world W is the unnormalized probabilistic measure of W divided by the sum of the unnormalized probabilistic measures of all possible worlds of Π, i.e.,

$$\mu(W) =_{def} \frac{\hat{\mu}(W)}{\sum_{W_i \in \Omega} \hat{\mu}(W_i)}.$$

For this definition to be correct, program Π must satisfy several natural conditions. There should be possible worlds (i.e., program $\tau(\Pi)$ should be consistent). In the process of construction of a possible world, a random attribute $a(\bar{t})$ should be defined by a unique random selection rule. There are a few others. It is not difficult to show that if these conditions are satisfied then the function μ defined earlier is indeed a probabilistic measure.

Finally, we introduce the two remaining syntactic constructs of P-log – statements recording observations of the results of random experiments

$$obs(a(\bar{t}) = y) \tag{11.4}$$

$$obs(a(\bar{t}) \neq y) \tag{11.5}$$

and of deliberate interventions in these experiments, setting the value of the corresponding attribute to some given y,

$$do(a(\bar{t}) = y). \tag{11.6}$$

Here $a(\bar{t})$ is a random attribute, and y is a possible value of $a(\bar{t})$. For instance, statement $obs(roll(d_1) = 6)$ says that the random experiment consisting of rolling the first die shows 6; $do(roll(d_1) = 6)$ says that, instead of throwing the die at random, it was deliberately put on the table showing 6. To give formal semantics of these statements we need to expand our translation $\tau(\Pi)$. This is done as follows. First, for every atom of the form (11.4), (11.5), and (11.6), we write, respectively,

$$obs(a(\bar{t}, y))$$
$$\neg obs(a(\bar{t}, y))$$
$$do(a(\bar{t}, y)).$$

Next, we add the following rules:

$$\leftarrow obs(a(\bar{t}, y)), \neg a(\bar{t}, y) \tag{11.7}$$
$$\leftarrow \neg obs(a(\bar{t}, y)), a(\bar{t}, y) \tag{11.8}$$
$$a(\bar{t}, y) \leftarrow do(a(\bar{t}, y)) \tag{11.9}$$
$$intervene(a(\bar{t})) \leftarrow do(a(\bar{t}, y)). \tag{11.10}$$

The first two rules eliminate possible worlds of the program that fail to satisfy the observation. The third rule makes sure that interventions affect their intervened-on variables in the expected way. The fourth rule defines the relation $intervene$ that, for each intervention, cancels the randomness of the corresponding attribute (see rule [11.1]). This completes the definition of $\tau(\Pi)$ for programs with observations and deliberate interventions.

As before, possible worlds of Π are defined as collections of atoms from answer sets of $\tau(\Pi)$. The definition of probability does not change either, but one should pay careful attention to the process of normalization and the possible change of an attribute from random to deterministic.

We illustrate the semantics of the new statements by revisiting our dice example.

Example 11.3.3. *(Dice Revisited)*
Consider the P-log program from Example 11.3.1 expanded by the observation that John rolled a 3:

$$obs(roll(d_2) = 3).$$

It is not difficult to see that the resulting program $\Pi_{dice} \cup \{obs(roll(d_2) = 3)\}$, has six possible worlds, which are obtained from possible worlds of Π_{dice} by removing those containing $roll(d_2) = y$ where $y \neq 3$. For each such world W, the unnormalized probabilistic measure is

$$\hat{\mu}(W) = \frac{1}{6} \times \frac{1}{6} = \frac{1}{36}$$

After normalization we have

$$\mu(W) = \frac{1/36}{6/36} = \frac{1}{6}$$

and hence, because $high$ is true in three worlds, we have

$$P_{\Pi_{dice}}(high) = \frac{3}{6} = \frac{1}{2}$$

(We hope the reader noticed the importance of normalization.)

Conditioning on observations has been extensively studied in classical probability theory. If P is a probability function, then the conditional probability $P(A|B)$ is defined as $P(A \wedge B)/P(B)$, provided $P(B)$ is not 0. Intuitively, $P(A|B)$ is understood as the probability of a formula A with respect to the background theory and a set B of all of the agent's additional observations of the world. The new evidence B simply eliminates the possible worlds that do not satisfy B. It can be shown that, under reasonable conditions on program Π, the same formula can be used to compute the corresponding P-log probability, i.e., we have

$$P_{\Pi}(A|B) = P_{\Pi \cup obs(B)}(A).$$

Let us now consider program $\Pi_{dice} \cup \{do(roll(d_2) = 3)\}$, which is identical to $\Pi_{dice} \cup \{obs(roll(d_2) = 3)\}$ except that the observation is replaced by the deliberate action of putting John's die on the table showing three dots. Possible worlds of the new program are the same as that of $\Pi_{dice} \cup \{obs(roll(d_2) = 3)\}$, but now, by rule (11.10), $intervene(roll(d_2))$ is true and hence the value of attribute $roll(d_2)$ is not selected at random by rule (11.1). Instead it is deterministically assigned the value 3, and is not used in the computation of the probabilistic measure. For each such world W, the normalized probabilistic measure

$$\mu(W) = \frac{1}{6}.$$

With d_2 deliberately set to 3, $high$ holds in three out of six possible worlds; hence,

$$P_{\Pi_{dice} \cup \{do(roll(d_2)=3)\}}(high) = \frac{1}{2}.$$

Even though the computations of the corresponding probabilities for observations and deliberate actions were different, they led to the same results. This is not always the case. The substantial differences between the results of updating your knowledge base by observations and interventions are discussed in later examples.

There is one more concept left to demonstrate, and that is the effect of the dynamic range on possible worlds. When an attribute has a dynamic range, it means that the sample from which its values are drawn can change with time. For example, if we draw a card from a deck and then draw another card without replacing the first, our sample has changed.

Example 11.3.4. *(Aces in Succession)*
Suppose we are interested in the probability that two consecutive draws from a deck of 52 cards result in both cards being aces. We begin by representing the relevant knowledge. One possible way to do that is to enumerate our cards

$$card = \{1 \ldots 52\}.$$

Without loss of generality we can assume that the first four cards are aces:

$$ace = \{1, 2, 3, 4\}.$$

We also need two tries and a random attribute $draw$ that maps each try into the card selected by this draw:

$$try = \{1, 2\}.$$
$$draw : try \rightarrow card.$$

If we were to simply draw a card from the deck and then put it back, we could define $draw$ as random ($random(draw(T))$). However, because the card we drew is not returned to the deck, the second selection is made from a smaller deck. To encode this we use a dynamic range and define the randomness of $draw$ as follows:

$$draw(T) : \{C : available(C, T)\}.$$

We define $available(C, T)$ to be the set of cards without the one we drew in the previous try:

$$\begin{aligned} available(C, 1) &\leftarrow card(C). \\ available(C, T+1) &\leftarrow available(C, T), \\ &\quad\ draw(T) \neq C. \end{aligned}$$

Finally we add

$$\begin{aligned} two\_aces \leftarrow\ & draw(1) = Y1, \\ & draw(2) = Y2, \\ & 1 \leq Y1 \leq 4, \\ & 1 \leq Y2 \leq 4. \end{aligned}$$

Note that because of the dynamic range of our selection the two cards chosen by the two draws cannot be the same. Possible worlds of the program are of the form

$$W_k = \{draw(1) = c_1, draw(2) = c_2, \dots\}$$

where $c_1 \neq c_2$. There are 52 possible outcomes of the first draw and 51 possible outcomes of the second. Thus, the program has 52×51 possible worlds – each world corresponding to outcomes of the two successive draws. By the Principle of Indifference the unnormalized probabilistic measure of each such world is equal to $\frac{1}{52} \times \frac{1}{51} = \frac{1}{2652}$. One can easily check that, as expected, the unnormalized measure is equal to the normalized one. To compute the number of possible worlds containing two_aces, let us notice that the number of aces that may be selected by the first draw is 4, whereas the number of aces that may be selected by the second draw is 3. Thus, there are 12 possible worlds containing two_aces, and the probability of this event is $12 \times \frac{1}{2652} = \frac{12}{2652} = \frac{1}{221}$.

11.4 Representing Knowledge in P-log

The previous examples addressed rather simple probabilistic problems, which could be easily solved without building a knowledge base of the corresponding domain. In this section we consider several examples where the use of P-log allows substantial clarification of the modeling process.

11.4.1 The Monty Hall Problem

This example, known as the Monty Hall Problem, is a nontrivial puzzle that is frequently solved incorrectly even by people with some knowledge of probability theory. The problem gets its name from the TV game show, *Let's Make a Deal*, hosted by Monty Hall. Here is a description of the problem:

> Monty's show involves a player who is given the opportunity to select one of three closed doors, behind one of which there is a prize. Behind the other two doors are empty rooms. Once the player has made a selection, Monty is obligated to open one of the remaining closed doors that does not contain the prize, showing that the room behind it is empty. He then asks the player if she would like to switch her selection to the other unopened door or stay with her original choice. Does it matter if she switches?

The answer is *yes*. In fact switching doubles the player's chance to win. This problem is quite interesting, because many people – often including mathematicians – feel the answer to be counter-intuitive. These people almost immediately come up with a (wrong) negative answer and are not easily persuaded that they made a mistake. We believe that part of the reason for the difficulty is that there is some disconnect between modeling probabilistic and nonprobabilistic knowledge about the problem. Let us show that in a P-log solution this disconnect disappears.

We start writing the P-log program representing knowledge about the show by declaring the set of three doors and three zero-arity attributes $selected$, $open$, and $prize$:

$$(1)\ \ doors = \{1, 2, 3\}.$$
$$(2)\ \ open, selected, prize : doors.$$

(The numbers are not part of the program; we number statements so that we can refer back to them.)

The regular part of the program contains rules that state that Monty can open any door to a room that is not selected by the player and that does not contain the prize.

$$(3)\ \ \neg can\_open(D) \leftarrow\ selected = D.$$
$$(4)\ \ \neg can\_open(D) \leftarrow\ prize = D.$$
$$(5)\ \ can\_open(D) \leftarrow\ not\ \neg can\_open(D).$$

The first two rules are self-explanatory. The last rule, which uses both classical and default negations, is a typical ASP representation of the closed world assumption – Monty can open any door except those that are explicitly prohibited.

We assume that both Monty and the player act randomly. This information is expressed as follows:

$$(6)\ random(prize).$$
$$(7)\ random(selected).$$
$$(8)\ random(open : \{X : can\_open(X)\}).$$

Notice that rule (8) uses a dynamic range; this guarantees that Monty selects only those doors that can be opened according to rules (3)–(5). The logical knowledge expressed by these rules (which can be extracted from the specification of the problem) is not explicitly represented in standard probabilistic formalisms that lack the expressive power to do that. The absence of this explicit knowledge may explain mistakes that are sometimes made in the process of solving the problem by traditional methods.

P-log program Π_{monty0} consisting of logical rules (1)–(8) represents our general knowledge of the problem domain. To proceed with our particular story, let us encode the results of random events that occurred before the player was asked if she wanted to change her selection. Without loss of generality we can assume that the player has already selected door 1 and that Monty opened door 2, revealing that it did not contain the prize. This is recorded by the following statements:

$$obs(selected = 1).$$
$$obs(open = 2).$$
$$obs(prize \neq 2).$$

Let us denote the resulting P-log program by Π_{monty1}. The program represents the player's complete knowledge relevant to her behavior in the game. To make an informed decision about switching, the player should compute the probability of the prize being behind door 1 and the prize being behind door 3. To do that, she must consider the possible worlds of Π_{monty1} and find their measures. Once the measures are found, she must sum up the measures of the worlds in which the prize is behind door 1 and do the same for worlds where the prize is behind door 3.

Because of the observations, Π_{monty1} has two possible worlds:

$$W_1 = \{selected = 1, prize = 1, open = 2, can\_open(2), can\_open(3)\}.$$

$$W_2 = \{selected = 1, prize = 3, open = 2, can\_open(2)\}.$$

In W_1 the player would lose if she switched; in W_2 she would win. Note that the possible worlds contain information not only about where the prize is but also which doors Monty can open. This is the key to correct calculation!

Now the player is ready to compute the probabilistic measures of W_1 and W_2. To do that she needs to compute the likelihood of random events within each world. In this case, these random events are which door is selected, where the prize is, and which door is opened by Monty. By the principle of indifference the causal probability of selecting particular values of $prize$ and $selected$ during the construction of both possible worlds is $\frac{1}{3}$ each. The situation, however, is different for the random attribute $open$. Recall that Monty can only open a door satisfying relation can_open. There are two such doors in W_1; hence the causal probability of selecting the second door in W_1 is $\frac{1}{2}$. But because Monty can open only one door in W_2, the causal probability of his choosing door 2 in W_2 is 1. The probabilistic measure of a possible world is the product of likelihoods of the random events that it comprises. It follows that

$$\hat{\mu}(W_1) = \tfrac{1}{3} \times \tfrac{1}{3} \times \tfrac{1}{2} = \tfrac{1}{18}$$
$$\hat{\mu}(W_2) = \tfrac{1}{3} \times \tfrac{1}{3} \times 1 = \tfrac{1}{9}.$$

Normalization gives us

$$\mu(W_1) = \frac{1/18}{1/18 + 1/9} = \frac{1}{3}$$

$$\mu(W_2) = \frac{1/9}{1/18 + 1/9} = \frac{2}{3}.$$

Finally, since $prize = 1$ is true in only W_1,

$$P_{\Pi_{monty1}}(prize = 1) = \mu(W_1) = \frac{1}{3}.$$

Similarly for $prize = 3$:

$$P_{\Pi_{monty1}}(prize = 3) = \mu(W_2) = \frac{2}{3}.$$

Changing doors doubles the player's chance to win.

Now consider a situation when the player assumes (either consciously or without consciously realizing it) that Monty could have opened any one of the doors not selected by her (including the one that contains the prize). Then the corresponding program has a new definition of can_open. Rules (3)–(5) are replaced by

$$\neg can\_open(D) \leftarrow \ selected = D.$$
$$can\_open(D) \leftarrow \ not\ \neg can\_open(D).$$

The resulting program Π_{monty2} also has two possible worlds containing $prize = 1$ and $prize = 3$, respectively, each with an unnormalized probabilistic measure of $\frac{1}{18}$, and therefore, $P_{\Pi_{monty2}}(prize = 1) = \frac{1}{2}$ and $P_{\Pi_{monty2}}(prize = 3) = \frac{1}{2}$. In that case, changing the door does not increase the probability of getting the prize. Since the rules are explicitly written, it makes the discussion about correctness of the answer much easier.

Program Π_{monty1} has no explicit probabilistic information, and so the possible results of each random selection are assumed to be equally likely. If we learn, for example, that given a choice between opening doors 2 and 3, Monty opens door 2 four times out of five, we can incorporate this information by the following statement:

$$(9)\ \ pr(open = 2 \mid_c can\_open(2), can\_open(3)) = \frac{4}{5}.$$

A computation similar to this one shows that changing doors still increases the player's chances of winning. (In fact changing doors is advisable as long as each of the available doors can be opened with some positive probability.)

The next example demonstrates the causal character of probabilistic statements of P-log.

11.4.2 Death of a Rat?

Consider the following program Π_{rat} representing knowledge about whether a certain rat will eat arsenic today and whether it will die today.

$$\begin{array}{l} arsenic, death : boolean. \\ random(arsenic). \\ random(death). \\ pr(arsenic) = 0.4. \\ pr(death \mid_c arsenic) = 0.8. \\ pr(death \mid_c \neg arsenic) = 0.01. \end{array}$$

This program tells us that the rat eating arsenic and the rat dying are viewed as random events and that the rat is more likely to die if it eats arsenic. In addition, the intuitive semantics of the pr-atoms expresses that the rat's consumption of arsenic carries information about the cause of its death (as opposed to, say, the rat's death being informative about the causes of its eating arsenic).

An intuitive consequence of this reading is that seeing the rat die raises our suspicion that it has eaten arsenic, whereas killing the rat (say, with a pistol) does not affect our degree of belief that arsenic has been consumed.

The following computations show that this implication is reflected in the probabilities computed under our semantics.

The possible worlds of this program, with their unnormalized probabilistic measures, are as follows (we show only *arsenic* and *death* literals):

$$\begin{array}{lll} W_1: & \{arsenic, death\}. & \hat{\mu}(W_1) = 0.4 \times 0.8 = 0.32 \\ W_2: & \{arsenic, \neg death\}. & \hat{\mu}(W_2) = 0.4 \times 0.2 = 0.08 \\ W_3: & \{\neg arsenic, death\}. & \hat{\mu}(W_3) = 0.6 \times 0.01 = 0.006 \\ W_4: & \{\neg arsenic, \neg death\}. & \hat{\mu}(W_4) = 0.6 \times 0.99 = 0.594 \end{array}$$

Since the unnormalized probabilistic measures add up to 1, they are the same as the normalized measures. Hence,

$$P_{\Pi_{rat}}(arsenic) = \mu(W_1) + \mu(W_2) = 0.32 + 0.08 = 0.4.$$

To compute the probability of $arsenic$ after the observation of $death$, we consider the program created from Π_{rat} and the statement $obs(death)$. The resulting program has two possible worlds, W_1 and W_3, with their unnormalized probabilistic measures shown earlier. Normalization yields

$$P_{\Pi_{rat} \cup \{obs(death)\}}(arsenic) = \frac{0.32}{0.32 + 0.006} = 0.982.$$

Notice that the observation of death raised our degree of belief that the rat had eaten arsenic.

To compute the effect of deliberately killing the rat on the agent's belief in $arsenic$, we augment the original program with the literal *do(death)*. The resulting program has the same two possible worlds, W_1 and W_3. However, the action of deliberate killing defeats the randomness of death so that W_1 has the unnormalized probabilistic measure 0.4 and W_3 has the unnormalized probabilistic measure 0.6. These sum up to 1, so the measures are also 0.4 and 0.6, respectively, and we get

$$P_{\Pi_{rat} \cup \{do(death)\}}(arsenic) = 0.4.$$

Note that this is identical to the initial probability $P_{\Pi_{rat}}(arsenic)$ computed earlier.

This last example shows that, in contrast to the case when the effect was passively observed, deliberately bringing about the effect did not change our degree of belief about the propositions relevant to its cause. On the other hand, propositions relevant to a cause, give equal evidence for the attendant effects whether they are forced to happen or are passively observed. For example, if we feed the rat arsenic, this increases its chance of death, just as if we had observed the rat eating the arsenic on its own. The conditional

probabilities computed under our semantics bear this out. Similarly to the above example, we can compute

$$P_{\Pi_{rat}}(death) = 0.362$$

$$P_{\Pi_{rat} \cup \{do(arsenic)\}}(death) = 0.8$$

$$P_{\Pi_{rat} \cup \{obs(arsenic)\}}(death) = 0.8$$

11.4.3 The Spider Bite

Here is another example of subtle probabilistic reasoning that can be done in P-log with the help of interventions. The story we would like to formalize follows:

In Stan's home town there are two kinds of poisonous spiders – the creeper and the spinner. Bites from the two are equally common in Stan's area, although spinner bites are more common on a worldwide basis. An experimental antivenom has been developed to treat bites from either kind of spider, but its effectiveness is questionable.

One morning Stan wakes to find he has a bite on his ankle and drives to the emergency room. A doctor examines the bite and concludes it is from either a creeper or a spinner. In deciding whether to administer the antivenom, the doctor examines the data he has on bites from the two kinds of spiders: Of 416 people bitten by the creeper worldwide, 312 received the antivenom and 104 did not. Among those who received the antivenom, 187 survived, whereas 73 survived among those who did not receive antivenom. The spinner is more deadly and tends to inhabit areas where the treatment is less available. Of 924 people bitten by the spinner, 168 received the antivenom, 34 of whom survived. Of the 756 spinner bite victims who did not receive the experimental treatment, only 227 survived. Should Stan take the antivenom treatment?

Let us formalize relevant parts of the story in a P-log program Π_{spider} from the point of view of the doctor. We use boolean attributes $survive$ – a random patient survived, $antivenom$ – a random patient was administered antivenom, and attribute $spider$ where $spider = creeper$ indicates that a random person was bitten by $creeper$; we do so similarly for $spinner$. From the standpoint of the doctor all three attributes (including $antivenom$) are random. Hence we have

$$survive, antivenom \ : \ boolean.$$
$$spider \ : \ \{creeper, spinner\}.$$

$$random(spider).$$
$$random(survive).$$
$$random(antivenom).$$

Since bites from the two spiders are equally common in the area, the doctor assumes that

$$pr(spider = creeper) = 0.5.$$

Statistical information available in the story allows the doctor to estimate the corresponding probabilities:

$$pr(antivenom|spider = creeper) = 312/416 = 0.75$$
$$pr(antivenom|spider = spinner) = 168/924 = 0.18$$
$$pr(survive \mid_c spider = creeper, antivenom) = 187/312 = 0.6$$
$$pr(survive \mid_c spider = creeper, \neg antivenom) = 73/104 = 0.7$$
$$pr(survive \mid_c spider = spinner, antivenom) = 34/168 = 0.2$$
$$pr(survive \mid_c spider = spinner, \neg antivenom) = 227/756 = 0.3$$

How should the doctor decide whether giving the antivenom to Stan would be beneficial? A natural answer is to compare the result of the deliberate action of taking the antivenom on Stan's chance of survival with that of not taking the antivenom. To do that let us first compute $P_{\Pi_{spider} \cup \{do(antivenom)\}}(survive)$. The program $\Pi_{spider} \cup \{do(antivenom)\}$ has the following possible worlds:

$$W_1 = \{spider = creeper, antivenom, survive\}$$
$$W_2 = \{spider = creeper, antivenom, \neg survive\}$$
$$W_3 = \{spider = spinner, antivenom, survive\}$$
$$W_4 = \{spider = spinner, antivenom, \neg survive\}$$

To compute the probabilistic measures of the possible worlds, let us first notice that, because of the intervention, each possible world contains only two random attributes, $spider$ and $survive$. Accordingly,

$$\mu(W_1) = 0.5 \times 0.6 = 0.3$$
$$\mu(W_2) = 0.5 \times 0.4 = 0.2$$
$$\mu(W_3) = 0.5 \times 0.2 = 0.1$$
$$\mu(W_4) = 0.5 \times 0.8 = 0.4$$

(Note that, because the measures sum up to 1, unnormalized and normalized measures coincide.)

It follows that the probability of survival after the treatment is

$$P_{\Pi_{spider} \cup \{do(antivenom)\}}(survive) = 0.4.$$

Now let us compute $P_{\Pi_{spider} \cup \{do(\neg antivenom)\}}(survive)$. Possible worlds of the associated program are

$$W_5 = \{spider = creeper, \neg antivenom, survive\}$$
$$W_6 = \{spider = creeper, \neg antivenom, \neg survive\}$$
$$W_7 = \{spider = spinner, \neg antivenom, survive\}$$
$$W_8 = \{spider = spinner, \neg antivenom, \neg survive\}$$

The measures are

$$\mu(W_5) = 0.5 \times 0.7 = 0.35$$
$$\mu(W_6) = 0.5 \times 0.3 = 0.15$$
$$\mu(W_7) = 0.5 \times 0.3 = 0.15$$
$$\mu(W_8) = 0.5 \times 0.7 = 0.35$$

and the probability of survival without the treatment is

$$P_{\Pi_{spider} \cup \{do(\neg antivenom)\}}(survive) = 0.5.$$

Thus, to maximize Stan's chance of survival, it is better not to administer the antivenom.

But what would happen if, instead of conditioning on intervention, the doctor decided to use what he believes to be a more traditional approach and condition on observations; i.e., compare $P_{\Pi_{spider} \cup \{obs(antivenom)\}}(survive)$ and $P_{\Pi_{spider} \cup \{obs(\neg antivenom)\}}(survive)$? The program $P_{\Pi_{spider} \cup \{obs(antivenom)\}}$ would have the possible worlds $W_1, \ldots, W_4$ as defined earlier, but the measures would be different. In this case, $antivenom$ would be a random attribute, so the likelihood of antivenom being given would be taken into account.

The unnormalized probabilistic measures induced by $P_{\Pi_{spider} \cup \{obs(antivenom)\}}$ are

$$\hat{\mu}(W_1) = 0.5 \times 0.75 \times 0.6 = 0.225$$
$$\hat{\mu}(W_2) = 0.5 \times 0.75 \times 0.4 = 0.15$$
$$\hat{\mu}(W_3) = 0.5 \times 0.18 \times 0.2 = 0.018$$
$$\hat{\mu}(W_4) = 0.5 \times 0.18 \times 0.8 = 0.072$$

The normalized measures are

$$\mu(W_1) = 0.225/0.465 = 0.484$$
$$\mu(W_2) = 0.15/0.465 = 0.322$$
$$\mu(W_3) = 0.018/0.465 = 0.039$$
$$\mu(W_4) = 0.072/0.465 = 0.155$$

and the probability is

$$P_{\Pi_{spider} \cup \{obs(antivenom)\}}(survive) = \mu(W_1) + \mu(W_3) = 0.523.$$

Similarly, we can compute the unnormalized probabilistic measures for $P_{\Pi_{spider} \cup \{obs(\neg antivenom)\}}$:

$$\hat{\mu}(W_5) = 0.5 \times 0.25 \times 0.7 = 0.088$$
$$\hat{\mu}(W_6) = 0.5 \times 0.25 \times 0.3 = 0.038$$
$$\hat{\mu}(W_7) = 0.5 \times 0.82 \times 0.3 = 0.123$$
$$\hat{\mu}(W_8) = 0.5 \times 0.82 \times 0.7 = 0.287$$

Normalization gives us

$$P_{\Pi_{spider} \cup \{obs(\neg antivenom)\}}(survive) = \mu(W_5) + \mu(W_7) = 0.164 + 0.229 = 0.393.$$

This different computation suggests that taking antivenom would increase Stan's chance of survival; i.e., it would lead to a different (and wrong) conclusion. We hope that this example shows how conditioning on interventions – a possibility not available in classical probability – helps avoid an unpleasant and costly mistake caused by the confusion between observations and interventions.

11.4.4 The Bayesian Squirrel

In this section we consider an example from Hilborn and Mangel (1997) used to illustrate the notion of Bayesian learning. One common type of learning problem consists of selecting from a set of models of a random phenomenon by observing repeated occurrences of that phenomenon. The Bayesian approach to this problem is to begin with a "prior density" on the set of candidate models and update it in light of our observations.

As an example, Hilborn and Mangel describe the Bayesian squirrel. The squirrel has hidden its acorns in one of two patches, say Patch 1 and Patch 2, but cannot remember which. The squirrel is 80% certain that the food is hidden in Patch 1. Also, it knows there is a 20% chance of finding food per day when it is looking in the right patch (and, of course, a 0% chance if it is looking in the wrong patch).

To represent this knowledge in P-log program Π_{sq}, we introduce sorts

$$patch = \{p1, p2\}.$$
$$day = \{1 \ldots n\}.$$

(where n is some constant, say, 5) and attributes

$$hidden\_in : patch.$$
$$found : day \rightarrow boolean.$$
$$look : day \rightarrow patch.$$

Attribute $hidden\_in$ is always random. Hence we include

$$random(hidden\_in).$$

Attribute $found$, however, is random only if the squirrel is looking for food in the right patch; i.e., we have

$$random(found(D)) \leftarrow hidden\_in = P, \; look(D) = P.$$

Otherwise we have

$$\neg found(D) \leftarrow hidden\_in = P_1, \; look(D) = P_2, \; P_1 \neq P_2.$$

The value of attribute $look(D)$ is decided by the squirrel's deliberation. Hence this attribute is not random.

Probabilistic information of the story is given by these statements:

$$pr(hidden\_in = p1) = 0.8.$$
$$pr(found(D)) = 0.2.$$

This knowledge, in conjunction with the description of the squirrel's activity, can be used to compute probabilities of possible outcomes of the next search for food.

Consider, for instance, program $\Pi_{sq1} = \Pi_{sq} \cup \{look(1) = p_1\}$, which represents that the squirrel decided to look for food in Patch 1 on the first day.

The program has three possible worlds

$$W_1^1 = \{look(1) = p_1, hidden\_in = p_1, found(1), \dots\}$$
$$W_2^1 = \{look(1) = p_1, hidden\_in = p_1, \neg found(1), \dots\}$$
$$W_3^1 = \{look(1) = p_1, hidden\_in = p_2, \neg found(1), \dots\}$$

with probabilistic measures

$$\mu(W_1^1) = 0.16$$
$$\mu(W_2^1) = 0.64$$
$$\mu(W_3^1) = 0.2$$

As expected,

$$P_{\Pi_{sq1}}(hidden\_in = p_1) = 0.16 + 0.64 = 0.8$$

and

$$P_{\Pi_{sq1}}(found(1)) = 0.16.$$

Suppose now that the squirrel failed to find its food during the first day and decided to continue its search in Patch 1 the next morning. Failure to find food on the first day should decrease the squirrel's degree of belief that the food is hidden in Patch 1 and, consequently, decrease its degree of belief that it will find food by looking in the first patch again. This is reflected in the following computation:

Let $\Pi_{sq2} = \Pi_{sq1} \cup \{obs(\neg found(1)), look(2) = p_1\}$.

The possible worlds of Π_{sq2} are

$$W_1^2 = \{look(1) = p_1, \neg found(1), hidden\_in = p_1, look(2) = p_1, found(2) \dots\}$$
$$W_2^2 = \{look(1) = p_1, \neg found(1), hidden\_in = p_1, look(2) = p_1, \neg found(2) \dots\}$$
$$W_3^2 = \{look(1) = p_1, \neg found(1), hidden\_in = p_2, look(2) = p_1, \neg found(2) \dots\}$$

Their probabilistic measures are

$$\mu(W_1^2) = 0.128/0.84 = 0.152$$
$$\mu(W_2^2) = 0.512/0.84 = 0.61$$
$$\mu(W_3^2) = 0.2/0.84 = 0.238$$

Consequently,

$$P_{\Pi_{sq2}}(hidden\_in = p_1) = 0.762$$

and

$$P_{\Pi_{sq2}}(found(2)) = 0.152$$

and so on.

After a number of unsuccessful attempts to find food in the first patch, the squirrel can come to the conclusion that the food is probably hidden in the second patch and can change its search strategy accordingly. Notice that each new experiment changes the squirrel's probabilistic model in a nonmonotonic way. That is, the set of possible worlds resulting from each successive experiment is not merely a subset of the possible worlds of the previous model. The program, however, is changed only by the addition of new actions and observations. Distinctive features of P-log such as the ability to represent observations and actions, as well as conditional randomness, play an important role in allowing the squirrel to learn new probabilistic models from experience.

11.5 (*) P-log + CR-Prolog and the Wandering Robot

There is no reason why P-log must be limited to standard ASP. In fact, its semantics is naturally defined on top of CR-Prolog, with the only difference being that possible worlds correspond to CR-Prolog answer sets instead of to ASP ones. In this section we give an example of a P-log program with a cr-rule.

A robot is located in a circular hall connected to three rooms, say r_0, r_1, r_2. The robot can enter any room. Normally this works pretty well, but in a rare case the robot might have a malfunction and end up in any of the three rooms. If it does malfunction, the probability of the robot ending up in the intended room is $\frac{1}{2}$.

The formalization of this domain in P-log requires the initial and final moments of time

$$time = \{0, 1\}.$$

three rooms

$$room = \{r_0, r_1, r_2\}.$$

and four places

$$position = \{h, r_0, r_1, r_2\}.$$

where h stands for the hall.

There are two attributes that we refer to as actions:

$$enter : room.$$

$$break : boolean.$$

The value of action $enter$ is R if at time 0 the robot *attempts* to enter room R. This action succeeds unless exogenous action $break$, which corresponds to the robot's malfunctioning, occurs at the same time. Since the value of $enter$ is determined by the deliberating robot, the attribute is not random. We know that $break$ normally does not happen, and we have no information on the likelihood of this rare event. Hence $break$ is not random either.

An attribute in

$$in : time \rightarrow postion.$$

returns the position of the robot at a given point in time. Initially, the robot is in the hall:

$$in(0) = h.$$

The following rules govern changes in the domain:

1. If no malfunction occurs, the robot gets to where it is going:

$$in(1) = R \leftarrow enter = R, \neg break.$$

2. A malfunction can cause the robot to end up in any of the rooms:

$$random(in(1) : room) \leftarrow enter = R, break.$$

3. If there is a malfunction, the probability of the robot winding up in the room it intended to enter is $\frac{1}{2}$:

$$pr(in(1) = R \mid_c enter = R, break) = \frac{1}{2}.$$

4. We assume that normally the robot does not malfunction:

$$\neg break \leftarrow \ not\ break.$$

However, if no other consistent state of the world exists, we must assume that the robot malfunctioned:

$$break \stackrel{+}{\leftarrow} .$$

(For the sake of brevity, we also assume that the program is used in conjunction with collections of atoms that make sense. For example, we do not allow a robot to be in two rooms at once, to try to enter more than one room at a time, and so on.)

Let's consider some inputs and the corresponding possible worlds of the program. Let's add the line

$$enter = r_0$$

to our program. In this case, there is a unique possible world that contains $\neg break$ and $in(1) = r_0$. Since we do not consider malfunctions unless we are told about them or there is some sort of inconsistency, we have this expected result. Now let's tell the program that a malfunction occurred by adding this line:

$$break.$$

The program with the two new inputs has three possible worlds:

$$W_0 = \{in(1) = r_0, \dots\}$$
$$W_1 = \{in(1) = r_1, \dots\}$$
$$W_2 = \{in(1) = r_2, \dots\}$$

According to P-log semantics, the first world is assigned the probability of $\frac{1}{2}$, whereas the next two get $\frac{1}{4}$. Notice that the addition of $break$ changed the reasoner's degree of belief in the robot's being in r_0 from 1 to $\frac{1}{2}$ – a feat not possible in classical Bayesian updating.

Let's return to the original program and assume no given knowledge of a break occurring. Given CR-Prolog semantics, we can have inputs such as

$$enter = r_0.$$
$$obs(in(1) = r_2).$$

without contradiction because the program will simply fire the CR-rule and deduce $break$. (Note, that we have $obs(in(1) = r_2)$ instead of $in(1) = r_2$ because the value of $in(1)$ can be selected randomly.)

Consider another situation. Suppose we learn that the robot tried to enter r_0, but failed to do so. This knowledge can be recorded by adding the

following lines to the original program:

$$enter = r_0.$$
$$obs(in(1) \neq r_0).$$

Now there are two possible worlds:

$$W_0 = \{in(1) = r_1, break, \dots\}$$

$$W_1 = \{in(1) = r_2, break, \dots\}$$

each with probability $\frac{1}{2}$, which conforms to intuition. We hope this example demonstrates the utility of combining CR-Prolog and P-log and the simplicity with which this can be done.

Summary

In this chapter we defined the syntax and semantics of knowledge representation language P-log, which is capable of combining logical and probabilistic reasoning. The logical part of the language is based on ASP and its extensions. The probabilistic part adopts the idea of causal probability from Pearl (2000). The interpretation of the probability of an event as a measure of the degree of a rational agent's belief in the truth of that event allows for a natural combination of the two formalisms. Several examples showed how the language can be used to formalize subtle forms of reasoning including both probability and logic.

There has been a substantial number of other attempts to combine logical and probabilistic reasoning. However, P-log programs have a collection of properties that are not seen together in other languages. These include the following:

- P-log probabilities are defined with respect to an explicitly stated knowledge base. In many cases this greatly facilitates the creation of probabilistic models.
- In addition to logical nonmonotonicity, P-log is "probabilistically nonmonotonic" – the addition of new information can add new possible worlds and substantially change the original probabilistic model, allowing for Bayesian learning.
- Possible knowledge base updates include defaults, rules introducing new terms, observations, and deliberate actions in the sense of Pearl (2000).

The language is comparatively new and is currently the subject of extensive investigation. In particular, we can expect substantial progress in the automation of P-log reasoning.

References and Further Reading

The material in this chapter is based primarily on Baral, Gelfond, and Rushton (2009) and Gelfond and Rushton (2010). Some information on P-log inference engines and applications can be found in Gelfond, Rushton, and Zhu (2006), Zhu (2010, 2012), Anh et al. (2008), Pereira and Ramli (2010), and Baral and Hunsaker (2007). The reader may be especially interested in the use of P-log for finding most probable diagnosis discussed in Zhu (2012), which combines logical knowledge about dynamic systems in the style presented in this book with some probabilistic information about the likelihood of possible faults. Our understanding of the nature of probability and causality, their role in AI, and the importance of the representation of probability distributions by Bayesian nets was greatly influenced by Pearl (1988, 2000). E.T. Jaynes (2003) gives an account of probability that views probabilistic reasoning as commonsense reasoning of a rational agent. There are numerous attempts to combine logical and probabilistic knowledge via logic programming; see, for instance, Poole (2008), Sato and Kameya (1997), and Vennekens, Denecker, and Bruynooghe (2009). A more traditional approach to combining logic and probability uses classical logic. For an early example of this approach one can see Nilsson (1986). A contemporary review can be found in Domingos and Lowd (2009). The recent book by Koller and Friedman (2009) is a good example of approaches that combine classical logic with graphical probabilistic models.

Exercises

1. Consider the following P-log program Π:

$$
\begin{aligned}
&num = \{1, 2, 3, 4\}.\\
&draw : num.\\
&random(draw).\\
&success : boolean.\\
&even(N) \leftarrow N \; mod \; 2 = 0.\\
&success \leftarrow draw = N, even(N).
\end{aligned}
$$

   (a) Construct possible worlds of Π and compute probability P_Π of $success$.

(b) Expand the program by statement

$$pr(draw = 1) = \frac{1}{2}.$$

What is the probability of $success$ now?

2. Expand the program Π_{dice} from Example 11.3.1 by a new boolean attribute max_score, which holds iff each die shows 6. Compute the probability of max_score defined by the new program.
3. Consider a P-log program Π:

$$\begin{array}{l} p : \{y_1, y_2\}. \\ random(p). \\ q : boolean. \\ q \leftarrow p = y_1. \\ \neg q \leftarrow not\ q. \end{array}$$

Show possible worlds of programs Π, $\Pi \cup \{obs(q)\}$, and $\Pi \cup \{q\}$. Compute and compare probabilities of q defined by each one of the programs.

4. Consider a game of Russian roulette with two six-chamber guns. Each of the guns is loaded with a single bullet. Write a P-log program to represent this knowledge. What is the probability of the player dying if he fires both guns simultaneously? (*Hint:* Use the following attributes: $pull\_trigger(G)$, which says that the player pulls the trigger of gun G; $fatal(G)$, which says that the bullet from gun G is sufficient to kill the player, and is_dead, which says that the player is dead.)
5. A parallel system (see Fig. 11.1) consists of three components, and it functions if at least one component works. The probability that each component works is independent of the others.

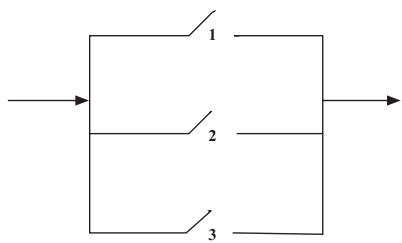

Figure 11.1. Parallel System

(a) Write a P-log program to describe this system. Use the notion of probabilistic measure defined by this program to compute the probability that the system functions.
(b) How would you use P-log to write that the first component was observed to be broken?
(c) How would you use P-log to write that the first component was broken on purpose?
(d) Suppose the system consists of components a, b, and c. Part a works $\frac{1}{2}$ of the time, part b works $\frac{1}{3}$ of the time, and part c works $\frac{2}{5}$ of the time. Write statements that would allow the program to take this new information into account. Compute the probability that the system functions.
(e) Change the program to describe a serial system (see Fig. 11.2). Compute the probability that the system functions. (A serial system requires all components to work.)

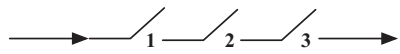

Figure 11.2. Serial System

6. Write a P-log program to represent the information about cards from Example 11.3.4 that does not require the use of a dynamic range. *Hint:* Use attribute

$$draw\_ace : try \rightarrow boolean.$$

and define pr-atoms for $draw\_ace(1)$ and $draw\_ace(2)$.

7. At a party names of dance partners are drawn at random from two hats. The first hat contains the names of all the boys, and the second contains those of the girls. There are three names in each hat. Write a P-log program to compute the odds that boy 1 and girl 1 will end up as dance partners.

8. Write a P-log program (with cr-rules) to represent the following information: A circuit has a motor, a breaker, and a switch. The switch may be open or closed, the breaker may be tripped or not, and the motor may be turning or not. The operator may toggle the switch or reset the breaker. If the switch is closed and the system is functioning normally, the motor turns. The motor never turns when the switch is open, the breaker is tripped, or the motor is burned out. The system may break,

and if so the break could consist of a tripped breaker, a burned-out motor, or both, with respective probabilities 0.9, 0.09, and 0.01. Breaking, however, is rare, and should be considered only in the absence of other explanations. Design several scenarios to test your representation.

9. Prove that for any disjoint subsets E_1 and E_2 of Ω, $P(E_1 \cup E_2) = P(E_1) + P(E_2)$.