7. An object is in the intersection of two sets if and only if it is a member of both sets:

$$\forall x, s_1, s_2 \;\; x \in (s_1 \cap s_2) \;\Leftrightarrow\; (x \in s_1 \wedge x \in s_2) \; .$$

8. An object is in the union of two sets if and only if it is a member of either set:

$$\forall x, s_1, s_2 \;\; x \in (s_1 \cup s_2) \;\Leftrightarrow\; (x \in s_1 \vee x \in s_2) \; .$$

LIST

Lists are similar to sets. The differences are that lists are ordered and the same element can appear more than once in a list. We can use the vocabulary of Lisp for lists: Nil is the constant list with no elements; $Cons$, $Append$, $First$, and $Rest$ are functions; and $Find$ is the predicate that does for lists what $Member$ does for sets. $List?$ is a predicate that is true only of lists. As with sets, it is common to use syntactic sugar in logical sentences involving lists. The empty list is $[\,]$. The term $Cons(x, y)$, where y is a nonempty list, is written $[x|y]$. The term $Cons(x, Nil)$ (i.e., the list containing the element x) is written as $[x]$. A list of several elements, such as $[A, B, C]$, corresponds to the nested term $Cons(A, Cons(B, Cons(C, Nil)))$. Exercise 8.16 asks you to write out the axioms for lists.

8.3.4 The wumpus world

Some propositional logic axioms for the wumpus world were given in Chapter 7. The first-order axioms in this section are much more concise, capturing in a natural way exactly what we want to say.

Recall that the wumpus agent receives a percept vector with five elements. The corresponding first-order sentence stored in the knowledge base must include both the percept and the time at which it occurred; otherwise, the agent will get confused about when it saw what. We use integers for time steps. A typical percept sentence would be

$$Percept([Stench, Breeze, Glitter, None, None], 5) \; .$$

Here, $Percept$ is a binary predicate, and $Stench$ and so on are constants placed in a list. The actions in the wumpus world can be represented by logical terms:

$$Turn(Right), \;\; Turn(Left), \;\; Forward, \;\; Shoot, \;\; Grab, \;\; Climb \; .$$

To determine which is best, the agent program executes the query

$$\text{ASKVARS}(\exists a \;\; BestAction(a, 5)) \; ,$$

which returns a binding list such as $\{a/Grab\}$. The agent program can then return $Grab$ as the action to take. The raw percept data implies certain facts about the current state. For example:

$$\forall t, s, g, m, c \;\; Percept([s, Breeze, g, m, c], t) \;\Rightarrow\; Breeze(t) \; ,$$
$$\forall t, s, b, m, c \;\; Percept([s, b, Glitter, m, c], t) \;\Rightarrow\; Glitter(t) \; ,$$

and so on. These rules exhibit a trivial form of the reasoning process called **perception**, which we study in depth in Chapter 24. Notice the quantification over time t. In propositional logic, we would need copies of each sentence for each time step.

Simple "reflex" behavior can also be implemented by quantified implication sentences. For example, we have

$$\forall t \;\; Glitter(t) \;\Rightarrow\; BestAction(Grab, t) \; .$$

Given the percept and rules from the preceding paragraphs, this would yield the desired conclusion $BestAction(Grab, 5)$—that is, $Grab$ is the right thing to do.

We have represented the agent's inputs and outputs; now it is time to represent the environment itself. Let us begin with objects. Obvious candidates are squares, pits, and the wumpus. We could name each square—$Square_{1,2}$ and so on—but then the fact that $Square_{1,2}$ and $Square_{1,3}$ are adjacent would have to be an "extra" fact, and we would need one such fact for each pair of squares. It is better to use a complex term in which the row and column appear as integers; for example, we can simply use the list term $[1, 2]$. Adjacency of any two squares can be defined as

$$\forall\, x, y, a, b \;\; Adjacent([x, y], [a, b]) \;\Leftrightarrow \\ (x = a \wedge (y = b - 1 \vee y = b + 1)) \vee (y = b \wedge (x = a - 1 \vee x = a + 1)) \; .$$

We could name each pit, but this would be inappropriate for a different reason: there is no reason to distinguish among pits.[10] It is simpler to use a unary predicate Pit that is true of squares containing pits. Finally, since there is exactly one wumpus, a constant $Wumpus$ is just as good as a unary predicate (and perhaps more dignified from the wumpus's viewpoint).

The agent's location changes over time, so we write $At(Agent, s, t)$ to mean that the agent is at square s at time t. We can fix the wumpus's location with $\forall t \; At(Wumpus, [2, 2], t)$. We can then say that objects can only be at one location at a time:

$$\forall\, x, s_1, s_2, t \;\; At(x, s_1, t) \wedge At(x, s_2, t) \;\Rightarrow\; s_1 = s_2 \; .$$

Given its current location, the agent can infer properties of the square from properties of its current percept. For example, if the agent is at a square and perceives a breeze, then that square is breezy:

$$\forall\, s, t \;\; At(Agent, s, t) \wedge Breeze(t) \;\Rightarrow\; Breezy(s) \; .$$

It is useful to know that a *square* is breezy because we know that the pits cannot move about. Notice that $Breezy$ has no time argument.

Having discovered which places are breezy (or smelly) and, very important, *not* breezy (or *not* smelly), the agent can deduce where the pits are (and where the wumpus is). Whereas propositional logic necessitates a separate axiom for each square (see R_2 and R_3 on page 247) and would need a different set of axioms for each geographical layout of the world, first-order logic just needs one axiom:

$$\forall\, s \;\; Breezy(s) \;\Leftrightarrow\; \exists\, r \;\; Adjacent(r, s) \wedge Pit(r) \; . \tag{8.4}$$

Similarly, in first-order logic we can quantify over time, so we need just one successor-state axiom for each predicate, rather than a different copy for each time step. For example, the axiom for the arrow (Equation (7.2) on page 267) becomes

$$\forall\, t \;\; HaveArrow(t + 1) \;\Leftrightarrow\; (HaveArrow(t) \wedge \neg Action(Shoot, t)) \; .$$

From these two example sentences, we can see that the first-order logic formulation is no less concise than the original English-language description given in Chapter 7. The reader

[10] Similarly, most of us do not name each bird that flies overhead as it migrates to warmer regions in winter. An ornithologist wishing to study migration patterns, survival rates, and so on *does* name each bird, by means of a ring on its leg, because individual birds must be tracked.

EXERCISES

8.1 A logical knowledge base represents the world using a set of sentences with no explicit structure. An **analogical** representation, on the other hand, has physical structure that corresponds directly to the structure of the thing represented. Consider a road map of your country as an analogical representation of facts about the country—it represents facts with a map language. The two-dimensional structure of the map corresponds to the two-dimensional surface of the area.

a. Give five examples of *symbols* in the map language.

b. An *explicit* sentence is a sentence that the creator of the representation actually writes down. An *implicit* sentence is a sentence that results from explicit sentences because of properties of the analogical representation. Give three examples each of *implicit* and *explicit* sentences in the map language.

c. Give three examples of facts about the physical structure of your country that cannot be represented in the map language.

d. Give two examples of facts that are much easier to express in the map language than in first-order logic.

e. Give two other examples of useful analogical representations. What are the advantages and disadvantages of each of these languages?

8.2 Consider a knowledge base containing just two sentences: $P(a)$ and $P(b)$. Does this knowledge base entail $\forall\, x\ P(x)$? Explain your answer in terms of models.

8.3 Is the sentence $\exists\, x, y\ \ x = y$ valid? Explain.

8.4 Write down a logical sentence such that every world in which it is true contains exactly one object.

8.5 Consider a symbol vocabulary that contains c constant symbols, p_k predicate symbols of each arity k, and f_k function symbols of each arity k, where $1 \leq k \leq A$. Let the domain size be fixed at D. For any given model, each predicate or function symbol is mapped onto a relation or function, respectively, of the same arity. You may assume that the functions in the model allow some input tuples to have no value for the function (i.e., the value is the invisible object). Derive a formula for the number of possible models for a domain with D elements. Don't worry about eliminating redundant combinations.

8.6 Which of the following are valid (necessarily true) sentences?

a. $(\exists x\ x = x)\ \Rightarrow\ (\forall\, y\ \ \exists z\ y = z)$.

b. $\forall\, x\ \ P(x) \vee \neg P(x)$.

c. $\forall\, x\ \ \mathit{Smart}(x) \vee (x = x)$.

8.7 Consider a version of the semantics for first-order logic in which models with empty domains are allowed. Give at least two examples of sentences that are valid according to the

standard semantics but not according to the new semantics. Discuss which outcome makes more intuitive sense for your examples.

8.8 Does the fact $\neg Spouse(George, Laura)$ follow from the facts $Jim \neq George$ and $Spouse(Jim, Laura)$? If so, give a proof; if not, supply additional axioms as needed. What happens if we use $Spouse$ as a unary function symbol instead of a binary predicate?

8.9 This exercise uses the function $MapColor$ and predicates $In(x, y)$, $Borders(x, y)$, and $Country(x)$, whose arguments are geographical regions, along with constant symbols for various regions. In each of the following we give an English sentence and a number of candidate logical expressions. For each of the logical expressions, state whether it (1) correctly expresses the English sentence; (2) is syntactically invalid and therefore meaningless; or (3) is syntactically valid but does not express the meaning of the English sentence.

a. Paris and Marseilles are both in France.

(i) $In(Paris \wedge Marseilles, France)$.
(ii) $In(Paris, France) \wedge In(Marseilles, France)$.
(iii) $In(Paris, France) \vee In(Marseilles, France)$.

b. There is a country that borders both Iraq and Pakistan.

(i) $\exists\, c \quad Country(c) \wedge Border(c, Iraq) \wedge Border(c, Pakistan)$.
(ii) $\exists\, c \quad Country(c) \Rightarrow [Border(c, Iraq) \wedge Border(c, Pakistan)]$.
(iii) $[\exists\, c \quad Country(c)] \Rightarrow [Border(c, Iraq) \wedge Border(c, Pakistan)]$.
(iv) $\exists\, c \quad Border(Country(c), Iraq \wedge Pakistan)$.

c. All countries that border Ecuador are in South America.

(i) $\forall\, c \;\; Country(c) \wedge Border(c, Ecuador) \Rightarrow In(c, SouthAmerica)$.
(ii) $\forall\, c \;\; Country(c) \Rightarrow [Border(c, Ecuador) \Rightarrow In(c, SouthAmerica)]$.
(iii) $\forall\, c \;\; [Country(c) \Rightarrow Border(c, Ecuador)] \Rightarrow In(c, SouthAmerica)$.
(iv) $\forall\, c \;\; Country(c) \wedge Border(c, Ecuador) \wedge In(c, SouthAmerica)$.

d. No region in South America borders any region in Europe.

(i) $\neg[\exists\, c, d \;\; In(c, SouthAmerica) \wedge In(d, Europe) \wedge Borders(c, d)]$.
(ii) $\forall\, c, d \;\; [In(c, SouthAmerica) \wedge In(d, Europe)] \Rightarrow \neg Borders(c, d)]$.
(iii) $\neg\forall\, c \;\; In(c, SouthAmerica) \Rightarrow \exists\, d \;\; In(d, Europe) \wedge \neg Borders(c, d)$.
(iv) $\forall\, c \;\; In(c, SouthAmerica) \Rightarrow \forall\, d \;\; In(d, Europe) \Rightarrow \neg Borders(c, d)$.

e. No two adjacent countries have the same map color.

(i) $\forall\, x, y \;\; \neg Country(x) \vee \neg Country(y) \vee \neg Borders(x, y) \vee$
$\neg(MapColor(x) = MapColor(y))$.
(ii) $\forall\, x, y \;\; (Country(x) \wedge Country(y) \wedge Borders(x, y) \wedge \neg(x = y)) \Rightarrow$
$\neg(MapColor(x) = MapColor(y))$.
(iii) $\forall\, x, y \;\; Country(x) \wedge Country(y) \wedge Borders(x, y) \wedge$
$\neg(MapColor(x) = MapColor(y))$.
(iv) $\forall\, x, y \;\; (Country(x) \wedge Country(y) \wedge Borders(x, y)) \Rightarrow MapColor(x \neq y)$.