EXERCISES 10

KNOWLEDGE REPRESENTATION

10.1 Ontological Engineering

[[need exercises]]

10.2 Categories and Objects

Exercise 10.OTTT

Define an ontology in first-order logic for tic-tac-toe. The ontology should contain situations, actions, squares, players, marks (X, O, or blank), and the notion of winning, losing, or drawing a game. Also define the notion of a forced win (or draw): a position from which a player can force a win (or draw) with the right sequence of actions. Write axioms for the domain. (Note: The axioms that enumerate the different squares and that characterize the winning positions are rather long. You need not write these out in full, but indicate clearly what they look like.)

Sortal predicates:
$Player(p)$
$Mark(m)$
$Square(q)$

Constants:
Xp, Op: Players.
$X, O, Blank$: Marks.
$Q11, Q12 \ldots Q33$: Squares.
$S0$: Situation.

Atemporal:
$MarkOf(p)$: Function mapping player p to his/her mark.
$Winning(q1, q2, q3)$: Predicate. Squares $q1, q2, q3$ constitute a winning position.
$Opponent(p)$: Function mapping player p to his opponent.

Situation Calculus:
$Result(a, s)$.
$Poss(a, s)$.

State:
$TurnAt(s)$: Function mapping situation s to the player whose turn it is.
$Marked(q, s)$: Function mapping square q and situation s to the mark in q at s.

$Wins(p, s)$. Player p has won in situation s.

Action:

$Play(p, q)$: Function mapping player p and square q to the action of p marking q.

Atemporal axioms:

A1. $MarkOf(Xp) = X$.
A2. $MarkOf(Op) = O$.
A3. $Opponent(Xp) = Op$.
A4. $Opponent(Op) = Xp$.
A5. $\forall p \;\; Player(p) \;\Leftrightarrow\; p = Xp \vee p = Op$.
A6. $\forall m \;\; Mark(m) \;\Leftrightarrow\; m = X \vee m = O \vee m = Blank$.
A7. $\forall q \;\; Square(q) \;\Leftrightarrow\; q = Q11 \vee q = Q12 \vee \ldots \vee q = Q33$.
A8. $\forall q1, q2, q3 \;\; WinningPosition(q1, q2, q3) \;\Leftrightarrow$
$[q1 = Q11 \wedge q2 = Q12 \wedge q3 = Q13] \vee$
$[q1 = Q21 \wedge q2 = Q22 \wedge q3 = Q23] \vee$
... (Similarly for the other six winning positions $\vee$
$[q1 = Q31 \wedge q2 = Q22 \wedge q3 = Q13]$.

Definition of winning:

A9. $\forall p, s \;\; Wins(p, s) \;\Leftrightarrow$
$\exists q1, q2, q3 \;\; WinningPosition(q1, q2, q3) \wedge$
$MarkAt(q1, s) = MarkAt(q2, s) = MarkAt(q3, s) = MarkOf(p)$

Causal Axioms:

A10. $\forall p, q \;\; Player(p) \wedge Square(q) \;\Rightarrow$
$MarkAt(q, Result(Play(p, q), s)) = MarkOf(p)$.
A11. $\forall p, a, s \;\; TurnAt(p, s) \;\Rightarrow\; TurnAt(Opponent(p), Result(a, s))$.

Precondition Axiom:

A12. $Poss(Play(p, q), s) \;\Rightarrow\; TurnAt(s) = p \wedge MarkAt(q, s) = Blank$.

Frame Axiom: A13. $q1 \neq q2 \Rightarrow MarkAt(q1, Result(Play(p, q2), s)) = MarkAt(q1, s)$.

Unique names:

A14. $X \neq O \neq Blank$.

(Note: the unique property on players $Xp \neq Op$ follows from A14, A1, and A2.)

A15-A50. For each i, j, k, m between 1 and 3 such that either $i \neq k$ or $j \neq m$ assert the axiom $Qij \neq Qkm$.

Note: In many theories it is useful to posit unique names axioms between entities of different sorts e.g. $\forall p, q \;\; Player(p) \wedge Square(q) \;\Rightarrow\; p \neq q$. In this theory these are not actually necessary; if you want to imagine a circumstance in which player Xp is actually the same entity as square $Q23$ or the same as the action $Play(Xp, Q23)$ there is no harm in it.

Exercise 10.CSUG

You are to create a system for advising computer science undergraduates on what courses to take over an extended period in order to satisfy the program requirements. (Use whatever requirements are appropriate for your institution.) First, decide on a vocabulary for repre-

senting all the information, and then represent it; then formulate a query to the system that will return a legal program of study as a solution. You should allow for some tailoring to individual students, in that your system should ask what courses or equivalents the student has already taken, and not generate programs that repeat those courses.

Suggest ways in which your system could be improved—for example to take into account knowledge about student preferences, the workload, good and bad instructors, and so on. For each kind of knowledge, explain how it could be expressed logically. Could your system easily incorporate this information to find all feasible programs of study for a student? Could it find the *best* program?

Most schools distinguish between required courses and elected courses, and between courses inside the department and outside the department. For each of these, there may be requirements for the number of courses, the number of units (since different courses may carry different numbers of units), and on grade point averages. We show our chosen vocabulary by example:

- Student Jones' complete course of study for the whole college career consists of Math1, CS1, CS2, CS3, CS21, CS33 and CS34, and some other courses outside the major.

$$Take(Jones,\\ \{Math1, EE1, Bio24, CS1, CS2, CS3, CS21, CS33, CS34|others\})$$

- Jones meets the requirements for a major in Computer Science

$$Major(Jones, CS)$$

- Courses Math1, CS1, CS2, and CS3 are required for a Computer Science major.

$$Required(\{Math1, CS1, CS2, CS3\}, CS)\\ \forall\, s, d \;\; Required(s, d) \;\Leftrightarrow\\ \quad (\forall\, p \;\; \exists\, others \;\; Major(p, d) \;\Rightarrow\; Take(p, Union(s, others)))$$

- A student must take at least 18 units in the CS department to get a degree in CS.

$$Department(CS1) = CS \wedge Department(Math1) = Math \wedge \ldots\\ Units(CS1) = 3 \wedge Units(CS2) = 4 \wedge \ldots\\ RequiredUnitsIn(18, CS, CS)\\ \forall\, u, d \;\; RequiredUnitsIn(u, d) \;\Leftrightarrow\\ \quad (\forall\, p \;\; \exists\, s, others \;\; Major(p, d) \;\Rightarrow\; Take(p, Union(s, others))\\ \quad\quad \wedge\, AllInDepartment(s, d) \wedge TotalUnits(s) \geq u\\ \forall\, s, d \;\; AllInDepartment(s, d) \;\Leftrightarrow\; (\forall\, c \;\; c \in s \;\Rightarrow\; Department(c) = d)\\ \forall\, c \;\; TotalUnits(\{\}) = 0\\ \forall\, c, s \;\; TotalUnits(\{c|s\}) = Units(c) + TotalUnits(s)$$

One can easily imagine other kinds of requirements; these just give you a flavor.

In this solution we took "over an extended period" to mean that we should recommend a set of courses to take, without scheduling them on a semester-by-semester basis. If you wanted to do that, you would need additional information such as when courses are taught, what is a reasonable course load in a semester, and what courses are prerequisites for what others. For example:

$$Taught(CS1, Fall)$$
$$Prerequisites(\{CS1, CS2\}, CS3)$$
$$TakeInSemester(Jones, Fall95, \{Math1, CS1, English1, History1\})$$
$$MaxCoursesPerSemester(5)$$

The problem with finding the *best* program of study is in defining what *best* means to the student. It is easy enough to say that all other things being equal, one prefers a good teacher to a bad one, or an interesting course to a boring one. But how do you decide which is best when one course has a better teacher and is expected to be easier, while an alternative is more interesting and provides one more credit? Chapter 15 uses utility theory to address this. If you can provide a way of weighing these elements against each other, then you can choose a best program of study; otherwise you can only eliminate some programs as being worse than others, but can't pick an absolute best one. Complexity is a further problem: witha general-purpose theorem-prover it's hard to do much more than enumerate legal programs and pick the best.

Exercise 10.EVER

Figure 10.1 shows the top levels of a hierarchy for everything. Extend it to include as many real categories as possible. A good way to do this is to cover all the things in your everyday life. This includes objects and events. Start with waking up, and proceed in an orderly fashion noting everything that you see, touch, do, and think about. For example, a random sampling produces music, news, milk, walking, driving, gas, Soda Hall, carpet, talking, Professor Fateman, chicken curry, tongue, $7, sun, the daily newspaper, and so on.

You should produce both a single hierarchy chart (on a large sheet of paper) and a listing of objects and categories with the relations satisfied by members of each category. Every object should be in a category, and every category should be in the hierarchy.

This exercise might be suitable for a term project. At this point, we want to strongly urge that you do assign some of these exercises (or ones like them) to give your students a feeling of what it is really like to do knowledge representation. In general, students find classification hierarchies easier than other representation tasks. A recent twist is to compare one's hierarchy with online ones such as `yahoo.com`.

Exercise 10.WATR

Represent the following seven sentences using and extending the representations developed in the chapter:

a. Water is a liquid between 0 and 100 degrees.

b. Water boils at 100 degrees.
c. The water in John's water bottle is frozen.
d. Perrier is a kind of water.
e. John has Perrier in his water bottle.
f. All liquids have a freezing point.
g. A liter of water weighs more than a liter of alcohol.

Remember that we defined substances so that $Water$ is a category whose elements are all those things of which one might say "it's water." One tricky part is that the English language is ambiguous. One sense of the word "water" includes ice ("that's frozen water"), while another sense excludes it: ("that's not water—it's ice"). The sentences here seem to use the first sense, so we will stick with that. It is the sense that is roughly synonymous with H_2O.

The other tricky part is that we are dealing with objects that change (freeze and melt) over time. Thus, it won't do to say $w \in Liquid$, because w (a mass of water) might be a liquid at one time and a solid at another. For simplicity, we will use a situation calculus representation, with sentences such as $T(w \in Liquid, s)$. There are many possible correct answers to each of these. The key thing is to be *consistent* in the way that information is represented. For example, do not use $Liquid$ as a predicate on objects if $Water$ is used as a substance category.

a. "Water is a liquid between 0 and 100 degrees." We will translate this as "For any water and any situation, the water is liquid iff and only if the water's temperature in the situation is between 0 and 100 centigrade."

$$\begin{array}{l}\forall\, w, s \;\; w \in Water \;\Rightarrow \\ \quad (Centigrade(0) < Temperature(w, s) < Centigrade(100)) \;\Leftrightarrow \\ \quad\quad T(w \in Liquid, s)\end{array}$$

b. "Water boils at 100 degrees." It is a good idea here to do some tool-building. On page 261 we used $MeltingPoint$ as a predicate applying to individual instances of a substance. Here, we will define $SBoilingPoint$ to denote the boiling point of all instances of a substance. The basic meaning of boiling is that instances of the substance becomes gaseous above the boiling point:

$$\begin{array}{l}SBoilingPoint(c, bp) \;\Leftrightarrow \\ \quad \forall\, x, s \;\; x \in c \;\Rightarrow \\ \quad\quad (\forall\, t \;\; T(Temperature(x, t), s) \wedge t > bp \;\Rightarrow\; T(x \in Gas, s))\end{array}$$

Then we need only say $SBoilingPoint(Water, Centigrade(100))$.

c. "The water in John's water bottle is frozen."

We will use the constant Now to represent the situation in which this sentence holds. Note that it is easy to make mistakes in which one asserts that only some of the water

in the bottle is frozen.

$$\exists b \ \forall w \ \ w \in Water \wedge b \in WaterBottles \wedge Has(John, b, Now) \\ \wedge Inside(w, b, Now) \ \Rightarrow \ (w \in Solid, Now)$$

d. "Perrier is a kind of water."

$$Perrier \subset Water$$

e. "John has Perrier in his water bottle."

$$\exists b \ \forall w \ \ w \in Water \wedge b \in WaterBottles \wedge Has(John, b, Now) \\ \wedge Inside(w, b, Now) \ \Rightarrow \ w \in Perrier$$

f. "All liquids have a freezing point."

Presumably what this means is that all substances that are liquid at room temperature have a freezing point. If we use $RTLiquidSubstance$ to denote this class of substances, then we have

$$\forall c \ \ RTLiquidSubstance(c) \ \Rightarrow \ \exists t \ \ SFreezingPoint(c, t)$$

where $SFreezingPoint$ is defined similarly to $SBoilingPoint$. Note that this statement is false in the real world: we can invent categories such as "blue liquid" which do not have a unique freezing point. An interesting exercise would be to define a "pure" substance as one all of whose instances have the same chemical composition.

g. "A liter of water weighs more than a liter of alcohol."

$$\forall w, a \ \ w \in Water \wedge a \in Alcohol \wedge Volume(w) = Liters(1) \\ \wedge Volume(a) = Liters(1) \ \Rightarrow \ Mass(w) > Mass(a)$$

Exercise 10.DECM

Write definitions for the following:

a. *ExhaustivePartDecomposition*

b. *PartPartition*

c. *PartwiseDisjoint*

These should be analogous to the definitions for *ExhaustiveDecomposition*, *Partition*, and *Disjoint*. Is it the case that $PartPartition(s, BunchOf(s))$? If so, prove it; if not, give a counterexample and define sufficient conditions under which it does hold.

This is a fairly straightforward exercise that can be done in direct analogy to the corresponding definitions for sets.

a. $ExhaustivePartDecomposition$ holds between a set of parts and a whole, saying that anything that is a part of the whole must be a part of one of the set of parts.

$$\forall s, w \;\; ExhaustivePartDecomposition(s, w) \;\Leftrightarrow \\ (\forall p \;\; PartOf(p, w) \;\Rightarrow\; \exists p_2 \;\; p_2 \in s \wedge PartOf(p, p_2))$$

b. $PartPartition$ holds between a set of parts and a whole when the set is disjoint and is an exhaustive decomposition.

$$\forall s, w \;\; PartPartition(s, w) \;\Leftrightarrow \\ PartwiseDisjoint(s) \wedge ExhaustivePartDecomposition(s, w)$$

c. A set of parts is $PartwiseDisjoint$ if when you take any two parts from the set, there is nothing that is a part of both parts.

$$\forall s \;\; PartwiseDisjoint(s) \;\Leftrightarrow \\ \forall p_1, p_2 \;\; p_1 \in s \wedge p_2 \in s \wedge p_1 \neq p_2 \;\Rightarrow\; \neg\exists p_3 \; PartOf(p_3, p_1) \wedge PartOf(p_3, p_2)$$

It is *not* the case that $PartPartition(s, BunchOf(s))$ for any s. A set s may consist of physically overlapping objects, such as a hand and the fingers of the hand. In that case, $BunchOf(s)$ is equal to the hand, but s is not a partition of it. We need to ensure that the elements of s are partwise disjoint:

$$\forall s \;\; PartwiseDisjoint(s) \;\Rightarrow\; PartPartition(s, BunchOf(s)) \; .$$

Exercise 10.ALTM

An alternative scheme for representing measures involves applying the units function to an abstract length object. In such a scheme, one would write $Inches(Length(L_1)) = 1.5$. How does this scheme compare with the one in the chapter? Issues include conversion axioms, names for abstract quantities (such as "50 dollars"), and comparisons of abstract measures in different units (50 inches is more than 50 centimeters).

In the scheme in the chapter, a conversion axiom looks like this:

$$\forall x \;\; Centimeters(2.54 \times x) = Inches(x) \; .$$

"50 dollars" is just $\$(50)$, the name of an abstract monetary quantity. For any measure function such as \$, we can extend the use of $>$ as follows:

$$\forall x, y \;\; x > y \;\Rightarrow\; \$(x) > \$(y) \; .$$

Since the conversion axiom for dollars and cents has

$$\forall x \;\; Cents(100 \times x) = \$(x)$$

it follows immediately that $\$(50) > Cents(50)$.

In the new scheme, we must introduce objects whose lengths are converted:

$$\forall x \;\; Centimeters(Length(x)) = 2.54 \times Inches(Length(x)) \; .$$

There is no obvious way to refer directly to "50 dollars" or its relation to "50 cents". Again, we must introduce objects whose monetary value is 50 dollars or 50 cents:

$$\forall x, y \;\; \$(Value(x)) = 50 \wedge Cents(Value(y)) = 50 \;\Rightarrow\; \$(Value(x)) > \$(Value(y))$$

Exercise 10.TOMS

Write a set of sentences that allows one to calculate the price of an individual tomato (or other object), given the price per pound. Extend the theory to allow the price of a bag of tomatoes to be calculated.

For an instance i of a substance s with price per pound c and weight n pounds, the price of i will be $n \times c$, or in other words:

$$\forall i, s, n, c \;\; i \in s \wedge PricePer(s, Pounds(1)) = \$(c) \wedge Weight(i) = Pounds(n)$$
$$\Rightarrow \;\; Price(i) = \$(n \times c)$$

If b is the set of tomatoes in a bag, then $BunchOf(b)$ is the composite object consisting of all the tomatoes in the bag. Then we have

$$\forall i, s, n, c \;\; b \subset s \wedge PricePer(s, Pounds(1)) = \$(c)$$
$$\wedge Weight(BunchOf(b)) = Pounds(n)$$
$$\Rightarrow \;\; Price(BunchOf(b)) = \$(n \times c)$$

Exercise 10.NAME

Add sentences to extend the definition of the predicate $Name(s, c)$ so that a string such as "laptop computer" matches the appropriate category names from a variety of stores. Try to make your definition general. Test it by looking at ten online stores, and at the category names they give for three different categories. For example, for the category of laptops, we found the names "Notebooks," "Laptops," "Notebook Computers," "Notebook," "Laptops and Notebooks," and "Notebook PCs." Some of these can be covered by explicit $Name$ facts, while others could be covered by sentences for handling plurals, conjunctions, etc.

Plurals can be handled by a $Plural$ relation between strings, e.g.,

$$Plural(\text{“}computer\text{”}, \text{“}computers\text{”})$$

plus an assertion that the plural (or singular) of a name is also a name for the same category:

$$\forall\, c, s_1, s_2 \;\; Name(s_1, c) \wedge (Plural(s_1, s_2) \vee Plural(s_2, s_1)) \;\Rightarrow\; Name(s_2, c)$$

Conjunctions can be handled by saying that any conjunction string is a name for a category if one of the conjuncts is a name for the category:

$$\forall\, c, s, s_2 \;\; Conjunct(s_2, s) \wedge Name(s_2, c) \;\Rightarrow\; Name(s, c)$$

where $Conjunct$ is defined appropriately in terms of concatenation. Probably it would be better to redefine $RelevantCategoryName$ instead.

Exercise 10.SMKC

One part of the shopping process that was not covered in this chapter is checking for compatibility between items. For example, if a digital camera is ordered, what accessory batteries, memory cards, and cases are compatible with the camera? Write a knowledge base that can determine the compatibility of a set of items and suggest replacements or additional items if the shopper makes a choice that is not compatible. The knowledge base should works with at least one line of products and extend easily to other lines.

Here is an initial sketch of one approach. (Others are possible.) A given object to be purchased may *require* some additional parts (e.g., batteries) to be functional, and there may also be *optional* extras. We can represent requirements as a relation between an individual object and a class of objects, qualified by the number of objects required:

$$\forall\, x \;\; x \in Coolpix995DigitalCamera \;\Rightarrow\; Requires(x, AABattery, 4)\ .$$

We also need to know that a particular object is compatible, i.e., fills a given role appropriately. For example,

$$\forall\, x, y \;\; x \in Coolpix995DigitalCamera \wedge y \in DuracellAABattery \\ \Rightarrow\; Compatible(y, x, AABattery)$$

Then it is relatively easy to test whether the set of ordered objects contains compatible required objects for each object.

Exercise 10.SMKG

A complete solution to the problem of inexact matches to the buyer's description in shopping is very difficult and requires a full array of natural language processing and information retrieval techniques. (See Chapters 24 and **??**.) One small step is to allow the user to spec-

ify minimum and maximum values for various attributes. The buyer must use the following grammar for product descriptions:

$$
\begin{array}{lll}
\mathit{Description} & \rightarrow & \mathit{Category}\ [\mathit{Connector}\ \mathit{Modifier}]* \\
\mathit{Connector} & \rightarrow & \text{“}\mathit{with}\text{”} \mid \text{“}\mathit{and}\text{”} \mid \text{“,”} \\
\mathit{Modifier} & \rightarrow & \mathit{Attribute} \mid \mathit{Attribute}\ \mathit{Op}\ \mathit{Value} \\
\mathit{Op} & \rightarrow & \text{“}=\text{”} \mid \text{“}>\text{”} \mid \text{“}<\text{”}
\end{array}
$$

Here, *Category* names a product category, *Attribute* is some feature such as “CPU” or “price,” and *Value* is the target value for the attribute. So the query “computer with at least a 2.5 GHz CPU for under $500” must be re-expressed as “computer with CPU $>$ 2.5 GHz and price $<$ $500.” Implement a shopping agent that accepts descriptions in this language.

Chapter 24 explains how to use logic to parse text strings and extract semantic information. The outcome of this process is a definition of what objects are acceptable to the user for a specific shopping request; this allows the agent to go out and find offers matching the user's requirements. We omit the full definition of the agent, although a skeleton may appear on the AIMA project web pages.

10.3 Events

Exercise 10.WNDO

Develop a representational system for reasoning about windows in a window-based computer interface. In particular, your representation should be able to describe:

- The state of a window: minimized, displayed, or nonexistent.
- Which window (if any) is the active window.
- The position of every window at a given time.
- The order (front to back) of overlapping windows.
- The actions of creating, destroying, resizing, and moving windows; changing the state of a window; and bringing a window to the front. Treat these actions as atomic; that is, do not deal with the issue of relating them to mouse actions. Give axioms describing the effects of actions on fluents. You may use either event or situation calculus.

Assume an ontology containing *situations, actions, integers* (for x and y coordinates) and *windows*. Define a language over this ontology; that is, a list of constants, function symbols, and predicates with an English description of each. If you need to add more categories to the ontology (e.g., pixels), you may do so, but be sure to specify these in your write-up. You may (and should) use symbols defined in the text, but be sure to list these explicitly.

A plausible language might contain the following primitives:
Temporal Predicates:

$Poss(a, s)$ – Predicate: Action a is possible in situation s. As in section 10.3
$Result(a, s)$ – Function from action a and situation s to situation. As in section 10.3.

Arithmetic: $x < y$, $x \leq y$, $x + y$, 0.
Window State:

$Minimized(w, s)$, $Displayed(w, s)$, $Nonexistent(w, s)$, $Active(w, s)$ – Predicates. In all these w is a window and s is a situation. ("$Displayed(w, s)$" means existent and non-minimized; it includes the case where all of w is actually occluded by other windows.)

Window Position:

$RightEdge(w, s)$, $LeftEdge(w, s)$, $TopEdge(w, s)$, $BottomEdge(w, s)$: Functions from a window w and situation s to a coordinate.
$ScreenWidth$, $ScreenHeight$: Constants.

Window Order:

$InFront(w1, w2, s)$: Predicate. Window $w1$ is in front of window $w2$ in situation s.

Actions:

$Minimize(w)$, $MakeVisible(w)$, $Destroy(w)$, $BringToFront(w)$ – Functions from a window w to an action.
$Move(w, dx, dy)$ – Move window w by dx to the left and dy upward. (Quantities dx and dy may be negative.)
$Resize(w, dxl, dxr, dyb, dyt)$ – Resize window w by dxl on the left, dxr on the right, dyb on bottom, and dyt on top.

Exercise 10.WNDP

State the following in the language you developed for the previous exercise:

a. In situation S_0, window W_1 is behind W_2 but sticks out on the left and right. Do *not* state exact coordinates for these; describe the *general* situation.
b. If a window is displayed, then its top edge is higher than its bottom edge.
c. After you create a window w, it is displayed.
d. A window can be minimized if it is displayed.

a $LeftEdge(W1, S0) < LeftEdge(W2, S0) \wedge RightEdge(W2, S0) < RightEdge(W1, S0) \wedge TopEdge(W1, S0) \leq TopEdge(W2, S0) \wedge BottomEdge(W2, S0) \leq BottomEdge(W1, S0) \wedge InFront(W2, W1, S0)$.

b $\forall w, s \;\; Displayed(w, s) \Rightarrow BottomEdge(w, s) < TopEdge(w, s)$.

c $\forall w, s \;\; Poss(Create(w), s) \Rightarrow Displayed(w, Result(Create(w), s))$.

d $Displayed(w, s) \Rightarrow Poss(Minimize(w), s)$

Exercise 10.SPMK

(Adapted from an example by Doug Lenat.) Your mission is to capture, in logical form, enough knowledge to answer a series of questions about the following simple scenario:

> Yesterday John went to the North Berkeley Safeway supermarket and bought two pounds of tomatoes and a pound of ground beef.

Start by trying to represent the content of the sentence as a series of assertions. You should write sentences that have straightforward logical structure (e.g., statements that objects have certain properties, that objects are related in certain ways, that all objects satisfying one property satisfy another). The following might help you get started:

- Which classes, objects, and relations would you need? What are their parents, siblings and so on? (You will need events and temporal ordering, among other things.)
- Where would they fit in a more general hierarchy?
- What are the constraints and interrelationships among them?
- How detailed must you be about each of the various concepts?

To answer the questions below, your knowledge base must include background knowledge. You'll have to deal with what kind of things are at a supermarket, what is involved with purchasing the things one selects, what the purchases will be used for, and so on. Try to make your representation as general as possible. To give a trivial example: don't say "People buy food from Safeway," because that won't help you with those who shop at another supermarket. Also, don't turn the questions into answers; for example, question (c) asks "Did John buy any meat?"—not "Did John buy a pound of ground beef?"

Sketch the chains of reasoning that would answer the questions. If possible, use a logical reasoning system to demonstrate the sufficiency of your knowledge base. Many of the things you write might be only approximately correct in reality, but don't worry too much; the idea is to extract the common sense that lets you answer these questions at all. A truly complete answer to this question is *extremely* difficult, probably beyond the state of the art of current knowledge representation. But you should be able to put together a consistent set of axioms for the limited questions posed here.

a. Is John a child or an adult? [Adult]
b. Does John now have at least two tomatoes? [Yes]
c. Did John buy any meat? [Yes]
d. If Mary was buying tomatoes at the same time as John, did he see her? [Yes]
e. Are the tomatoes made in the supermarket? [No]
f. What is John going to do with the tomatoes? [Eat them]
g. Does Safeway sell deodorant? [Yes]
h. Did John bring some money or a credit card to the supermarket? [Yes]
i. Does John have less money after going to the supermarket? [Yes]

This is the most involved representation problem. It is suitable for a group project of 2

or 3 students over the course of at least 2 weeks. Solutions should include a taxonomy, a choice of situation calculus, fluent calculus, or event calculus for handling time and change, and enough background knowledge. If a logic programming system or theorem prover is not used, students might want to write out the proofs for at least some of the answers.

Exercise 10.SPML

Make the necessary additions or changes to your knowledge base from the previous exercise so that the questions that follow can be answered. Include in your report a discussion of your changes, explaining why they were needed, whether they were minor or major, and what kinds of questions would necessitate further changes.

a. Are there other people in Safeway while John is there? [Yes—staff!]
b. Is John a vegetarian? [No]
c. Who owns the deodorant in Safeway? [Safeway Corporation]
d. Did John have an ounce of ground beef? [Yes]
e. Does the Shell station next door have any gas? [Yes]
f. Do the tomatoes fit in John's car trunk? [Yes]

Normally one would assign the preceding exercise in one assignment, and then when it is done, add this exercise (posibly varying the questions). That way, the students see whether they have made sufficient generalizations in their initial answer, and get experience with debugging and modifying a knowledge base.

Exercise 10.EVEW

Write event calculus axioms to describe the actions in the wumpus world.

Section 10.3 includes a couple of axioms for the wumpus world:

$$Initiates(e, HaveArrow(a), t) \Leftrightarrow e = Start$$
$$Terminates(e, HaveArrow(a), t) \Leftrightarrow e \in Shootings(a)$$

Here is an axiom for turning; the others are similar albeit more complex. Let the term $TurnRight(a)$ denote the event category of the agent turning right. We want to say about it that if (say) the agent is facing south up to the beginning of the action, then it is facing west after the action ends, and so on.

$$\begin{aligned} T(TurnRight(a), i) \quad \Leftrightarrow \quad & \\ & [\exists h \;\; Meets(h, i) \wedge T(FacingSouth(a), h) \;\Rightarrow \\ & \quad Clipped(FacingSouth(a), i) \wedge Restored(FacingWest(a), i)] \\ & \vee \quad \ldots \end{aligned}$$

Exercise 10.INAL

State the interval-algebra relation that holds between every pair of the following real-world events:

LK: The life of President Kennedy.
IK: The infancy of President Kennedy.
PK: The presidency of President Kennedy.
LJ: The life of President Johnson.
PJ: The presidency of President Johnson.
LO: The life of President Obama.

$Starts(IK, LK)$.
$Finishes(PK, LK)$.
$During(LK, LJ)$.
$Meets(LK, PJ)$.
$Overlap(LK, LC)$.
$Before(IK, PK)$.
$During(IK, LJ)$.
$Before(IK, PJ)$.
$Before(IK, LC)$.
$During(PK, LJ)$.
$Meets(PK, PJ)$.
$During(PK, LC)$.
$During(PJ, LJ)$.
$Overlap(LJ, LC)$.
$During(PJ, LC)$.

Exercise 10.RBGO

This exercise concerns the problem of planning a route for a robot to take from one city to another. The basic action taken by the robot is $Go(x, y)$, which takes it from city x to city y if there is a route between those cities. $Road(x, y)$ is true if and only if there is a road connecting cities x and y; if there is, then $Distance(x, y)$ gives the length of the road. See the map on page 82 for an example. The robot begins in Arad and must reach Bucharest.

a. Write a suitable logical description of the initial situation of the robot.

b. Write a suitable logical query whose solutions provide possible paths to the goal.

c. Write a sentence describing the Go action.

d. Now suppose that the robot consumes fuel at the rate of .02 gallons per mile. The robot starts with 20 gallons of fuel. Augment your representation to include these considerations.

e. Now suppose some of the cities have gas stations at which the robot can fill its tank. Extend your representation and write all the rules needed to describe gas stations, in-

cluding the *Fillup* action.

This question takes the student through the initial stages of developing a logical representation for actions that incorporates more and more realism. Implementing the reasoning tasks in a theorem-prover is also a good idea. Although the use of logical reasoning for the initial task—finding a route on a graph—may seem like overkill, the student should be impressed that we can keep making the situation more complicated simply by describing those added complications, with no additions to the reasoning system.

a. $At(Robot, Arad, S_0)$.

b. $\exists\, s \;\; At(Robot, Bucharest, s)$.

c. The successor-state axiom should be mechanical by now. $\forall\, a, x, y, s$:

$$\begin{aligned} At(Robot, y, Result(a, s)) \quad \Leftrightarrow \quad & [(a = Go(x, y) \\ & \wedge Road(x, y) \wedge At(Robot, x, s)) \\ \vee \quad & (At(Robot, y, s) \\ & \wedge \neg(\exists\, z \;\; a = Go(y, z) \wedge z \neq y))] \end{aligned}$$

d. To represent the fuel the robot has in a given situation, use the function $FuelLevel(Robot, s)$; this refers not to actual fuel or to a number, but to an abstract amount. Let $Full = Gallons(20)$ be a constant denoting the fuel capacity of the tank. The initial situation is then described by $At(Robot, Arad, S_0) \wedge FuelLevel(Robot, S_0) = Full$. Also, $Distance(x, y)$ will return an abstract length object.

Perhaps the trickiest part is handling the measures properly. To simplify things, we have used a little trick, defining "inverse" unit functions such as $Miles^{-1}$, which returns the number of miles in a given length object:

$$\forall\, x \;\; Miles^{-1}(Miles(x)) = x\ .$$

Now, the successor-state axiom for location is extended as follows (note that we do not

say what happens if the robot runs out of gas):

$$
\begin{aligned}
&At(Robot, y, Result(a, s)) \\
&\quad \Leftrightarrow \quad [(a = Go(x, y) \\
&\qquad \wedge Road(x, y) \wedge At(Robot, x, s) \\
&\qquad \wedge 0.02 \times Miles^{-1}(Distance(x, y)) \leq Gallons^{-1}(FuelLevel(Robot, s))) \\
&\qquad \vee \quad (At(Robot, y, s) \\
&\qquad\quad \wedge \neg(\exists z \;\; a = Go(y, z) \wedge z \neq y))] \\
&FuelLevel(Robot, Result(a, s)) = f \\
&\quad \Leftrightarrow \quad [(a = Go(x, y) \\
&\qquad \wedge Road(x, y) \wedge At(Robot, x, s) \\
&\qquad \wedge 0.02 \times Miles^{-1}(Distance(x, y)) \leq Gallons^{-1}(FuelLevel(Robot, s)) \\
&\qquad \wedge f = FuelLevel(Robot, s) - Gallons(0.02 \times Miles^{-1}(Distance(x, y))) \\
&\qquad \vee \quad (f = Fuel(Robot, s) \\
&\qquad\quad \wedge \neg(\exists v, w \;\; a = Go(v, w) \wedge v \neq w))]
\end{aligned}
$$

e. The simplest way to extend the representation is to add the predicate $GasStation(x)$, which is true of cities with gas stations. The $Fillup$ action is described by adding another clause to the above axiom for $Fuel$, saying that $f = Full$ when $a = FillUp$.

Exercise 10.EVES

Investigate ways to extend the event calculus to handle *simultaneous* events. Is it possible to avoid a combinatorial explosion of axioms?

The main difficulty with simultaneous (also called concurrent) events and actions is how to account correctly for possible interference. A good starting point is the expository paper by Shanahan (1999). Section 5 of that paper shows how to manage concurrent actions by the introduction of additional generic predicates $Cancels$ and $Canceled$, describing circumstances in which actions may interfere with each other. We avoid lots of "non-cancellation" assertions using the same predicate-completion trick as in successor-state axioms, and the meaning of cancellation is defined once and for all through its connection to clipping, restoring, etc.

Exercise 10.EXRT

Construct a representation for exchange rates between currencies that allows for daily fluctuations.

For quantities such as length and time, the conversion axioms such as

$$Centimeters(2.54 \times d) = Inches(d)$$

are absolutes that hold (with a few exceptions) for all time. The same is true for conversion axioms within a given currency; for example, $US\$(1) = US¢(100)$. When it comes to

conversion *between* currencies, we make the simplifying assumption that at any given time t there is a prevailing exchange rate:

$$T(ExchangeRate(UK\pounds(1), US\$(1)) = 1.55, t)$$

and the rate is reciprocal:

$$ExchangeRate(UK\pounds(1), US\$(1)) = 1/ExchangeRate(US\$(1), UK\pounds(1)) .$$

What we cannot do, however, is write

$$T(UK\pounds(1) = US\$(1.55), t)$$

thereby *equating* abstract amounts of money in different currencies. At any given moment, prevailing exchange rates across the world's currencies need not be consistent, and using equality across currencies would therefore introduce a *logical* inconsistency. Instead, exchange rates should be interpreted as indicating a willingness to exchange, perhaps with some commission; and exchange rate inconsistency is an opportunity for arbitrage. A more sophisticated model would include the entity offering the rate, limits on amounts and forms of payment, etc.

Exercise 10.FIXD

Define the predicate $Fixed$, where $Fixed(Location(x))$ means that the location of object x is fixed over time.

Any object x is an event, and $Location(x)$ is the event that for every subinterval of time, refers to the place where x is. For example, $Location(Peter)$ is the complex event consisting of his home from midnight to about 9:00 today, then various parts of the road, then his office from 10:00 to 1:30, and so on. To say that an event is fixed is to say that any two moments of the event have the same spatial extent:

$$\begin{aligned}&\forall e \;\; Fixed(e) \;\Leftrightarrow \\ &\quad (\forall a, b \;\; a \in Moments \wedge b \in Moments \wedge Subevent(a, e) \wedge Subevent(b, e) \\ &\qquad \Rightarrow \;\; SpatialExtent(a) = SpatialExtent(b))\end{aligned}$$

Exercise 10.TRAD

Describe the event of trading something for something else. Describe buying as a kind of trading in which one of the objects traded is a sum of money.

Let $Trade(b, x, a, y)$ denote the class of events where person b trades object y to person

a for object x:

$$T(Trade(b,x,a,y),i) \Leftrightarrow \\ T(Owns(b,y),Start(i)) \wedge T(Owns(a,x),Start(i)) \wedge \\ T(Owns(b,x),End(i)) \wedge T(Owns(a,y),End(i))$$

Now the only tricky part about defining buying in terms of trading is in distinguishing a price (a measurement) from an actual collection of money.

$$T(Buy(b,x,a,p),i) \Leftrightarrow \exists m \;\; Money(m) \wedge Trade(b,x,a,m) \wedge Value(m) = p$$

Exercise 10.OWNR

The two preceding exercises assume a fairly primitive notion of ownership. For example, the buyer starts by *owning* the dollar bills. This picture begins to break down when, for example, one's money is in the bank, because there is no longer any specific collection of dollar bills that one owns. The picture is complicated still further by borrowing, leasing, renting, and bailment. Investigate the various commonsense and legal concepts of ownership, and propose a scheme by which they can be represented formally.

There are many possible approaches to this exercise. The idea is for the students to think about doing knowledge representation for real; to consider a host of complications and find some way to represent the facts about them. Some of the key points are:

- Ownership occurs over time, so we need either a situation-calculus or interval-calculus approach.
- There can be joint ownership and corporate ownership. This suggests the owner is a group of some kind, which in the simple case is a group of one person.
- Ownership provides certain rights: to use, to resell, to give away, etc. Much of this is outside the definition of ownership *per se*, but a good answer would at least consider how much of this to represent.
- Own can own abstract obligations as well as concrete objects. This is the idea behind the futures market, and also behind banks: when you deposit a dollar in a bank, you are giving up ownership of that particular dollar in exchange for ownership of the right to withdraw another dollar later. (Or it could coincidentally turn out to be the exact same dollar.) Leases and the like work this way as well. This is tricky in terms of representation, because it means we have to reify transactions of this kind. That is, $Withdraw(person, money, bank, time)$ must be an object, not a predicate.

Exercise 10.SMKI

Our description of Internet shopping omitted the all-important step of actually *buying* the product. Provide a formal logical description of buying, using event calculus. That is,

define the sequence of events that occurs when a buyer submits a credit-card purchase and then eventually gets billed and receives the product.

Here is a simple version of the answer; it can be elaborated *ad infinitum*. Let the term $Buy(b, x, s, p)$ denote the event category of buyer b buying object x from seller s for price p. We want to say about it that b transfers the money to s, and s transfers ownership of x to b.

$$
\begin{aligned}
&T(Buy(b, x, s, p), i) \;\Leftrightarrow \\
&\quad T(Owns(s, x), Start(i)) \wedge \\
&\quad \exists m \;\; Money(m) \wedge p = Value(m) \wedge T(Owns(b, m), Start(i)) \wedge \\
&\quad T(Owns(b, x), End(i)) \wedge T(Owns(s, m), End(i))
\end{aligned}
$$

10.4 Mental Objects and Modal Logic

Exercise 10.CDFH

Consider a game played with a deck of just 8 cards, 4 aces and 4 kings. The three players, Alice, Bob, and Carlos, are dealt two cards each. Without looking at them, they place the cards on their foreheads so that the other players can see them. Then the players take turns either announcing that they know what cards are on their own forehead, thereby winning the game, or saying "I don't know." Everyone knows the players are truthful and are perfect at reasoning about beliefs.

a. Game 1. Alice and Bob have both said "I don't know." Carlos sees that Alice has two aces (A-A) and Bob has two kings (K-K). What should Carlos say? (*Hint*: consider all three possible cases for Carlos: A-A, K-K, A-K.)

b. Describe each step of Game 1 using the notation of modal logic.

c. Game 2. Carlos, Alice, and Bob all said "I don't know" on their first turn. Alice holds K-K and Bob holds A-K. What should Carlos say on his second turn?

d. Game 3. Alice, Carlos, and Bob all say "I don't know" on their first turn, as does Alice on her second turn. Alice and Bob both hold A-K. What should Carlos say?

e. Prove that there will always be a winner to this game.

(Adapted from Fagin *et al.* (1995).)

Just to get you started: In Game 1, Alice says "I don't know." If Carlos had K-K, and given that Alice can see Bob's K-K, then she would know that Bob and Carlos had all four kings between them and she would announce A-A. Therefore, Carlos does not have K-K. Then Bob says "I don't know." If Carlos had A-A, and given that Bob can see Alice's A-A, then he would know that Alice and Carlos had all four aces between them and he would announce A-A. Therefore, Carlos does not have A-A. Therefore Carlos should announce A-K.

Exercise 10.LGOM

The assumption of *logical omniscience,* discussed on page 346, is of course not true of any actual reasoners. Rather, it is an *idealization* of the reasoning process that may be more or less acceptable depending on the applications. Discuss the reasonableness of the assumption for each of the following applications of reasoning about knowledge:

a. Partial knowledge adversary games, such as card games. Here one player wants to reason about what his opponent knows about the state of the game.

b. Chess with a clock. Here the player may wish to reason about the limits of his opponent's or his own ability to find the best move in the time available. For instance, if player A has much more time left than player B, then A will sometimes make a move that greatly complicates the situation, in the hopes of gaining an advantage because he has more time to work out the proper strategy.

c. A shopping agent in an environment in which there are costs of gathering information.

d. Reasoning about public key cryptography, which rests on the intractability of certain computational problems.

A. The logical omniscience assumption is a reasonable idealization. The limiting factor here is generally the information available to the players, not the difficulty of making inferences.

B. This kind of reasoning cannot be accommodated in a theory with logical omniscience. If logical omniscience were true, then every player could always figure out the optimal move instantaneously.

C. Logical omniscience is a reasonable idealization. The costs of getting the information are almost always much greater than the costs of reasoning with it.

D. It depends on the kind of reasoning you want to do. If you want to reason about the relation of cryptography to particular computational problems, then logical omniscience cannot be assumed, because the assumption entails that any computational problem can be solved instantly. On the other hand, if you are willing to idealize the encryption/decryption as a magical process with no computational basis, then it may be reasonable to apply a theory with logical omniscience to other aspects of the theory.

10.5 Reasoning Systems for Categories

Exercise 10.DSCL

Translate the following description logic expression (from page 350) into first-order logic, and comment on the result:

$$\begin{aligned}&And(Man, AtLeast(3, Son), AtMost(2, Daughter),\\ &\quad All(Son, And(Unemployed, Married, All(Spouse, Doctor))),\\ &\quad All(Daughter, And(Professor, Fills(Department, Physics, Math)))) \ .\end{aligned}$$

This corresponds to the following open formula:

$$
\begin{aligned}
&Man(x) \wedge \exists\, s_1, s_2, s_3 \;\; Son(s_1, x) \wedge Son(s_2, x) \wedge Son(s_3, x) \\
&\qquad \wedge s_1 \neq s_2 \wedge s_1 \neq s_3 \wedge s_2 \neq s_3 \\
&\wedge \neg \exists\, d_1, d_2, d_3 \;\; Daughter(d_1, x) \wedge Daughter(d_2, x) \wedge Daughter(d_3, x) \\
&\qquad \wedge d_1 \neq d_2 \wedge d_1 \neq d_3 \wedge d_2 \neq d_3 \\
&\wedge \forall\, s \;\; Son(s, x) \;\Rightarrow\; Unemployed(s) \wedge Married(s) \wedge Doctor(Spouse(s)) \\
&\wedge \forall\, d \;\; Daughter(d, x) \;\Rightarrow\; Professor(d) \wedge \\
&\qquad (Department(d) = Physics \vee Department(d) = Math) \; .
\end{aligned}
$$

Exercise 10.INHX

Recall that inheritance information in semantic networks can be captured logically by suitable implication sentences. This exercise investigates the efficiency of using such sentences for inheritance.

a. Consider the information in a used-car catalog such as Kelly's Blue Book—for example, that 1973 Dodge vans are (or perhaps were once) worth $575. Suppose all this information (for 11,000 models) is encoded as logical sentences, as suggested in the chapter. Write down three such sentences, including that for 1973 Dodge vans. How would you use the sentences to find the value of a *particular* car, given a backward-chaining theorem prover such as Prolog?

b. Compare the time efficiency of the backward-chaining method for solving this problem with the inheritance method used in semantic nets.

c. Explain how forward chaining allows a logic-based system to solve the same problem efficiently, assuming that the KB contains only the 11,000 sentences about prices.

d. Describe a situation in which neither forward nor backward chaining on the sentences will allow the price query for an individual car to be handled efficiently.

e. Can you suggest a solution enabling this type of query to be solved efficiently in all cases in logic systems? (*Hint:* Remember that two cars of the same year and model have the same price.)

In many AI and Prolog textbooks, you will find it stated plainly that implications suffice for the implementation of inheritance. This is true in the logical but not the practical sense.

a. Here are three rules, written in Prolog. We actually would need many more clauses on the right hand side to distinguish between different models, different options, etc.

```
worth(X,575) :- year(X,1973), make(X,dodge), style(X,van).
worth(X,27000) :- year(X,1994), make(X,lexus), style(X,sedan).
worth(X,5000) :- year(X,1987), make(X,toyota), style(X,sedan).
```

To find the value of JB, given a data base with `year(jb,1973)`, `make(jb,dodge)`

and `style(jb,van)` we would call the backward chainer with the goal `worth(jb,D)`, and read the value for `D`.

b. The time efficiency of this query is $O(n)$, where n in this case is the 11,000 entries in the Blue Book. A semantic network with inheritance would allow us to follow a link from `JB` to `1973-dodge-van`, and from there to follow the `worth` slot to find the dollar value in $O(1)$ time.

c. With forward chaining, as soon as we are told the three facts about JB, we add the new fact `worth(jb,575)`. Then when we get the query `worth(jb,D)`, it is $O(1)$ to find the answer, assuming indexing on the predicate and first argument. This makes logical inference seem just like semantic networks except for two things: the logical inference does a hash table lookup instead of pointer following, and logical inference explicitly stores `worth` statements for each individual car, thus wasting space if there are a lot of individual cars. (For this kind of application, however, we will probably want to consider only a few individual cars, as opposed to the 11,000 different models.)

d. If each category has many properties—for example, the specifications of all the replacement parts for the vehicle—then forward-chaining on the implications will also be an impractical way to figure out the price of a vehicle.

e. If we have a rule of the following kind:

```
worth(X,D) :- year-make-style(X,Yr,Mk,St),
              year-make-style(Y,Yr,Mk,St), worth(Y,D).
```

together with facts in the database about some other specific vehicle of the same type as JB, then the query `worth(jb,D)` will be solved in $O(1)$ time with appropriate indexing, regardless of how many other facts are known about that type of vehicle and regardless of the number of types of vehicle.

Exercise 10.NATS

One might suppose that the syntactic distinction between unboxed links and singly boxed links in semantic networks is unnecessary, because singly boxed links are always attached to categories; an inheritance algorithm could simply assume that an unboxed link attached to a category is intended to apply to all members of that category. Show that this argument is fallacious, giving examples of errors that would arise.

When categories are reified, they can have properties as individual objects (such as $Cardinality$ and $Supersets$) that do not apply to their elements. Without the distinction between boxed and unboxed links, the sentence $Cardinality(SingletonSets, 1)$ might mean that every singleton set has one element, or that there si only one singleton set.

10.6 Reasoning with Default Information

[[need exercises]]