9.17 This exercise looks at the recursive application of rewrite rules, using logic programming. A rewrite rule (or demodulator in OTTER terminology) is an equation with a specified direction. For example, the rewrite rule \( x + 0 \rightarrow x \) suggests replacing any expression that matches \( r + 0 \) with the expression \( x \). Rewrite rules are a key component of equational reasoning systems. Use the predicate rewrite \( (X, Y) \) to represent rewrite rules. For example, the earlier rewrite rule is written as rewrite \( (X+0, X) \). Some terms are primitive and cannot be further simplified; thus, we write \( \text{primitive}(0) \) to say that 0 is a primitive term.

a. Write a definition of a predicate \( \text{simplify}(X, Y) \), that is true when \( Y \) is a simplified version of \( X \)—that is, when no further rewrite rules apply to any subexpression of \( Y \).

b. Write a collection of rules for the simplification of expressions involving arithmetic operators, and apply your simplification algorithm to some sample expressions.

c. Write a collection of rewrite rules for symbolic differentiation, and use them along with your simplification rules to differentiate and simplify expressions involving arithmetic expressions, including exponentiation.

9.18 This exercise considers the implementation of search algorithms in Prolog. Suppose that \( \text{successor}(X, Y) \) is true when state \( Y \) is a successor of state \( X \); and that \( \text{goal}(X) \) is true when \( X \) is a goal state. Write a definition for \( \text{solve}(X, P) \), which means that \( P \) is a path (list of states) beginning with \( X \), ending in a goal state, and consisting of a sequence of legal steps as defined by \( \text{successor} \). You will find that depth-first search is the easiest way to do this. How easy would it be to add heuristic search control?

9.19 Suppose a knowledge base contains just the following first-order Horn clauses:

\[
\begin{align*}
\text{Ancestor}(\text{Mother}(x), x) \\
\text{Ancestor}(x, y) \land \text{Ancestor}(y, z) \land \text{Ancestor}(x, z)
\end{align*}
\]

Consider a forward chaining algorithm that, on the \( j \)th iteration, terminates if the KB contains a sentence that unifies with the query, else adds to the KB every atomic sentence that can be inferred from the sentences already in the KB after iteration \( j - 1 \).

a. For each of the following queries, say whether the algorithm will (1) give an answer (if so, write down that answer); or (2) terminate with no answer, or (3) never terminate.

(i) \( \text{Ancestor}([\text{Mother}(y), \text{John}]) \)

(ii) \( \text{Ancestor}([\text{Mother}(\text{Mother}(y)), \text{John}]) \)

(iii) \( \text{Ancestor}([\text{Mother}(\text{Mother}(\text{Mother}(y))), \text{Mother}(y)]) \)

(iv) \( \text{Ancestor}([\text{Mother}(\text{John}), \text{Mother}(\text{Mother}(\text{Mother}(\text{John})�)])]) \)

b. Can a resolution algorithm prove the sentence \( \neg \text{Ancestor}(\text{John}, \text{John}) \) from the original knowledge base? Explain how, or why not.

c. Suppose we add the assertion that \( \neg (\text{Mother}(x) = x) \) and augment the resolution algorithm with inference rules for equality. Now what is the answer to (b)?

9.20 Let \( C \) be the first-order language with a single predicate \( S(p, q) \), meaning "\( p \) shaves \( q \)." Assume a domain of people.
Exercises 365

a. Consider the sentence "There exists a person $P$ who shaves every one $\text{who does not shave themselves, and only people that do not shave themselves.}" Express this in

b. Convert the sentence in (a) to clausal form.

c. Construct a resolution proof to show that the clauses in (b) are inherently inconsistent. (Note: you do not need any additional axioms.)

9.21 How can resolution be used to show that a sentence is valid? Unsatisfiable?

9.22 Construct an example of two clauses that can be resolved together in two different ways giving two different outcomes.

9.23 From "Horses are animals," it follows that "The head of a horse is the head of an animal." Demonstrate that this inference is valid by carrying out the following steps:

a. Translate the premise and the conclusion into the language of first-order logic. Use three predicates: $\text{HeadOf}(h, x)$ (meaning "$h$ is the head of $x"$), $\text{Horse}(x)$, and $\text{Animal}(x)$.

b. Negate the conclusion, and convert the premise and the negated conclusion into conjunctive normal form.

c. Use resolution to show that the conclusion follows from the premise.

9.24 Here are two sentences in the language of first-order logic:

(A) $\forall x \exists y (x \succ y)$

(B) $\exists y \forall x (x \succ y)$

a. Assume that the variables range over all the natural numbers $0, 1, 2, \ldots, \infty$ and that the $\succ$ predicate means "is greater than or equal to" Under this interpretation, translate (A) and (B) into English.

b. Is (A) true under this interpretation?

c. Is (B) true under this interpretation?

d. Does (A) logically entail (B)?

e. Does (B) logically entail (A)?

f. Using resolution, try to prove that (A) follows from (B). Do this even if you think that (B) does not logically entail (A); continue until the proof breaks down and you cannot proceed (if it does break down). Show the unifying substitution for each resolution step. If the proof fails, explain exactly where, how, and why it breaks down.

g. Now try to prove that (B) follows from (A).

9.25 Resolution can produce nonconstructive proofs for queries with variables, so we had to introduce special mechanisms to extract definite answers. Explain why this issue does not arise with knowledge bases containing only definite clauses.

9.26 We said in this chapter that resolution cannot be used to generate all logical consequences of a set of sentences. Can any algorithm do this?