```
function PL-FC-ENTAILS?($KB$, $q$) returns $true$ or $false$
  inputs: $KB$, the knowledge base, a set of propositional definite clauses
          $q$, the query, a proposition symbol
  $count \leftarrow$ a table, where $count[c]$ is initially the number of symbols in clause $c$'s premise
  $inferred \leftarrow$ a table, where $inferred[s]$ is initially $false$ for all symbols
  $queue \leftarrow$ a queue of symbols, initially symbols known to be true in $KB$

  while $queue$ is not empty do
    $p \leftarrow$ POP($queue$)
    if $p = q$ then return $true$
    if $inferred[p] = false$ then
      $inferred[p] \leftarrow true$
      for each clause $c$ in $KB$ where $p$ is in $c$.PREMISE do
        decrement $count[c]$
        if $count[c] = 0$ then add $c$.CONCLUSION to $queue$
  return $false$
```

Figure 7.15 The forward-chaining algorithm for propositional logic. The *queue* keeps track of symbols known to be true but not yet "processed." The *count* table keeps track of how many premises of each implication are not yet proven. Whenever a new symbol p from the queue is processed, the count is reduced by one for each implication in whose premise p appears (easily identified in constant time with appropriate indexing.) If a count reaches zero, all the premises of the implication are known, so its conclusion can be added to the queue. Finally, we need to keep track of which symbols have been processed; a symbol that is already in the set of inferred symbols need not be added to the queue again. This avoids redundant work and prevents loops caused by implications such as $P \Rightarrow Q$ and $Q \Rightarrow P$.

To see this, assume the opposite, namely that some clause $a_1 \wedge \ldots \wedge a_k \Rightarrow b$ is false in the model. Then $a_1 \wedge \ldots \wedge a_k$ must be true in the model and b must be false in the model. But this contradicts our assumption that the algorithm has reached a fixed point, because we would now be licensed to add b to the KB. We can conclude, therefore, that the set of atomic sentences inferred at the fixed point defines a model of the original KB. Furthermore, any atomic sentence q that is entailed by the KB must be true in all its models and in this model in particular. Hence, every entailed atomic sentence q must be inferred by the algorithm.

Data-driven

Forward chaining is an example of the general concept of **data-driven** reasoning—that is, reasoning in which the focus of attention starts with the known data. It can be used within an agent to derive conclusions from incoming percepts, often without a specific query in mind. For example, the wumpus agent might TELL its percepts to the knowledge base using an incremental forward-chaining algorithm in which new facts can be added to the agenda to initiate new inferences. In humans, a certain amount of data-driven reasoning occurs as new information arrives. For example, if I am indoors and hear rain starting to fall, it might occur to me that the picnic will be canceled. Yet it will probably not occur to me that the seventeenth petal on the largest rose in my neighbor's garden will get wet; humans keep forward chaining under careful control, lest they be swamped with irrelevant consequences.

The backward-chaining algorithm, as its name suggests, works backward from the query. If the query q is known to be true, then no work is needed. Otherwise, the algorithm finds