

```

function FOL-FC-ASK( $KB, \alpha$ ) returns a substitution or false
  inputs:  $KB$ , the knowledge base, a set of first-order definite clauses
            $\alpha$ , the query, an atomic sentence
  local variables: new, the new sentences inferred on each iteration

  repeat until new is empty
     $new \leftarrow \{ \}$ 
    for each rule in  $KB$  do
       $(p_1 \wedge \dots \wedge p_n \Rightarrow q) \leftarrow \text{STANDARDIZE-VARIABLES}(\textit{rule})$ 
      for each  $\theta$  such that  $\text{SUBST}(\theta, p_1 \wedge \dots \wedge p_n) = \text{SUBST}(\theta, p'_1 \wedge \dots \wedge p'_n)$ 
        for some  $p'_1, \dots, p'_n$  in  $KB$ 
           $q' \leftarrow \text{SUBST}(\theta, q)$ 
          if  $q'$  does not unify with some sentence already in  $KB$  or new then
            add  $q'$  to new
             $\phi \leftarrow \text{UNIFY}(q', \alpha)$ 
            if  $\phi$  is not fail then return  $\phi$ 
          add new to  $KB$ 
  return false

```

Figure 9.3 A conceptually straightforward, but very inefficient, forward-chaining algorithm. On each iteration, it adds to KB all the atomic sentences that can be inferred in one step from the implication sentences and the atomic sentences already in KB . The function STANDARDIZE-VARIABLES replaces all variables in its arguments with new ones that have not been used before.

```

function FOL-BC-ASK( $KB, query$ ) returns a generator of substitutions
  return FOL-BC-OR( $KB, query, \{ \}$ )



---


generator FOL-BC-OR( $KB, goal, \theta$ ) yields a substitution
  for each  $(lhs \Rightarrow rhs)$  in FETCH-RULES-FOR-GOAL( $KB, goal$ ) do
     $(lhs, rhs) \leftarrow \text{STANDARDIZE-VARIABLES}((lhs, rhs))$ 
    for each  $\theta'$  in FOL-BC-AND( $KB, lhs, \text{UNIFY}(rhs, goal, \theta)$ ) do
      yield  $\theta'$ 



---


generator FOL-BC-AND( $KB, goals, \theta$ ) yields a substitution
  if  $\theta = failure$  then return
  else if LENGTH( $goals$ ) = 0 then yield  $\theta$ 
  else do
     $first, rest \leftarrow \text{FIRST}(goals), \text{REST}(goals)$ 
    for each  $\theta'$  in FOL-BC-OR( $KB, \text{SUBST}(\theta, first), \theta$ ) do
      for each  $\theta''$  in FOL-BC-AND( $KB, rest, \theta'$ ) do
        yield  $\theta''$ 

```

Figure 9.6 A simple backward-chaining algorithm for first-order knowledge bases.