FORWARD & BACKWARD CHAINING

HORN CLAUSES AND DEFINITE CLAUSES

HORN CLAUSES

- Disjunction of literals of which at most one is positive.

1. Exactly one positive literal (one positive literal and at least one negative literal) : DEFINITE CLAUSE
2. Exactly one positive literal and Zero negative literal : FACT
3. Zero positive literal : NEGATED GOAL
4. Zero positive literal and Zero negative literal : Appears only as the end of a resolution proof

- Conjunction (logical AND); Disjunction (logical OR)

DEFINITE AND HORN CLAUSES

Definite clauses

- Disjunction of literals of which exactly one is positive.
- (¬L1,1 v ¬Breeze v B1,1) ✓
- (¬B1,1 v P1,2 v P2,1) ✗

Horn clauses

- Disjunction of literals of which at most one is positive.
- All definite clauses are Horn clauses
- Horn clauses with NO positive literals are called Goal clauses.

$$(\overset{\text{Premise}}{\underset{\text{Body}}{\neg L1,1 \lor \neg Breeze}} \lor \overset{\text{Conclusion}}{\underset{\text{Head}}{B1,1}}) \equiv \overset{\text{Premise}}{\underset{\text{Body}}{(L1,1 \land Breeze)}} \Rightarrow \overset{\text{Conclusion}}{\underset{\text{Head}}{B1,1}}$$

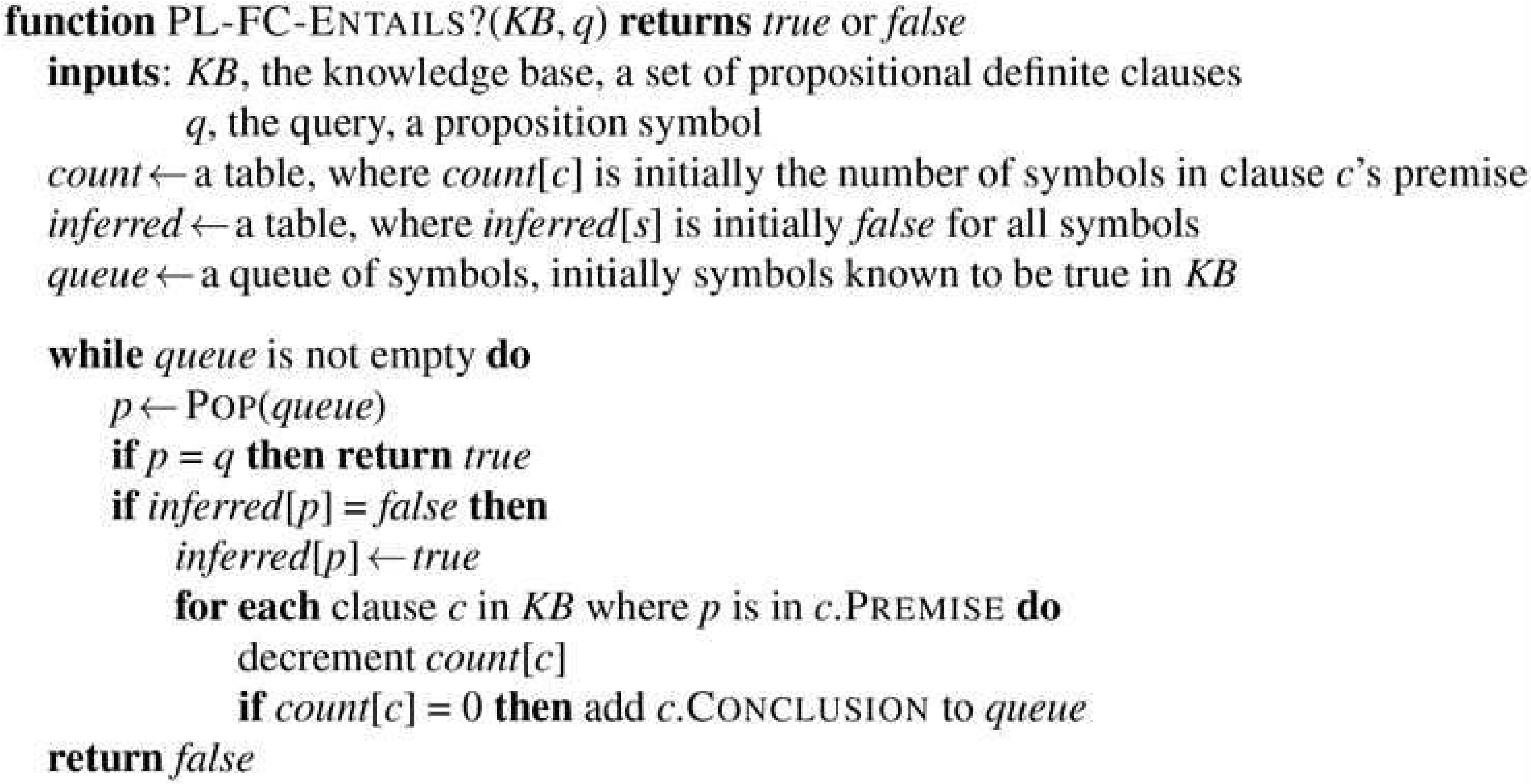
FORWARD CHAINING

- Starts with the known facts and asserts new facts.

- The forward-chaining algorithm PL-FC-ENTAILS? (KB, q) determines if a single proposition symbol q—the query—is entailed by a knowledge base of definite clauses.
- It begins from known facts (positive literals) in the knowledge base. If all the premises of an implication are known, then its conclusion is added to the set of known facts.

if $L_{1,1}$and Breeze are known and ($L_{1,1}$ ∧ Breeze) ⇒ $B_{1,1}$ is in the knowledge base, then $B_{1,1}$ can be added.

FORWARD CHAINING



```
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
```

FORWARD CHAINING

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

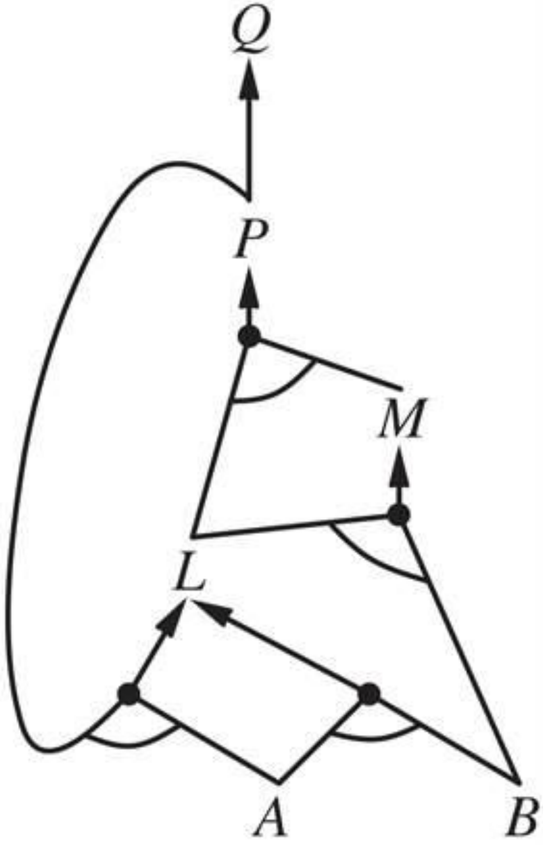
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

(a)



(b)

BACKWARD CHAINING

- Works backward from the query.
- If the query is known to be true, then no work is needed.
- Otherwise, the algorithm finds those implications in the knowledge base whose conclusion is q.
- If all the premises of one of those implications can be proved true (by backward chaining), then q is true.

- The algorithm is essentially identical to the AND-OR-GRAPH-SEARCH algorithm.
- Backward chaining is a form of **goal-directed reasoning**.