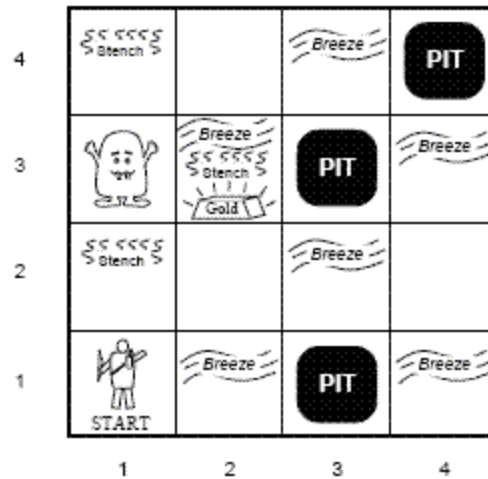


INFERENCE IN PROPOSITIONAL LOGIC



Outline

- Inference rules and theorem proving
 - Forward chaining
 - Backward chaining
 - Resolution

Proof Methods

- Proof methods divide into (roughly) two kinds:
- Application of inference rules
 - Legitimate (sound) generation of new sentences from old
 - Proof = a sequence of inference rule applications
 - Can use inference rules as operators in a standard search algorithm
 - Typically require translation of sentences into a normal form
- Model checking
 - Truth table enumeration (always exponential in n)
 - Heuristic search in model space (sound but incomplete)
 - e.g., min-conflicts-like hill-climbing algorithms

Forward and Backward Chaining

- Horn Form (restricted)
 - KB = conjunction of Horn clauses
 - Horn clause =
 - proposition symbol; or
 - (conjunction of symbols) \Rightarrow symbol
 - E.g., $C \wedge (B \Rightarrow A) \wedge (C \wedge D) \Rightarrow B$
- Modus Ponens (for Horn Form): complete for Horn KBs

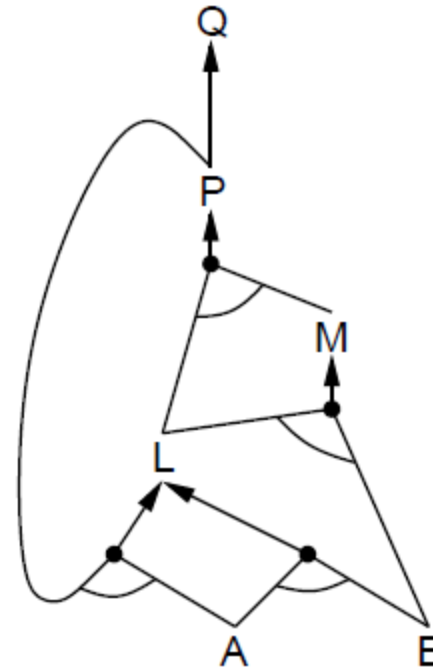
$$\frac{\alpha_1, \dots, \alpha_n, \quad \alpha_1 \wedge \dots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

- Can be used with forward chaining or backward chaining.
- These algorithms are very natural and run in linear time

Forward Chaining

- Idea: fire any rule whose premises are satisfied in the KB, add its conclusion to the KB, until query is found

- $P \Rightarrow Q$
- $L \wedge M \Rightarrow P$
- $B \wedge L \Rightarrow M$
- $A \wedge P \Rightarrow L$
- $A \wedge B \Rightarrow L$
- A
- B



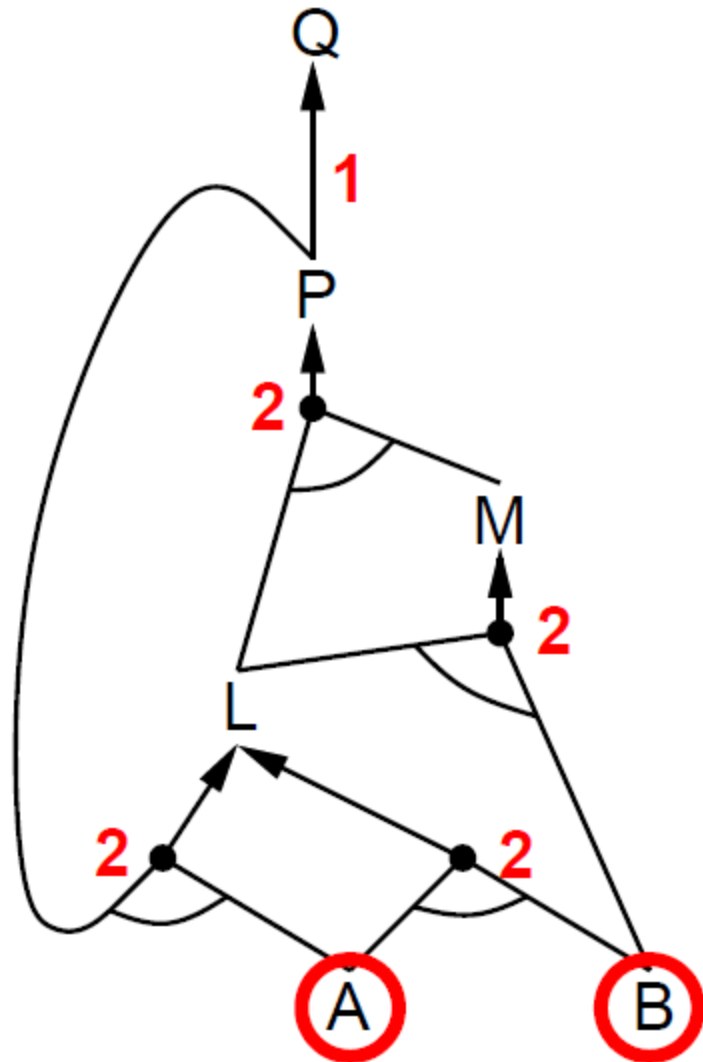
Forward Chaining Algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  inputs: KB, the knowledge base, a set of propositional Horn clauses
         q, the query, a proposition symbol
  local variables: count, a table, indexed by clause, initially the number of premises
                  inferred, a table, indexed by symbol, each entry initially false
                  agenda, a list of symbols, initially the symbols known in KB

  while agenda is not empty do
    p ← POP(agenda)
    unless inferred[p] do
      inferred[p] ← true
      for each Horn clause c in whose premise p appears do
        decrement count[c]
        if count[c] = 0 then do
          if HEAD[c] = q then return true
          PUSH(HEAD[c], agenda)

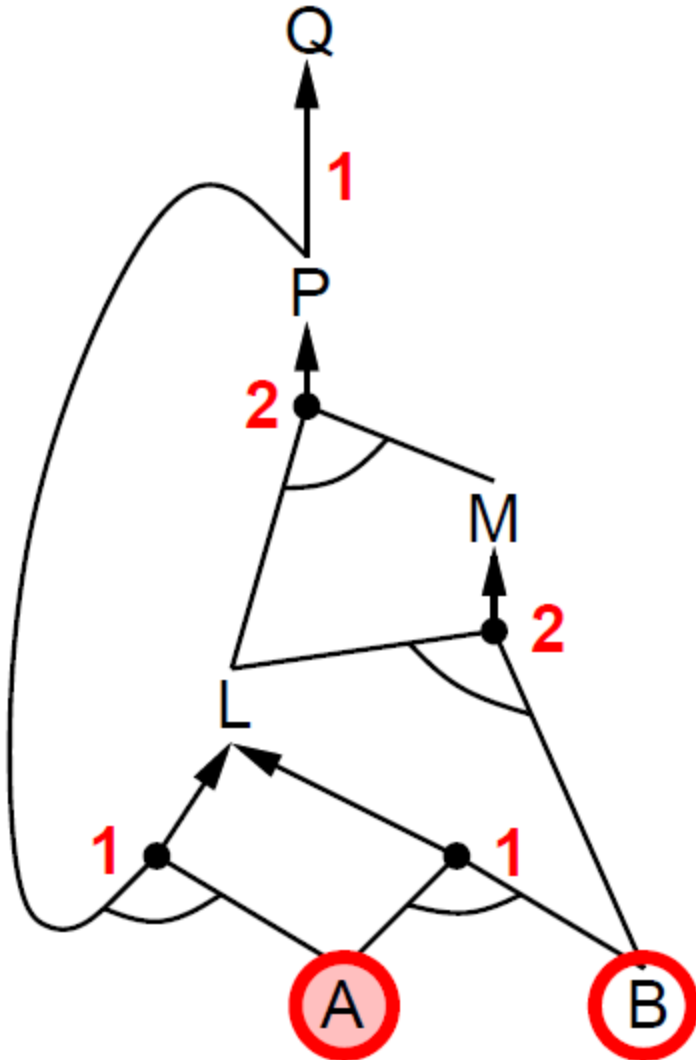
  return false
```

Forward Chaining Example



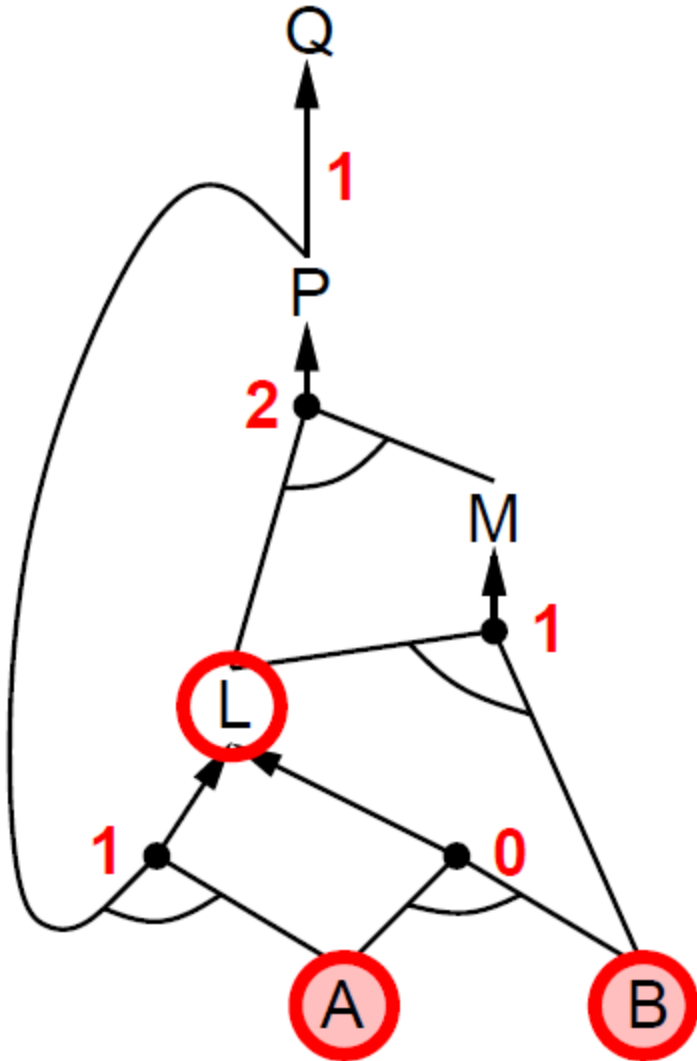
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Forward Chaining Example



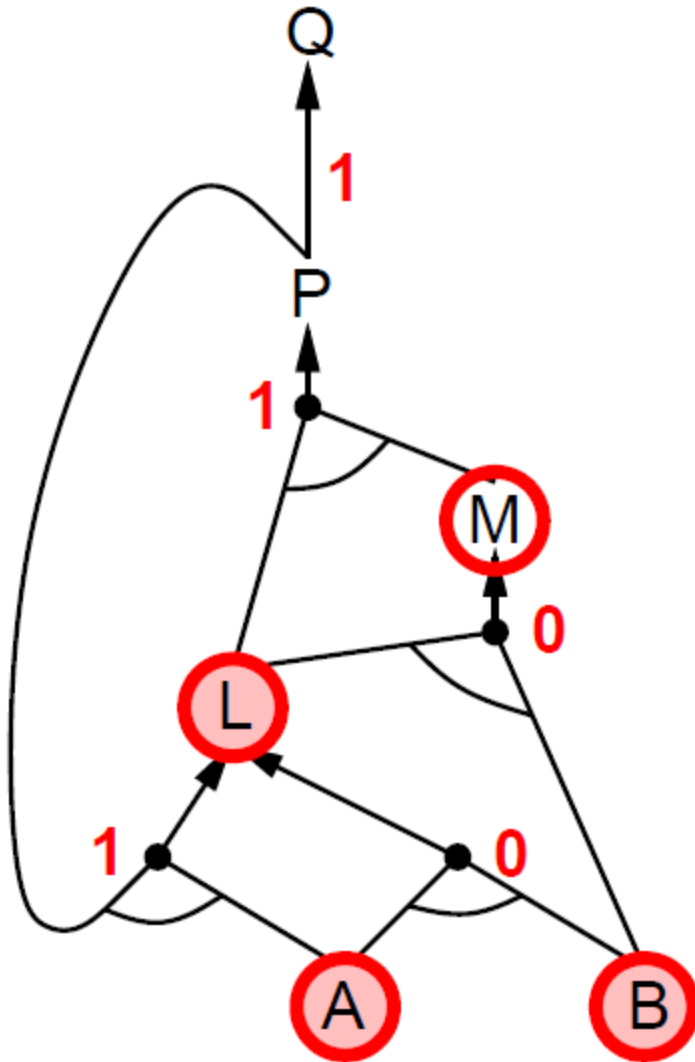
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Forward Chaining Example



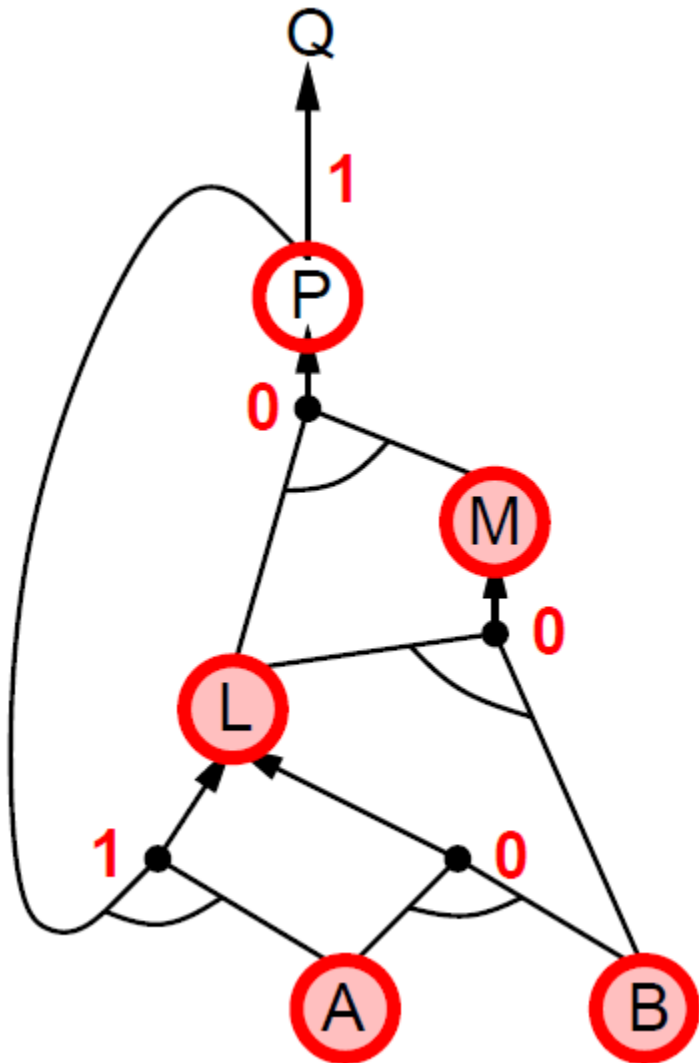
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Forward Chaining Example



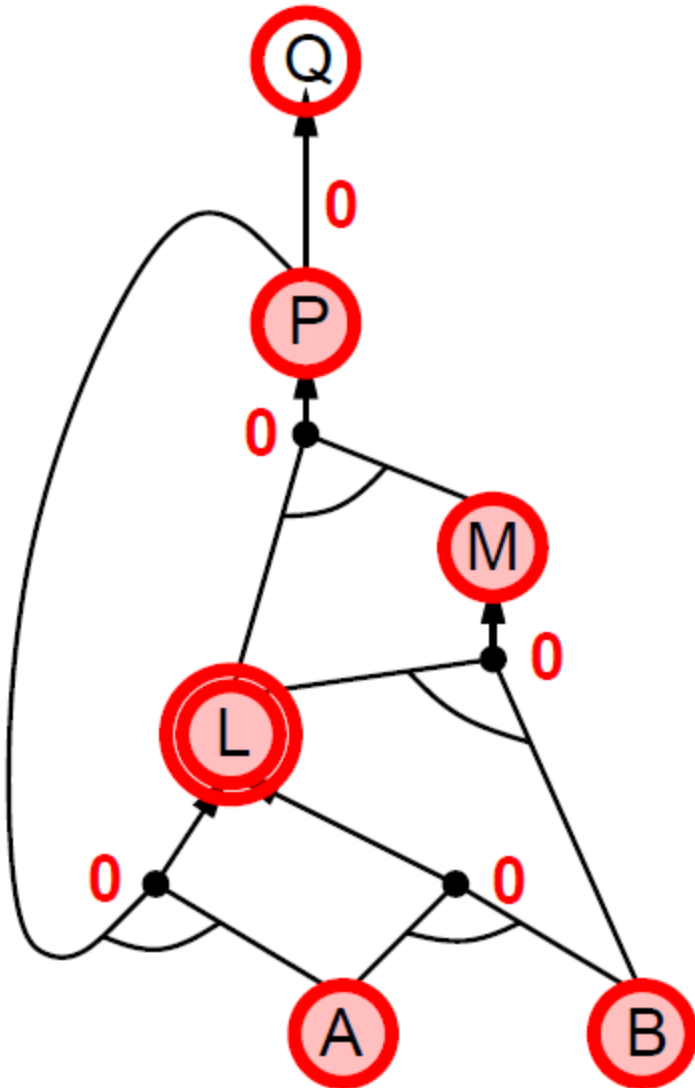
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Forward Chaining Example



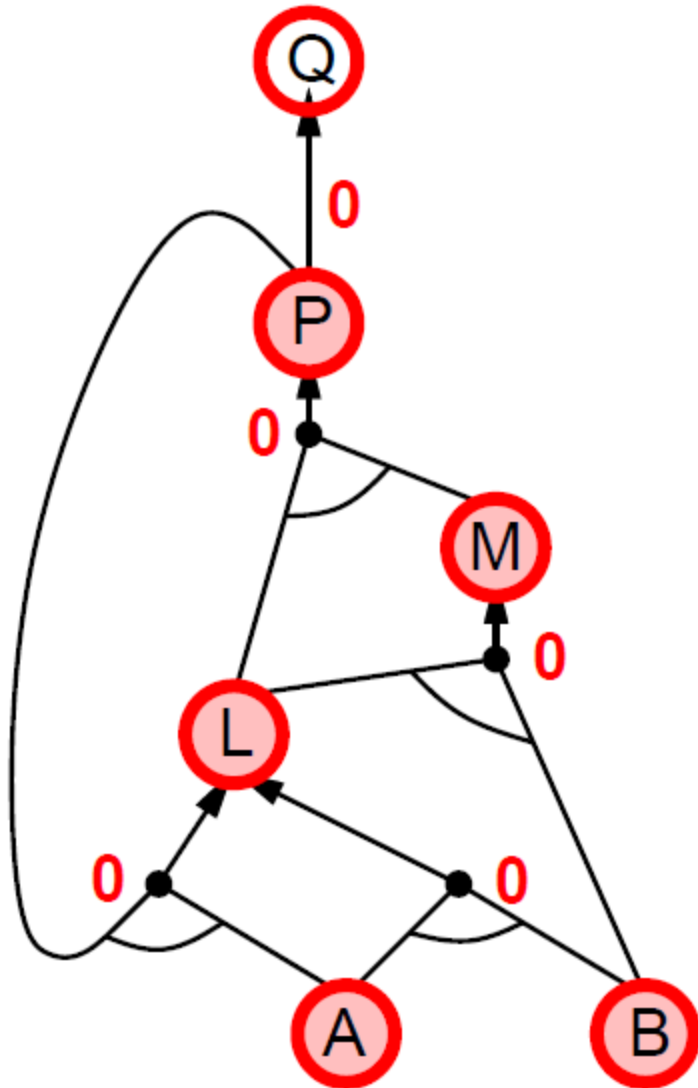
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Forward Chaining Example



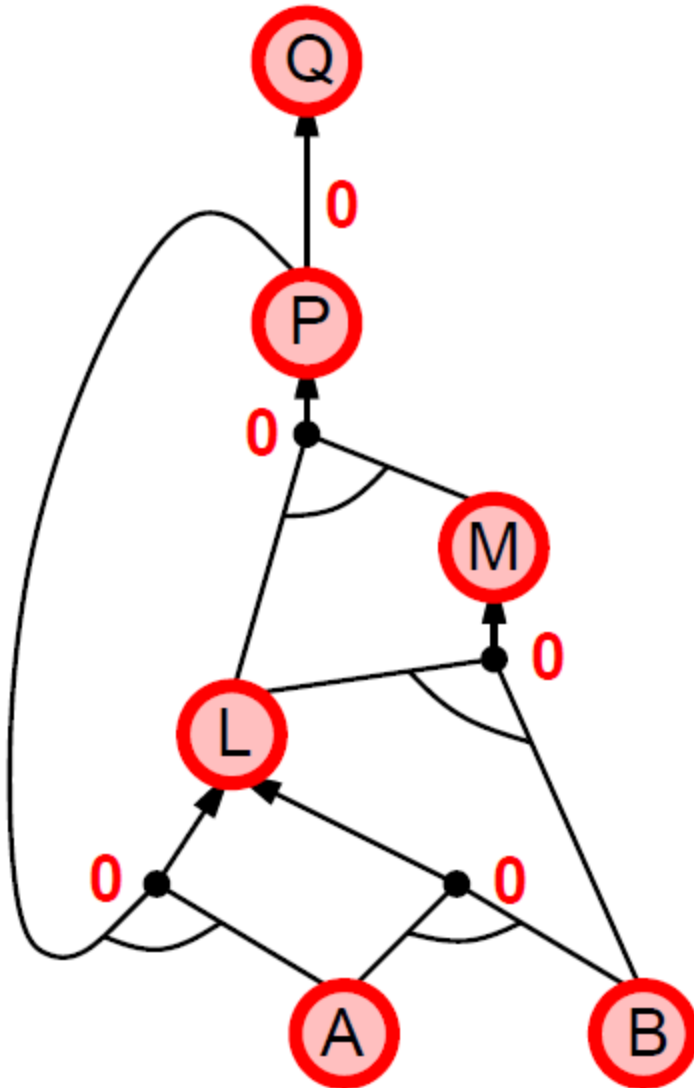
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Forward Chaining Example



$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Forward Chaining Example



$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

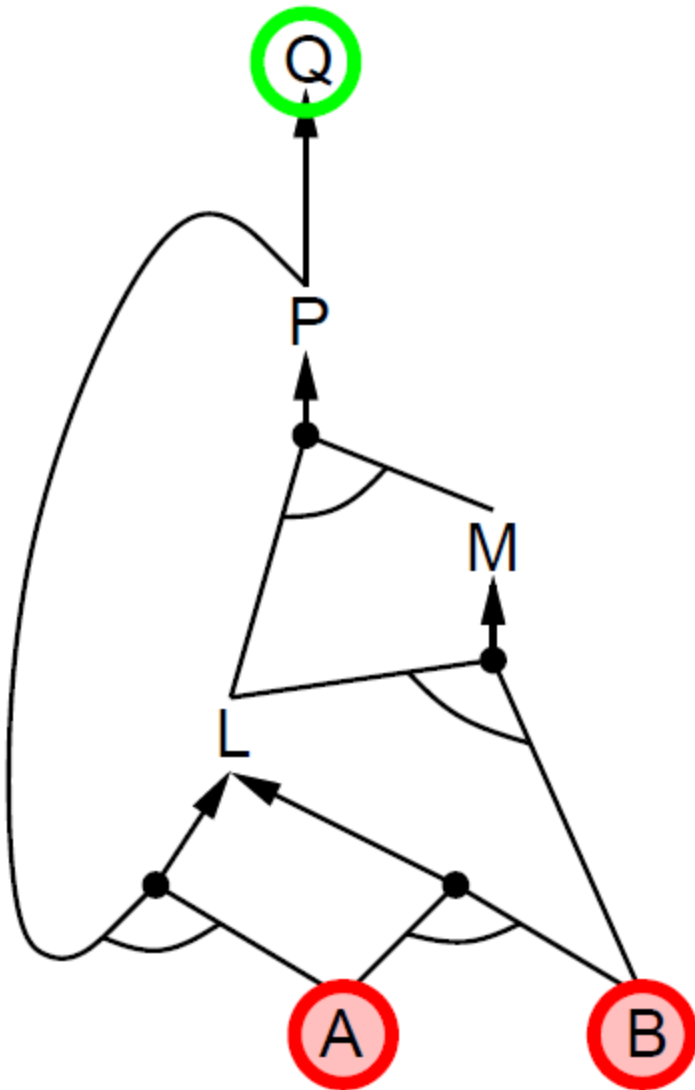
Proof of Completeness

- Forward chaining (FC) derives every atomic sentence that is entailed by KB
- 1. FC reaches a fixed point where no new atomic sentences are derived
- 2. Consider the final state as a model m , assigning true/false to symbols
- 3. Every clause in the original KB is true in m
 - Proof: Suppose a clause $a_1 \wedge \dots \wedge a_k \Rightarrow b$ is false in m
 - Then $a_1 \wedge \dots \wedge a_k$ is true in m and b is false in m
 - Therefore the algorithm has not reached a fixed point!
- 4. Hence m is a model of KB
- 5. If $\text{KB} \models q$, q is true in every model of KB, including m

Backward Chaining

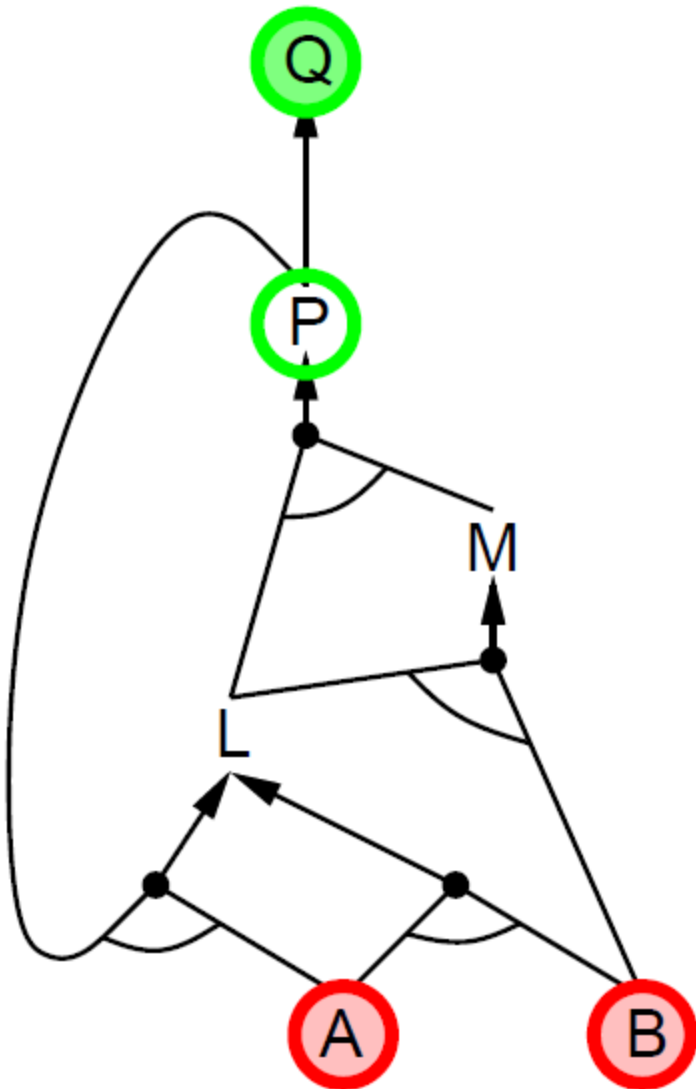
- Idea: work backwards from the query q :
 - To prove q by backward chaining,
 - Check if q is known already, or
 - Prove by backward chaining (BC) all premises of some rule concluding q
- Avoid loops: check if new subgoal is already on the goal stack
- Avoid repeated work: check if new subgoal
 - 1) has already been proved true, or
 - 2) has already failed

Backward Chaining Example



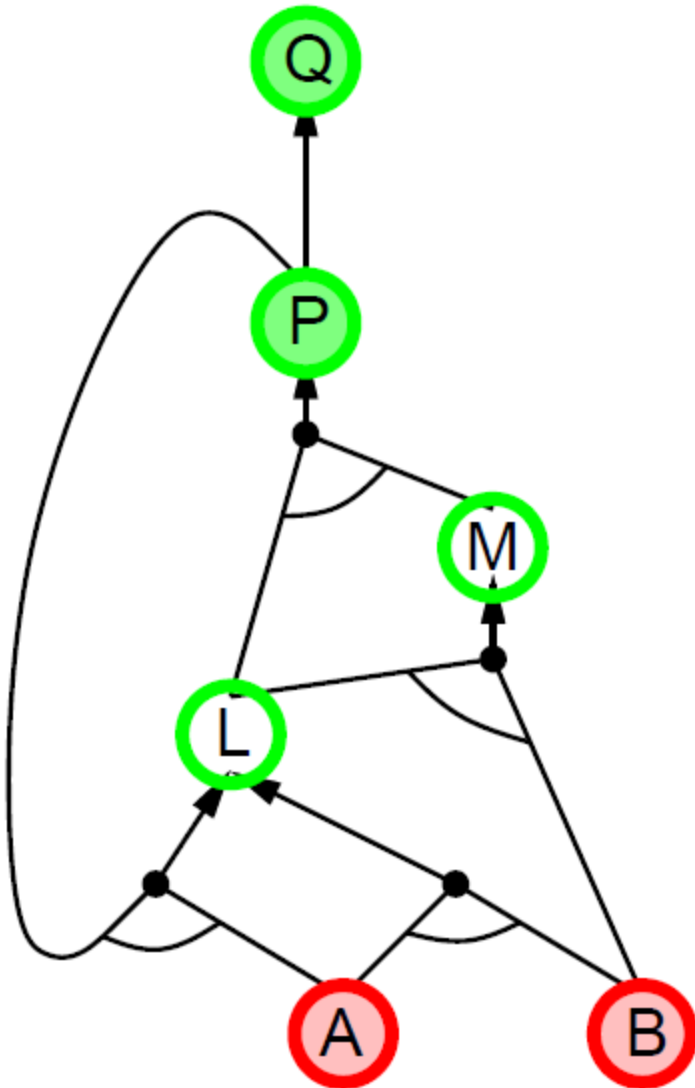
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Backward Chaining Example



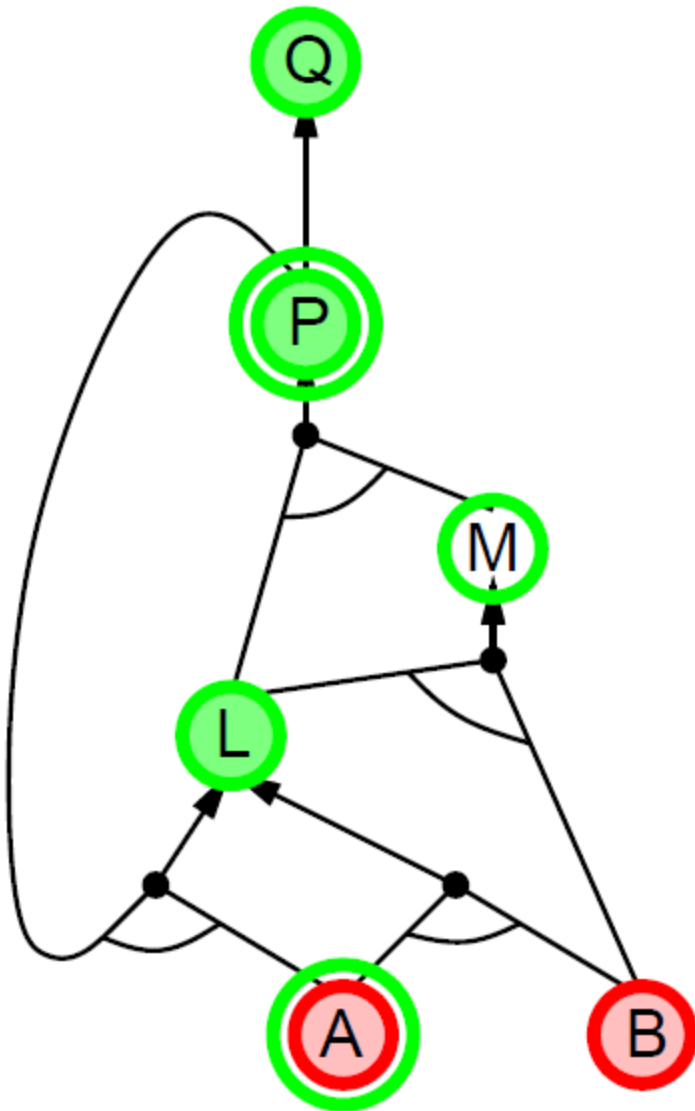
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Backward Chaining Example



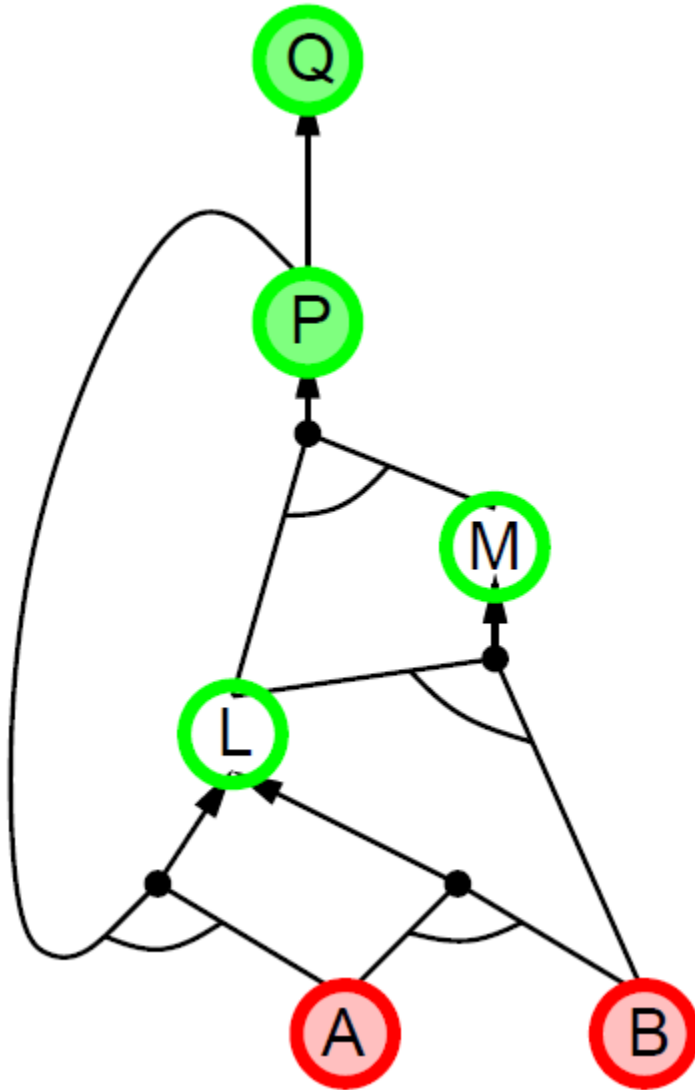
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Backward Chaining Example



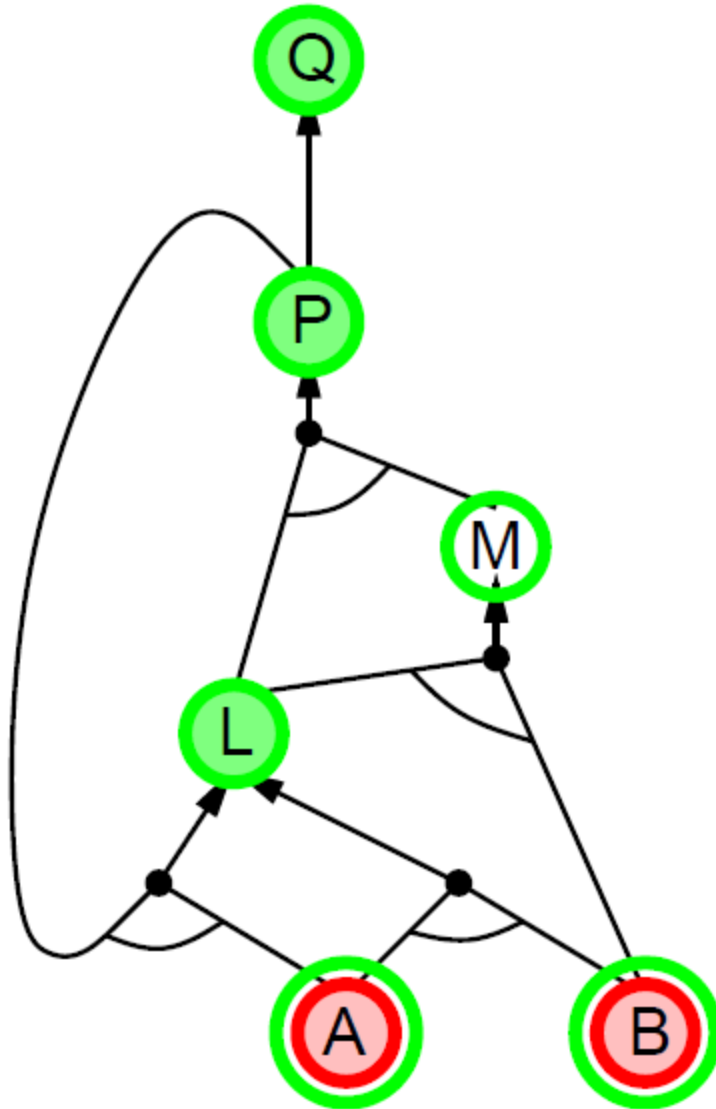
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Backward Chaining Example



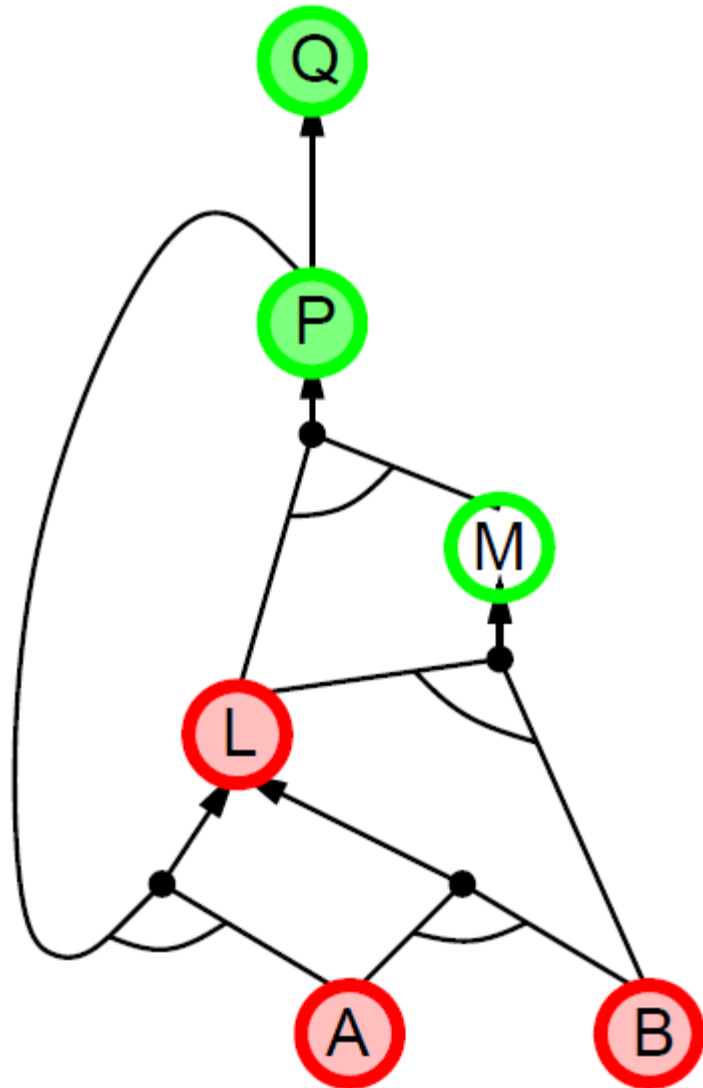
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Backward Chaining Example



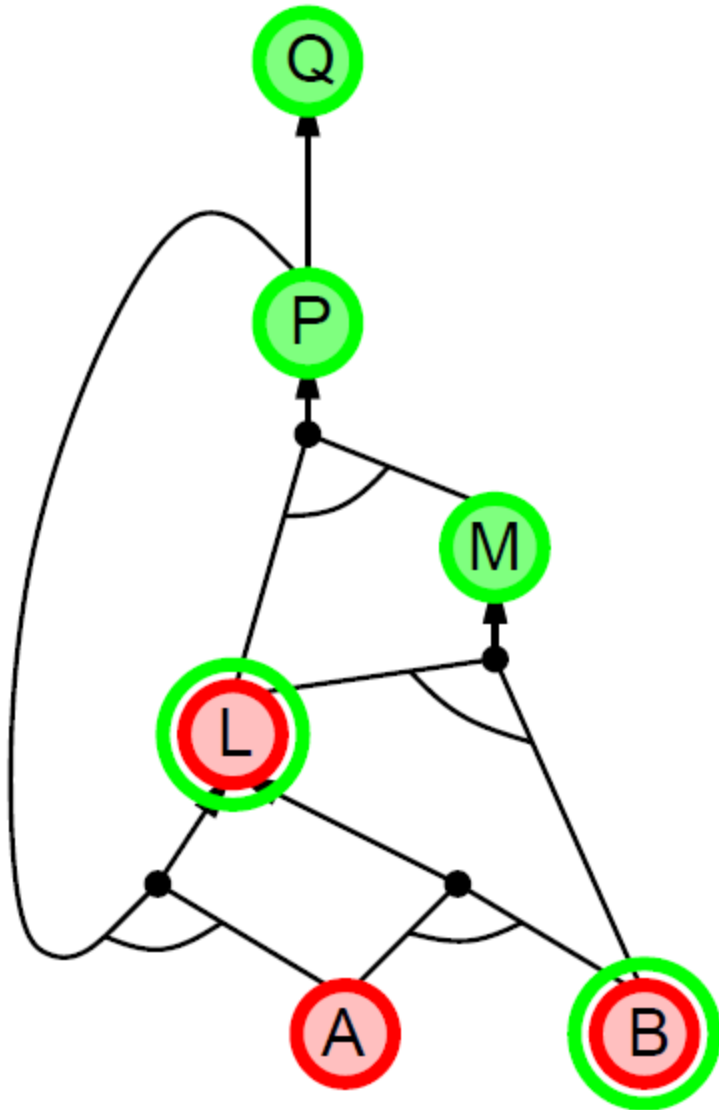
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Backward Chaining Example



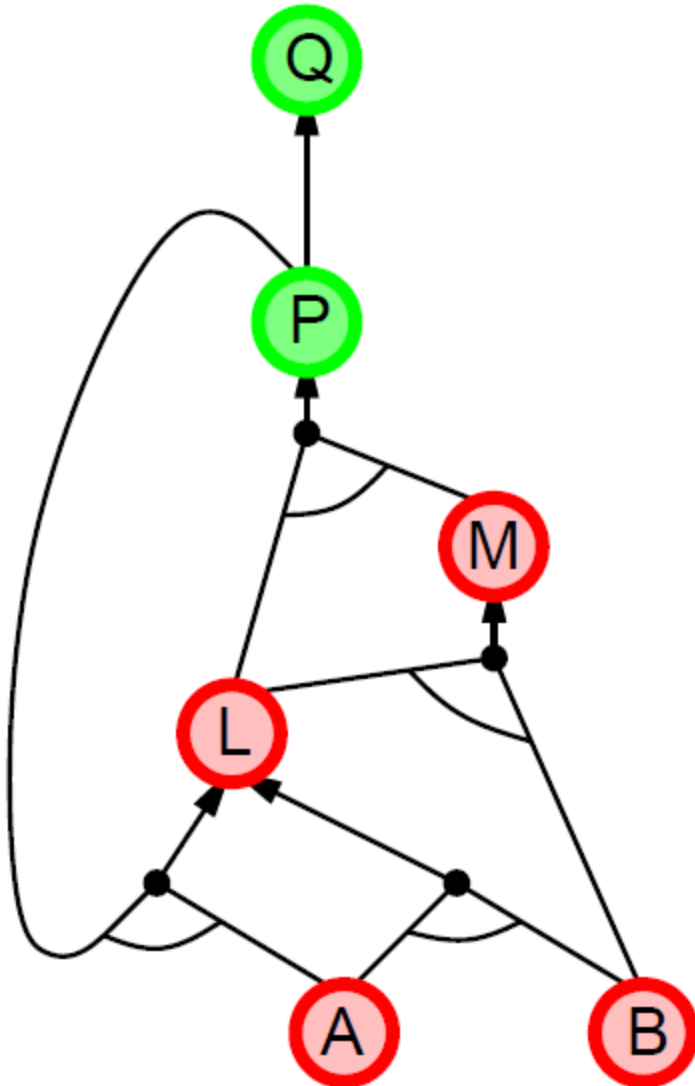
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Backward Chaining Example



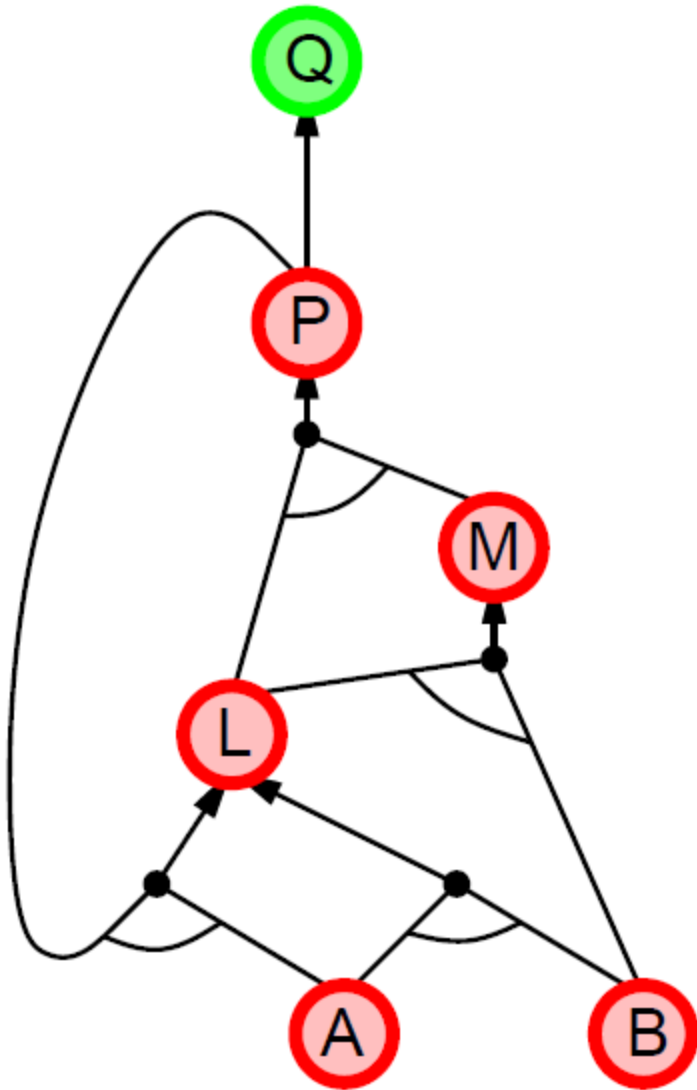
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Backward Chaining Example



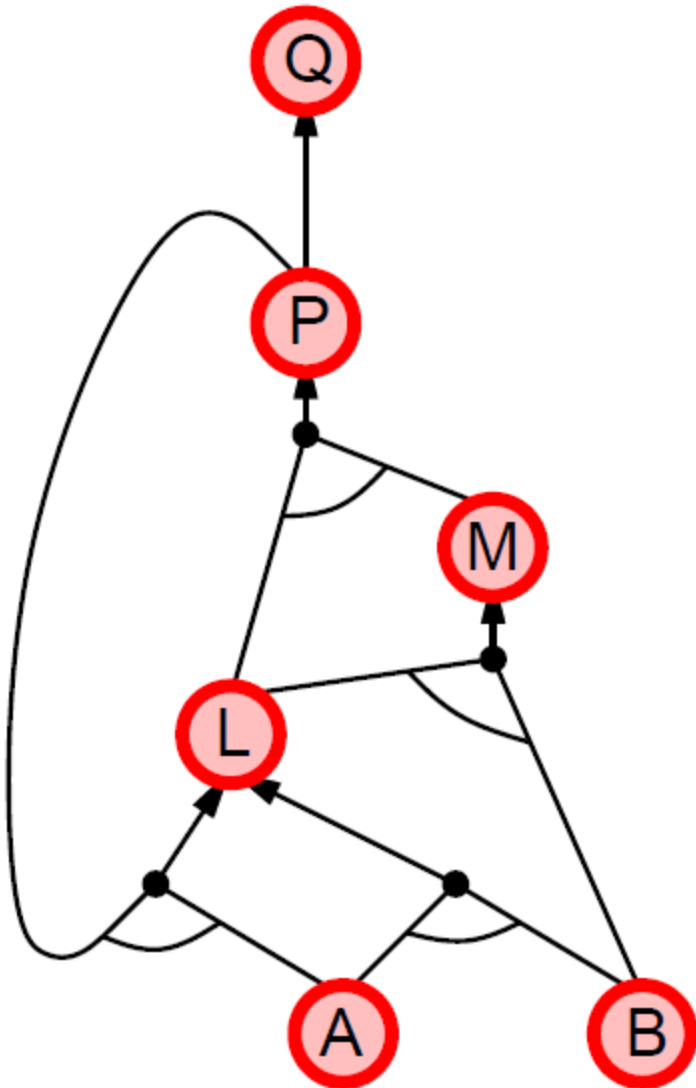
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Backward Chaining Example



$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Backward Chaining Example



$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
A
B

Forward vs. Backward Chaining

- FC is data-driven, automatic, unconscious processing,
 - e.g., object recognition, routine decisions
- May do lots of work that is irrelevant to the goal
- BC is goal-driven, appropriate for problem-solving,
 - e.g., Where are my keys? How do I get into a PhD program?
- Complexity of BC can be much less than linear in size of KB

Resolution

- Conjunctive Normal Form (CNF - universal)
 - Conjunction of disjunctions of literals
 - Disjunctions of literals means clauses
 - E.g.,
Resolution inference rule (for CNF): complete for propositional logic

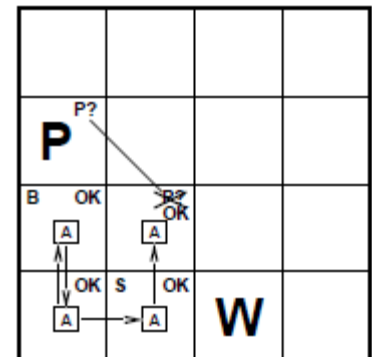
$$\frac{l_1 \vee \dots \vee l_k, \quad m_1 \vee \dots \vee m_n}{l_1 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n}$$

- where l_i and m_j are complementary literals.

E.g.,

$$\frac{P_{1,3} \vee P_{2,2}, \quad \neg P_{2,2}}{P_{1,3}}$$

- Resolution is sound and complete for propositional logic



Conversion to CNF

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$

1. Eliminate \Leftrightarrow , replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

$$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$$

2. Eliminate \Rightarrow , replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$$

3. Move \neg inwards using de Morgan's rules and double-negation:

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$$

4. Apply distributivity law (\vee over \wedge) and flatten:

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$$

- Proof by contradiction, i.e., show $KB \wedge \neg\alpha$: unsatisfiable

Resolution Algorithm

```
function PL-RESOLUTION( $KB, \alpha$ ) returns true or false
  inputs:  $KB$ , the knowledge base, a sentence in propositional logic
          $\alpha$ , the query, a sentence in propositional logic
   $clauses \leftarrow$  the set of clauses in the CNF representation of  $KB \wedge \neg\alpha$ 
   $new \leftarrow \{ \}$ 
  loop do
    for each  $C_i, C_j$  in  $clauses$  do
       $resolvents \leftarrow$  PL-RESOLVE( $C_i, C_j$ )
      if  $resolvents$  contains the empty clause then return true
       $new \leftarrow new \cup resolvents$ 
  if  $new \subseteq clauses$  then return false
   $clauses \leftarrow clauses \cup new$ 
```


Summary

- Logical agents apply inference to a knowledge base
- to derive new information and make decisions
- Basic concepts of logic:
 - Syntax: formal structure of sentences
 - Semantics: truth of sentences with respect to models
 - Entailment: necessary truth of one sentence given another
 - Inference: deriving sentences from other sentences
 - Soundness: derivations produce only entailed sentences
 - Completeness: derivations can produce all entailed sentences
- Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.
- Forward, backward chaining are linear-time, complete for Horn clauses
- Resolution is complete for propositional logic
- Propositional logic lacks expressive power