More Propositional Logic

- Inference
  - Forward and backward chaining
- Satisfiability problems.
  - Backtracking search (Davis-Putnam)
- A Propositional Logic based Wumpus World Agent

Forward and Backward Chaining

- Horn Clause:
  - Disjunction of literals in which at most one literal is positive.

$$\neg P \lor \neg Q \lor R$$

equivalent

  - Implication in which the premise is a conjunction of positive literals and the conclusion is a single positive literal.

$$(P \land Q) \Rightarrow R$$

  - There are many KBs which are naturally in this form.

Forward Chaining

- The KB is a conjunction of Horn Clauses
  - Each is an implication (rule).
- The goal is a single positive proposition Q.
- The answer tells us whether or not Q is entailed by the KB.
- Deciding entailment takes time linear in the size of the KB.

FC Algorithm (informal)

Do until Q is in the KB or nothing left to do:

- Find a rule whose premise is satisfied by the KB.
- Add the conclusion to the KB.

If we found Q (the conclusion of some rule whose premise was satisfied)

- KB entails Q

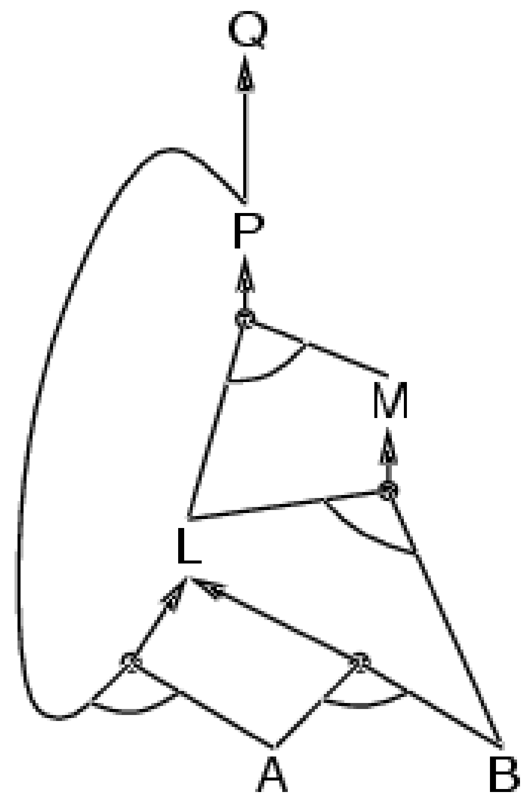
FC Algorithm (formal)

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  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 to be true

   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
```

FC Example

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



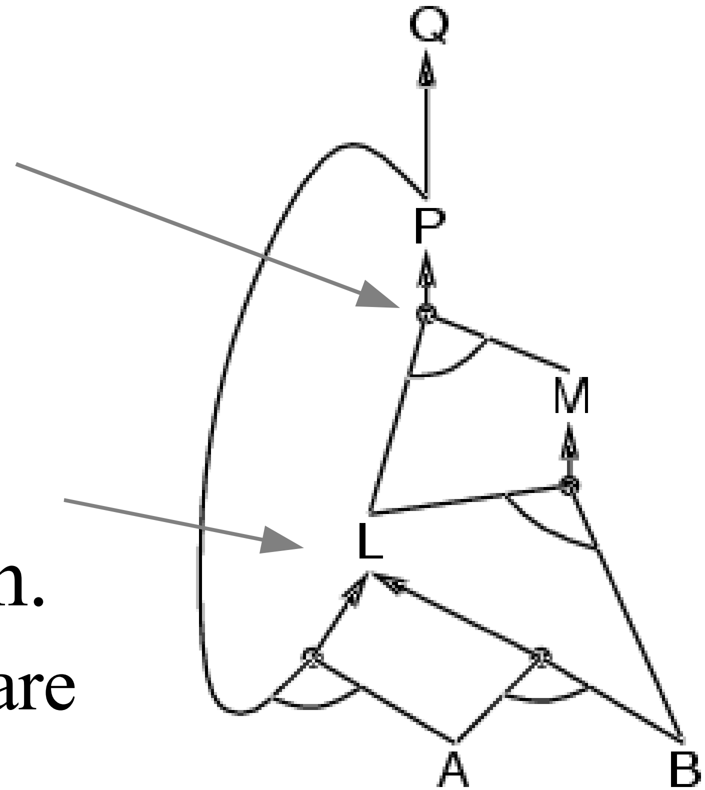
AND-OR Graph

Links joined by an arc indicate a conjunction

all must be true

Links joined with no arc indication a disjunction.

Proving any of the links are true is enough.



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

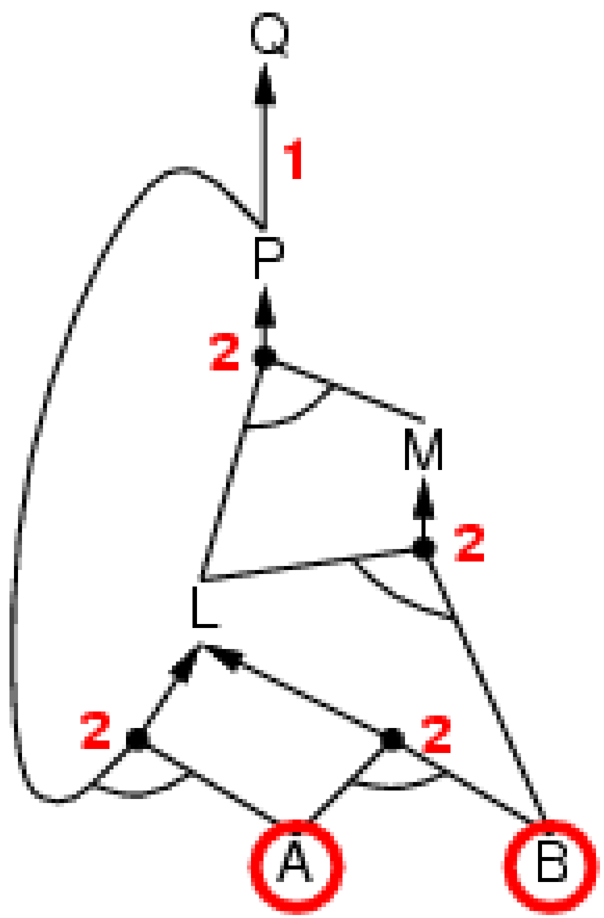
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

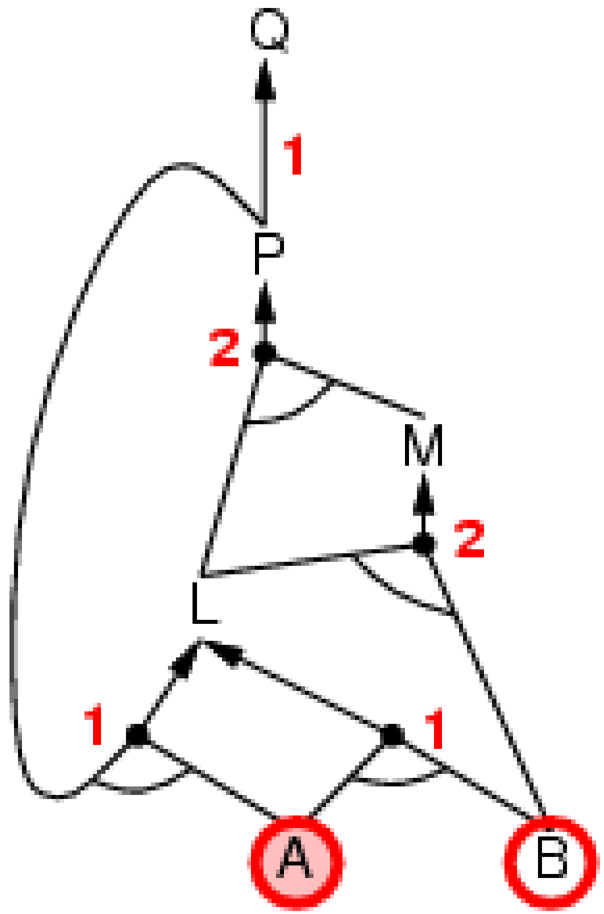
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

P ⇒ **Q**

L ∧ M ⇒ P

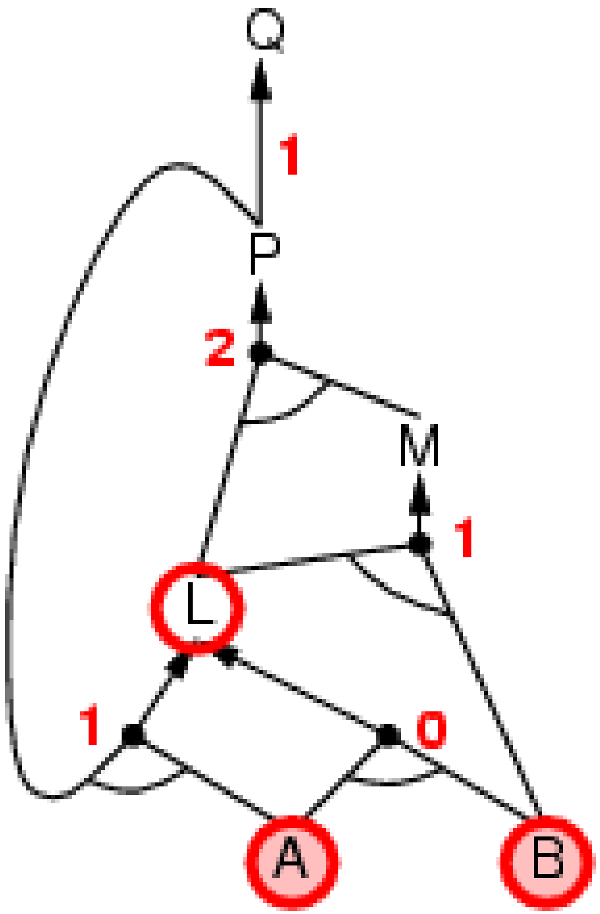
B ∧ L ⇒ M

A ∧ P ⇒ L

A ∧ B ⇒ L

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

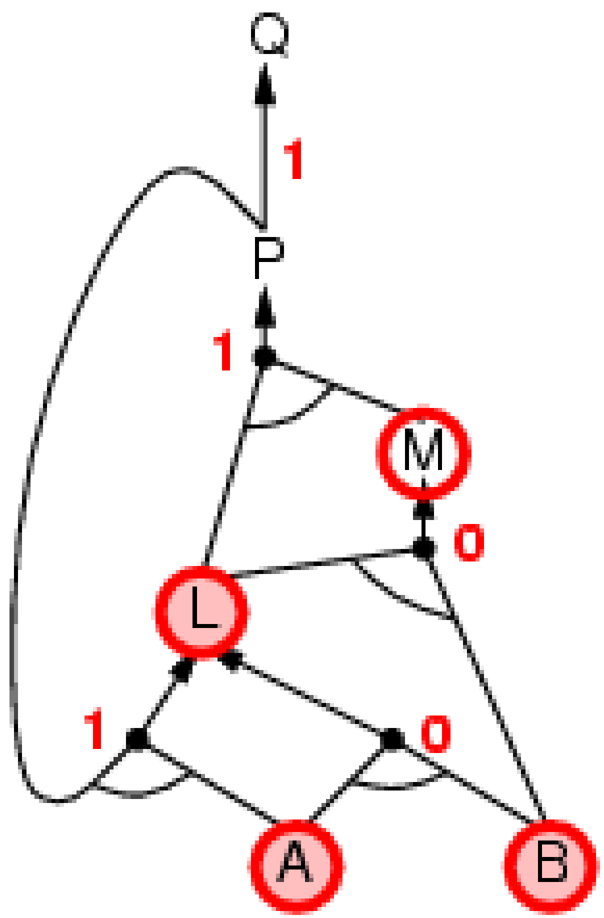
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

P ⇒ **Q**

L ∧ M ⇒ P

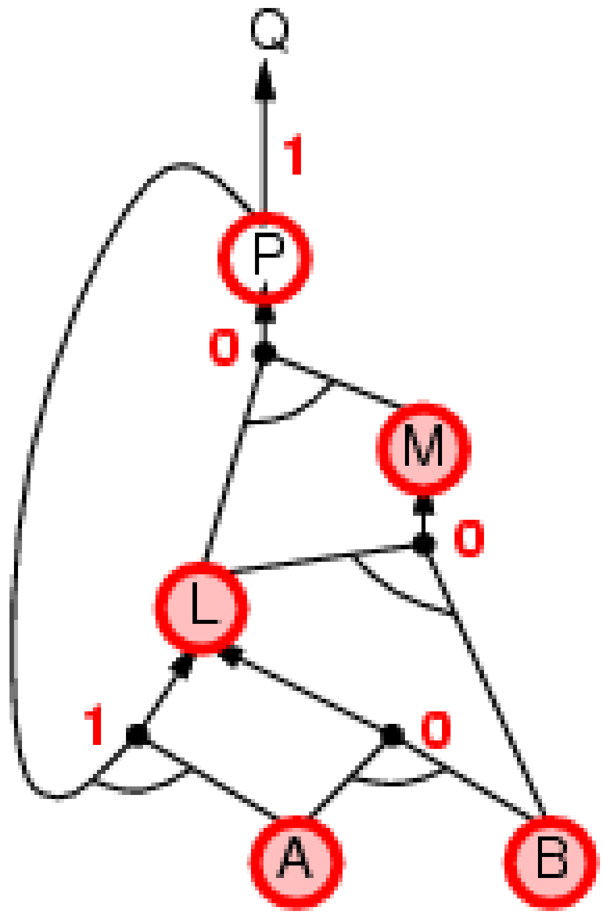
B ∧ L ⇒ M

A ∧ P ⇒ L

A ∧ B ⇒ L

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

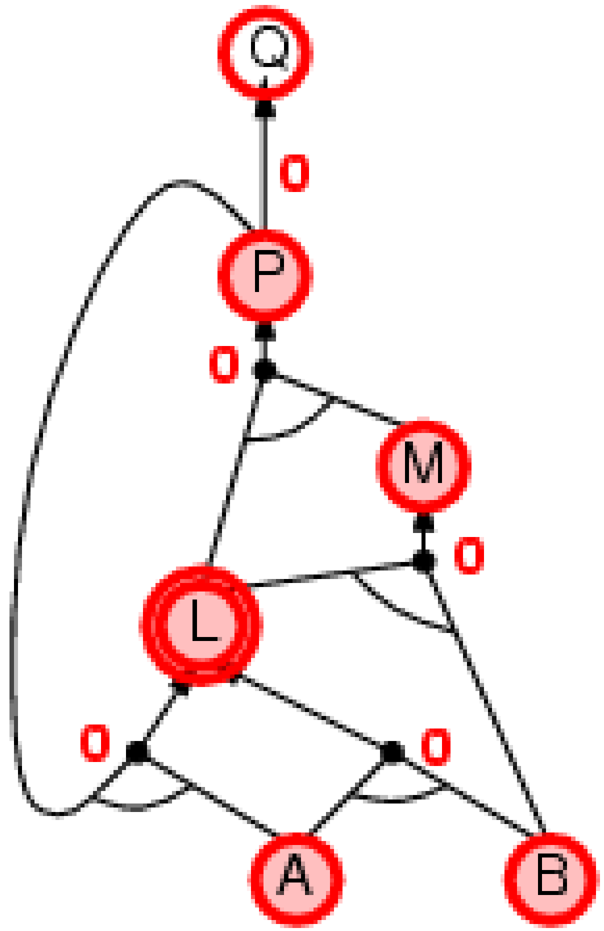
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

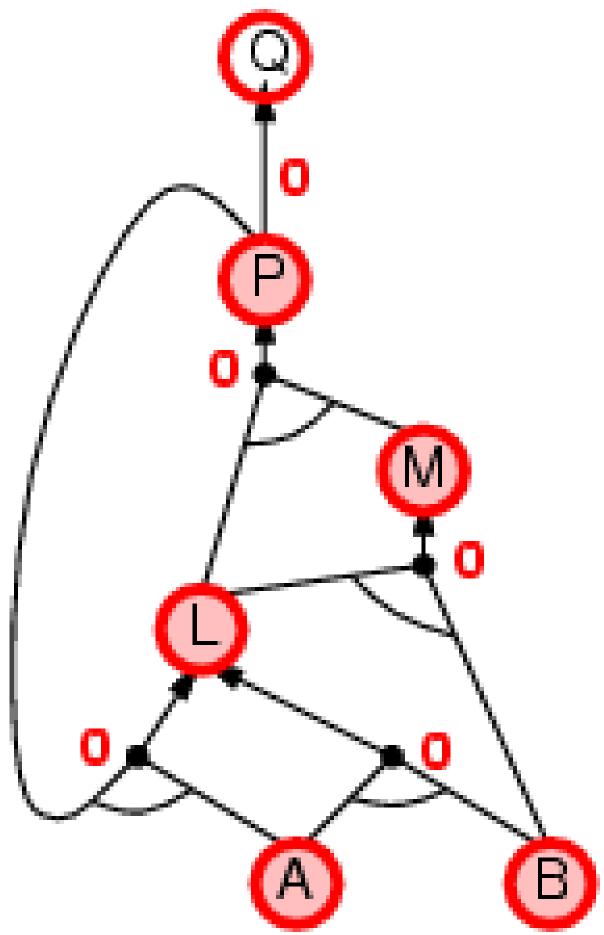
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

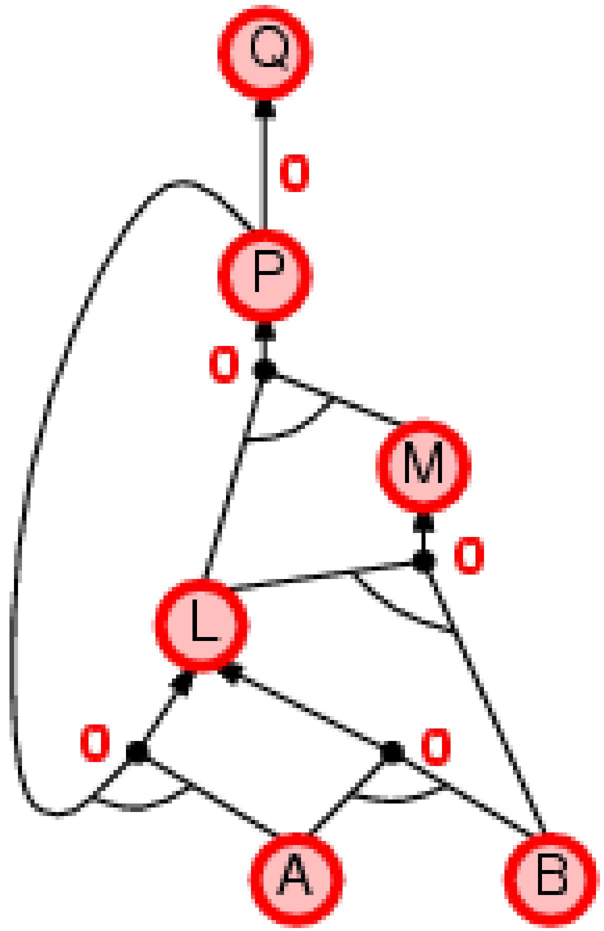
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

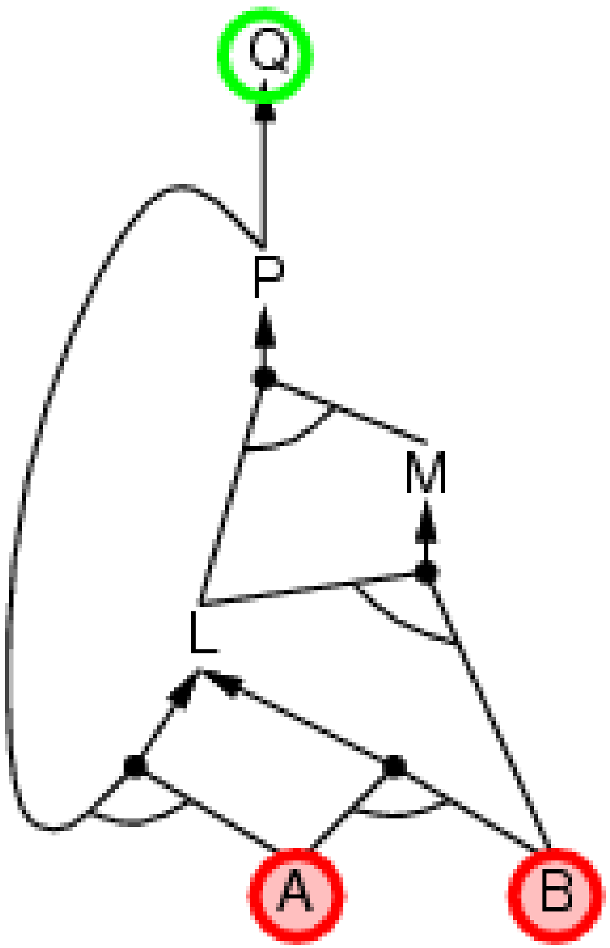
A

B

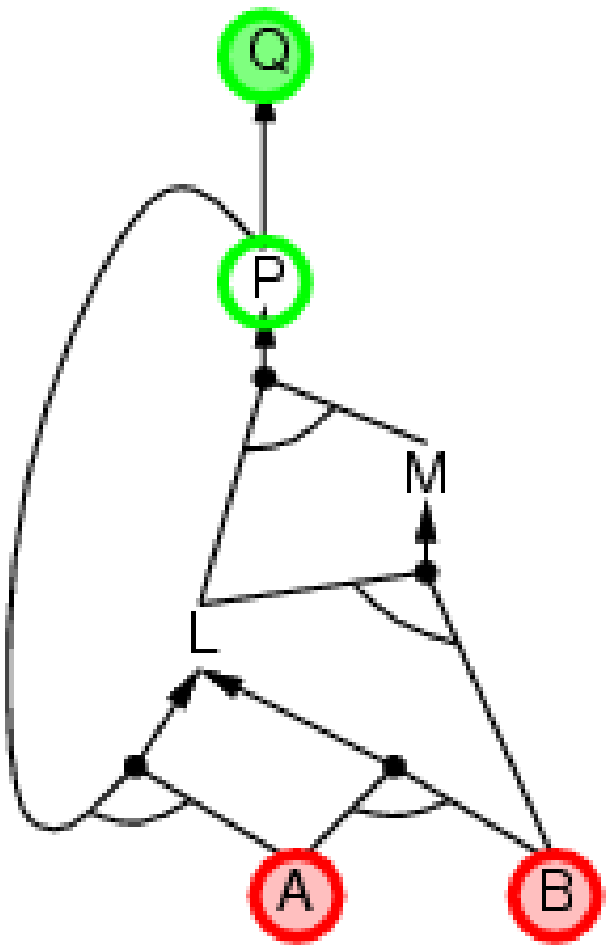


- Work backwards from the query Q.
- Find a clause whose conclusion is Q:
  - Prove each of the premises using backward chaining.
- Often requires less work than forward chaining
  - Using only the parts of the KB that can be relevant to the goal.

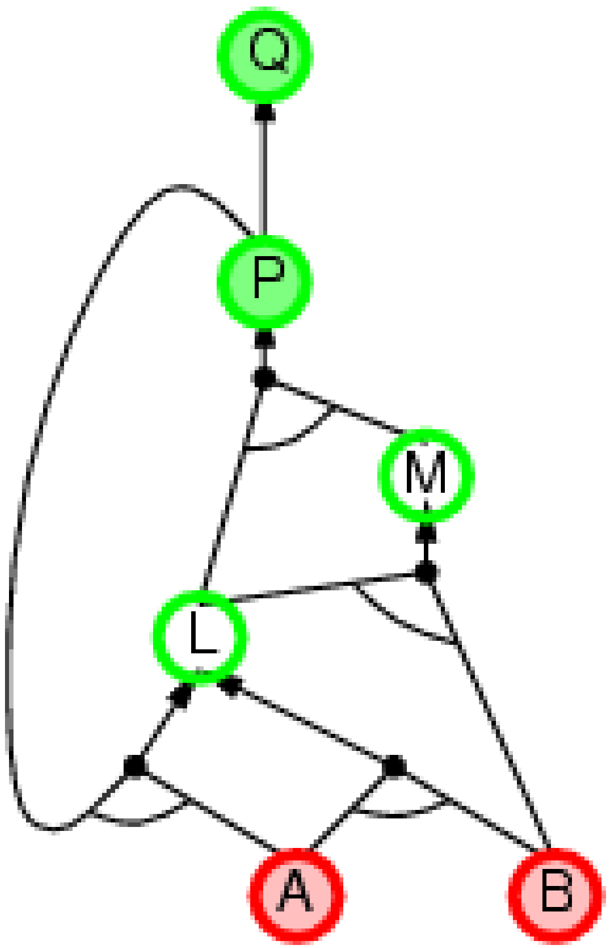
Backward chaining example



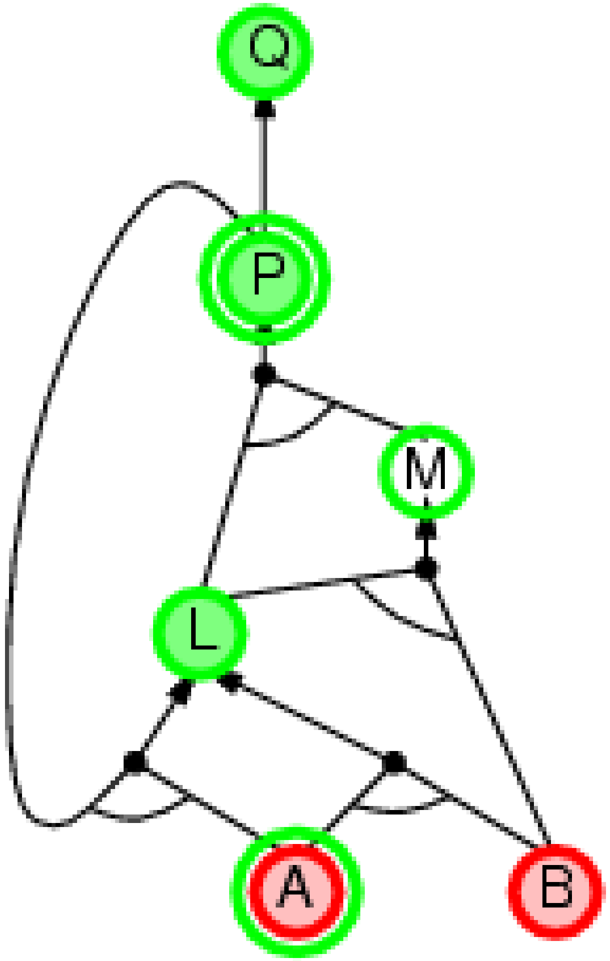
Backward chaining example



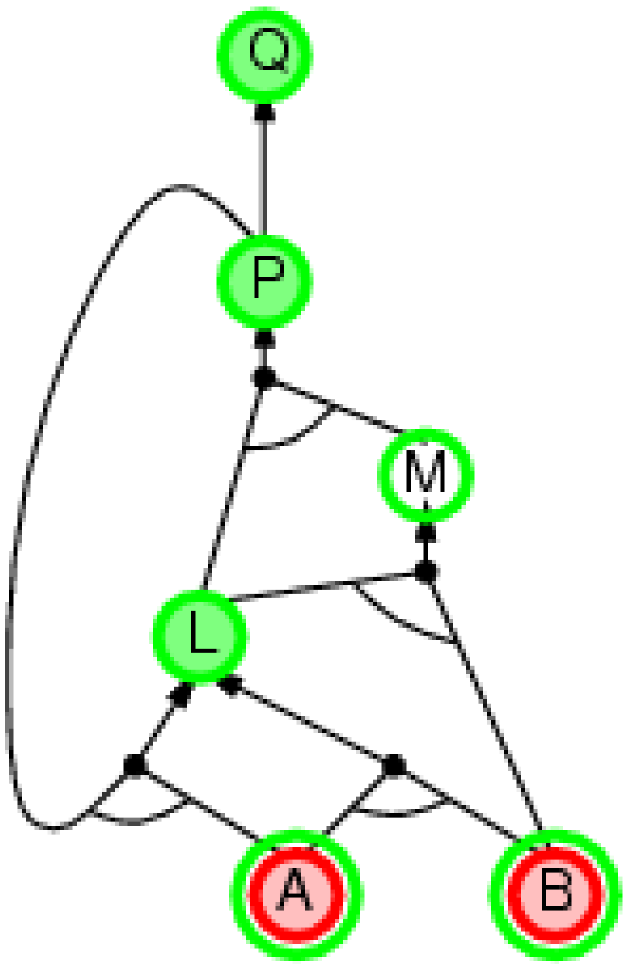
Backward chaining example



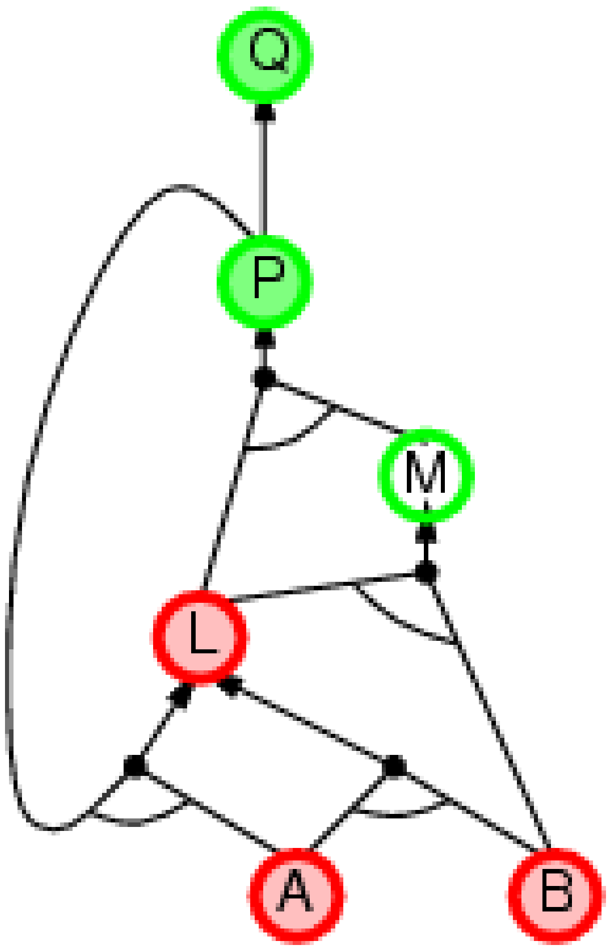
Backward chaining example



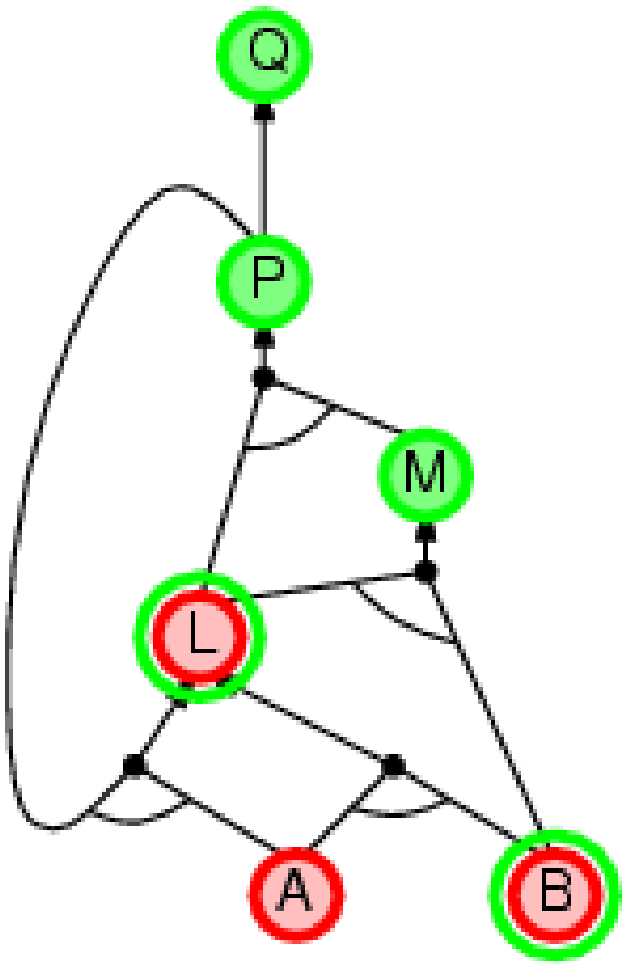
Backward chaining example



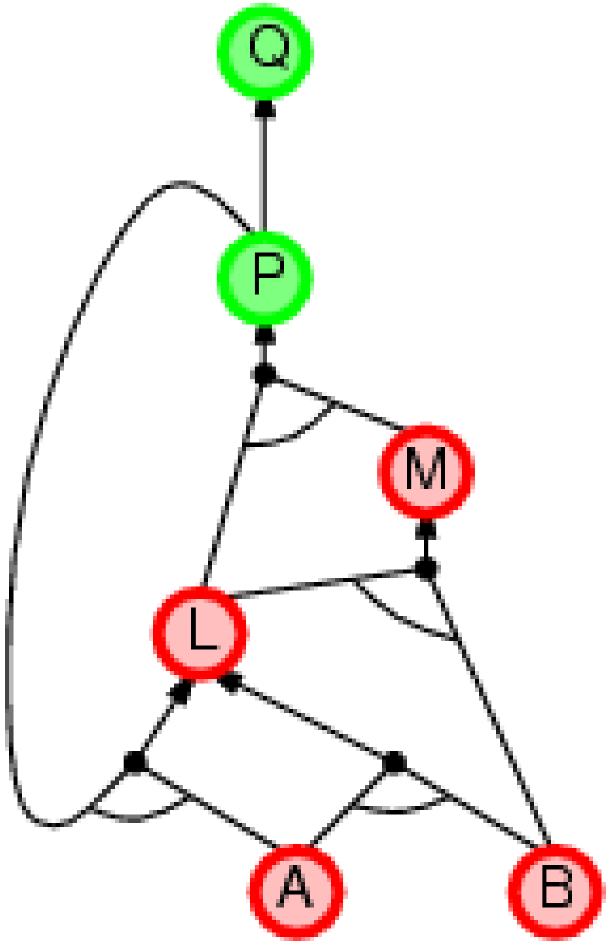
Backward chaining example



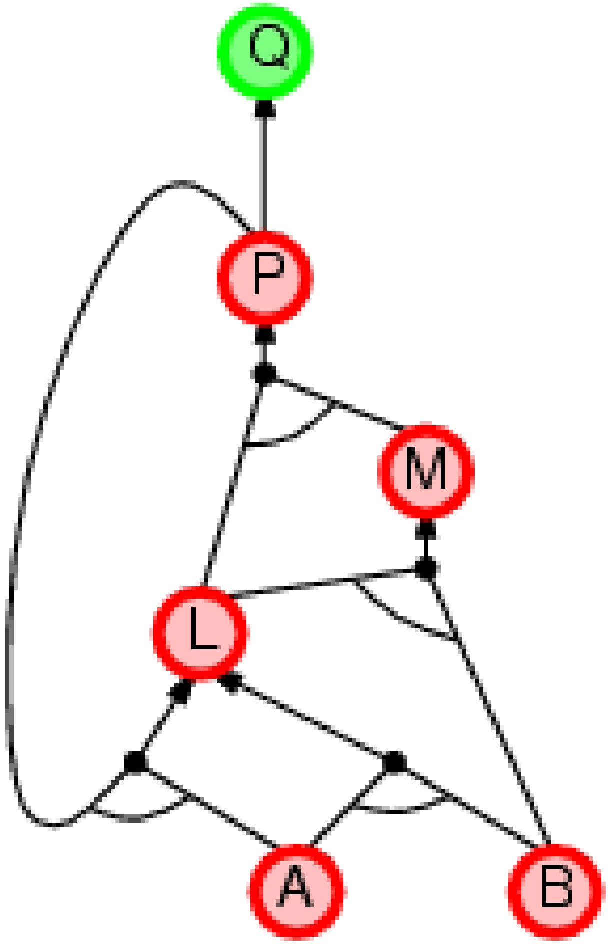
Backward chaining example



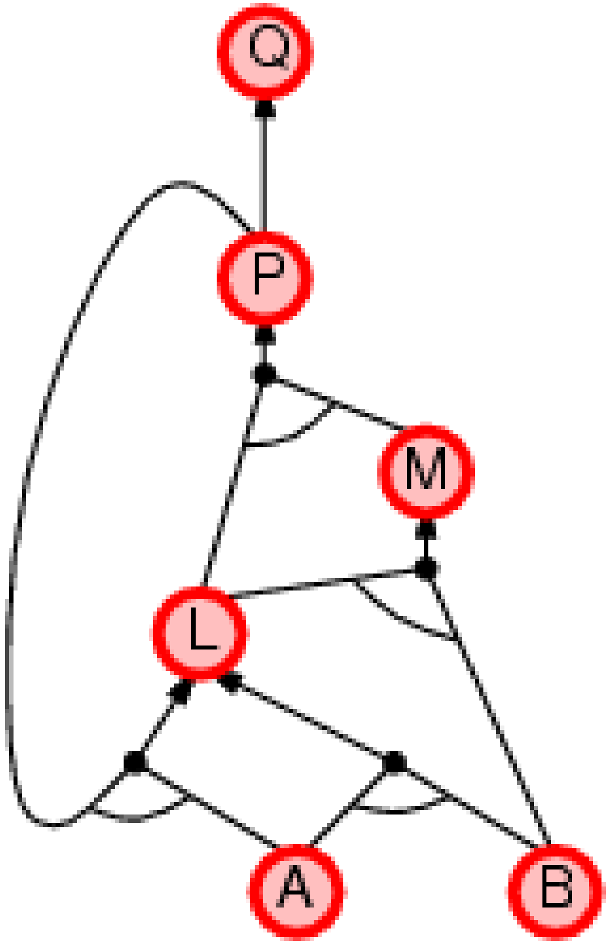
Backward chaining example



Backward chaining example



Backward chaining example



Satisfiability

- Very important class of problems
  - NP Complete
- Given a sentence in propositional logic, is there any model that satisfies the sentence?
  - Is there any model for which the sentence is true?
- Sounds easy enough...
  - The problem is how long it takes to check all possible models.

Satisfiability Search

- David-Putnam (Logemann and Loveland)
- Start with CNF clauses.
- Depth first enumeration of all possible models.
  - In general this would take O(2n) time.
  - David Putnam has some improvements that can shorten the search time significantly.

Davis-Putnam Improvements

- Early Termination – recognizes when a clause will be true based on partial assignment.
  - If we have assigned A=TRUE, then (A ∨ **B**∨ **C**) is true regardless of what we assign to B and C
  - For all models consistent with the current partial assignment, we can throw away the clause (A ∨ **B**∨ **C**)
    - Search tree below this point is dealing with a smaller problem.

Davis-Putnam Improvements (cont.)

- Heuristic: pure symbol
  - any symbol that appears with the same "sign" everywhere can be assigned a value without searching.

$(A \lor Q \lor \neg T) \land (\neg P \lor \mathbf{Q} \lor \mathbf{T}) \land (\neg \mathbf{P} \lor \mathbf{X} \lor \mathbf{Y} \lor \mathbf{Z})$

P and Q are pure.

Setting P to false can't hurt.

Setting Q to true can't hurt.

Davis-Putnam Improvements (cont.)

- Unit clause heuristics
  - Any clause containing a single literal defines what value that proposition should have.
  - This can propagate...
    - We propagate assignments to unit clauses as far as possible before branching in the search tree.
    - Example: $(\neg X) \wedge$ **$(\neg X \vee T) \wedge (Y \vee Z \vee Q \vee S)$ will become X=false, T=true , $(Y \vee Z \vee Q \vee S)$**

3SAT

CNF

Each clause has 3 literals.

(P ∨ Q∨ **R**) ∧ **(**¬P∨ **S**∨ **T)** ∧ **...** ∧ **(**¬**X**∨ **S**∨ ¬**Z)**

Find an assignment of values to all propositions P,Q,S,T,X,... such that the entire sentence is true.

If we have n propositions, there are 2^n possible assignments...

Wumpus KB

$$\neg P_{1,1} \; \neg W_{1,1}$$

- **For every square x,y we have rules:**

$$B_{x,y} \Leftrightarrow (P_{x+1,y} \vee P_{x-1,y} \vee P_{x,y+1} \vee P_{x,y-1})$$

$$S_{x,y} \Leftrightarrow (W_{x+1,y} \vee \mathbf{W}_{x-1,y} \vee W_{x,y+1} \vee W_{x,y-1})$$

- There is a wumpus somewhere:

$$(W_{1,1} \vee \mathbf{W}_{1,2} \vee \ldots \vee W_{4,4})$$

- There is only one wumpus:

$(\neg W_{1,1} \vee \neg \mathbf{W}_{1,2})$, $(\neg W_{1,1} \vee \neg \mathbf{W}_{1,3})$, $(\neg W_{1,1} \vee \neg \mathbf{W}_{1,4})$, ...

- We end up with 155 sentences using 64 propositions.