# Artificial Intelligence

Hai Thi Tuyet Nguyen

# Outline

# CHAPTER 5: LOGICAL AGENT

# 5.1 Knowledge-Based Agents

- ***Knowledge base (KB)*** = a set of sentences in a ***formal*** language (i.e., knowledge representation language)
- ***Declarative*** approach to build an agent:
    - *TELL* it what it needs to know
    - *ASK* itself what to do - answer should follow from the KB

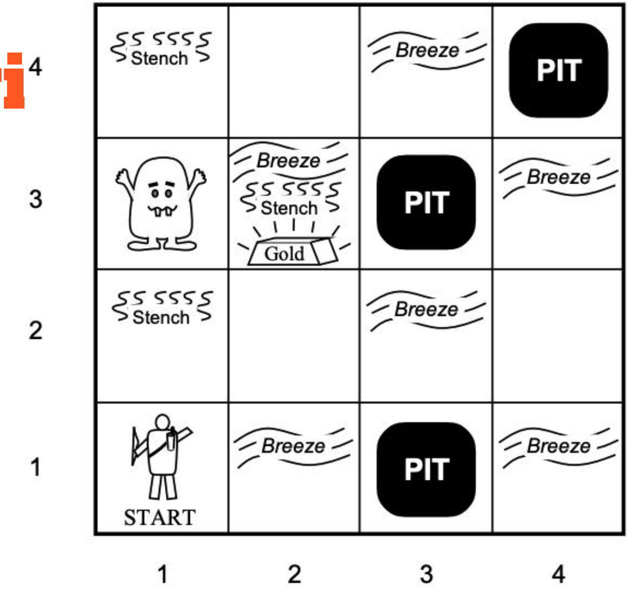
# 5.1 A simple knowledge-based agent.

- The agent takes a percept as input and returns an action.
  It maintains a knowledge base
- How it works:
  - TELLs the knowledge base what it perceives.
  - ASKs the knowledge base what action it should perform.
  - TELLs the knowledge base which action was chosen, and executes the action.

```
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time

    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

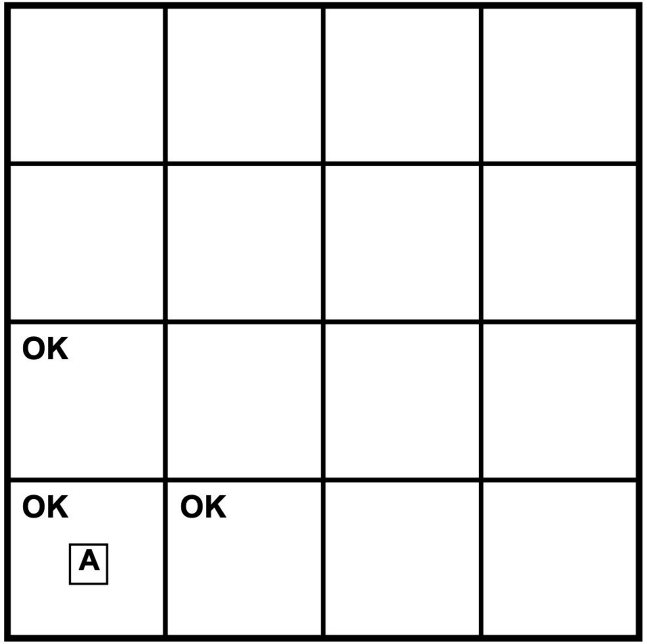
# 5.2 Wumpus World PEAS descri

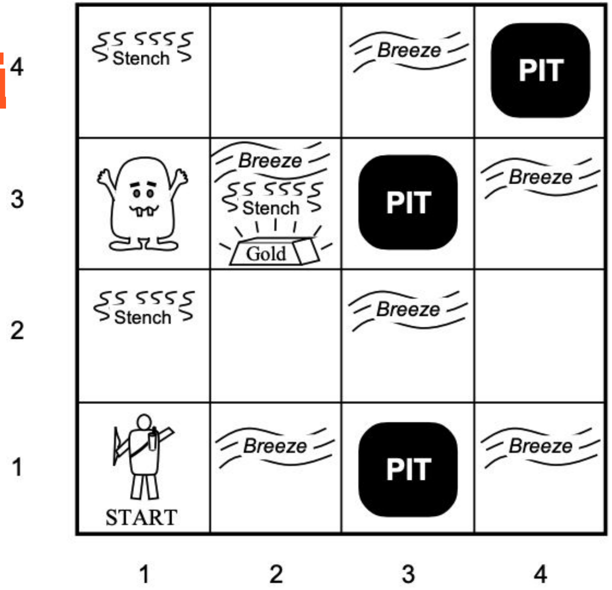



- **Performance measure:**
  - +1000: climb out of the cave with the gold,
  - –1000: fall into a pit or being eaten by the wumpus
  - –1: each action
  - –10: use up the arrow.
  - The game ends: the agent dies or it climbs out of the cave
- **Environment:** A 4 × 4 grid of rooms.
  - Start location of the agent: the square labeled [1,1]
  - Locations of the gold and the wumpus: random
- **Actuators**: Move Forward, Turn Left, Turn Right, Grab, Climb, Shoot
- **Sensors**: the agent will perceive
  - Stench: in the square containing the monster (called wumpus) and in the directly adjacent squares
  - Breeze: in the squares directly adjacent to a pit
  - Glitter: in the square where the gold is
  - Bump: into a wall
  - Scream: anywhere in the cave when the wumpus is killed

# 5.2 Wumpus World PEAS descri

- The first percept is [None,None,None,None,None] ~ [stench,breeze,glitter,bump,scream]
  => its neighboring squares, [1,2] and [2,1], are OK.



# 5.2 Wumpus World PEAS description

The agent decides to move forward to [2,1].

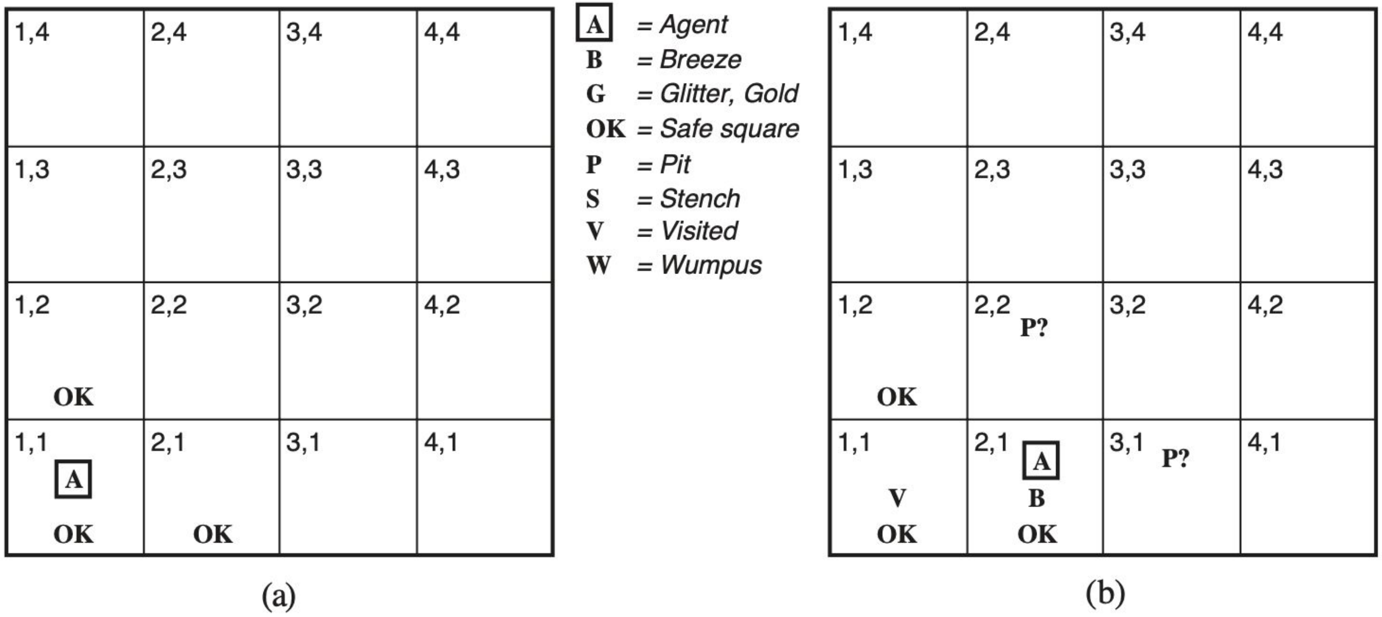



**Figure 7.3** The first step taken by the agent in the wumpus world. (a) The initial situation, after percept [*None*, *None*, *None*, *None*, *None*]. (b) After one move, with percept [*None*, *Breeze*, *None*, *None*, *None*].

# 5.2 Wumpus World PEAS description

- The agent perceives a breeze (denoted by “B”) in [2,1]
  [None,**breeze**,None,None,None]
  => there must be a pit in a neighboring square.
- The pit cannot be in [1,1] => so there must be a pit in [2,2] or [3,1] or both.
- The agent will turn around,
  go back to [1,1],
  and then proceed to [1,2].

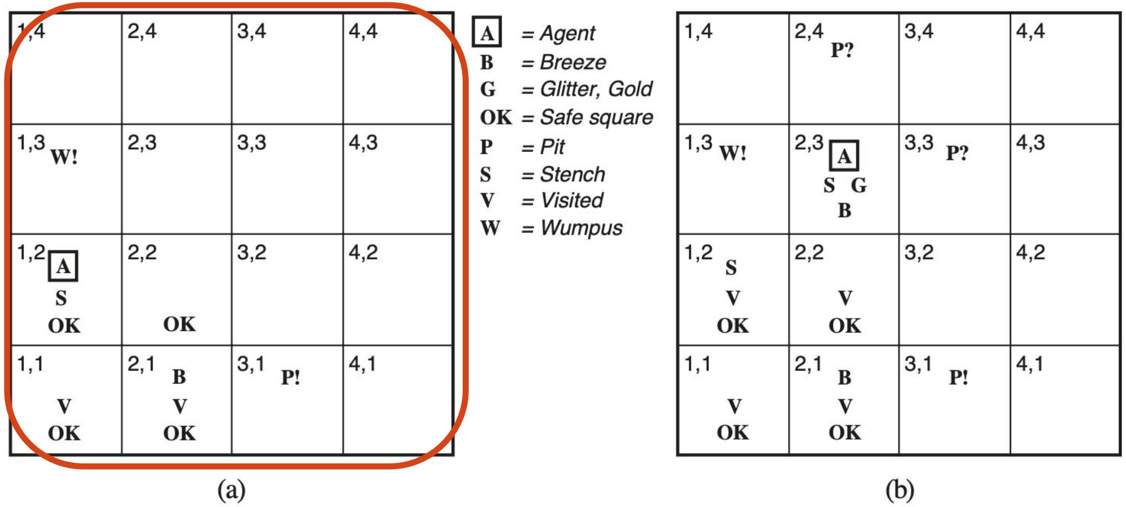



**Figure 7.4** Two later stages in the progress of the agent. (a) After the third move, with percept [*Stench*, *None*, *None*, *None*, *None*]. (b) After the fifth move, with percept [*Stench*, *Breeze*, *Glitter*, *None*, *None*].

# 5.2 Wumpus World PEAS description

- The agent perceives a stench in [1,2] ~ [stench,None,None,None,None]
  => there must be a wumpus nearby ([2,2] or [1,3])
- The lack of stench when the agent was in [2,1]
  => wumpus cannot be in [2,2]
  => wumpus is in [1,3]
- The lack of a breeze in [1,2]
  => there is no pit in [2,2]
  => [2,2]: safe, OK

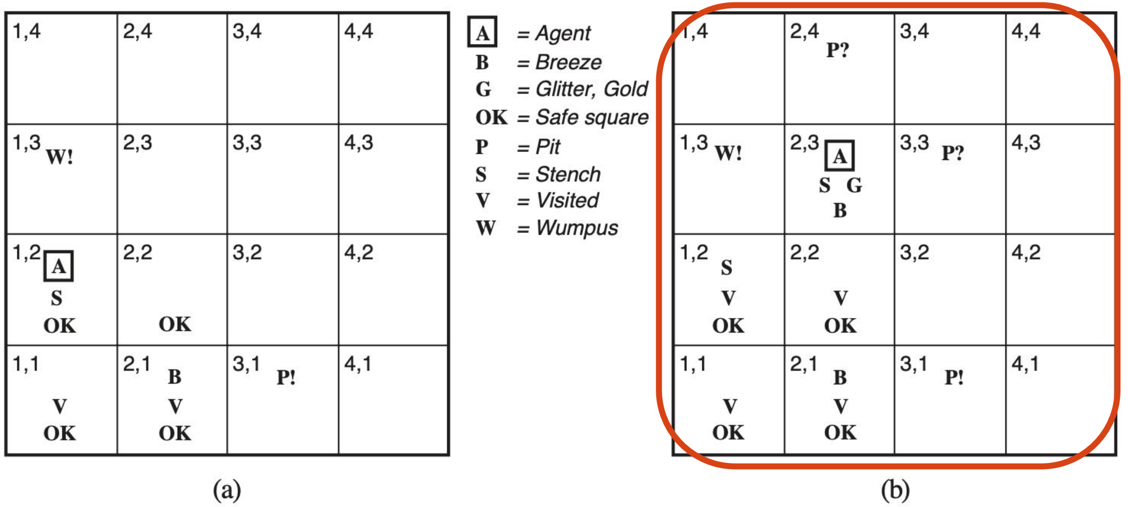



**Figure 7.4** Two later stages in the progress of the agent. (a) After the third move, with percept [*Stench*, *None*, *None*, *None*, *None*]. (b) After the fifth move, with percept [*Stench*, *Breeze*, *Glitter*, *None*, *None*].

# 5.2 Wumpus World PEAS description

- The agent draws a conclusion from the available information
- The conclusion is guaranteed to be correct if the available information is correct.

# 5.3 Logic

- ***Logics*** are formal languages for representing information
- ***Syntax*** defines the sentences in the language
  E.g., "x + y = 4" is a well-formed sentence, whereas "x4y+ =" is not
- ***Semantics*** defines the "meaning" of sentences OR the ***truth*** of each sentence with respect to each ***possible world*** (i.e., ***model***).
  E.g., the sentence "x + y = 4" is true in a world where $x$ is 2 and $y$ is 2,
  but false in a world where $x$ is 1 and $y$ is 1

# 5.3 Entailment

- Entailment means that ***one thing follows from another***:
- Knowledge base *KB* **entails** sentence $\alpha$ if and only if $\alpha$ is true in all possible worlds (models) where *KB* is true

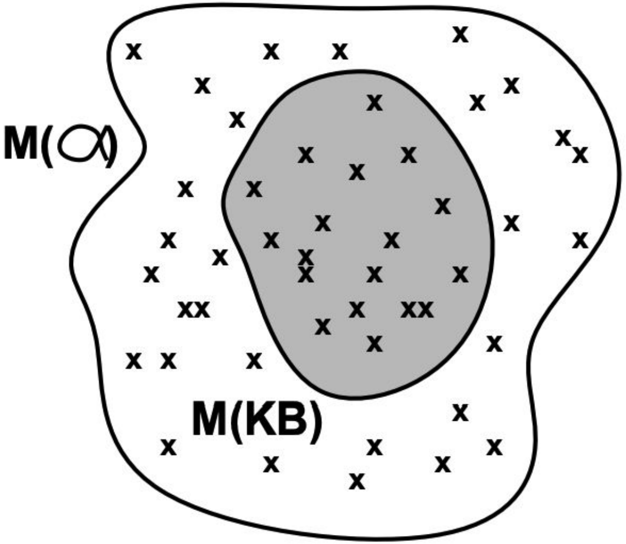
  $KB \models \alpha$
- E.g., KB containing "the Giants won" and "the Reds won" entails "Either the Giants won or the Reds won"

# 5.3 Models

- Logicians typically think in terms of **models**, which are formally structured worlds with respect to which truth can be evaluated
- We say $m$ is a **model** of a sentence $\alpha$ if $\alpha$ is true in $m$
- $M(\alpha)$ is the set of all models of $\alpha$
- Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$

E.g. $KB$ = Giants won and Reds won

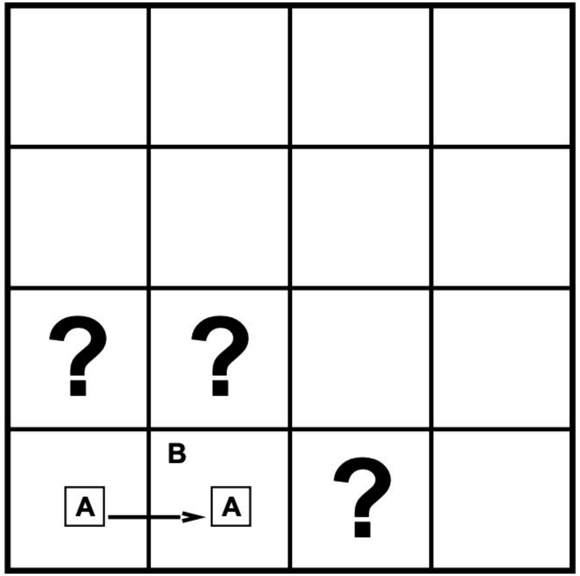
$\alpha$ = Giants won



# 5.3 Entailment in the wumpus world

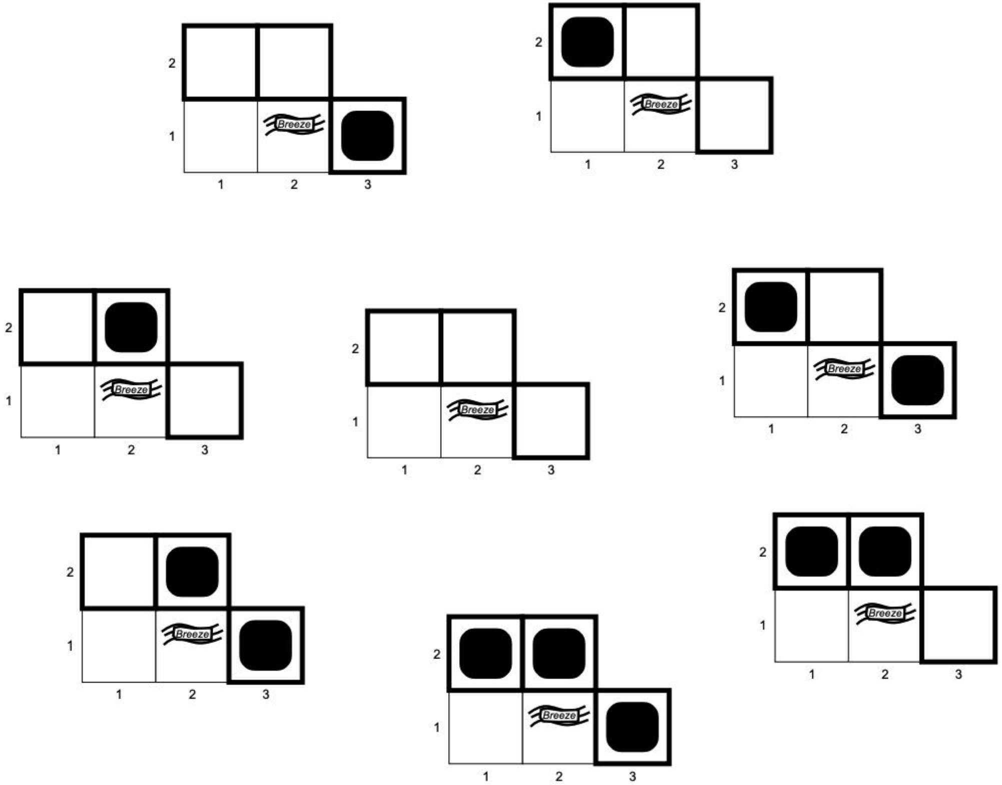
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

Consider possible models for ?s assuming only pits



# Wumpus models

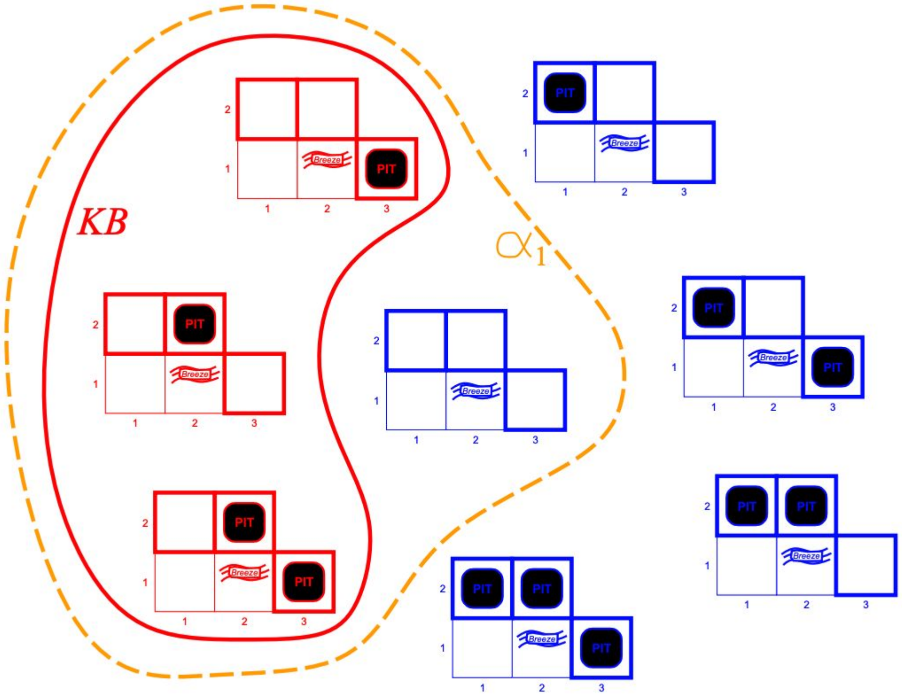
8 possible models



# Wumpus models

KB = wumpus-world rules + observations

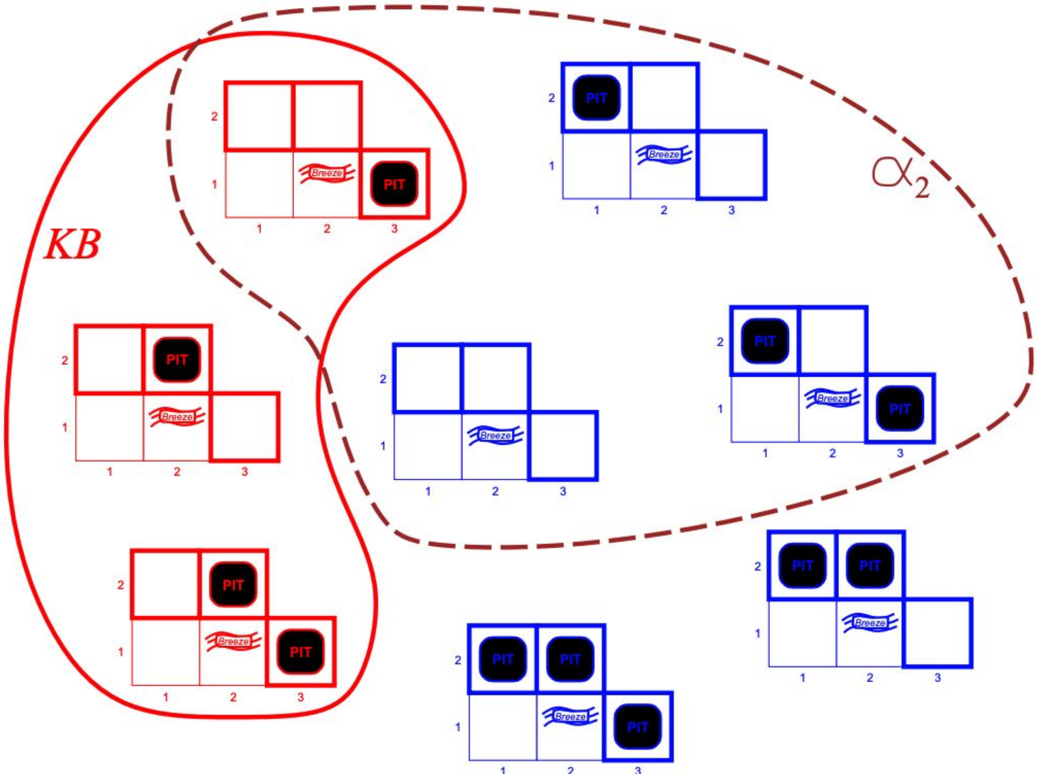
α1 = “[1,2] is safe”, KB |= α1



# Wumpus models

KB = wumpus-world rules + observations

α2 = “[2,2] is safe”, KB |= α2



# 5.3 Inference

- $KB \vdash_i \alpha$ = sentence α can be derived from KB by algorithm i
- Consequences of KB are a haystack; $\alpha$ is a needle.
  Entailment = needle in haystack; inference = finding it
- Soundness:
  i is sound if whenever $KB \vdash_i \alpha$, it is also true that KB |= α
- Completeness:
  i is complete if whenever KB |= α, it is also true that $KB \vdash_i \alpha$

# 5.4 Propositional logic: Syntax

- Propositional logic is the simplest logic - illustrates basic ideas
- The proposition symbols $P_1$, $P_2$, … are sentences
- Logical connectives
    - $\neg$ (not).
    - $\wedge$ (and).
    - $\vee$ (or).
    - $\Rightarrow$ (implies)
    - $\Leftrightarrow$ (if and only if)
- Operator precedence: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

# 5.4 Propositional logic: Syntax

- If $S$ is a sentence, $\neg S$ is a sentence (negation)
- If $S_1$ and $S_2$ are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)
- If $S_1$ and $S_2$ are sentences, $S_1 \vee S_2$ is a sentence (disjunction)
- If $S_1$ and $S_2$ are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)
- If $S_1$ and $S_2$ are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

# 5.4 Propositional logic: Semantics

- Each model specifies true/false for each proposition symbol

  E.g. $P_{1,2}$ $P_{2,2}$ $P_{3,1}$

  true true false

- Rules for evaluating truth with respect to a model m:

| | | | | | |
|---|---|---|---|---|---|
| $\neg S$ | is true iff | $S$ | is false | | |
| $S_1 \wedge S_2$ | is true iff | $S_1$ | is true **and** | $S_2$ | is true |
| $S_1 \vee S_2$ | is true iff | $S_1$ | is true **or** | $S_2$ | is true |
| $S_1 \Rightarrow S_2$ | is true iff | $S_1$ | is false **or** | $S_2$ | is true |
| i.e., | is false iff | $S_1$ | is true **and** | $S_2$ | is false |
| $S_1 \Leftrightarrow S_2$ | is true iff | $S_1 \Rightarrow S_2$ | is true **and** | $S_2 \Rightarrow S_1$ | is true |

# 5.4 Truth tables for connectives

| $P$ | $Q$ | $\neg P$ | $P \wedge Q$ | $P \vee Q$ | $P \Rightarrow Q$ | $P \Leftrightarrow Q$ |
|---|---|---|---|---|---|---|
| *false* | *false* | *true* | *false* | *false* | *true* | *true* |
| *false* | *true* | *true* | *false* | *true* | *true* | *false* |
| *true* | *false* | *false* | *false* | *true* | *false* | *false* |
| *true* | *true* | *false* | *true* | *true* | *true* | *true* |

# 5.4 A simple knowledge base - Wumpus world sentences

$P_{x,y}$ is true if there is a pit in [x, y].

$B_{x,y}$ is true if the agent perceives a breeze in [x, y].

There is no pit in [1,1]: $R_1 : \neg P_{1,1}$

A square is breezy if and only if there is a pit in a neighboring

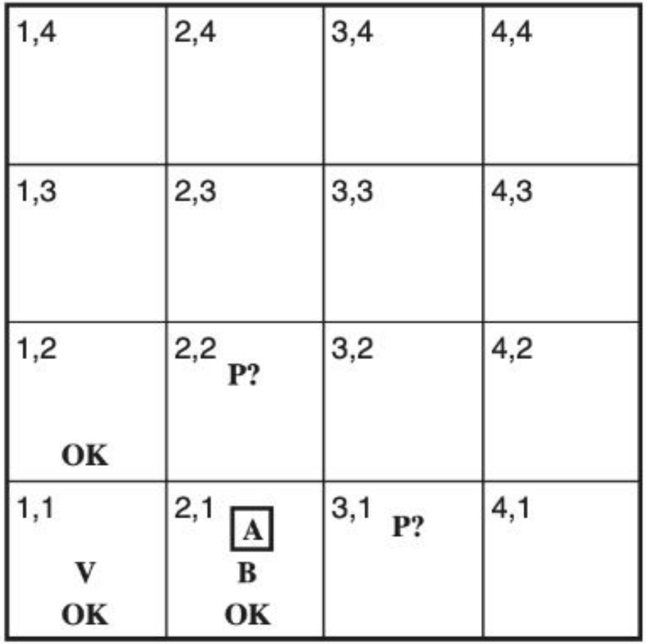
$R2: B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$

$R3 : B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

The breeze percepts on the first two squares of the agent

$R4 : \neg B_{1,1}$

$R5 : B_{2,1}$



# 5.4 A simple knowledge base - Wumpus world sentences

- Goal: to decide whether KB $\models \alpha$ for some sentence $\alpha$

  KB

  $\alpha$: $\neg P_{1,2}$

  prove: KB $\models \neg P_{1,2}$

- A simple inference procedure

  A model-checking approach:
  - enumerate the models
  - check that $\alpha$ is true in every model in which KB is true

# 5.4 A simple knowledge base - Wumpus world sentences

With 7 symbols, there are 2^7 = 128 possible models; in 3 of these, KB is true.

In those 3 models, $\neg P_{1,2}$ is true or there is no pit in [1,2].

| $B_{1,1}$ | $B_{2,1}$ | $P_{1,1}$ | $P_{1,2}$ | $P_{2,1}$ | $P_{2,2}$ | $P_{3,1}$ | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R_5$ | $KB$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *false* | *false* | *false* | *false* | *false* | *false* | *false* | *true* | *true* | *true* | *true* | *false* | *false* |
| *false* | *false* | *false* | *false* | *false* | *false* | *true* | *true* | *true* | *false* | *true* | *false* | *false* |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| *false* | *true* | *false* | *false* | *false* | *false* | *false* | *true* | *true* | *false* | *true* | *true* | *false* |
| *false* | *true* | *false* | *false* | *false* | *false* | *true* | *true* | *true* | *true* | *true* | *true* | *true* |
| *false* | *true* | *false* | *false* | *false* | *true* | *false* | *true* | *true* | *true* | *true* | *true* | *true* |
| *false* | *true* | *false* | *false* | *false* | *true* | *true* | *true* | *true* | *true* | *true* | *true* | *true* |
| *false* | *true* | *false* | *false* | *true* | *false* | *false* | *true* | *false* | *false* | *true* | *true* | *false* |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| *true* | *true* | *true* | *true* | *true* | *true* | *true* | *false* | *true* | *true* | *false* | *true* | *false* |

# 5.5 Propositional Theorem Proving

- Determine entailment by **theorem proving**:
  applying rules of inference directly to the sentences in our knowledge base to construct a proof of the desired sentence without consulting models

- Some additional concepts related to entailment:
  - Logical equivalence
  - Validity
  - Satisfiability

# 5.5 Logical equivalence

- Two sentences α and β are logically equivalent if they are true in the same set of models
- Two sentences α and β are equivalent only if each of them entails the other: $\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) \quad \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) \quad \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

**Figure 7.11** Standard logical equivalences. The symbols $\alpha$, $\beta$, and $\gamma$ stand for arbitrary sentences of propositional logic.

# 5.5 Validity and satisfiability

- A sentence is valid if it is true in all models, e.g., True, $A \lor \neg A$, $A \Rightarrow A$
  - Valid sentences are also known as tautologies
  - **Validity** is connected to **inference**:
    $KB \models \alpha$ if and only if ($KB \Rightarrow \alpha$) is valid

- A sentence is satisfiable if it is true in some models, e.g., $A \lor B$, $C$
  - A sentence is **unsatisfiable** if it is true in no models
    E.g., $A \land \neg A$
  - **Satisfiability** is connected to **inference**:
    $KB \models \alpha$ if and only if ($KB \land \neg\alpha$) is unsatisfiable

# 5.5 Inference rules and proofs

**Modus Ponens** $\dfrac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$

**And-Elimination** $\dfrac{\alpha \wedge \beta}{\alpha}$

E.g. Wumpus world

$R1 : \neg P_{1,1}$

$R2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

$R3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

$R4 : \neg B_{1,1}$

$R5 : B_{2,1}$

prove $\neg P_{1,2}$

Apply **biconditional elimination** to R2 to obtain

$R6: (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$

Apply **And-Elimination** to R6 to obtain

$R7 : ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

Apply **contraposition** to R7 to obtain

$R8 : (\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1}))$

Apply **Modus Ponens** with R8 and R4 to obtain

$R9: \neg(P_{1,2} \vee P_{2,1})$

Apply **De Morgan**'s rule, giving the conclusion

$R10 : \neg P_{1,2} \wedge \neg P_{2,1}$

# 5.5 Proof by resolution

- **Conjunctive Normal Form** (CNF): conjunction of clauses, clauses are disjunctions of literals

  E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

- **Resolution** inference rule (for CNF): complete for propositional logic

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

where $l_i$ and $m_j$ are complementary literals (i.e., one is the negation of the other).

# 5.5 Conversion to CNF

Convert R2 : $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$ into CNF

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

   $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$

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

   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1})$

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

   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1})$

4. Apply distributivity law ($\lor$ over $\land$):

   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$

# 5.5 Resolution algorithm

Proof by contradiction, i.e., show KB ∧ ¬α unsatisfiable

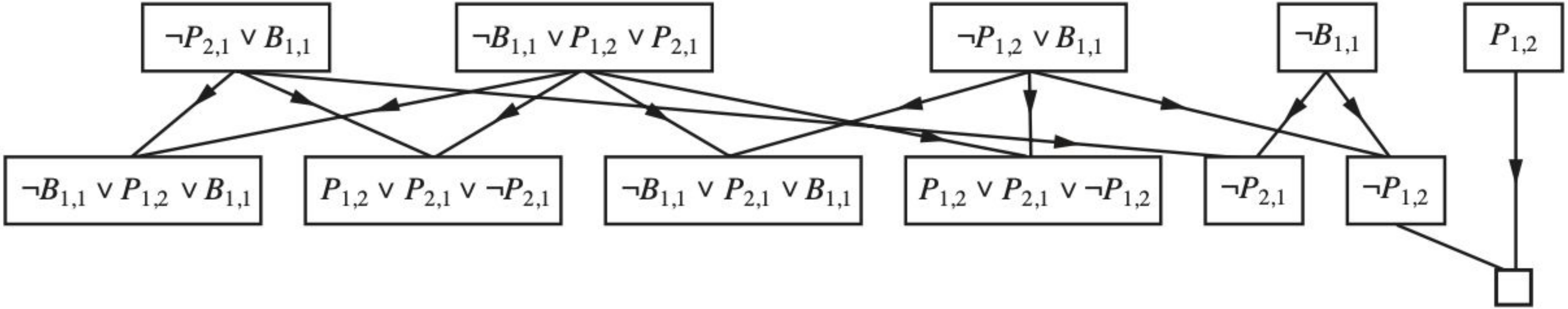
```
function PL-RESOLUTION(KB, α) returns true or false
    inputs: KB, the knowledge base, a sentence in propositional logic
            α, the query, a sentence in propositional logic

    clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
    new ← { }
    loop do
        for each Ci, Cj in clauses do
            resolvents ← PL-RESOLVE(Ci, Cj)
            if resolvents contains the empty clause then return true
            new ← new ∪ resolvents
        if new ⊆ clauses then return false
        clauses ← clauses ∪ new
```

# 5.5 Proof by resolution

$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$

$\alpha = \neg P_{1,2}$



# 5.5 Proof by resolution

E.g. Wumpus world

$R_1 : \neg P_{1,1}$.

$R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$.

$R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$.

$R_4 : \neg B_{1,1}$.

$R_5 : B_{2,1}$.

Prove $\neg P_{1,2}$ by resolution

# 5.5 Proof by resolution

$KB = R_2 \land R_4 = (B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$

$\alpha$ as $\neg P_{1,2}$

1. convert $(KB \land \neg\alpha)$ to CNF

$(\neg P_{1,2} \lor B_{1,1}) \land (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1})$
$\land (\neg P_{2,1} \lor B_{1,1}) \land \neg B_{1,1} \land \neg P_{1,2}$

2. resolve pairs

$(\neg P_{1,2} \lor B_{1,1}), (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1})$: $P_{2,1}$

$(\neg P_{1,2} \lor B_{1,1}), \neg B_{1,1}$: $\neg P_{1,2}$

$(\neg P_{2,1} \lor B_{1,1}), (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1})$: $P_{1,2}$

$(\neg P_{2,1} \lor B_{1,1}), \neg B_{1,1}$: $P_{2,1}$

3. resolve pairs

$\neg P_{1,2}, P_{1,2}$: empty

Result: $KB \models \neg P_{1,2}$

# 5.5 Horn clauses and definite clauses

- **Definite clause**: a disjunction of literals of which **exactly one** is positive. E.g., $(\neg L_{1,1} \vee \neg B \vee B_{1,1})$ is a definite clause
- **Horn clause**: a disjunction of literals of which **at most one** is positive
- **Goal clauses**: clauses with **no positive** literals

$$
\begin{aligned}
CNFSentence &\rightarrow Clause_1 \wedge \cdots \wedge Clause_n \\
Clause &\rightarrow Literal_1 \vee \cdots \vee Literal_m \\
Literal &\rightarrow Symbol \mid \neg Symbol \\
Symbol &\rightarrow P \mid Q \mid R \mid \ldots \\
HornClauseForm &\rightarrow DefiniteClauseForm \mid GoalClauseForm \\
DefiniteClauseForm &\rightarrow (Symbol_1 \wedge \cdots \wedge Symbol_l) \Rightarrow Symbol \\
GoalClauseForm &\rightarrow (Symbol_1 \wedge \cdots \wedge Symbol_l) \Rightarrow False
\end{aligned}
$$

Figure 7.14 A grammar for conjunctive normal form, Horn clauses, and definite clauses.

# 5.5 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

$$\frac{\alpha_1, \ldots, \alpha_n, \qquad \alpha_1 \wedge \cdots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

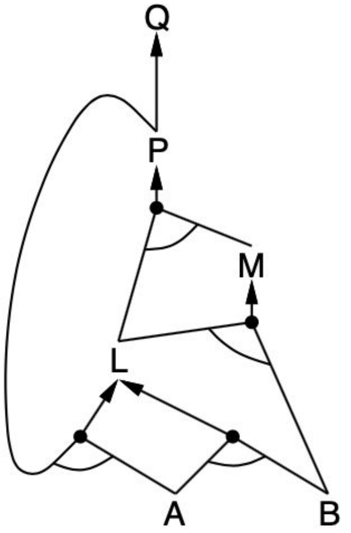
Can be used with forward chaining or backward chaining.

These algorithms are very natural and run in linear time

# 5.5 AND–OR graphs

- In AND–OR graphs,
  - multiple links joined by an arc indicate a conjunction
  - multiple links without an arc indicate a disjunction
- How the graphs work:
  - The known leaves are set, inference propagates up the graph as far as possible.
  - Where a conjunction appears, the propagation waits until all the conjuncts are known before proceeding.

$$
\begin{aligned}
P &\Rightarrow Q \\
L \wedge M &\Rightarrow P \\
B \wedge L &\Rightarrow M \\
A \wedge P &\Rightarrow L \\
A \wedge B &\Rightarrow L \\
A & \\
B &
\end{aligned}
$$



# 5.5 Forward chaining

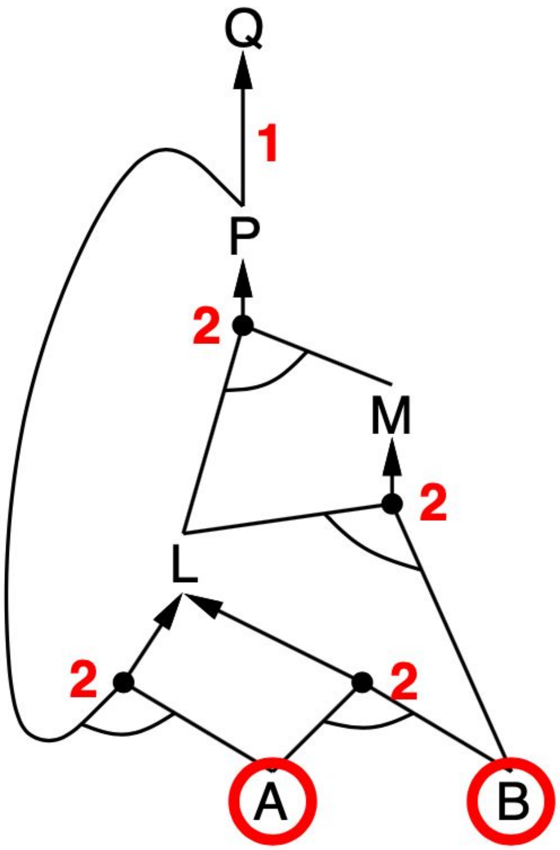
**Idea**: fire any rule whose premises are satisfied in the KB,
add its conclusion to the KB, until query is found or no further inferences can be made.

```
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
```

# 5.5 Forward chaining

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



# 5.5 Forward chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

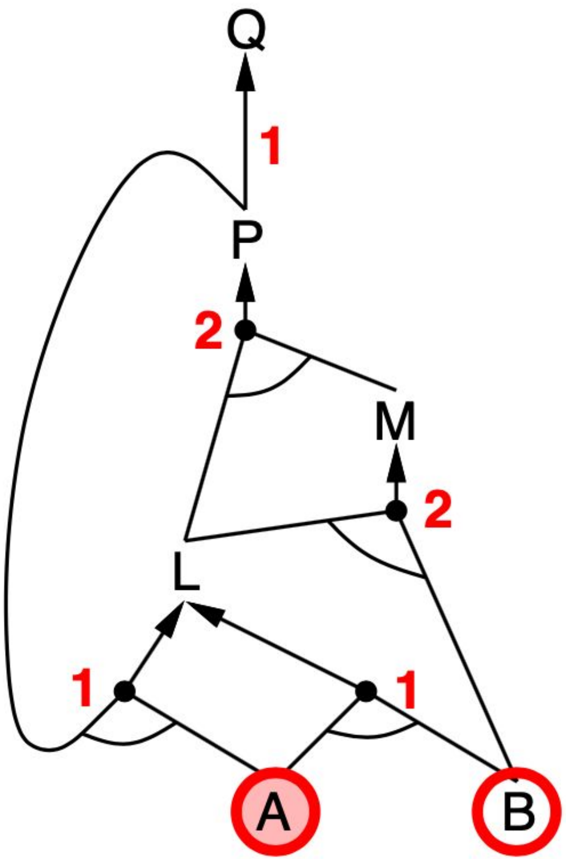
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

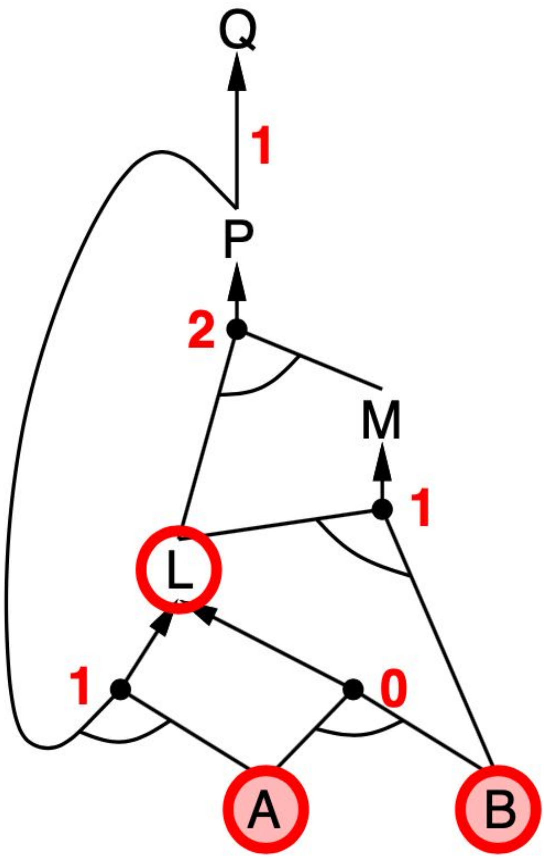
$A$

$B$



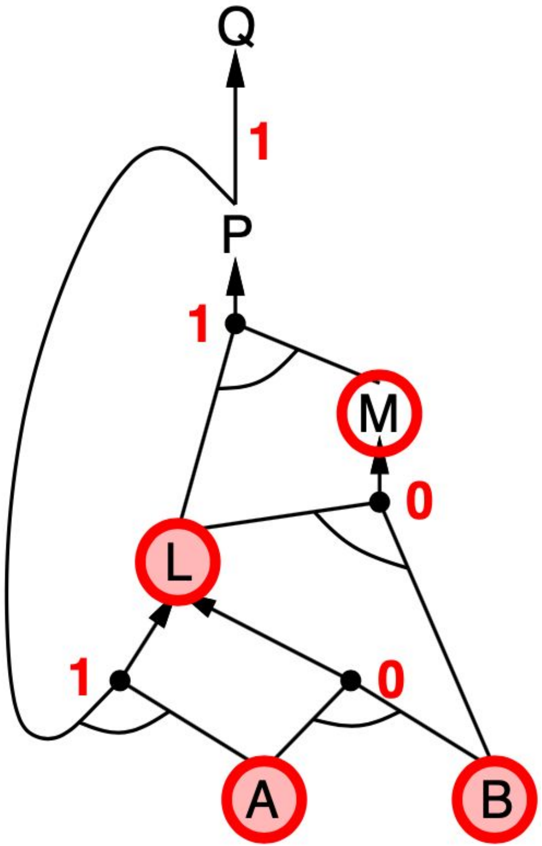
# 5.5 Forward chaining

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



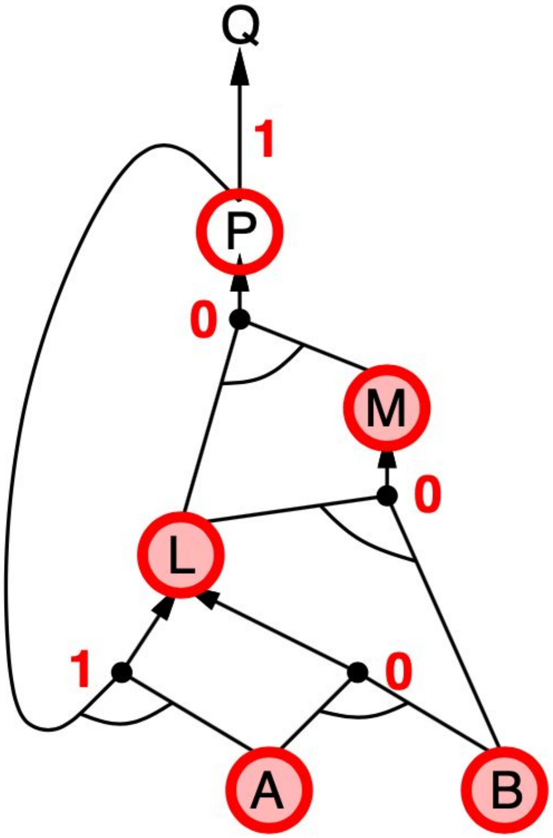
# 5.5 Forward chaining

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



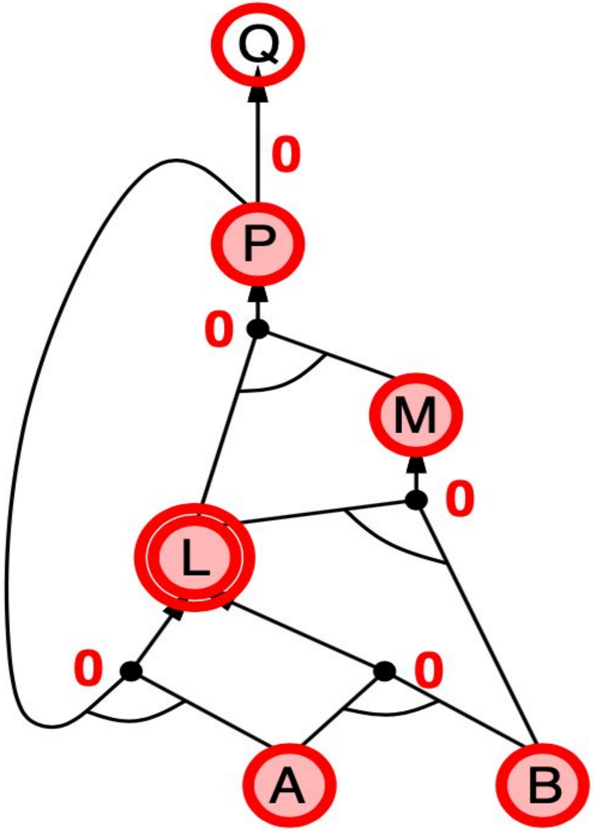
# 5.5 Forward chaining

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



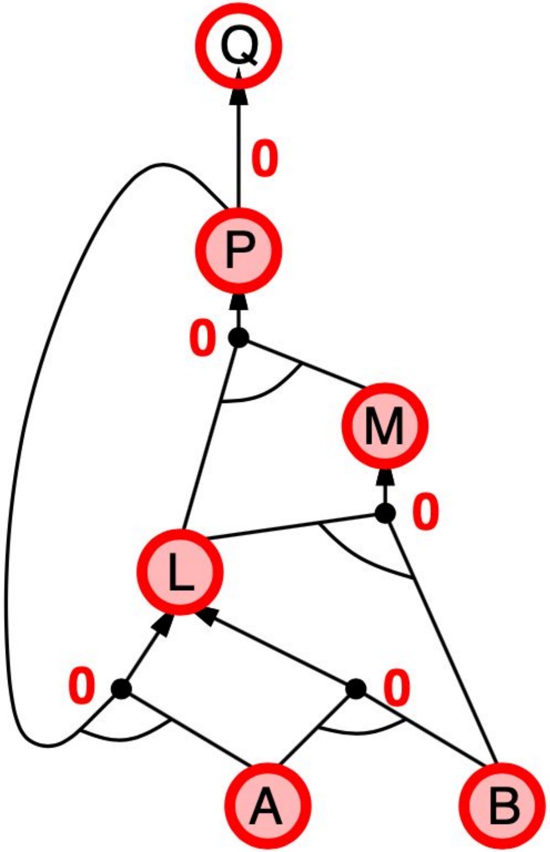
# 5.5 Forward chaining

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



# 5.5 Forward chaining

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



# 5.5 Forward chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

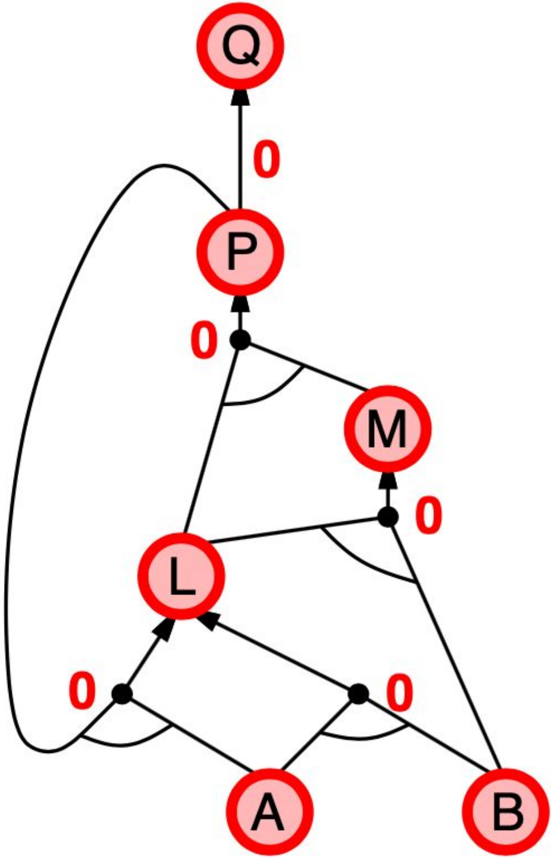
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

$A$

$B$



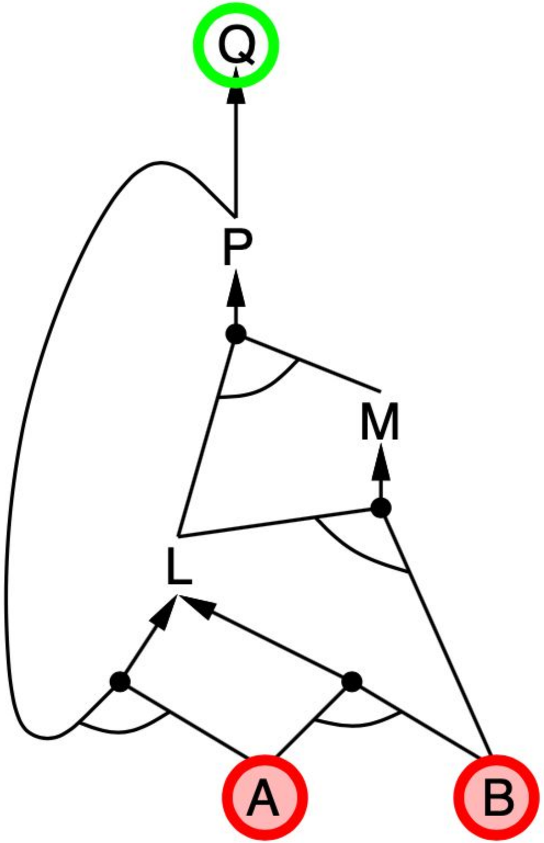
# 5.5 Backward chaining

- Idea: work backwards from the query q to prove q by BC,
  - check if q is known already, or
  - prove by 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

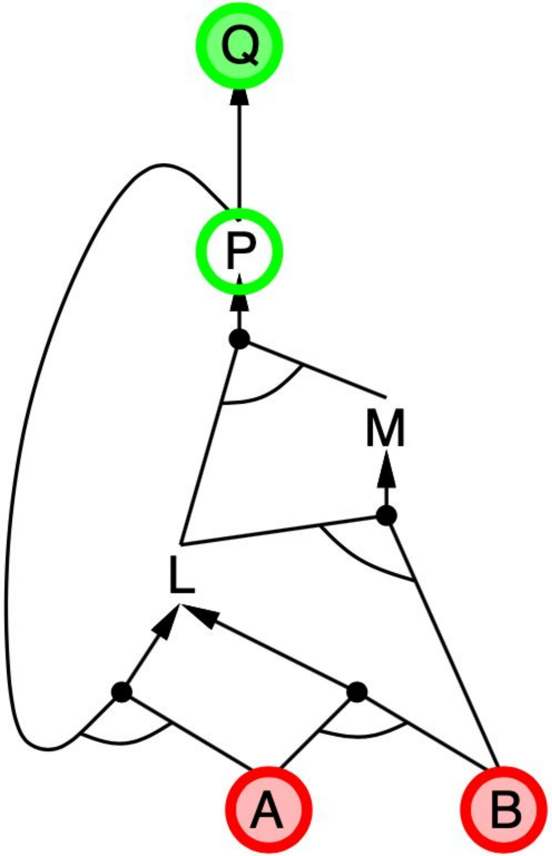
# 5.5 Backward chaining

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



# 5.5 Backward chaining

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



# 5.5 Backward chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

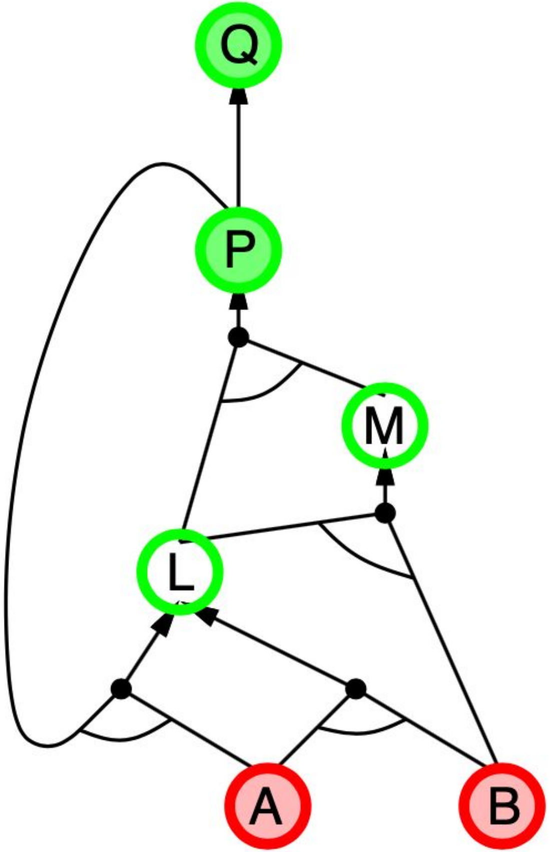
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

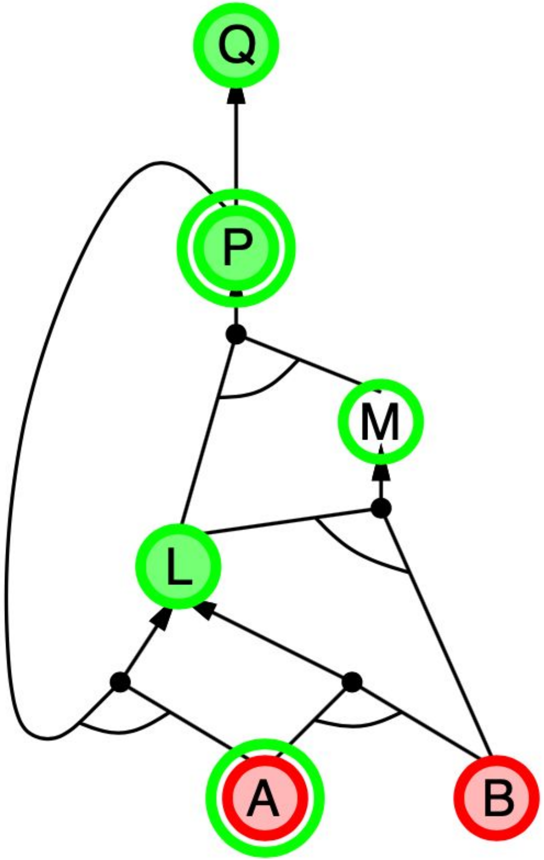
$A$

$B$



# 5.5 Backward chaining

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



# 5.5 Backward chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

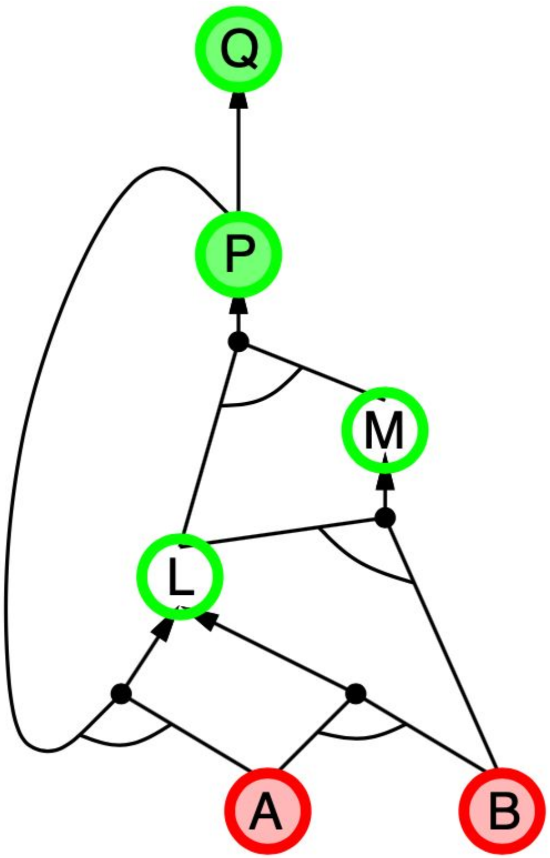
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

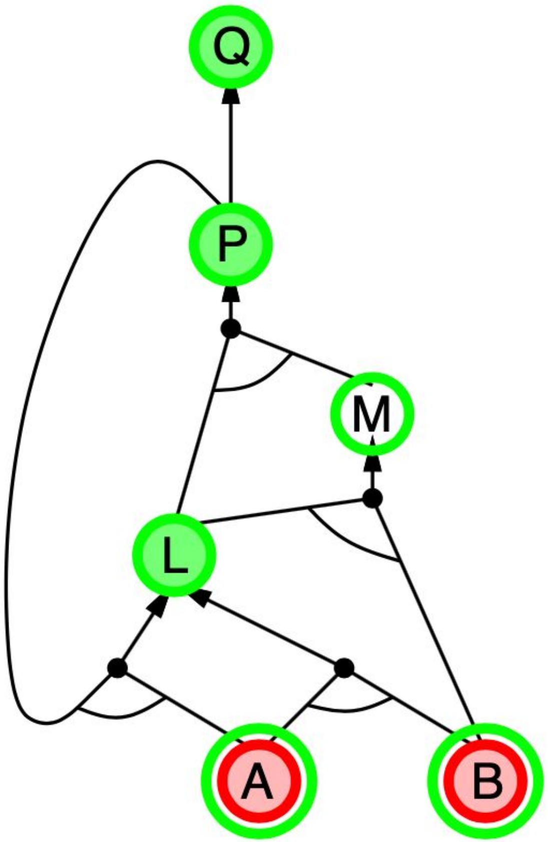
$A$

$B$



# 5.5 Backward chaining

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



# 5.5 Backward chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

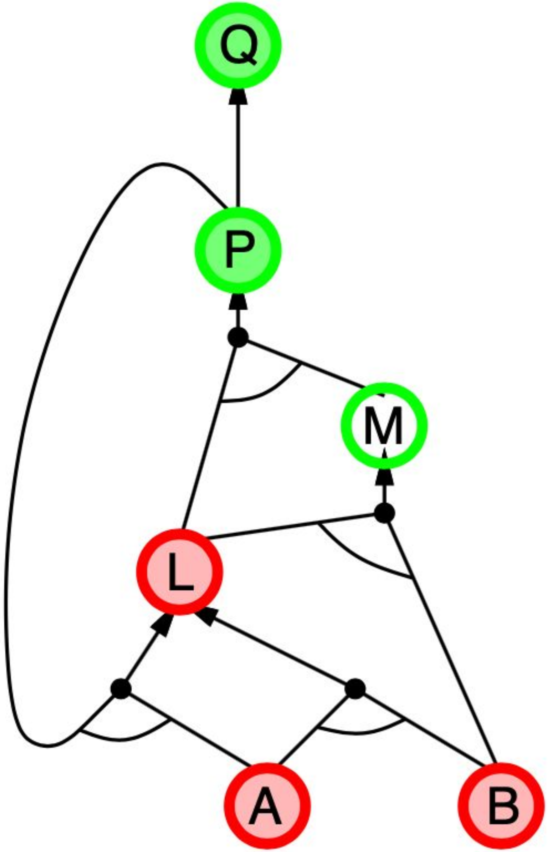
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

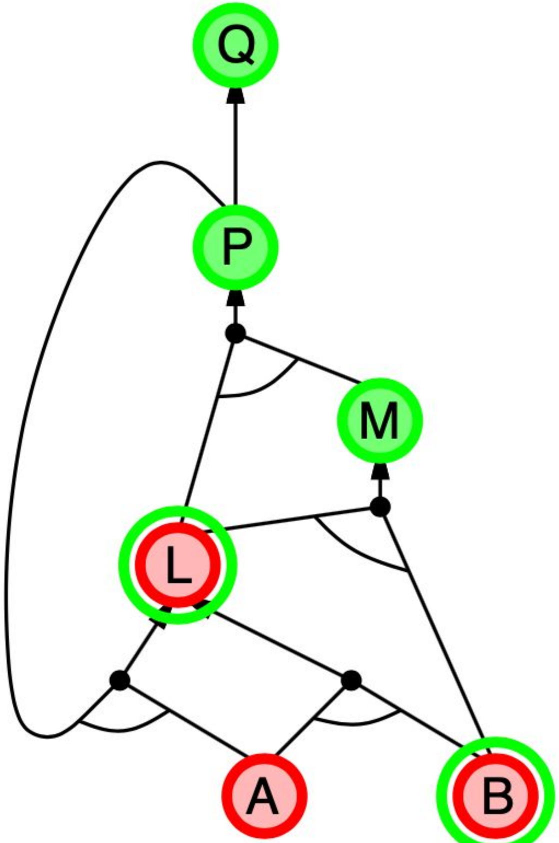
$A$

$B$



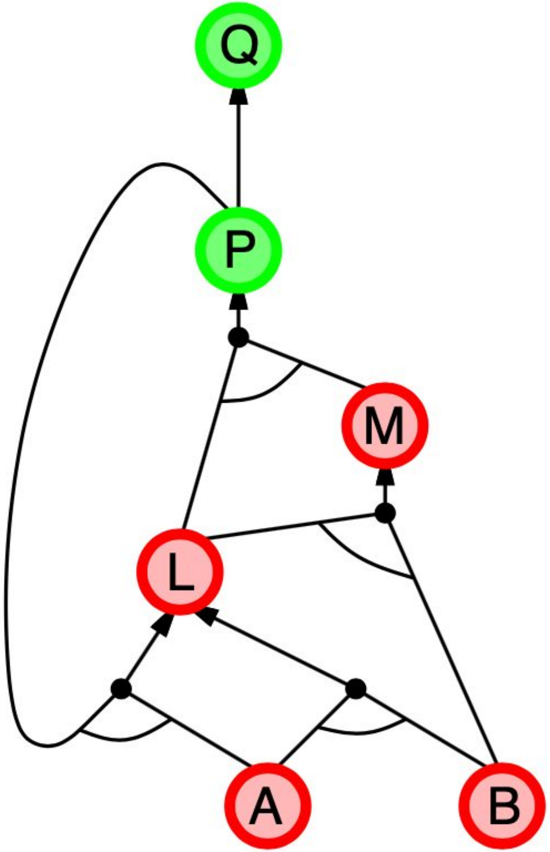
# 5.5 Backward chaining

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



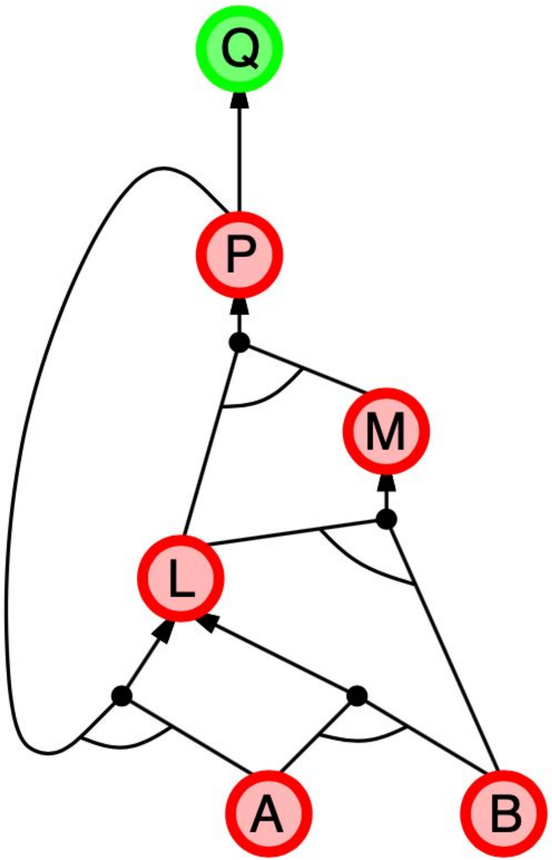
# 5.5 Backward chaining

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



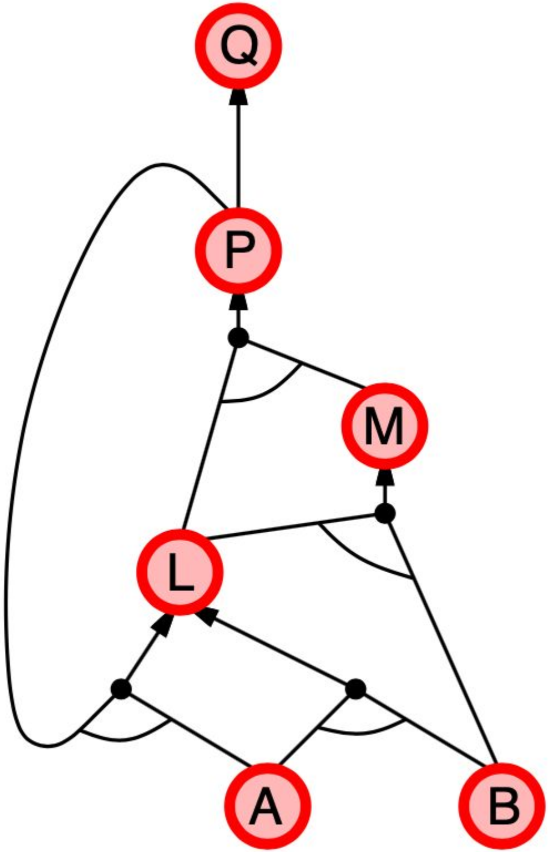
# 5.5 Backward chaining

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



# 5.5 Backward chaining

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



# 5.5 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?