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.

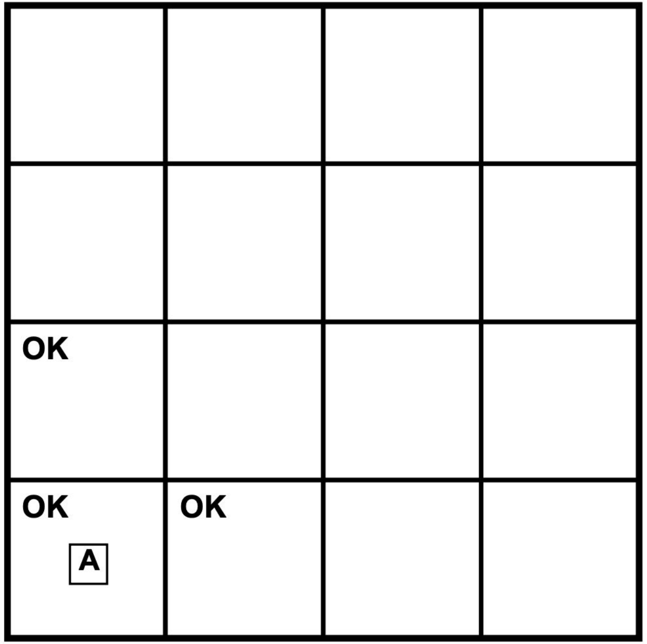
```
function KB-AGENT(percept) returns an action
   static: KB, a knowledge base
           t, a counter, initially 0, indicating time

   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 description

- **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

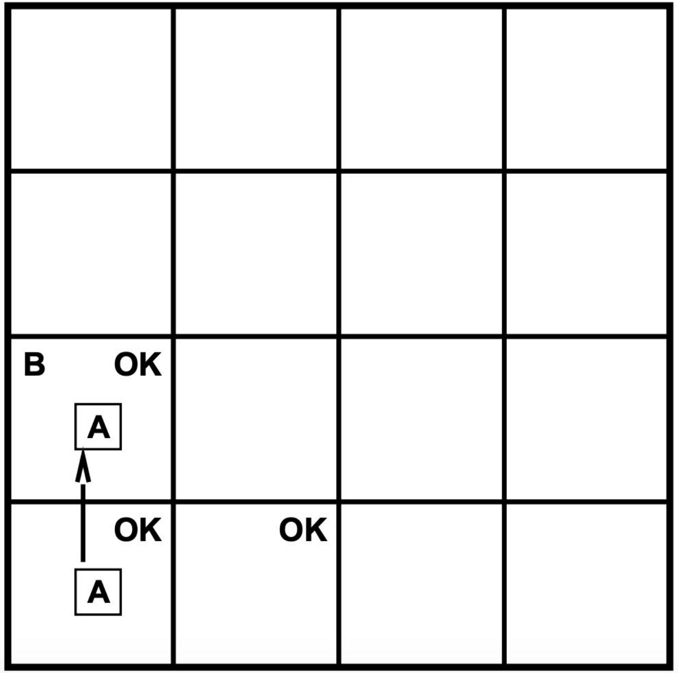
5.2 Wumpus World PEAS description

- The first percept is [None,None,None,None,None] => the agent can conclude that its neighboring squares, [1,2] and [2,1], are OK.



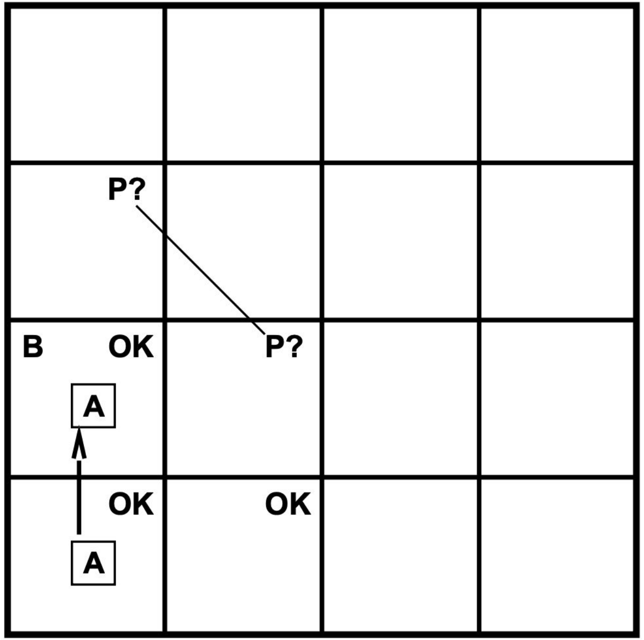
5.2 Wumpus World PEAS description

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



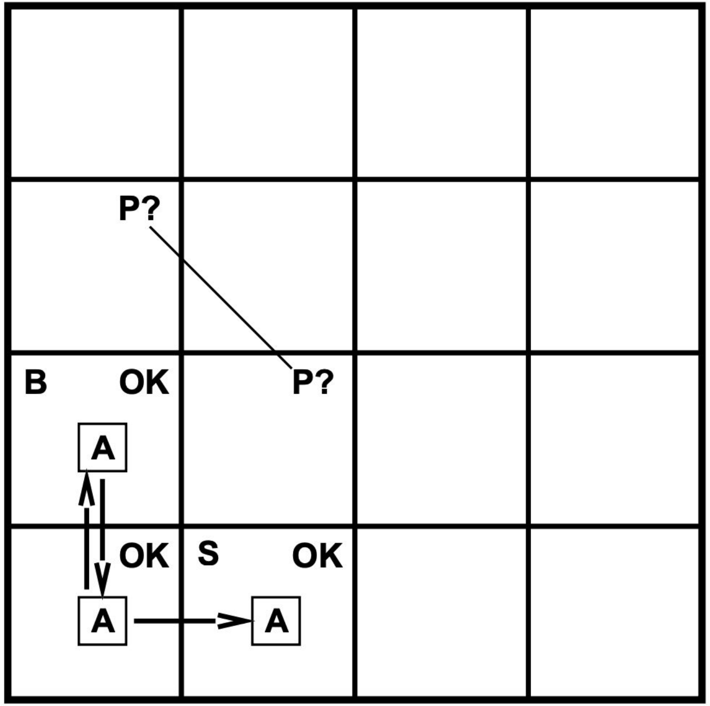
5.2 Wumpus World PEAS description

- The agent perceives a breeze (denoted by “B”) in [2,1] => 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.



5.2 Wumpus World PEAS description

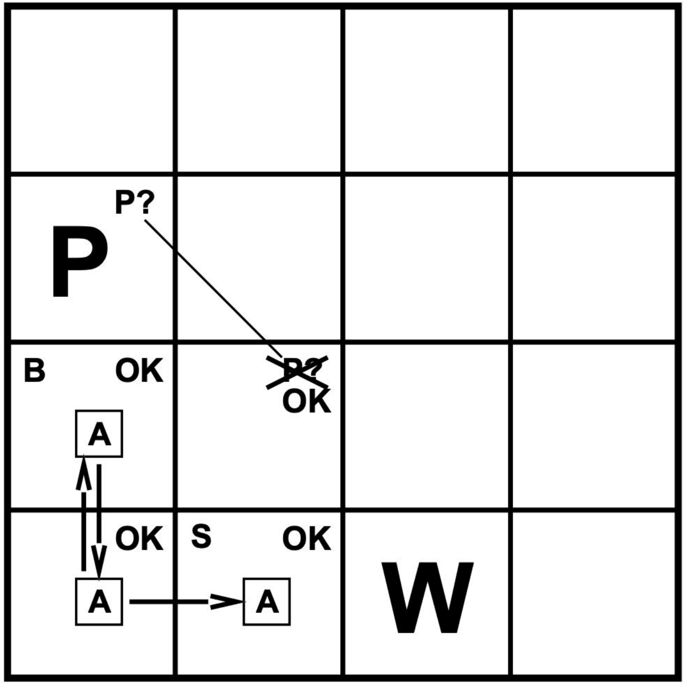
The agent will turn around, go back to [1,1], and then proceed to [1,2].



5.2 Wumpus World PEAS description

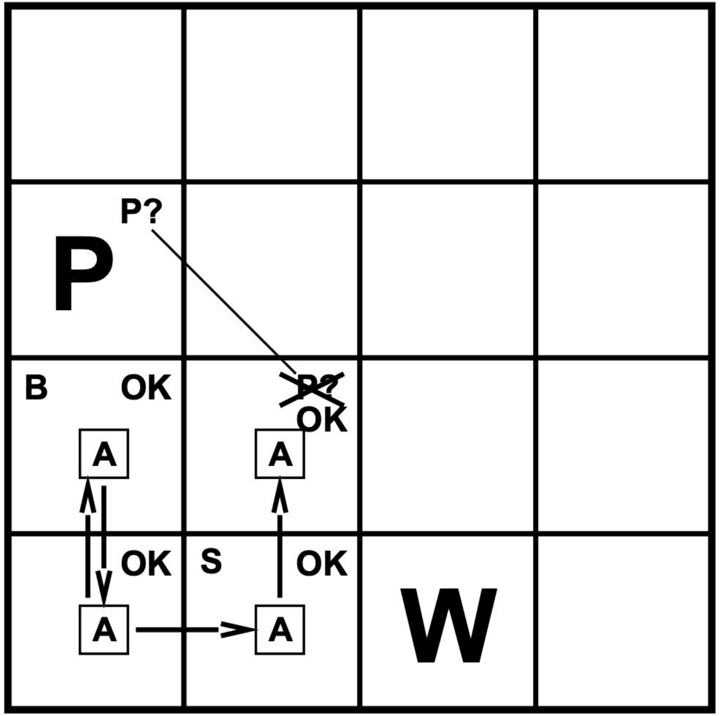
- The agent perceives a stench in [1,2] => there must be a wumpus nearby ([2,2] or [1,3])
- The lack of stench when the agent was in [2,1] => wumpus cannot be in [2,2] => it is in [1,3]
- the lack of a breeze in [1,2] => there is no pit in [2,2]

=> [2,2]: safe, OK



5.2 Wumpus World PEAS description

The agent draws a conclusion from the available information,
that 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
  - The semantics defines 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 α if and only if α is true in all worlds where KB is true

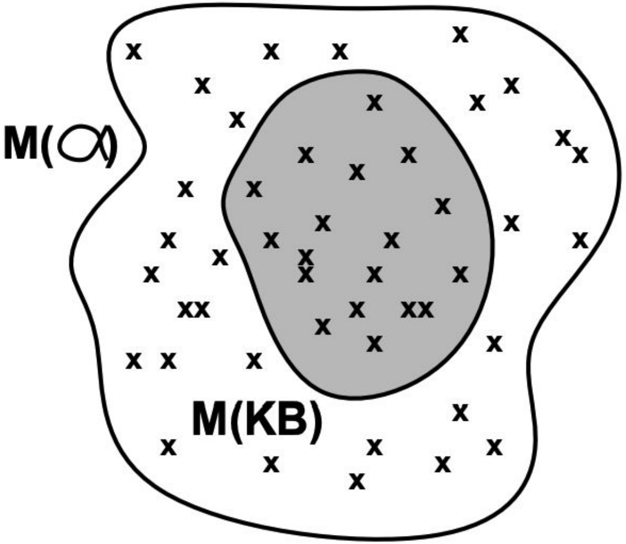
  $KB \models \alpha$
- E.g., KB containing “the Giants won” and “the Reds won” entails “Either the Giants won or the Reds won”

5.3 Models

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

E.g. KB = Giants won and Reds won

α = Giants won



5.3 Inference and Entailment

- An inference algorithm is a procedure for deriving a sentence from the KB
- If an inference algorithm i can derive α from KB, we write

  $$KB \vdash_i \alpha$$

  which is pronounced "α is derived from KB by i" or "i derives α from KB"

  OR: the sentence α is inferred from KB using algorithm i.

5.4 Propositional logic: Syntax

- Propositional logic is the simplest logic - illustrates basic ideas
- The proposition symbols P_1, P_2, … are sentences
- 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		
$S_1 \land S_2$	is true iff	S_1	is true **and**	S_2	is true
$S_1 \lor S_2$	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	$\neg P$	$P \land Q$	$P \lor Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
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 square.

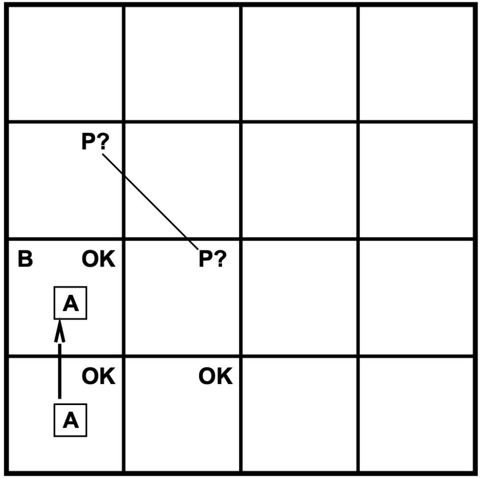
$R2: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$.

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

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 α
  KB, α as $\neg P_{1,2}$
  prove: $KB \models \neg P_{1,2}$
- A model-checking approach:
  - enumerate the models
  - check that α is true in every model in which KB is true

5.4 A simple knowledge base - Wumpus world sentences

With 7 symbols, there are 27 = 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.
- Some additional concepts related to entailment:
  - Logical equivalence
  - Validity
  - Satisfiability

5.5 Logical equivalence

Two sentences are logically equivalent iff true in same models:

$\alpha \equiv \beta$ 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 α, β, and γ 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 and proofs

Modus Ponens $\dfrac{\alpha \Rightarrow \beta, \qquad \alpha}{\beta}$

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

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

Logical equivalence for contrapositives gives

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

Apply Modus Ponens with R8 and the percept 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—universal)

  conjunction of clauses (i.e., disjunctions of literals)

  E.g., $(A \lor \neg B) \land (B \lor \neg C \lor \neg D)$

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

1. Eliminate $\Leftrightarrow$

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

2. Eliminate $\Rightarrow$

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

3. Apply logical equivalences

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

4. Apply the distributivity law

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

5.5 Proof by resolution

- **Resolution inference rule** (for CNF)

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

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

5.5 Proof by resolution

- Inference procedures based on a resolution algorithm uses the principle of proof by contradiction

  KB $\models \alpha$ if and only if (KB $\wedge \neg\alpha$) is unsatisfiable

- Steps:
  1. (KB $\wedge \neg\alpha$) is converted into CNF
  2. The resolution rule is applied to the resulting clauses, a new clause is added to the set if it is not already present
  3. The process continues until one of two things happens:
     i. no new clauses that can be added, in which case KB does not entail α;
     ii. two clauses resolve to yield the empty clause (~False), in which case KB entails α.

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

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

α 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 Proof by resolution

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

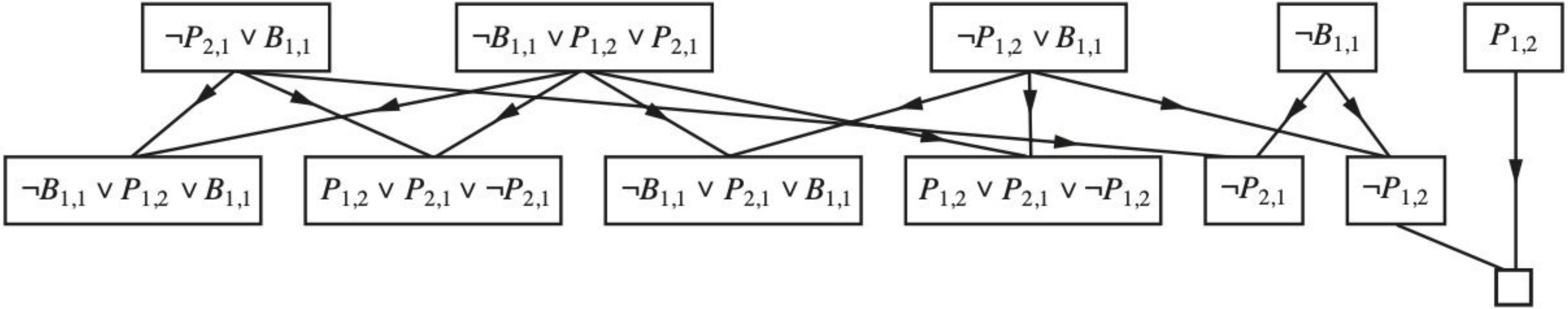
function PL-RESOLUTION(KB, α) **returns** *true* or *false*
  inputs: KB, the knowledge base, a sentence in propositional logic
    α, the query, a sentence in propositional logic

  $clauses \leftarrow$ 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*
    $clauses \leftarrow clauses \cup new$

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 Horn clauses and definite clauses

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

$$
\begin{array}{rcl}
\mathit{CNFSentence} & \rightarrow & \mathit{Clause}_1 \wedge \cdots \wedge \mathit{Clause}_n \\
\mathit{Clause} & \rightarrow & \mathit{Literal}_1 \vee \cdots \vee \mathit{Literal}_m \\
\mathit{Literal} & \rightarrow & \mathit{Symbol} \mid \neg \mathit{Symbol} \\
\mathit{Symbol} & \rightarrow & P \mid Q \mid R \mid \ldots \\
\mathit{HornClauseForm} & \rightarrow & \mathit{DefiniteClauseForm} \mid \mathit{GoalClauseForm} \\
\mathit{DefiniteClauseForm} & \rightarrow & (\mathit{Symbol}_1 \wedge \cdots \wedge \mathit{Symbol}_l) \Rightarrow \mathit{Symbol} \\
\mathit{GoalClauseForm} & \rightarrow & (\mathit{Symbol}_1 \wedge \cdots \wedge \mathit{Symbol}_l) \Rightarrow \mathit{False}
\end{array}
$$

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

5.5 Forward chaining

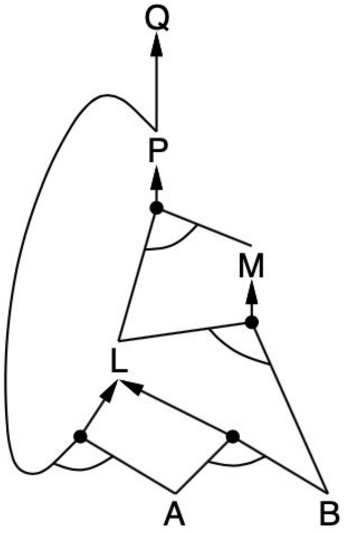
Idea: fire any rule whose premises are satisfied in the KB,
add its conclusion to the KB, until query is found or no further inferences can be made.

```
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

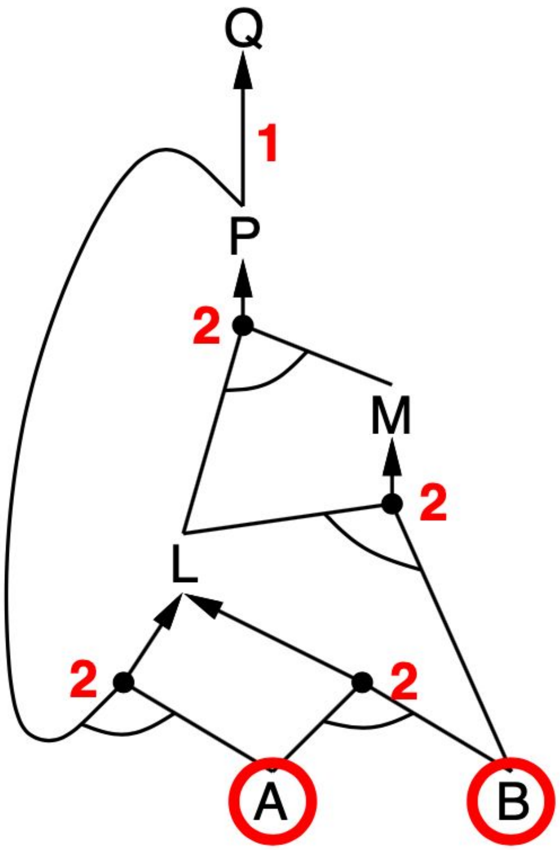
- In AND–OR graphs,
  - multiple links joined by an arc indicate a conjunction
  - multiple links without an arc indicate a disjunction
- How the graphs work:
  - The known leaves are set, inference propagates up the graph as far as possible.
  - Where a conjunction appears, the propagation waits until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

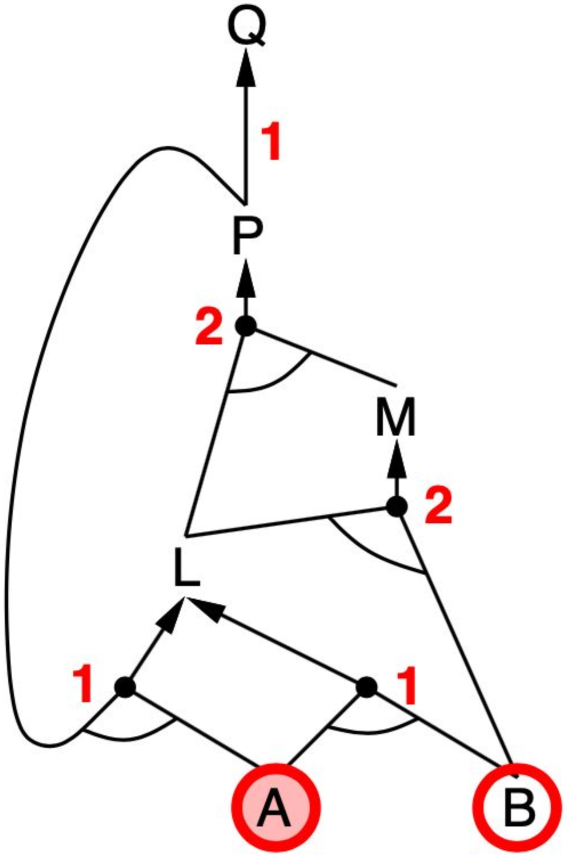
$$
\begin{aligned}
P &\Rightarrow Q \\
L \wedge M &\Rightarrow P \\
B \wedge L &\Rightarrow M \\
A \wedge P &\Rightarrow L \\
A \wedge B &\Rightarrow L \\
A & \\
B &
\end{aligned}
$$



5.5 Forward chaining

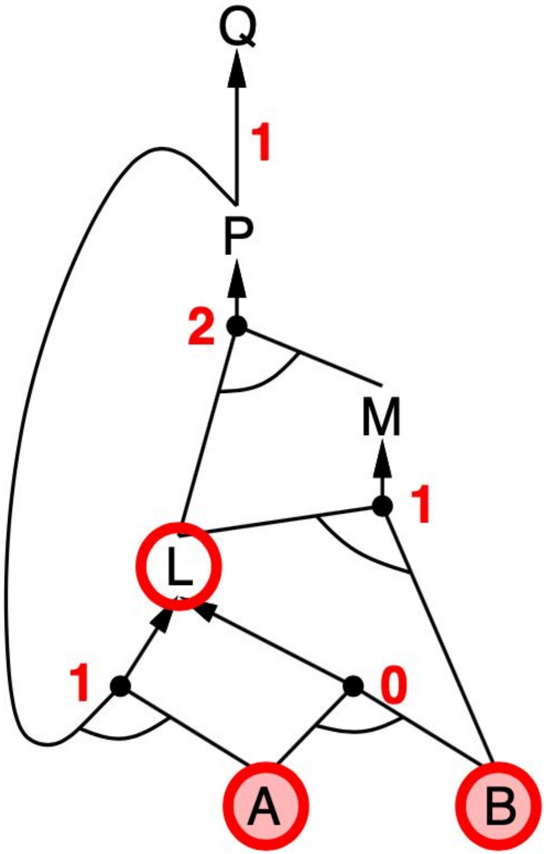
$$
\begin{aligned}
P &\Rightarrow Q \\
L \wedge M &\Rightarrow P \\
B \wedge L &\Rightarrow M \\
A \wedge P &\Rightarrow L \\
A \wedge B &\Rightarrow L \\
A & \\
B &
\end{aligned}
$$

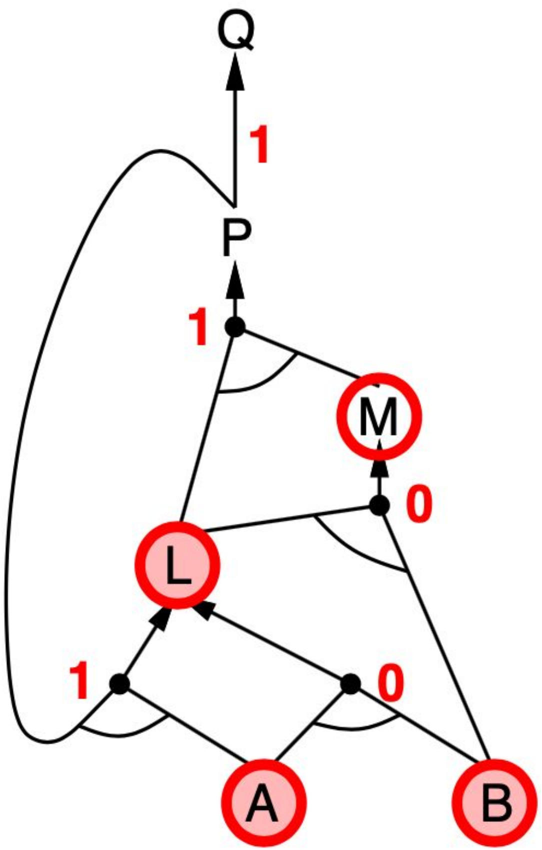




5.5 Forward chaining

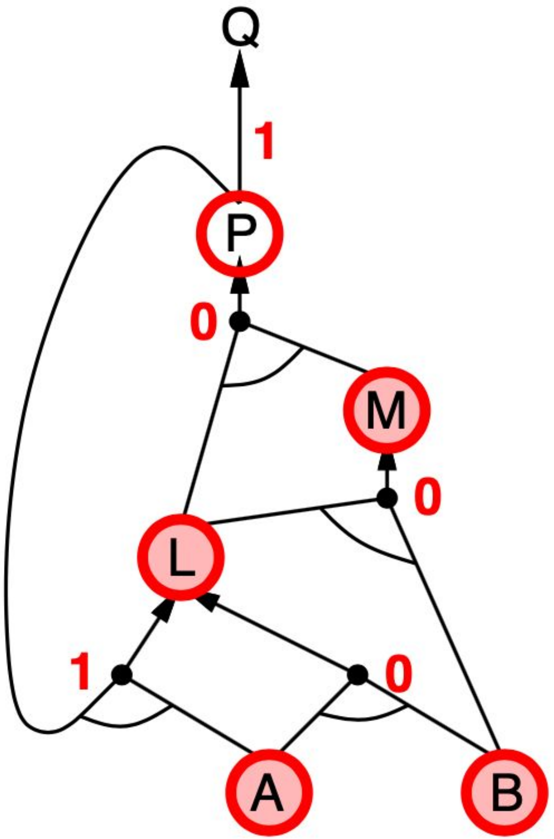
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

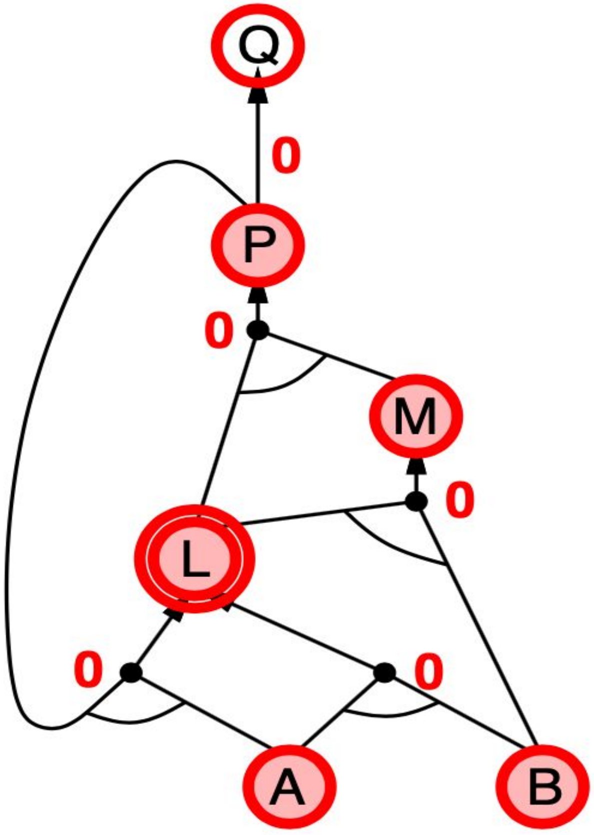




5.5 Forward chaining

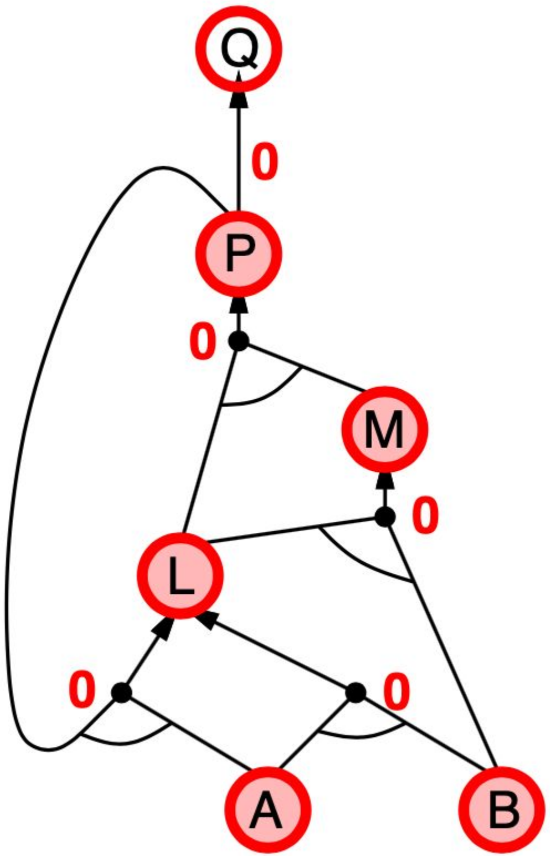
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

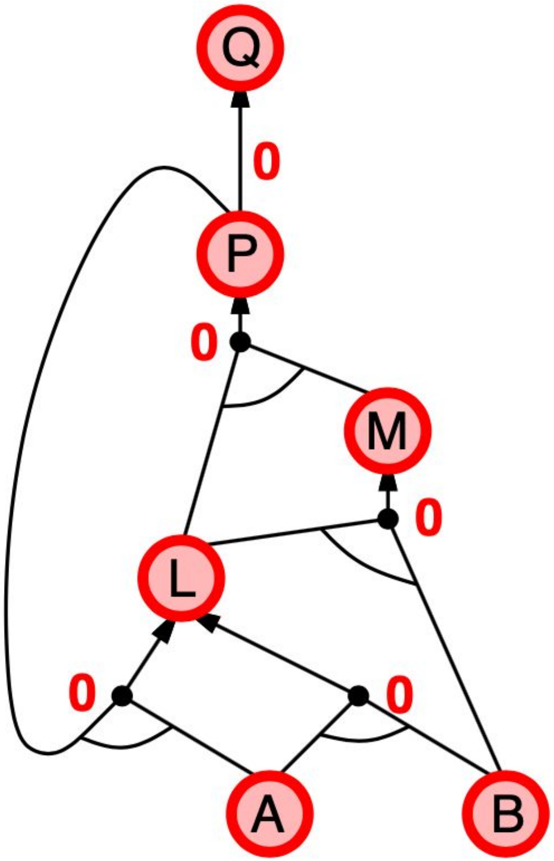




5.5 Forward chaining

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



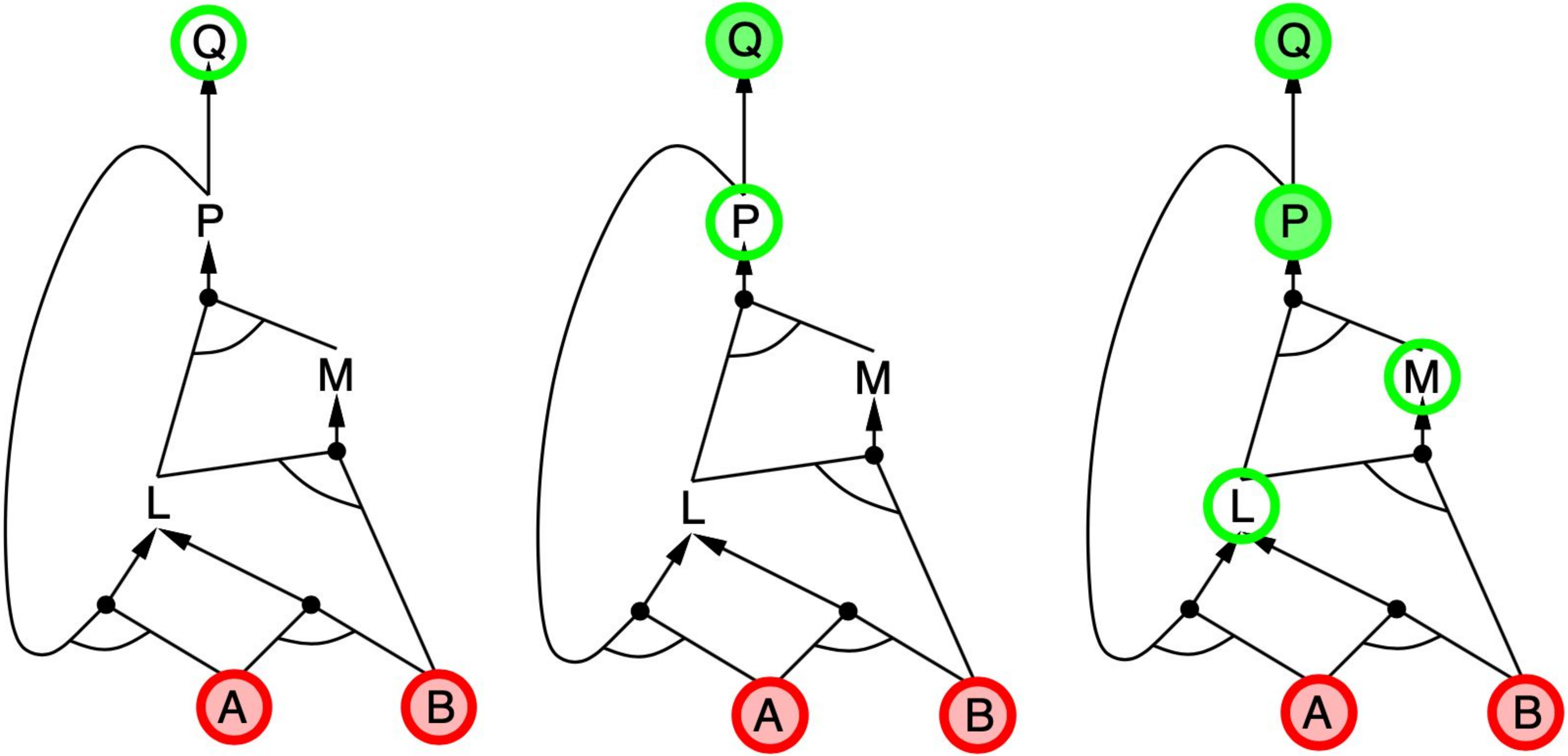


5.5 Backward chaining

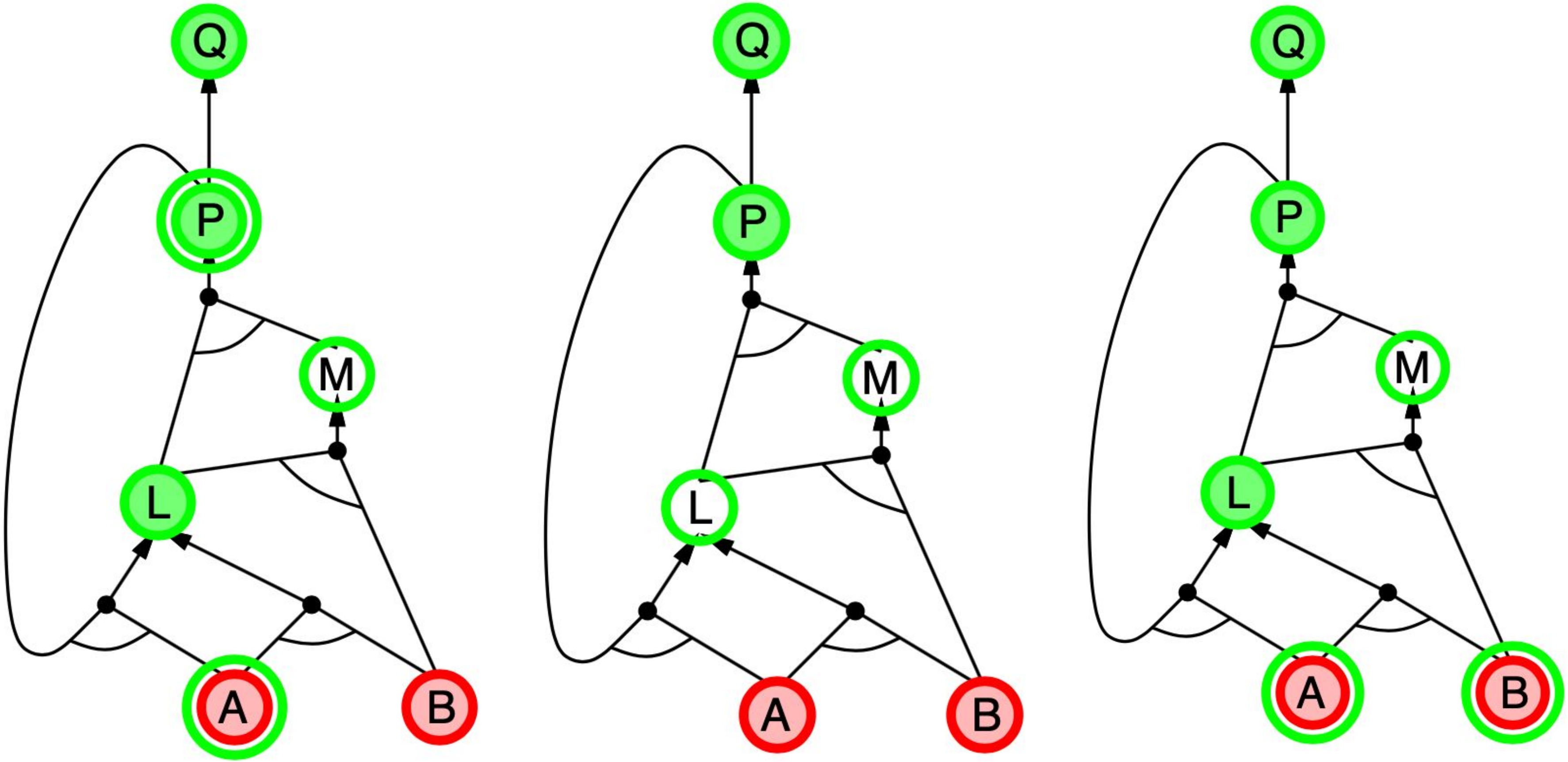
- Idea: work backwards from the query q to prove q by BC,
    - check if q is known already, or
    - prove by BC all premises of some rule concluding q

- 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

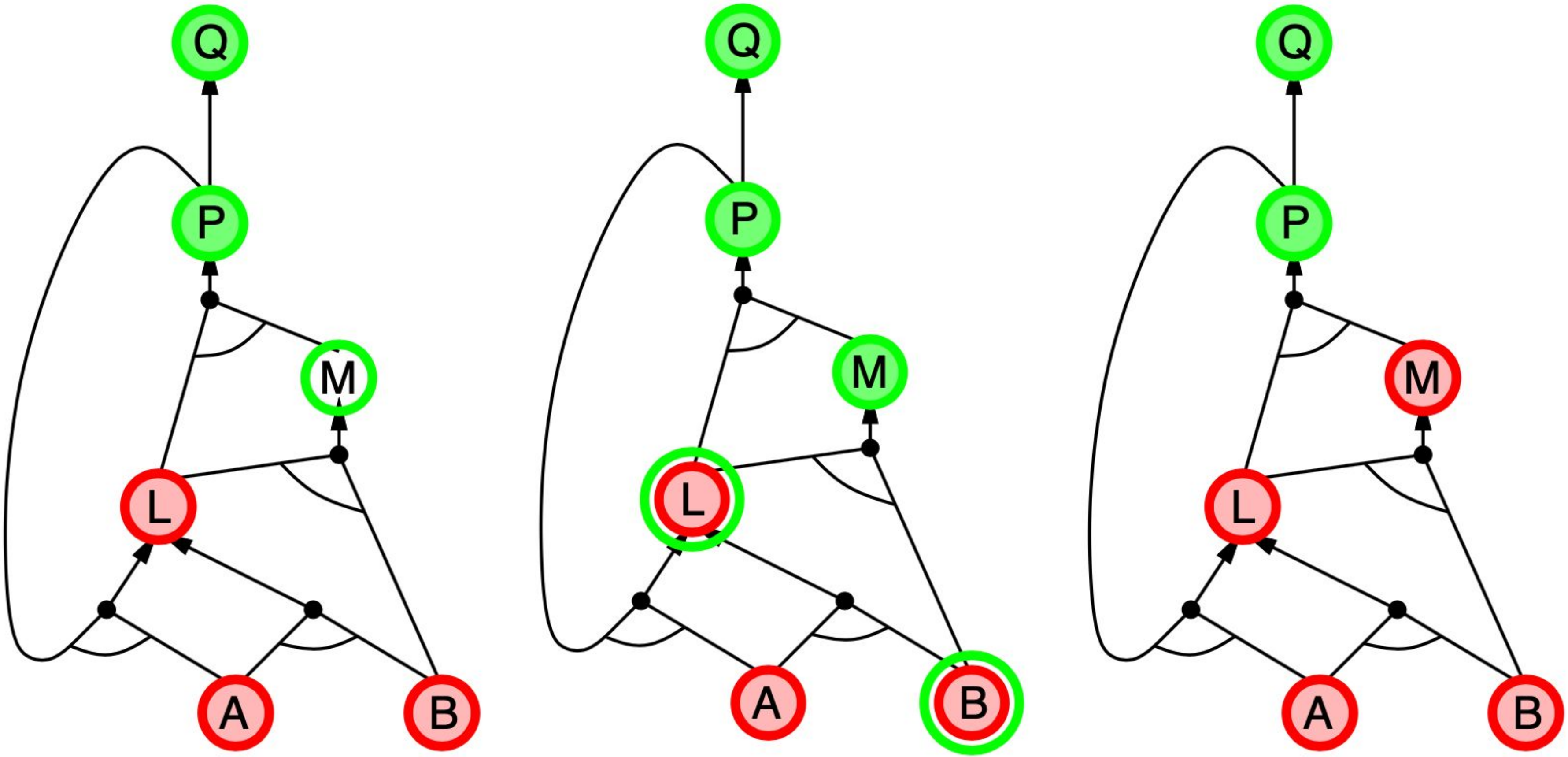
5.5 Backward chaining



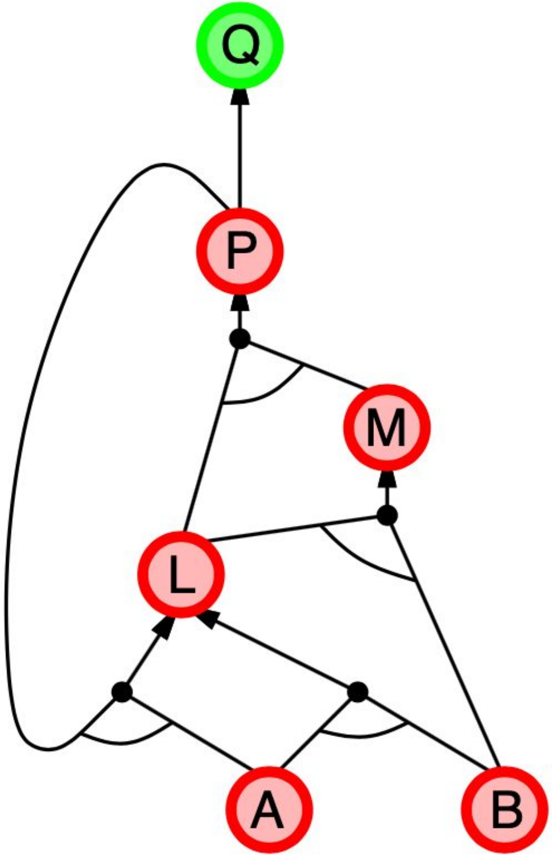
5.5 Backward chaining

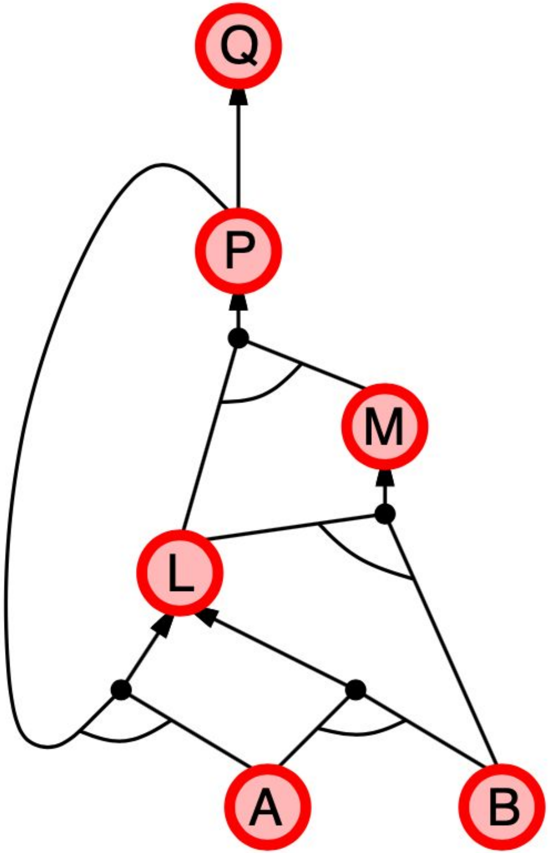


5.5 Backward chaining



5.5 Backward chaining





5.5 Forward vs. backward chaining

- FC is data-driven, cf. 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?