LOGICAL AGENTS

Dr. M. Saritha
AP/CSE

INTRODUCTION

- Agent perceives the environment through sensors and act on the environment through actuators
- An agent represents the knowledge about its environment, goals and the current situation.

KNOWLEDGE-BASED AGENTS

- Knowledge-based agents use a **process of reasoning** over an **internal representation of knowledge** to decide what actions to take.
- The central component of a knowledge-based agent is its **knowledge base, or KB.**
- A Knowledge base is a set of **sentences** (logical sentences).
- Each sentence is expressed in a language called a **knowledge representation language** and represents some assertion about the world.
- An **axiom** is basically a statement that the agent assumes to be true without needing to prove it.

KNOWLEDGE-BASED AGENTS

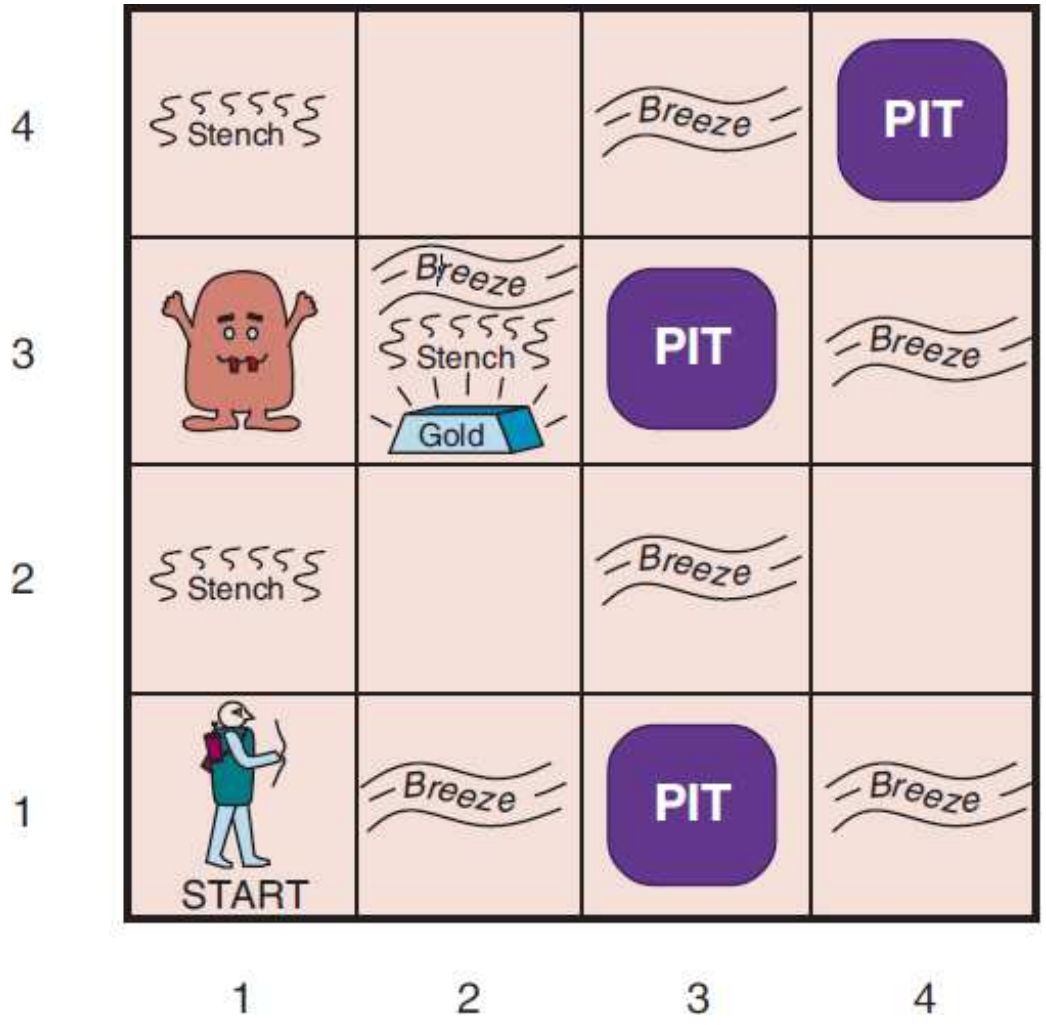
Generic Functions

- **TELL**
  - This operation tells the knowledge base what it perceives from the environment.
- **ASK**
  - This operation asks the knowledge base what action it should perform.

```
function KB-AGENT(percept) returns an action
  persistent: 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
```

THE WUMPUS WORLD



THE WUMPUS WORLD

PEAS

- **Performance**
  - +1000 for climbing out of the cave with the gold,
  - –1000 for falling into a pit or being eaten by the wumpus
  - –1 for each action taken,
  - –10 for using up the arrow.
  - The game ends either when the agent dies or when the agent climbs out of the cave.
- **Environment**
  - A 4×4 grid of rooms, with walls surrounding the grid.
  - The agent always starts in the square labeled [1,1], facing to the east.

THE WUMPUS WORLD

- **Actuators**
  - The agent can move **Forward, TurnLeft by 90◦, or TurnRight by 90◦.**
  - The agent dies a **miserable death** if it enters a square containing a pit or a live wumpus.
  - If an agent tries to move forward and **bumps into a wall**, then the agent does not move.
  - The action **Grab** can be used to pick up the gold if it is in the same square as the agent.
  - The action **Shoot** can be used to throw an arrow in a straight line in the direction the agent is facing. The agent has only one arrow, so only the first Shoot action has any effect.
  - Finally, the action **Climb** can be used to climb out of the cave, but only from square [1,1].

THE WUMPUS WORLD

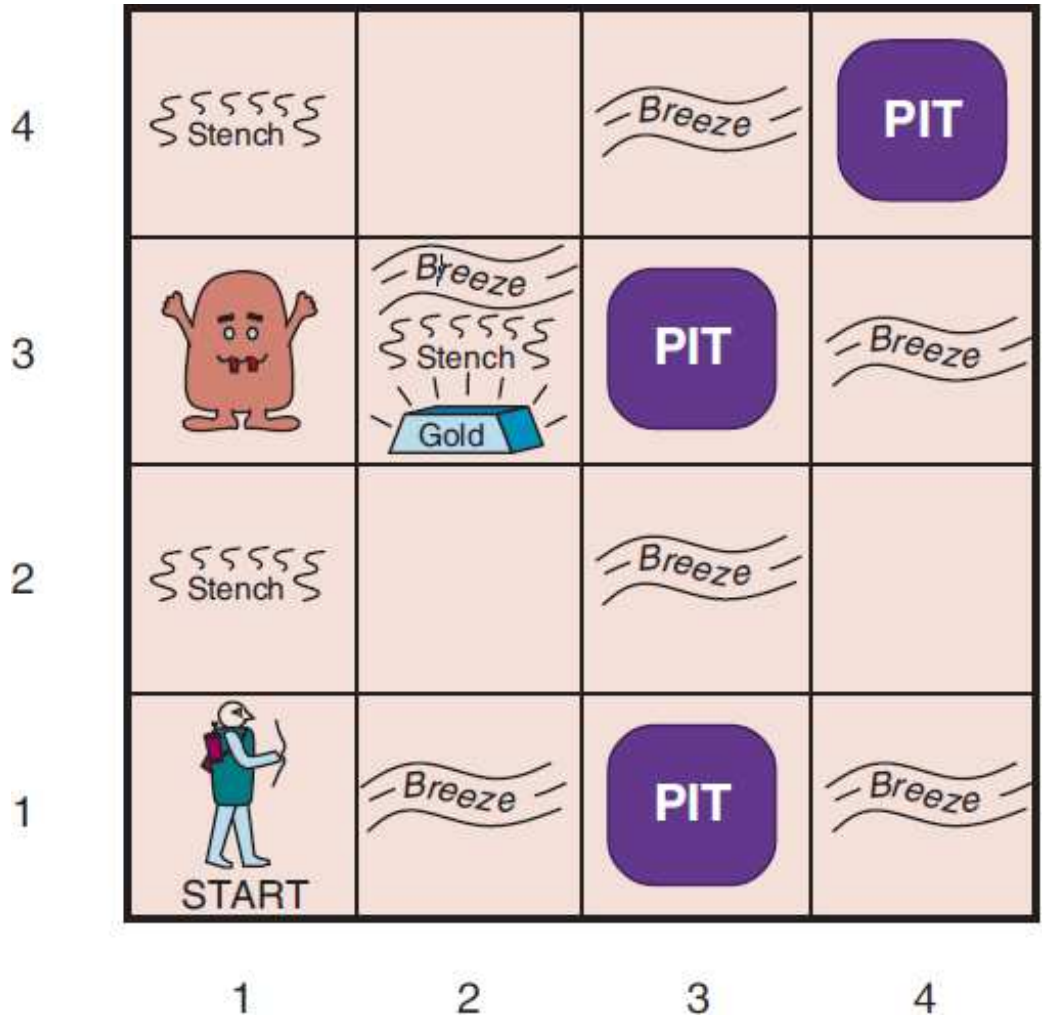
- **Sensors**

  The agent has five sensors, each of which gives a single bit of information:

  – In the squares directly (not diagonally) adjacent to the wumpus, the agent will perceive a **Stench**.
  – In the squares directly adjacent to a pit, the agent will perceive a **Breeze**.
  – In the square where the gold is, the agent will perceive a **Glitter**.
  – When an agent walks into a wall, it will perceive a **Bump**.
  – When the wumpus is killed, it emits a woeful **Scream** that can be perceived anywhere in the cave.
  – **[Stench,Breeze,Glitter,Bump,Scream]**

  Ex: [Stench,Breeze,None,None,None]

THE WUMPUS WORLD



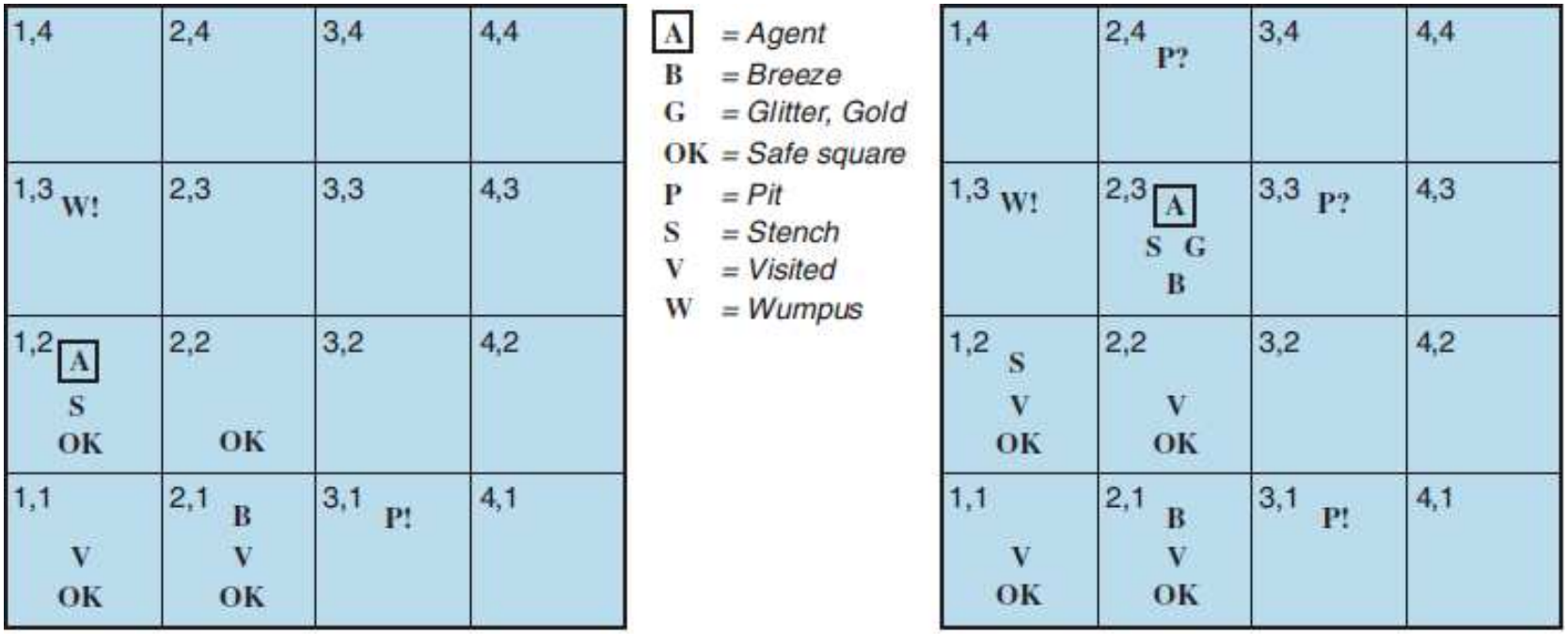
[Stench,Breeze,Glitter,Bump,Scream]

THE WUMPUS WORLD

(a) (b)

[Stench,Breeze,Glitter,Bump,Scream]

THE WUMPUS WORLD



(a) (b)

PROPOSITIONAL LOGIC

- Simplest form of logic
- All statements/sentences are made by propositions.
- A proposition is a declarative statement which is either **true or false but cannot be both**.
- It is a technique of knowledge representation in logical and mathematical form.
- Questions, opinion, comma are not allowed in propositional logic
- Ex:
  - Students are learning AI. (True proposition)
  - 3-2=4 (False proposition)
  - 3-2=1 (True proposition)
  - What do you want? (not accepted/ syntax error)
  - Some students like AI (True/False proposition. Therefore, not accepted)

SYNTAX

- **Syntax** defines the allowable sentences.
- Two types of sentences:

  Atomic sentences consist of a single proposition symbol.

  Each symbol represents a true or false.

  - Students are learning AI. (True proposition)
  - 3-2=4 (False proposition)
  - 3-2=1 (True proposition)

SYNTAX

- **Complex sentences** are constructed from simpler sentences, using parentheses and operators called logical connectives
- 5 connectives
  - conjunction (∧), and, its parts are conjuncts
  - disjunction (∨), or, its parts are disjuncts
  - negation ¬ not
  - implication ⇒implies, if then, or rules
  - biconditional ⇔ iff

SYNTAX

Peter can play tennis	P
Peter cannot play tennis	$\sim P$
Peter can play tennis and badminton	$P \wedge Q$
Peter can play tennis or badminton	$P \vee Q$
If Peter can play tennis then he can play badminton	$P \rightarrow Q$
Peter can play tennis if and only if he can play badminton	$P \leftrightarrow Q$

SYNTAX

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} &\rightarrow \textit{True} \mid \textit{False} \mid P \mid Q \mid R \mid \ldots \\
\textit{ComplexSentence} &\rightarrow (\textit{Sentence}) \\
&\mid \neg \textit{Sentence} \\
&\mid \textit{Sentence} \wedge \textit{Sentence} \\
&\mid \textit{Sentence} \vee \textit{Sentence} \\
&\mid \textit{Sentence} \Rightarrow \textit{Sentence} \\
&\mid \textit{Sentence} \Leftrightarrow \textit{Sentence}
\end{aligned}
$$

$$
\text{OPERATOR PRECEDENCE} \quad : \quad \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow
$$

BNF (BACKUS–NAUR FORM) GRAMMAR OF SENTENCES

SEMANTICS

- The semantics defines **the rules for determining the truth of a sentence** with respect to a particular model.
- In propositional logic, a model simply sets the truth value—true or false—for every proposition symbol.

$m_1 = \{P_{1,2} = false, P_{2,2} = false, P_{3,1} = true\}$

sample model

- Wumpus world probem
  - pit in 1,2 → false
  - pit in 2,2 → false
  - pit in 3,1 → true

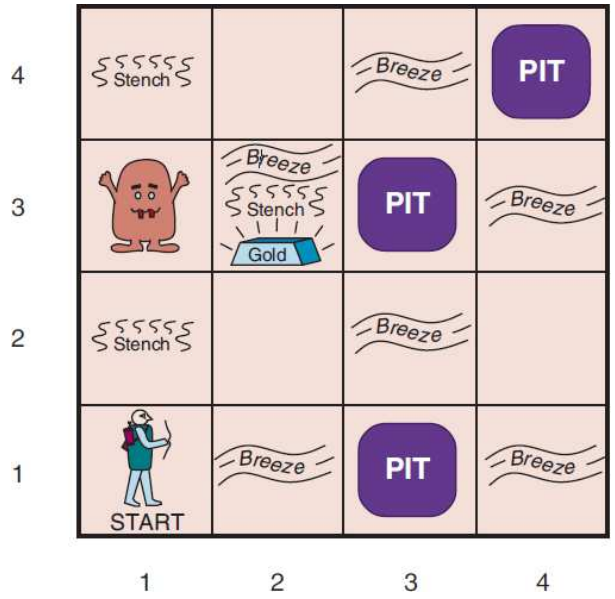
SEMANTICS

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*

TRUTH TABLES FOR THE FIVE LOGICAL CONNECTIVES

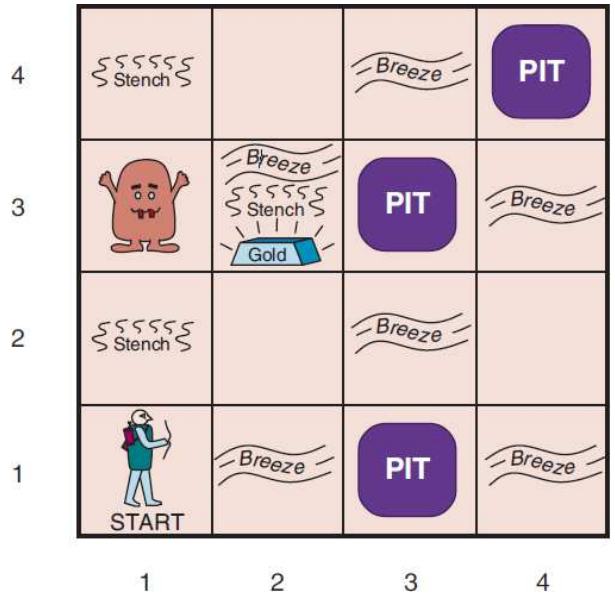
A SIMPLE KNOWLEDGE BASE

- $P_{x,y}$ is true if there is a pit in $[x, y]$.
- $W_{x,y}$ is true if there is a wumpus in $[x, y]$, dead or alive.
- $B_{x,y}$ is true if there is a breeze in $[x, y]$.
- $S_{x,y}$ is true if there is a stench in $[x, y]$.
- $L_{x,y}$ is true if the agent is in location $[x, y]$.



A SIMPLE KNOWLEDGE BASE

- There is no pit in [1,1]:
  - $R1: \neg P_{1,1}$.
- A square is breezy if and only if there is a pit in a neighboring square. This has to be stated for each square; for now, we include just the relevant squares:
  - $R2: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$.
  - $R3: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$.
- After visiting [1,1], [2,1] and [1,2]:
  - R4: $\neg B_{1,1}$.
  - R5: $B_{2,1}$.



A SIMPLE INFERENCE PROCEDURE

- Goal
  - KB $\models \alpha$ for some sentence α. (ie KB entails α)
  - The formal definition of entailment is this: α $\models$ β if and only if, in every model in which α is true, β is also true.
- KB consists of sentences R1 to R5.
- Single sentence KB = R1 ∧ R2 ∧ R3 ∧ R4 ∧ R5

A SIMPLE MODEL BASED INFERENCE PROCEDURE

- Find whether the KB says there is no pit in [1,2]
  - ie does KB $\models \neg P_{1,2}$
- $\neg P_{1,2}$ is a sentence α
- Goal
  - Decide whether KB $\models \alpha$
  - α can be much more complex query
- Enumerate the models.
- For each model check that:
  - if it is true in α it has to be true in KB

A SIMPLE MODEL BASED INFERENCE PROCEDURE

- 7 relevant symbols
- $B_{1,1}$, $B_{2,1}$, $P_{1,1}$, $P_{1,2}$, $P_{2,1}$, $P_{2,2}$, and $P_{3,1}$

- $2^7 = 128$ models. Only 3 are true

A SIMPLE MODEL BASED INFERENCE PROCEDURE

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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

- KB $\models \neg P_{1,2}$
- Does KB $\models \neg P_{1,1}$?

PROPOSITIONAL THEOREM PROVING

- Theorem proving
  - Applying rules of inference directly to the sentences in the knowledge base to construct a proof of the desired sentence without consulting models.
  - When there are many possible models to check, but the proof required is short, theorem proving can be faster than model checking.

PROPOSITIONAL THEOREM PROVING

- Logical equivalence
  - Two sentences α and β are logically equivalent if they are true in the same set of models.
  - **$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$**
- Validity
  - A sentence is valid if it is true in all possible models.
  - $P \vee \neg P$ is valid
- Tautology
  - A tautology is a sentence that is true in all models and is therefore logically valid.

PROPOSITIONAL THEOREM PROVING

- Deduction theorem
  - **For any sentences α and β, α |= β if and only if the sentence (α ⇒ β) is valid.**
- Satisfiability
  - A sentence is satisfiable if it is true in, or satisfied by some model
  - **(R1 ∧ R2 ∧ R3 ∧ R4 ∧ R5), is satisfiable because there are three models in which it is true.**
- Validity and satisfiability
  - Validity and satisfiability are connected.
  - **α is valid iff ¬α is unsatisfiable; contrapositively, α is satisfiable iff ¬α is not valid.**
  - **α |= β if and only if the sentence (α ∧ ¬β) is unsatisfiable**

SSN

STANDARD LOGICAL EQUIVALENCES

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

The symbols α, β, and γ stand for arbitrary sentences of propositional logic

INFERENCE AND PROOFS

- Inference rules can be applied to derive a proof
  - a chain of conclusions that leads to the desired goal
- Modus Ponens (Implication Elimination)
- And – Elimination
- And – Introduction
- Or – Introduction
- Unit Resolution
- Monotonicity
- Resolution

MODUS PONENS

- The best-known rule is called Modus Ponens and is written

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

- Any sentences of the form α ⇒ β and α are given, then the sentence β can be inferred.
  - **If it is raining, then Joe is inside. (α ⇒ β)**
  - **It is raining (α)**

    Infer → **Joe is inside. (β)**

  - **(WumpusAhead ∧ WumpusAlive) ⇒ Shoot**
  - **(WumpusAhead ∧ WumpusAlive)**
    - **Shoot can be inferred**

AND – ELIMINATION

- From a conjunction, any of the conjunct is inferred.

$$\frac{\alpha \wedge \beta}{\alpha} \qquad \frac{\alpha_1 \wedge \alpha_2 \wedge .... \wedge \alpha_n}{\alpha_1}$$

  - **Joe and Allen are friends of Joseph**
  - **α → Joe is a friend of Joseph**
  - **β → Allen is a friend of Joseph**
  - **α∧β Joe and Allen are friends of Joseph**
    - **Therefore infer either α or β**

AND – INTRODUCTION

- From a list of sentences their conjunction is inferred.

$$\frac{\alpha_1, \alpha_2 ..... \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge ..... \wedge \alpha_n}$$

  - **Joe pass AI exam.**
  - **Allen Pass AI exam.**
  - **α → Joe pass AI exam**
  - **β → Allen pass AI exam**
    - Infer α∧β **→ Joe and Allen pass AI exam**

- The equivalence for biconditional elimination yields the two inference rules

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)} \quad \text{and} \quad \frac{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}$$

OR – INTRODUCTION

- From a sentence, its disjunction with anything is inferred.

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots\ldots \vee \alpha_n}$$

  - **Pit in 3,1**
  - **$\alpha_i \rightarrow$ Pit in 3,1 (True)**
    - Infer $\rightarrow$ **$\alpha_i \vee \alpha_j$ (True)**

UNIT RESOLUTION

- From a disjunction, if one of the disjuncts is false, then the true one is inferred.

$$\frac{\alpha \vee \beta, \neg\beta}{\alpha}$$

  - **Joe or Allen pass AI exam (α ∨ β)**
  - **Allen did not pass AI exam. (¬β)**
  - **α → Joe pass AI exam**
  - **β → Allen pass AI exam**
  - **¬β → Allen did not pass AI exam**
    - Infer α → **Joe pass AI exam**

INFERENCE RULES, EQUIVALENCES IN WUMPUS WORLD

- The knowledge base containing R1 through R5 and show how to prove $\neg P_{1,2}$
- Goal
  - There is no pit in [1,2] ($\neg P_{1,2}$)
- Apply biconditional elimination to R2 to obtain

$$R_6 : \quad (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$$

- Then apply And-Elimination to R6 to obtain

$$R_7 : \quad ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$$

- Logical equivalence for contrapositives gives

$$R_8 : \quad (\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1}))$$

INFERENCE RULES, EQUIVALENCES IN WUMPUS WORLD

- Now apply Modus Ponens with R8 and the percept R4: $\neg B_{1,1}$

$$R_9 : \quad \neg(P_{1,2} \vee P_{2,1})$$

- Finally, we apply De Morgan's rule, giving the conclusion.

$$R_{10} : \quad \neg P_{1,2} \wedge \neg P_{2,1}$$

- Neither [1,2] nor [2,1] contains a pit.
- Derived proof by hand, but can also apply any of the search algorithms to find a sequence of steps that constitutes a proof.

MONOTONICITY

- **Monotonicity Property:** The set of entailed sentences can only increase as information is added to the knowledge base.

  For any sentences α and β, if KB $\models \alpha$ then KB$\wedge\beta \models \alpha$

- Monotonicity means that inference rules can be applied whenever suitable premises are found in the knowledge base; the conclusion of the rule must follow regardless of what else is in the knowledge base.

UNIT RESOLUTION

Unit Resolution

- From a disjunction, if one of the disjuncts is false, then the true one is inferred.

$$\frac{\alpha \vee \beta, \neg\beta}{\alpha}$$

or

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k}$$

- where each l is a literal and l_i and m are complementary literals

UNIT RESOLUTION

- The unit resolution rule can be generalized to the **full resolution rule**

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

where l_i and m_j are complementary literals.

- This says that resolution takes two clauses and produces a new clause containing all the literals of the two original clauses except the two complementary literals.

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

- Resolve only one pair of complementary literals at a time.

$$\frac{P \vee \neg Q \vee R, \qquad \neg P \vee Q}{\neg Q \vee Q \vee R}$$

PROOF BY RESOLUTION

- Resolution **yields a complete inference algorithm when coupled with any complete search algorithm.**
- The agent returns from [2,1] to [1,1] and then goes to [1,2], where it perceives a stench, but no breeze.
- Add the following facts to the knowledge base:

$$R_{11}: \quad \neg B_{1,2}.$$
$$R_{12}: \quad B_{1,2} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{1,3}).$$

- Derive similar to R10, the following rules

$$R_{13}: \quad \neg P_{2,2}.$$
$$R_{14}: \quad \neg P_{1,3}.$$

PROOF BY RESOLUTION

- Apply biconditional elimination to R3, followed by Modus Ponens with R5, to obtain the fact that there is a pit in [1,1], [2,2], or [3,1].

$$R_{15}: \quad P_{1,1} \vee P_{2,2} \vee P_{3,1} .$$

- Apply the resolution rule: the literal $\neg P_{2,2}$ in R13 resolves with the literal P2,2 in R15 to give the resolvent.

$$R_{16}: \quad P_{1,1} \vee P_{3,1} .$$

- Similarly, the literal ¬P1,1 in R1 resolves with the literal P1,1 in R16 to give

$$R_{17}: \quad P_{3,1} .$$

PROOF BY RESOLUTION

Conjunctive Normal Form

- A sentence expressed as a conjunction of clauses is said to be in conjunctive normal form or CNF.

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

- CNF → A conjunction of disjunction of literals

PROOF BY RESOLUTION

Procedure for converting to CNF

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. CNF requires $\neg$ to appear only in literals, so we "move $\neg$ inwards" by repeated application of the following equivalences from Figure 7.11:

$$\neg(\neg\alpha) \equiv \alpha \quad \text{(double-negation elimination)}$$
$$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \quad \text{(De Morgan)}$$
$$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \quad \text{(De Morgan)}$$

In the example, we require just one application of the last rule:

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

4. Now we have a sentence containing nested $\wedge$ and $\vee$ operators applied to literals. We apply the distributivity law from Figure 7.11, distributing $\vee$ over $\wedge$ wherever possible.

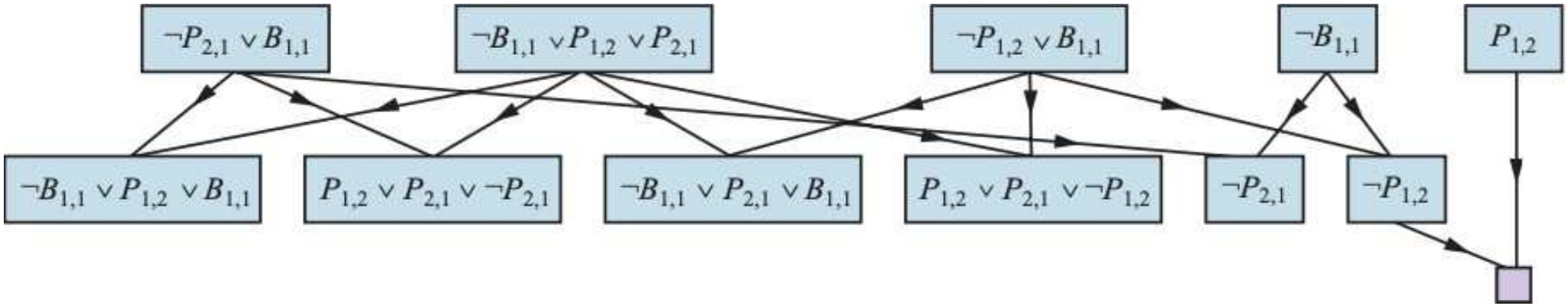
$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1}) .$$

PROOF BY RESOLUTION

A resolution algorithm

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

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



PROOF BY RESOLUTION

A resolution algorithm

- To show that KB |= α, we show that (KB ∧ ¬α) is unsatisfiable.
- PL-RESOLVE returns the set of all possible clauses obtained by resolving its two inputs.

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 $KB \wedge \neg\alpha$
  new ← { }
  while *true* **do**
    for each pair of clauses C_i, C_j **in** *clauses* **do**
      resolvents ← PL-RESOLVE(C_i, C_j)
      if *resolvents* contains the empty clause **then return** *true*
      new ← *new* ∪ *resolvents*
    if *new* ⊆ *clauses* **then return** *false*
    clauses ← *clauses* ∪ *new*

PROOF BY RESOLUTION

Completeness of resolution

- **Resolution closure RC(S)** of a set of clauses S, is the set of all clauses derivable by repeated application of the resolution rule to clauses in S or their derivatives.
- The resolution closure is what PL-RESOLUTION computes as the final value of the variable clauses.

PROOF BY RESOLUTION

Completeness of resolution

- The completeness theorem for resolution in propositional logic is called the **ground resolution theorem**:
  - If a set of clauses is unsatisfiable, then the resolution closure of those clauses contains the empty clause.

HORN CLAUSES AND DEFINITE CLAUSES

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

HORN CLAUSES AND DEFINITE CLAUSES

Definite clause

- Disjunction of literals of which **exactly one literal is positive.**
- For example, $(\neg L_{1,1} \vee \neg Breeze \vee B_{1,1})$ is a definite clause, whereas $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1})$ is not, because it has two positive clauses.

Horn Clause Form

- A disjunction of literals of which **atmost one literal is positive**

Ex: $\neg L_{1,1} \vee \neg Breeze \vee B_{1,1}$ **(one +ve)** (or)

$(\neg B_{1,1} \vee \neg P_{1,2} \vee \neg P_{2,1})$ **(zero +ve)**

- Horn clause can be rewritten as implications
  - $\neg L_{1,1} \vee \neg Breeze \vee B_{1,1}$ can be rewritten as $(L_{1,1} \wedge Breeze) \Rightarrow B_{1,1}$
- The premise is called the body and the conclusion is called the head
- All definite clauses are Horn clauses

SSN

FORWARD CHAINING

- Given the set of sentences, Prove Q
- Whenever the premises of a rule are satisfied, infer the conclusion

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

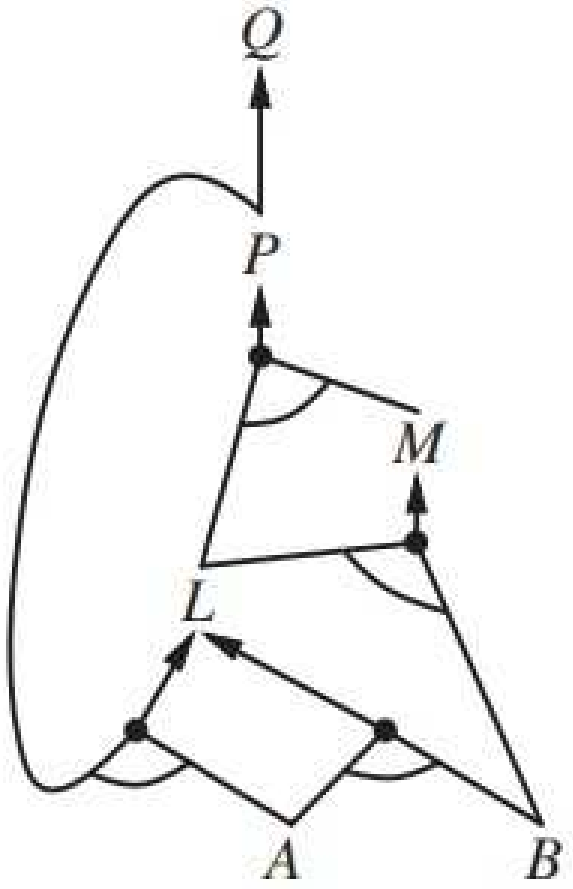
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

(a)



(b)

(a) A set of Horn clauses (b) The corresponding AND–OR graph

FORWARD CHAINING ALGORITHM

```
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 ← a table, where count[c] is initially the number of symbols in clause c's premise
  inferred ← a table, where inferred[s] is initially false for all symbols
  queue ← a queue of symbols, initially symbols known to be true in KB

  while queue is not empty do
      p ← POP(queue)
      if p = q then return true
      if inferred[p] = false then
          inferred[p] ← 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

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

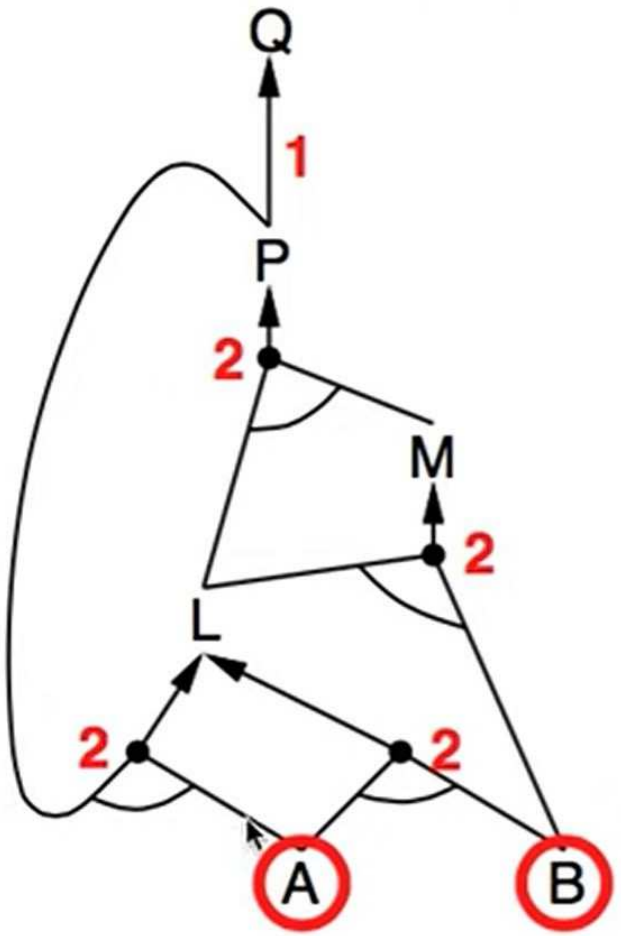
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

(a)



FORWARD CHAINING

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

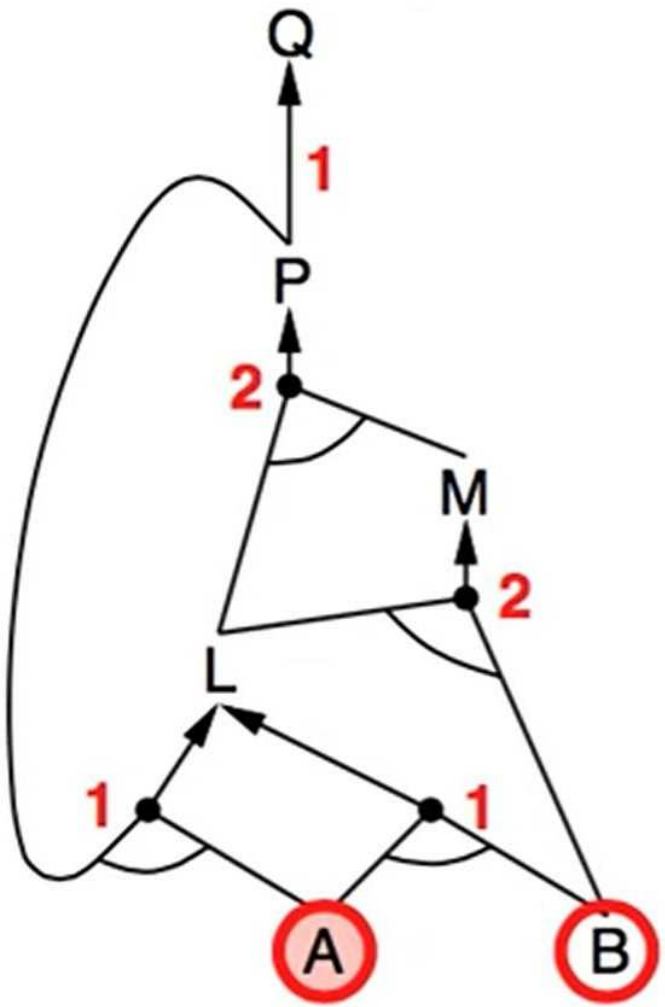
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

(a)



FORWARD CHAINING

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

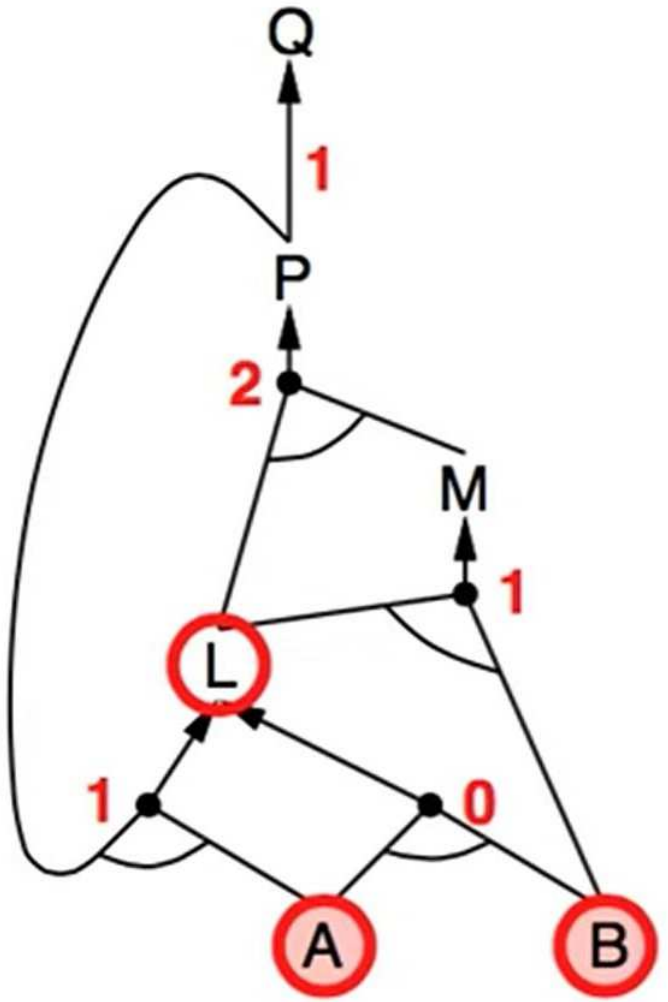
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

(a)



FORWARD CHAINING

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

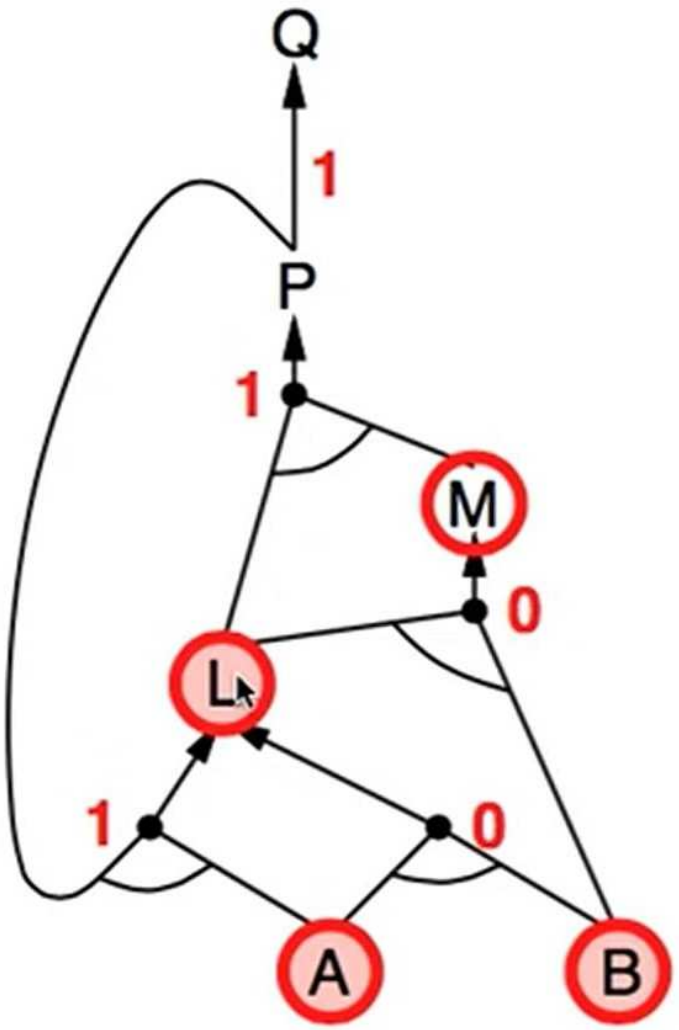
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

(a)



FORWARD CHAINING

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

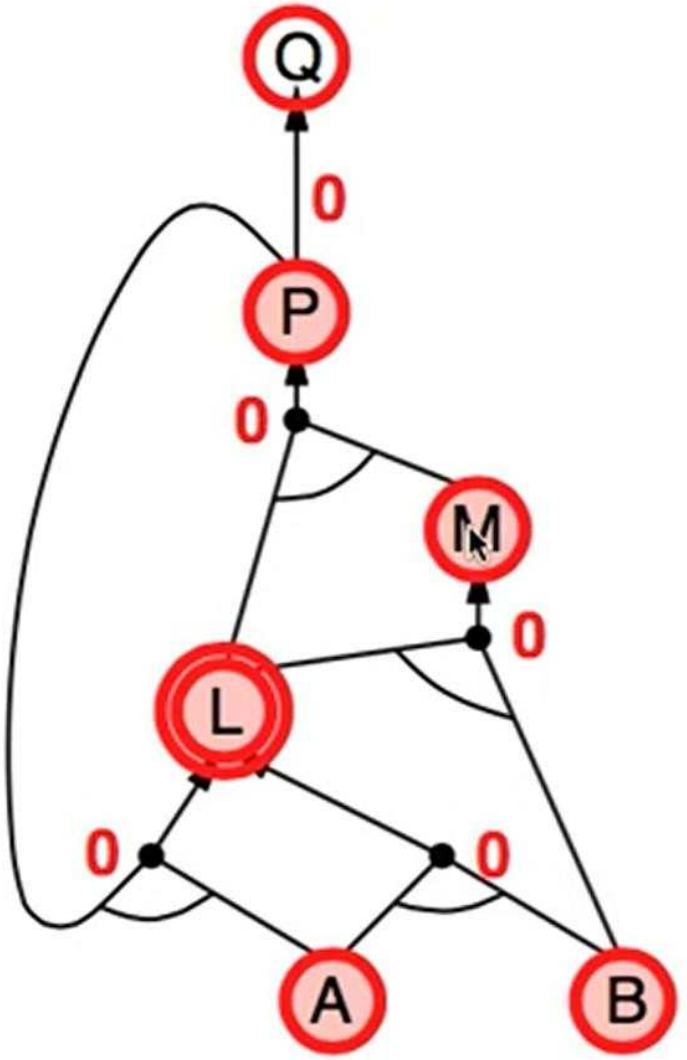
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

(a)



BACKWARD CHAINING

- Given the set of sentences, Prove Q
- Work backwards from the query q: To prove Q by BC, check if Q is already known or prove by C all premises of some rule concluding Q

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

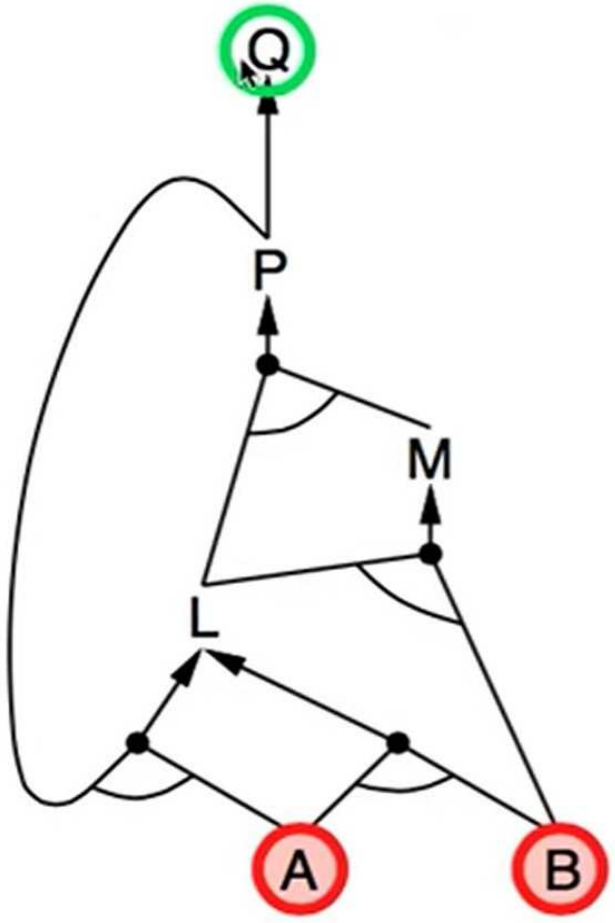
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

(a)



BACKWARD CHAINING

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

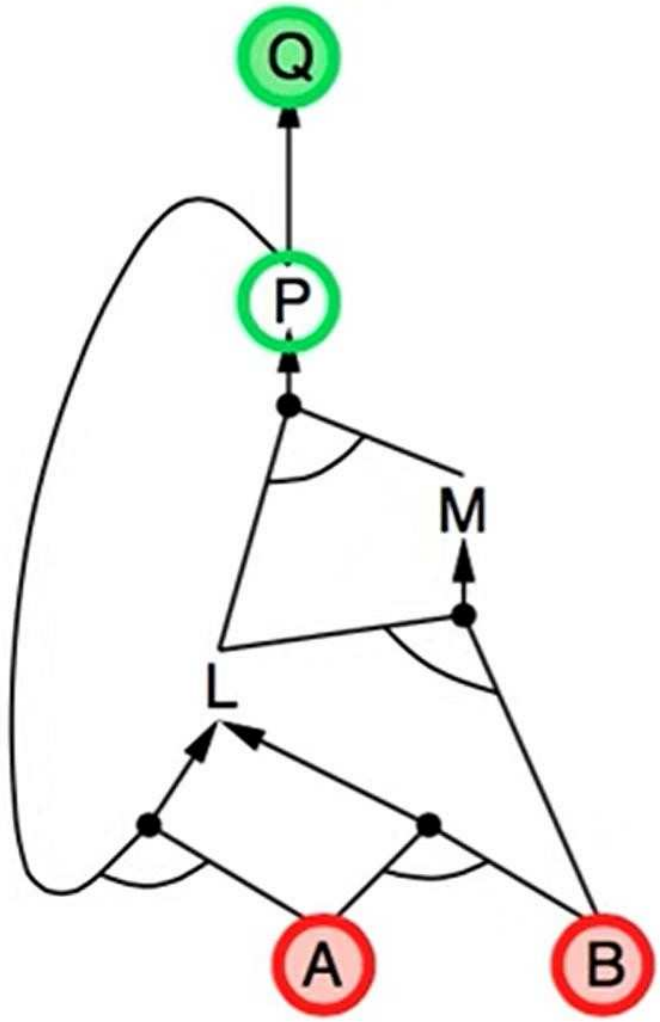
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

(a)



BACKWARD CHAINING

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

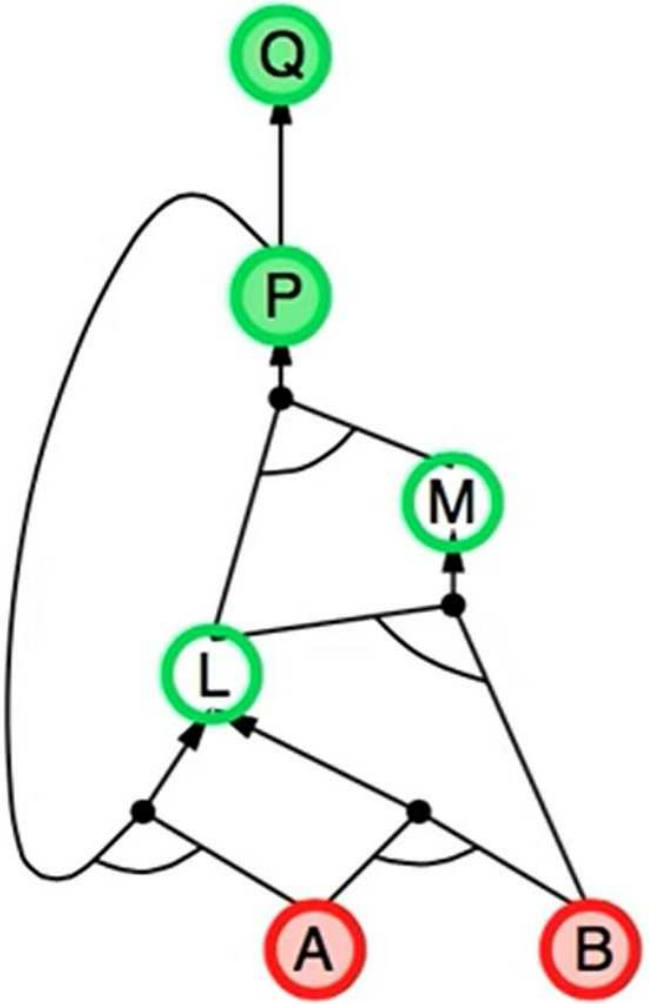
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

(a)



BACKWARD CHAINING

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

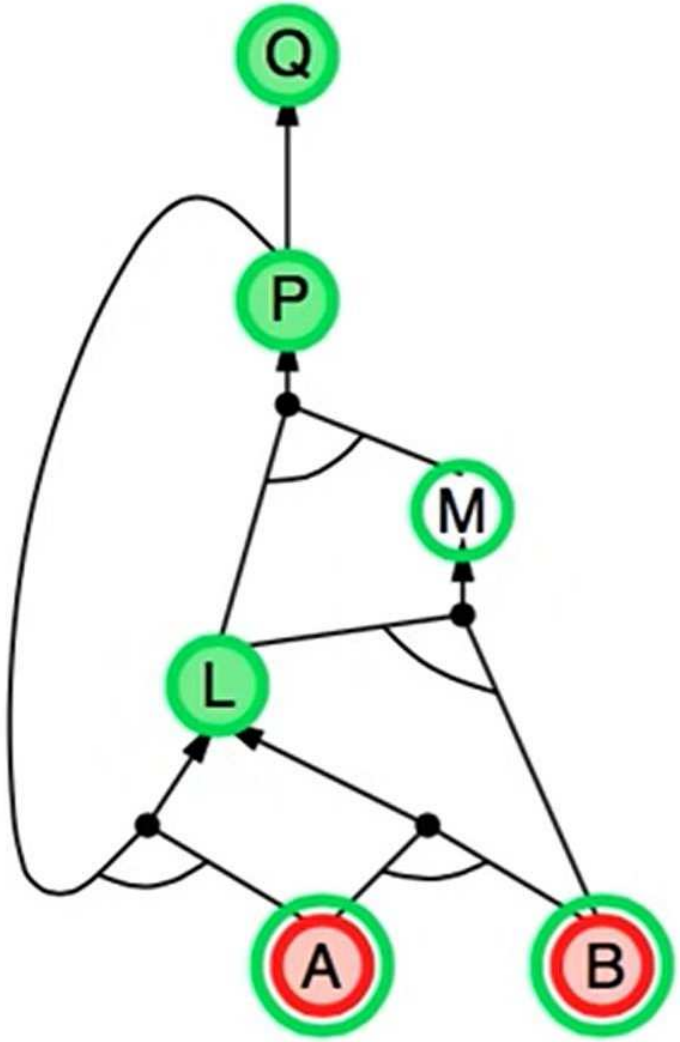
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

(a)



BACKWARD CHAINING

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

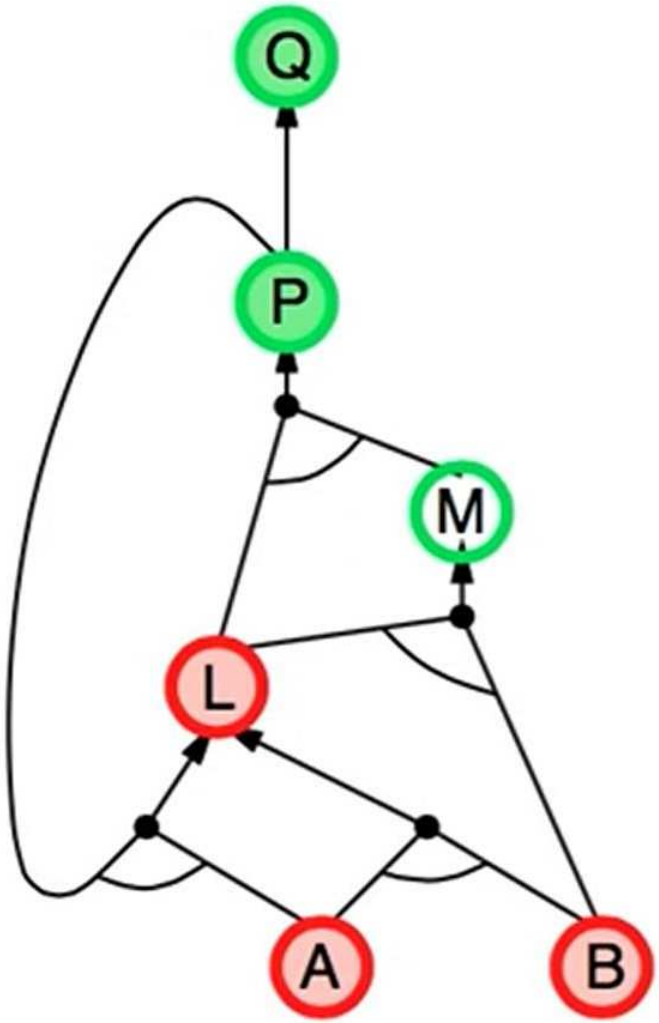
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

(a)



BACKWARD CHAINING

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

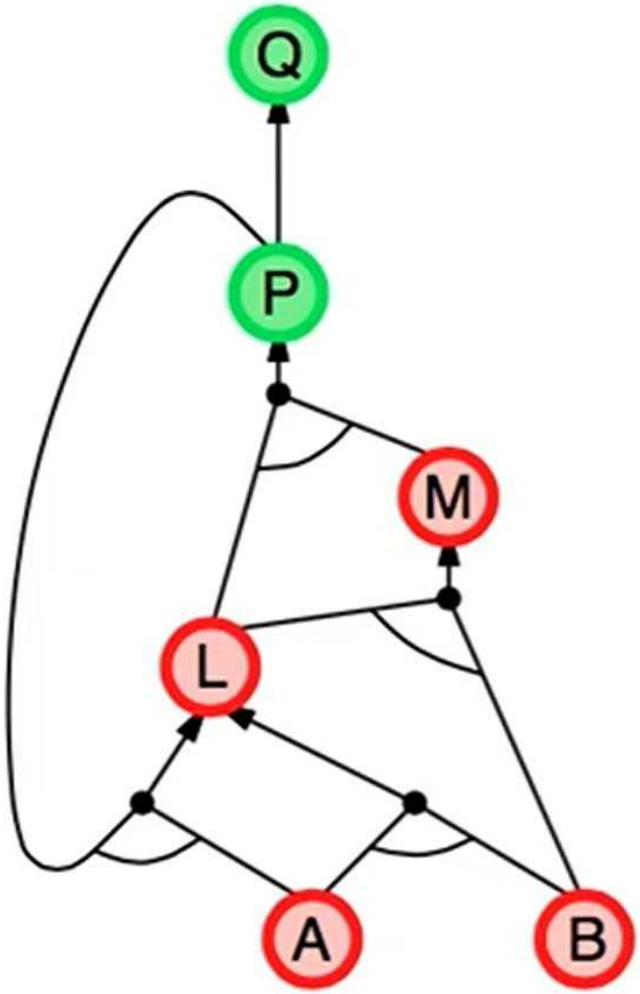
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

(a)



BACKWARD CHAINING

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

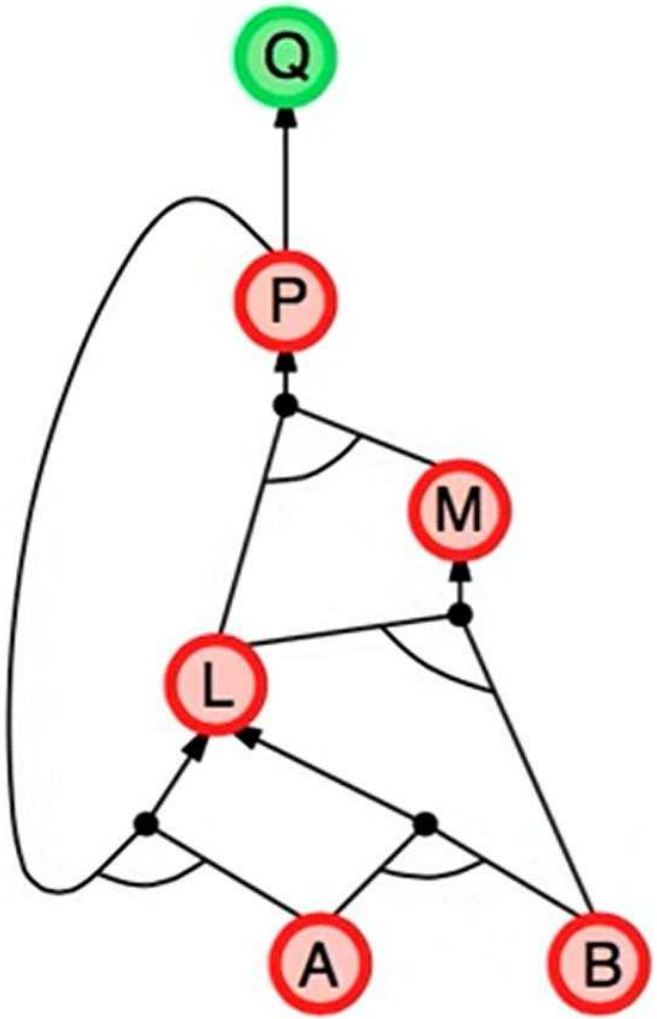
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

(a)



BACKWARD CHAINING

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

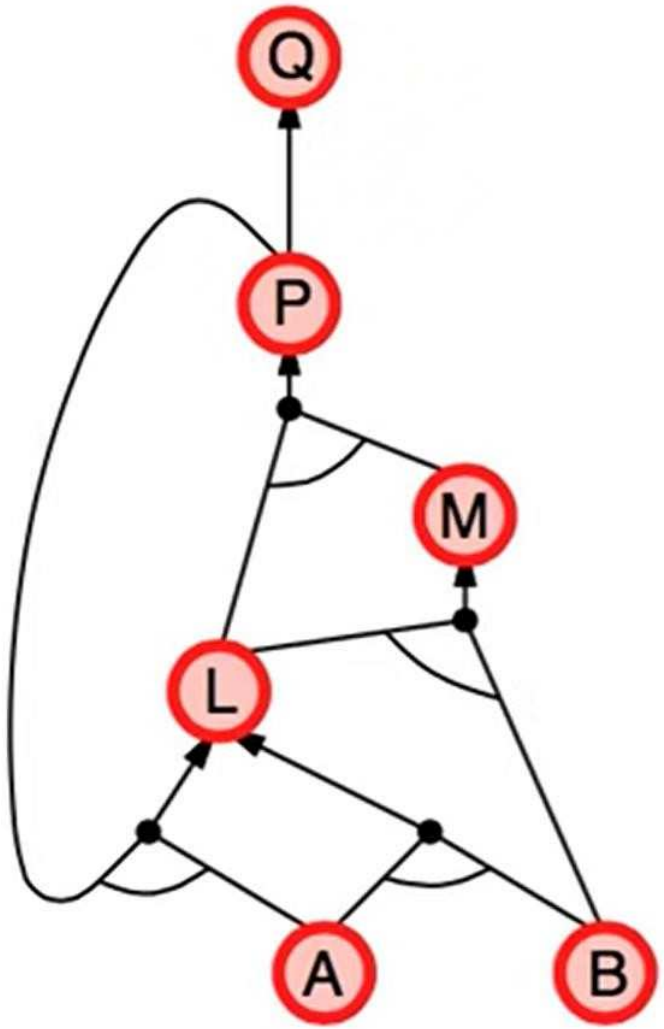
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

(a)



SUMMARY

- Propositional logic is a simple language consisting of proposition symbols and logical connectives.
- It can handle propositions that are known true, known false, or completely unknown.
- Inference rules are patterns of sound inference that can be used to find proofs.
- The resolution rule yields a complete inference algorithm for knowledge bases that are expressed in conjunctive normal form.
- Forward chaining and backward chaining are very natural reasoning algorithms for knowledge bases in Horn form

TEST YOUR KNOWLEDGE

- Given the following, can you prove that the unicorn is mythical? How about magical? Horned?

  If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.

TEST YOUR KNOWLEDGE

- Decide whether each of the following sentences is valid, unsatisfiable, or neither. Verify your decisions using truth tables or the equivalence rule

a. $Smoke \Rightarrow Smoke$
b. $Smoke \Rightarrow Fire$
c. $(Smoke \Rightarrow Fire) \Rightarrow (\neg Smoke \Rightarrow \neg Fire)$
d. $Smoke \vee Fire \vee \neg Fire$
e. $((Smoke \wedge Heat) \Rightarrow Fire) \Leftrightarrow ((Smoke \Rightarrow Fire) \vee (Heat \Rightarrow Fire))$
f. $(Smoke \Rightarrow Fire) \Rightarrow ((Smoke \wedge Heat) \Rightarrow Fire)$
g. $Big \vee Dumb \vee (Big \Rightarrow Dumb)$

7.18 Consider the following sentence:

$$[(Food \Rightarrow Party) \vee (Drinks \Rightarrow Party)] \Rightarrow [(Food \wedge Drinks) \Rightarrow Party] .$$

a. Determine, using enumeration, whether this sentence is valid, satisfiable (but not valid), or unsatisfiable.
b. Convert the left-hand and right-hand sides of the main implication into CNF, showing each step, and explain how the results confirm your answer to (a).
c. Prove your answer to (a) using resolution.