

Logical Agent & Propositional Logic

Berlin Chen 2005

References:

1. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Chapter 7
2. S. Russell's teaching materials


Introduction

- The representation of *knowledge* and the processes of *reasoning* will be discussed
 - Important for the design of artificial agents
 - Reflex agents
 - Rule-based, table-lookup
 - Problem-solving agents
 - Problem-specific and inflexible
 - Knowledge-based agents
 - Flexible
 - Combine knowledge with current percepts to infer hidden aspects of the current state prior to selecting actions
 - Logic is the primary vehicle for knowledge representation
 - Reasoning copes with different infinite variety of problem states using a finite store of knowledge


Introduction (cont.)

- Example: natural language understanding

John saw **the diamond** through **the window** and coveted **it**



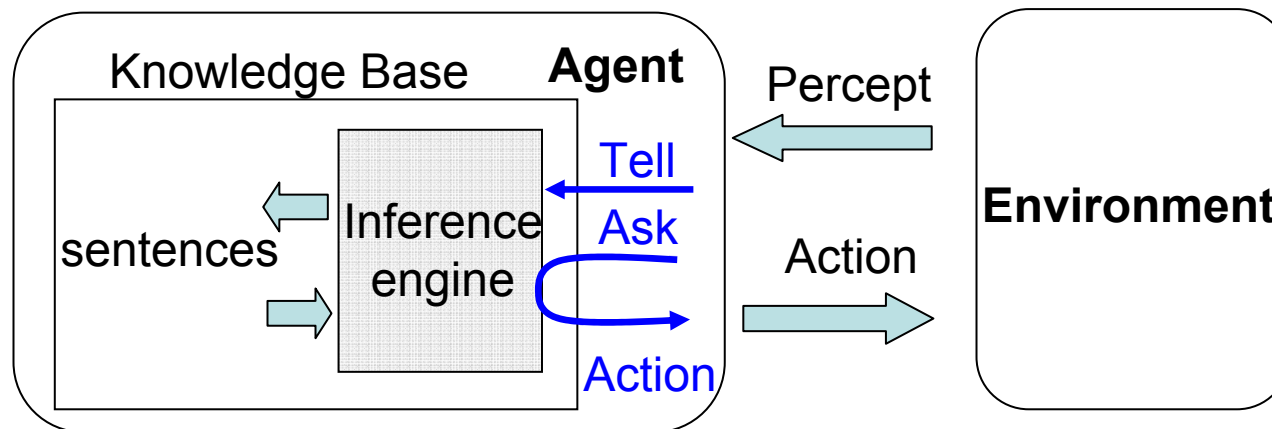
John threw **the brick** through **the window** and broke **it**



Knowledge-Based Agents

- Knowledge base (background knowledge)
 - A set of sentences of formal (or knowledge representation) language
 - Represent facts (assertions) about the world
 - Sentences have their syntax and semantics
- Declarative approach to building an agent
 - Tell: tell it what it needs to know (add new sentences to KB)
 - Ask: ask itself what to do (query what is known)

is a declarative approach



- Inference
 - Derive new sentences from old ones

Knowledge-Based Agents (cont.)

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*))

← extensive reasoning
may be taken here

TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))

t ← *t* + 1

return *action*

- KB initially contains some background knowledge
- Each time the agent function is called **the internal state**
 - It **Tells** KB what it perceives
 - It **Asks** KB what action it should perform
- Once the action is chosen
 - The agent records its choice with **Tell** and executes the action

Knowledge-Based Agents (cont.)

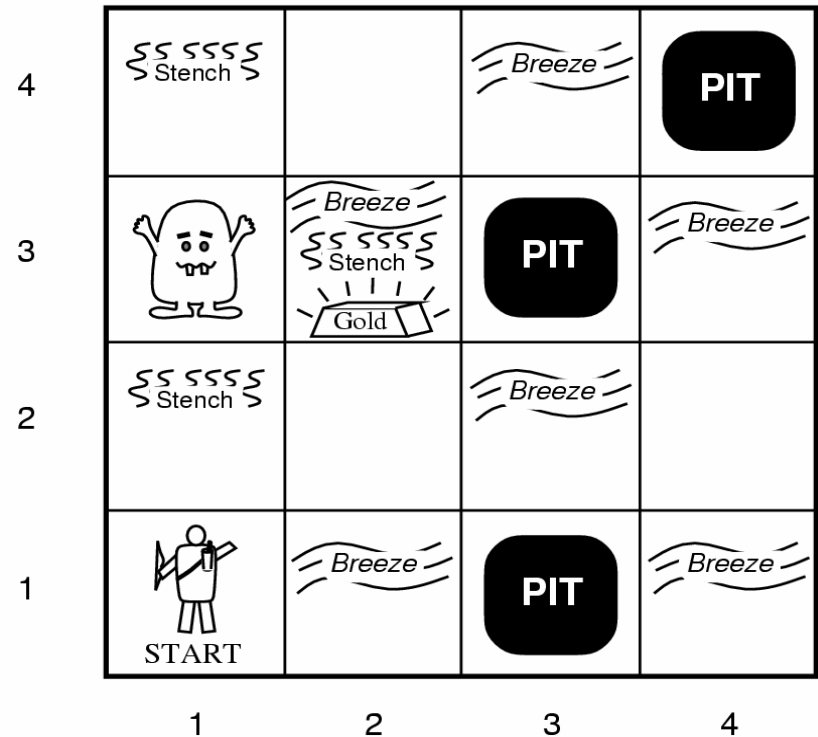
- Agents can be viewed at knowledge level
 - What they know, what the goals are, ...
- Or agents can be viewed at the implementation level
 - The data structures in KB and algorithms that manipulate them
- In summary, the agents must be able to
 - Represent states, actions, etc.
 - Incorporate new percepts
 - Update internal representations of the world
 - Deduce hidden properties of the world
 - Deduce appropriate actions

Wumpus World

- Wumpus world was an early computer game, based on an agent who explores a cave consisting of rooms connected by passageways
- Lurking somewhere in the cave is the wumpus, a beast that eats anyone who enters a room
- Some rooms contain bottomless pits that will trap anyone who wanders into these rooms (except the wumpus, who is too big to fall in)
- The only mitigating features of living in the environment is the probability of finding a heap of gold

Wumpus World PEAS Description

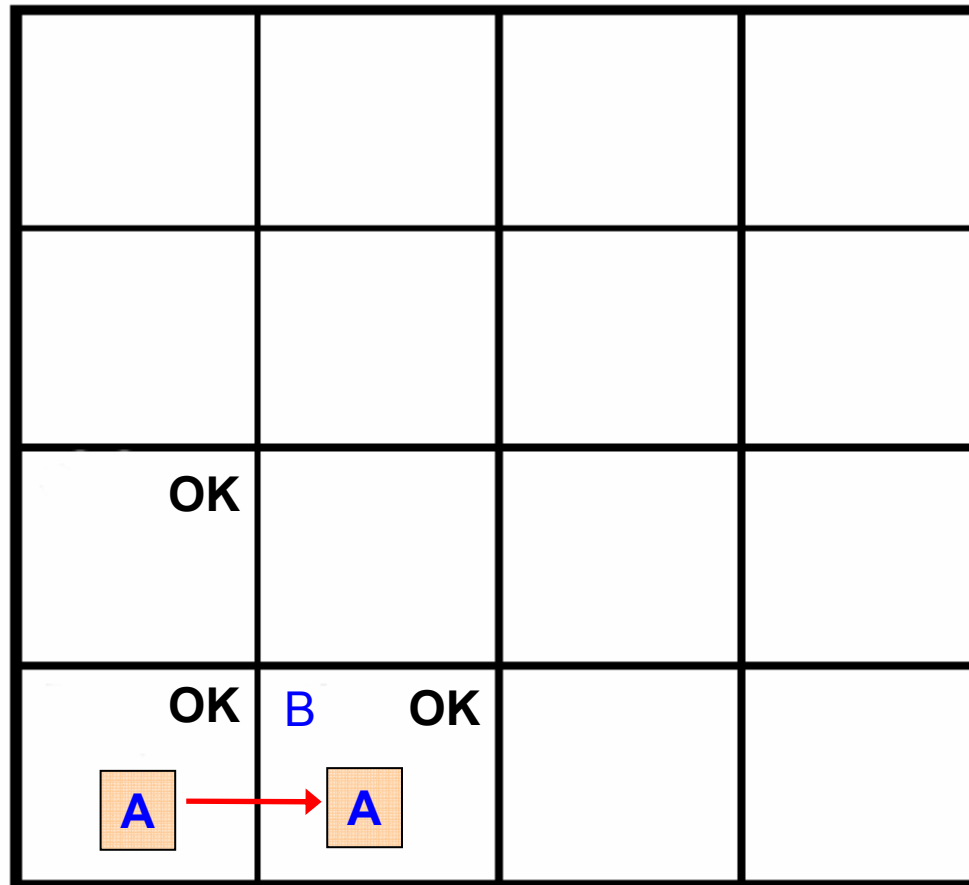
- Performance measure
 - gold +1000, death -1000, -1 per step, -10 for using the arrow
- Environment
 - Squares adjacent to wumpus are smelly
 - Squares adjacent to pits are breezy
 - Glitter if gold is in the same square
 - Shooting kills wumpus if you are facing it
 - Shooting uses up the only one arrow
 - Grabbing picks up gold if in same square
 - Releasing drops the gold in same square
- Actuators
 - Forward, Turn Right, Turn Left, Grab, Release, Shoot
- Sensors
 - Breeze, Glitter, Smell, ...



Wumpus World Characterization

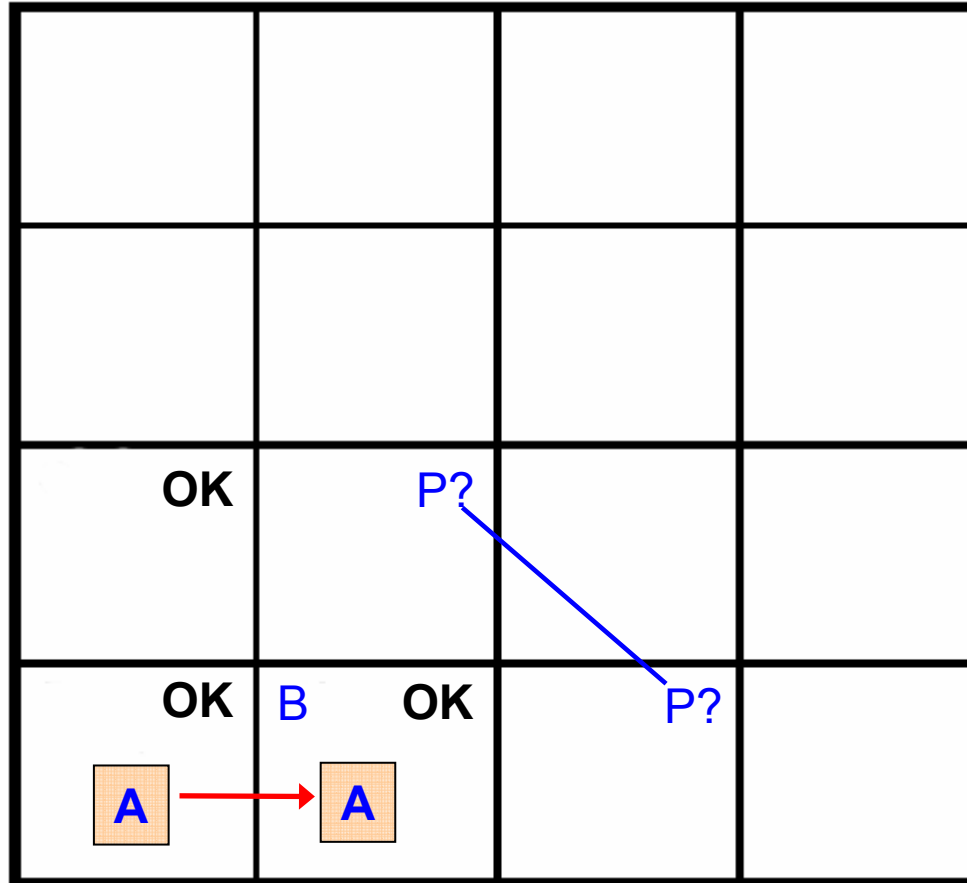
- **Observable??** No --- only local perception
- **Deterministic??** Yes --- outcomes exactly specified
- **Episodic??** No --- sequential at the level of actions
- **Static??** Yes --- Wumpus and pits can not move
- **Discrete??** Yes
- **Single-agent??** Yes --- Wumpus is essentially a nature feature

Exploring a Wumpus World (cont.)

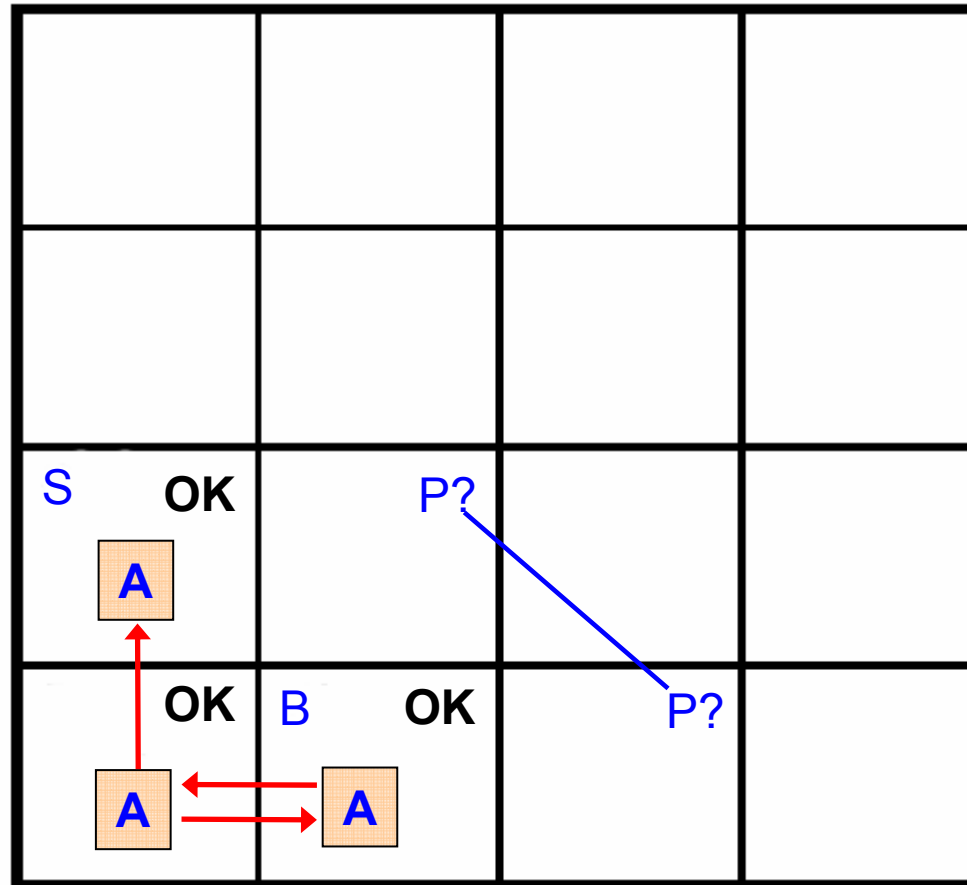


- After the first move, with percept
[None, Breeze, None, None, None]

Exploring a Wumpus World (cont.)

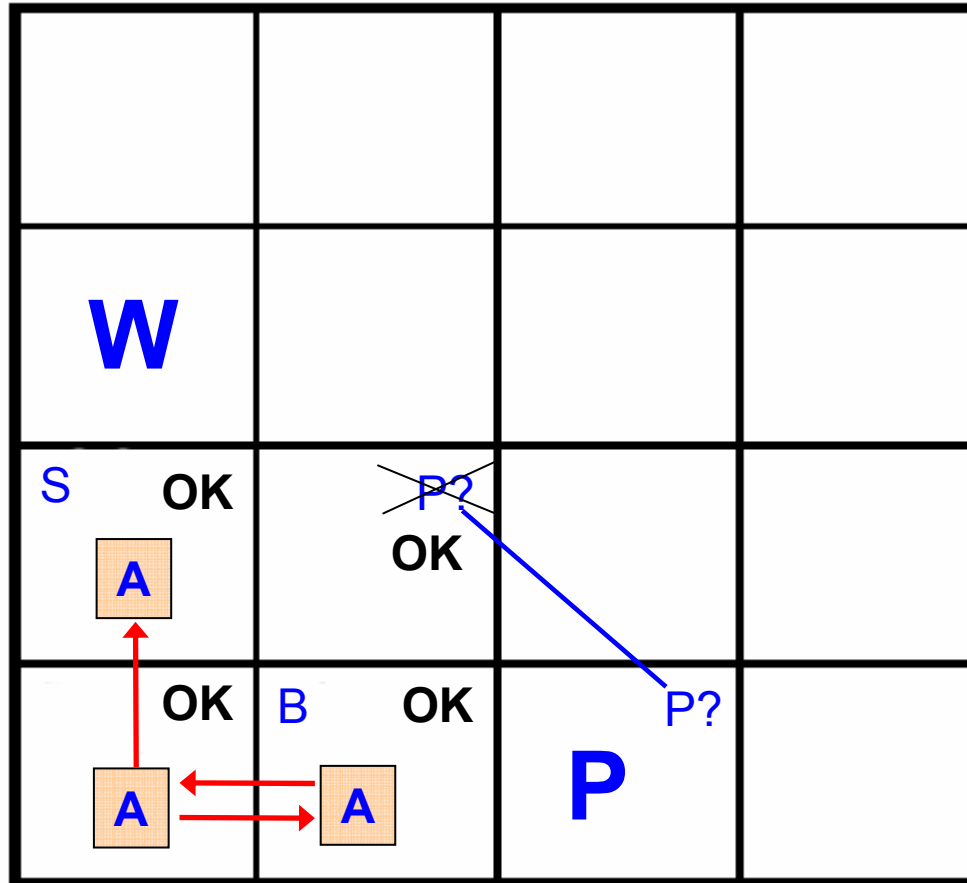


Exploring a Wumpus World (cont.)

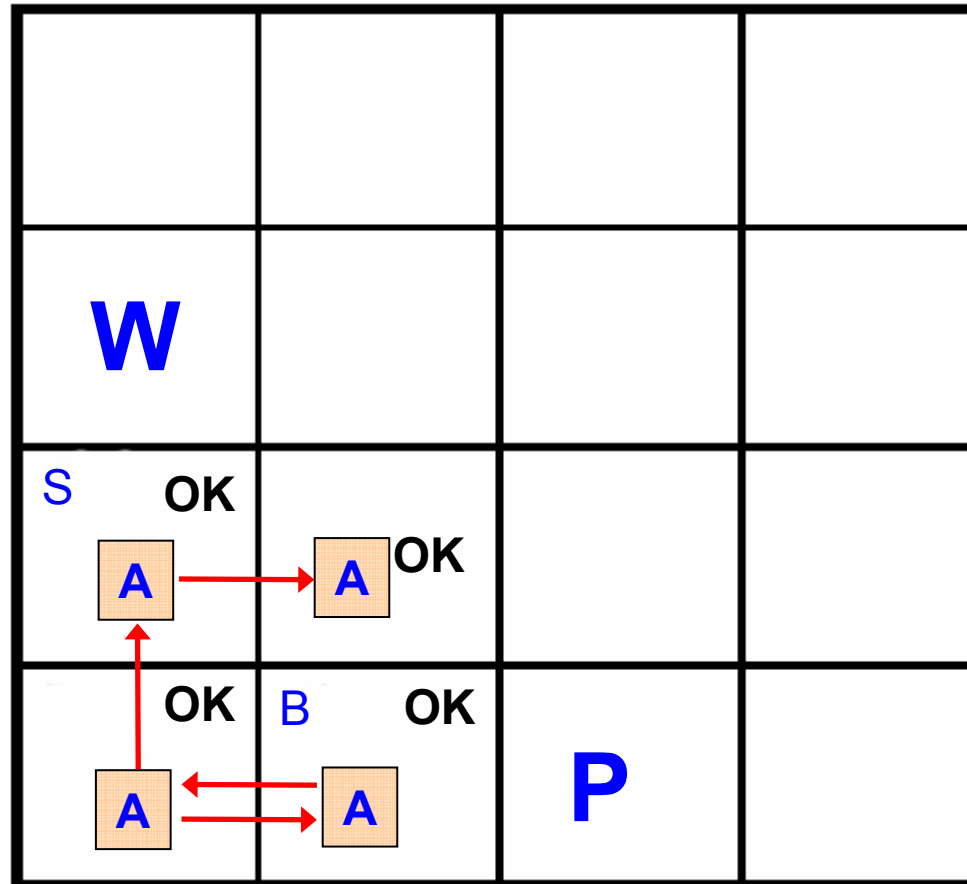


- After the third move, with percept
[Stench, None, None, None, None]

Exploring a Wumpus World (cont.)

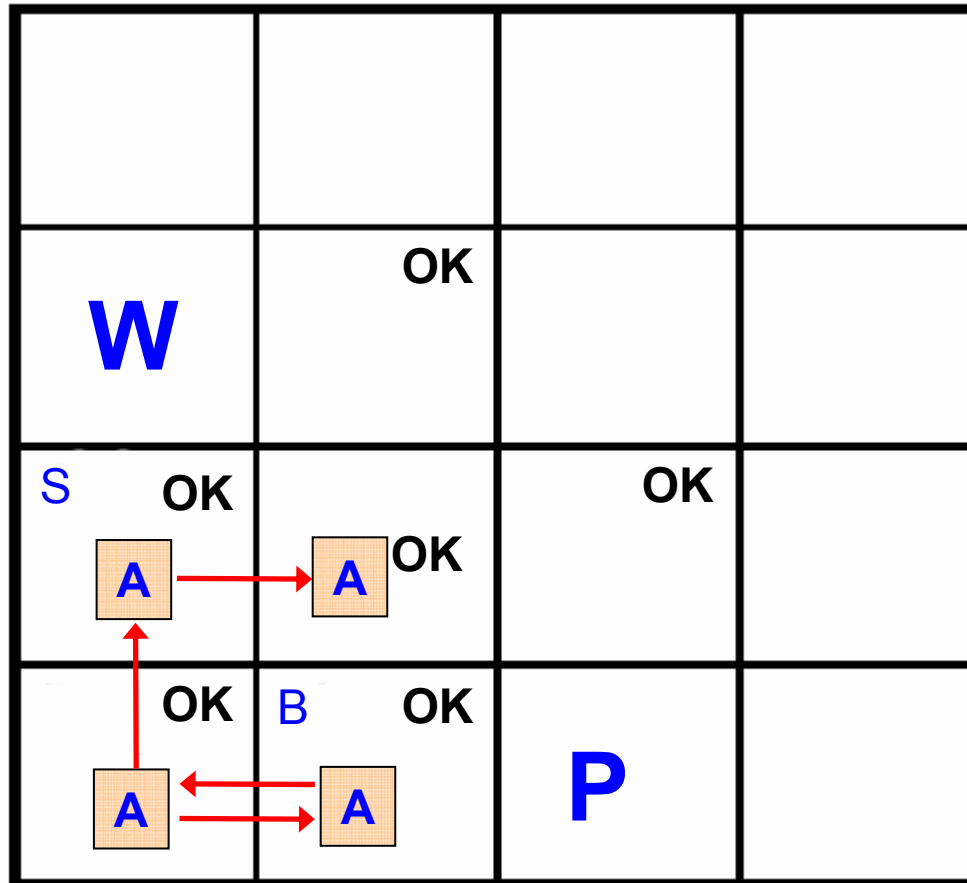


Exploring a Wumpus World (cont.)

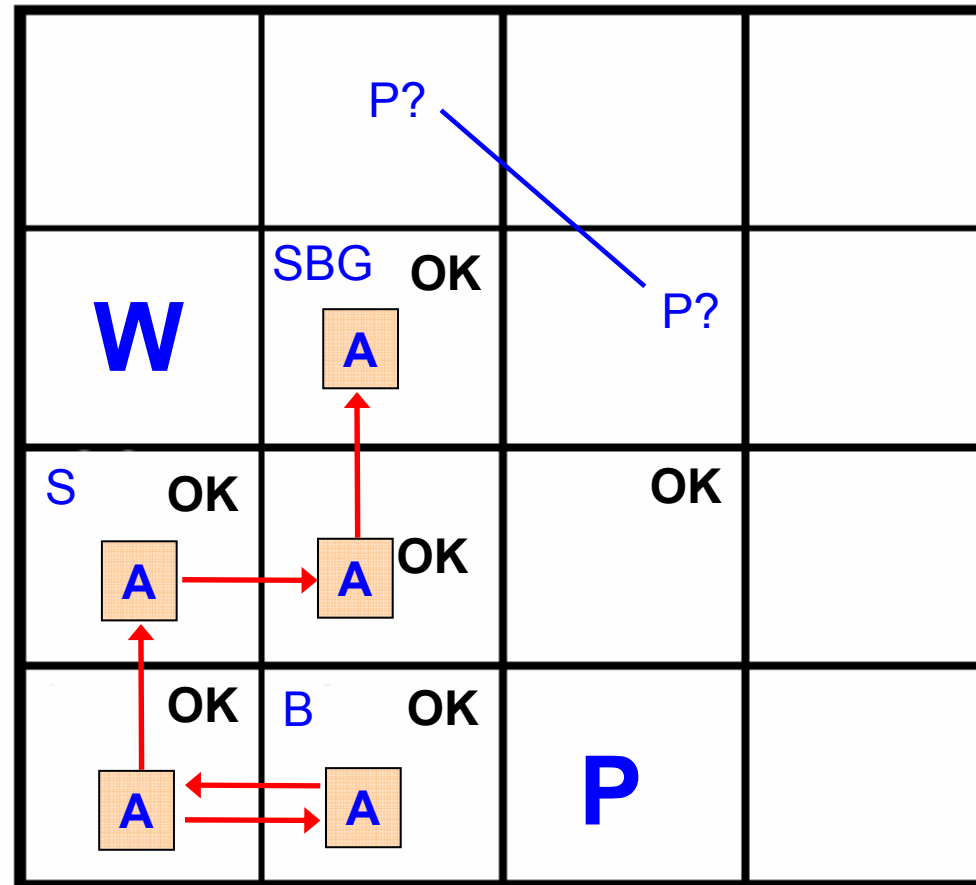


- After the fourth move, with percept
[None, None, None, None, None]

Exploring a Wumpus World (cont.)

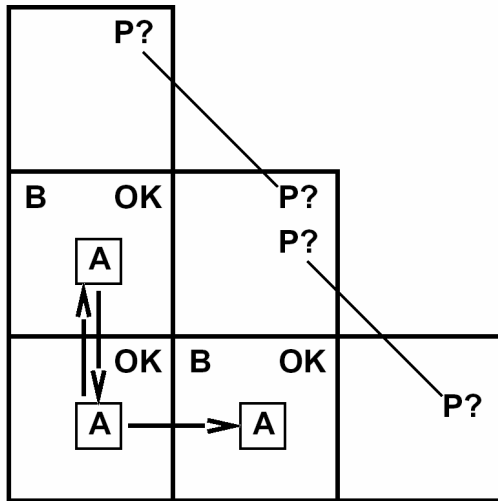


Exploring a Wumpus World (cont.)



- After the fifth move, with percept
[Stench, Breeze, Glitter, None, None]

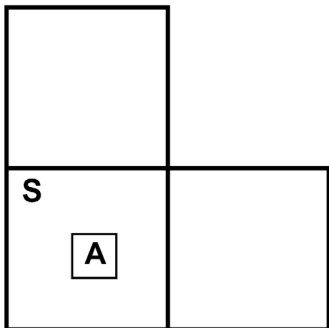
Other Tight Spots



Breeze in (1,2) and (2,1)
 ⇒ No safe actions

Smell in (1,1)
 ⇒ Cannot move

Can use a strategy of coercion
 shot straight ahead
 wumpus there → dead → safe
 wumpus wasn't there → safe



Logic in General

- Logics are formal languages for representing information such that conclusions can be drawn
- **Syntax** defines the sentences in the language
- **Semantics** define the “meaning” of sentences; i.e., define truth or falsity of a sentence in a world
- E.g., the language of arithmetic
 - $x+2 \geq y$ is a sentence; $x^2+y >$ is not a sentence
 - $x+2 \geq y$ is true iff the number $x+2$ is no less than the number y
 - $x+2 \geq y$ is true **in a world** where $x=7, y=1$
 - $x+2 \geq y$ is false **in a world** where $x=0, y=6$
- Sentences in an agent’s KB are real physical configurations of it

The term “model” will be used to replace the term “world”

Entailment

- **Entailment** means that one thing **follows from** another:

$$KB \models \alpha$$

- Knowledge base KB entails sentence α if α is true in all worlds where KB is true
 - E.g., the KB containing “the Giants won” and “the Reds won” entails “either the Giants or the Red won”
 - E.g., $x+y=4$ entails $4=x+y$
 - The knowledge base can be considered as a statement
- Entailment is a relationship between sentences (i.e., syntax) that is based on semantics
 - E.g., $\alpha \models \beta$
 - α entails β
 - $\alpha \models \beta$ iff in every model in which α is true, β is also true
 - Or, if α is true, β must be true

Models

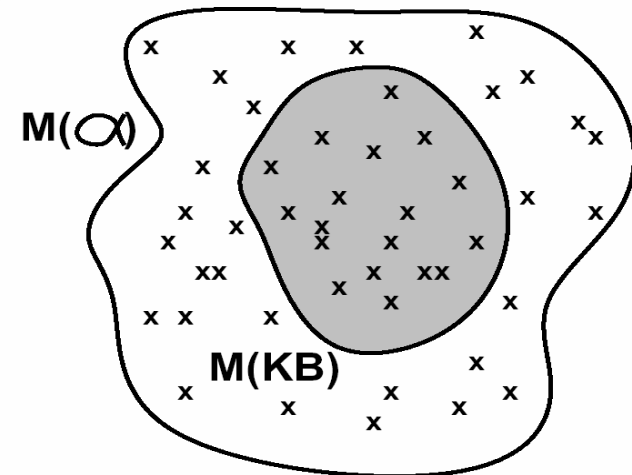
- Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated

m is a model of a sentence α iff α is true in m

- IF $M(\alpha)$ is the set of all models of α

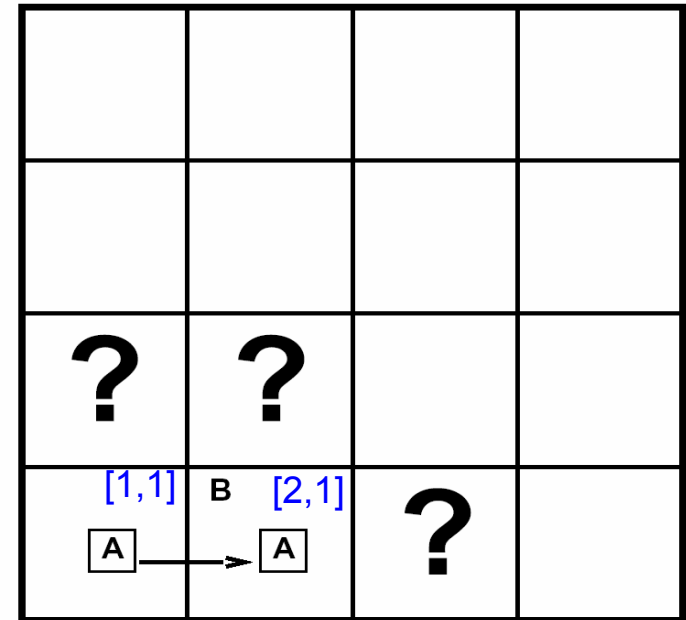
Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$

- I.e., every model in which KB is true, α is also true
- On the other hand, not every model in which α is true, KB is also true



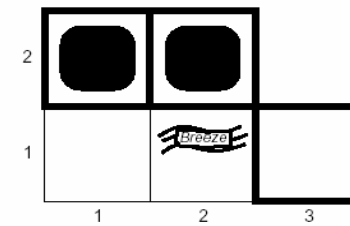
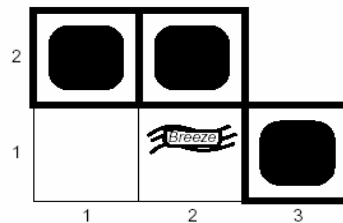
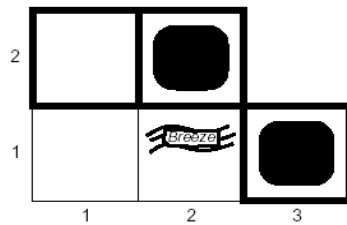
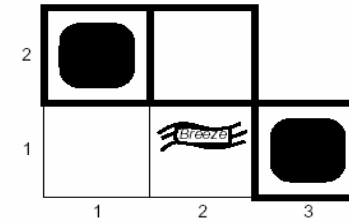
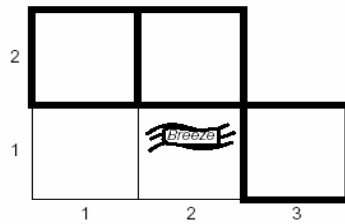
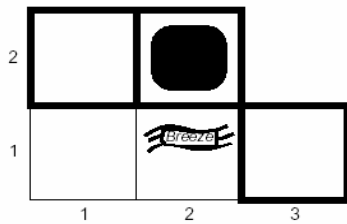
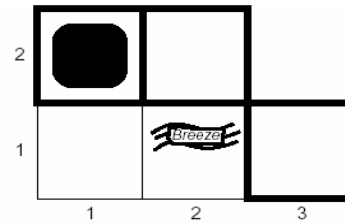
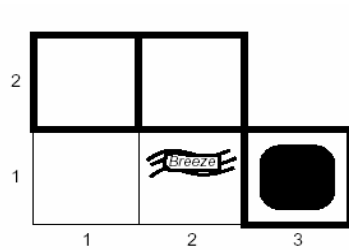
Entailment in the Wumpus World

- Situation after detecting nothing in [1,1], moving right, breeze in [2,1]
- Consider possible models for ?s assuming only pits
- 3 Boolean choices \Rightarrow 8 possible models



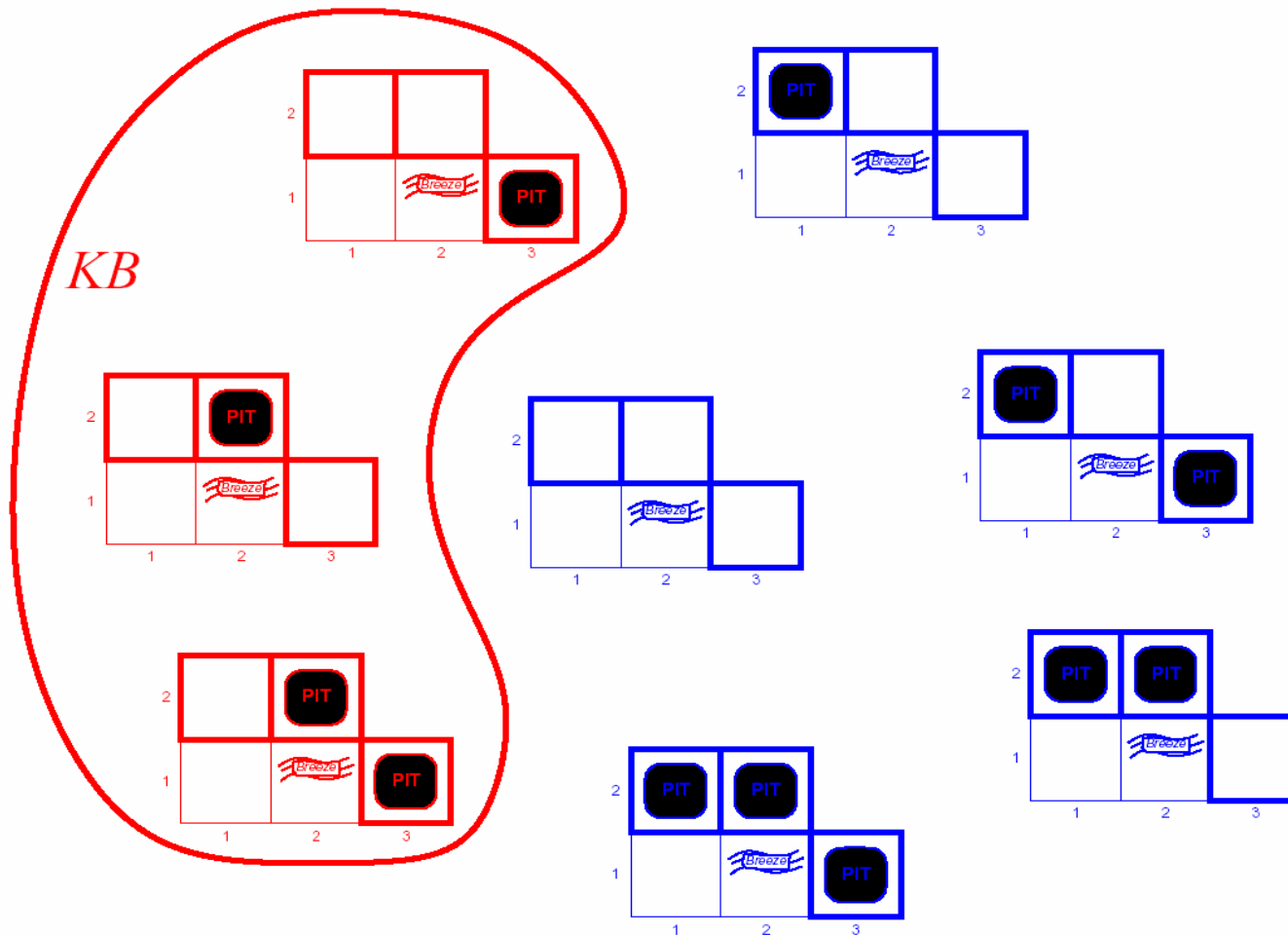
Wumpus Models

- 8 possible models



Wumpus Models (cont.)

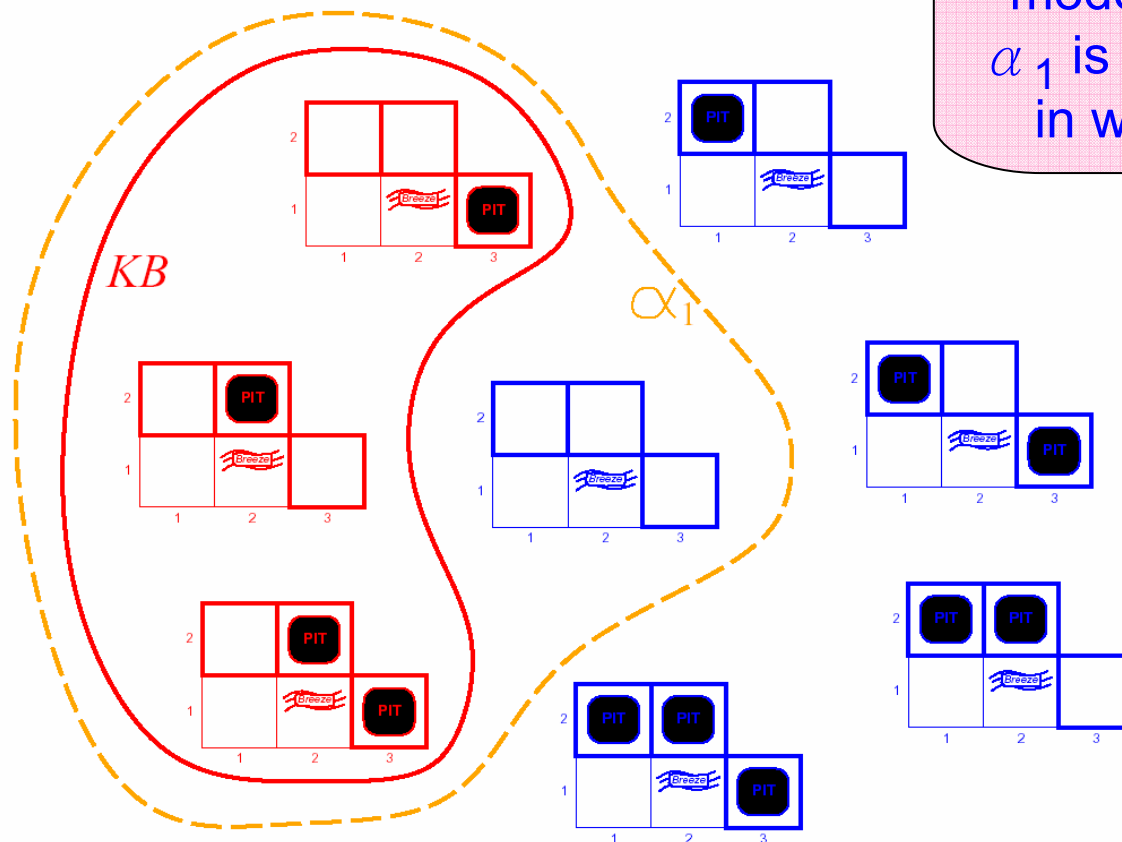
- $KB = \text{wumpus world-rules} + \text{observations}$



Wumpus Models (cont.)

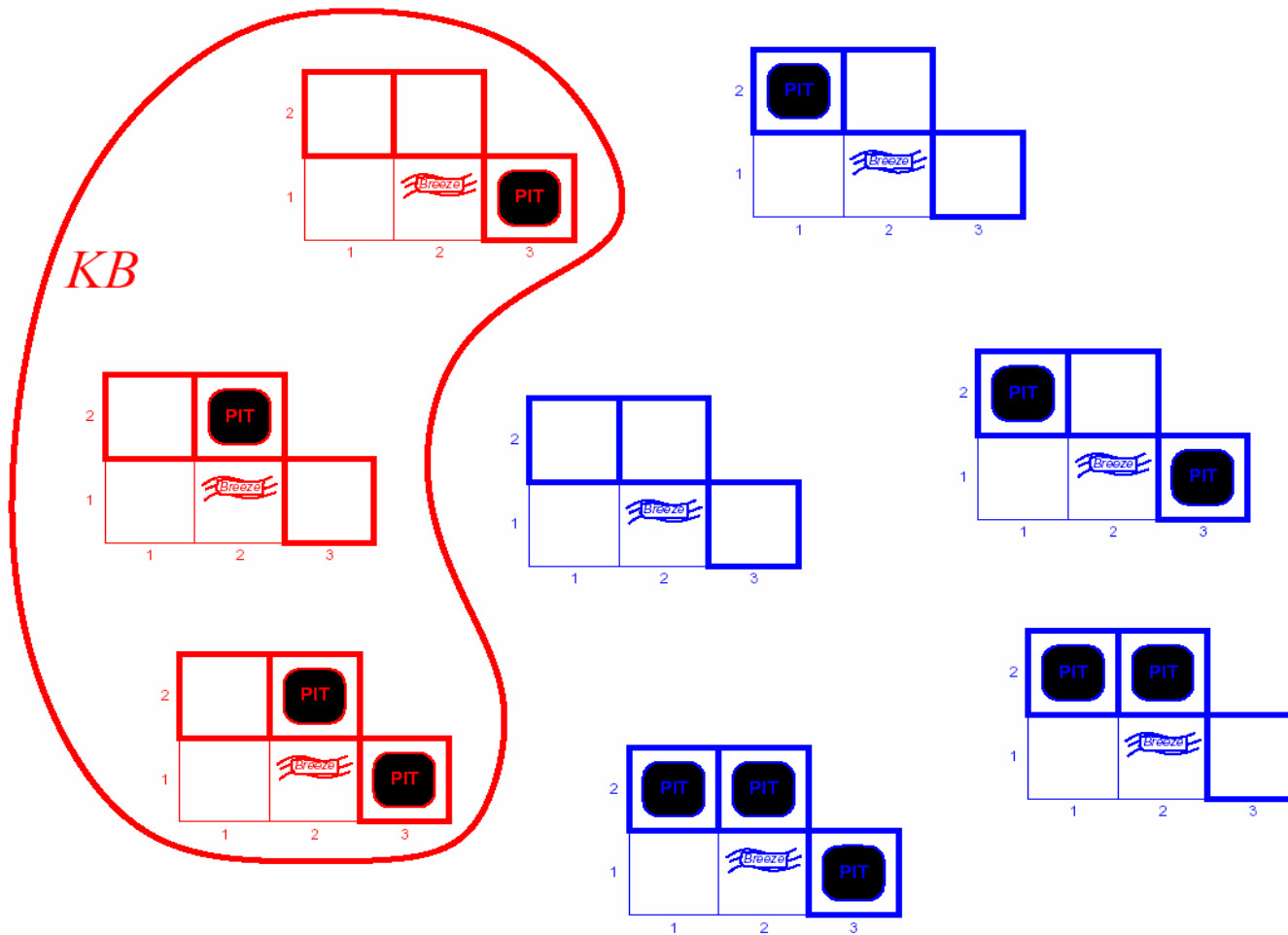
- KB = wumpus world-rules + observations
 - α_1 = “[1,2] is safe”
 - $KB \models \alpha_1$, proved by **model checking**

enumerate all possible models to check that α_1 is true in all models in which KB is true



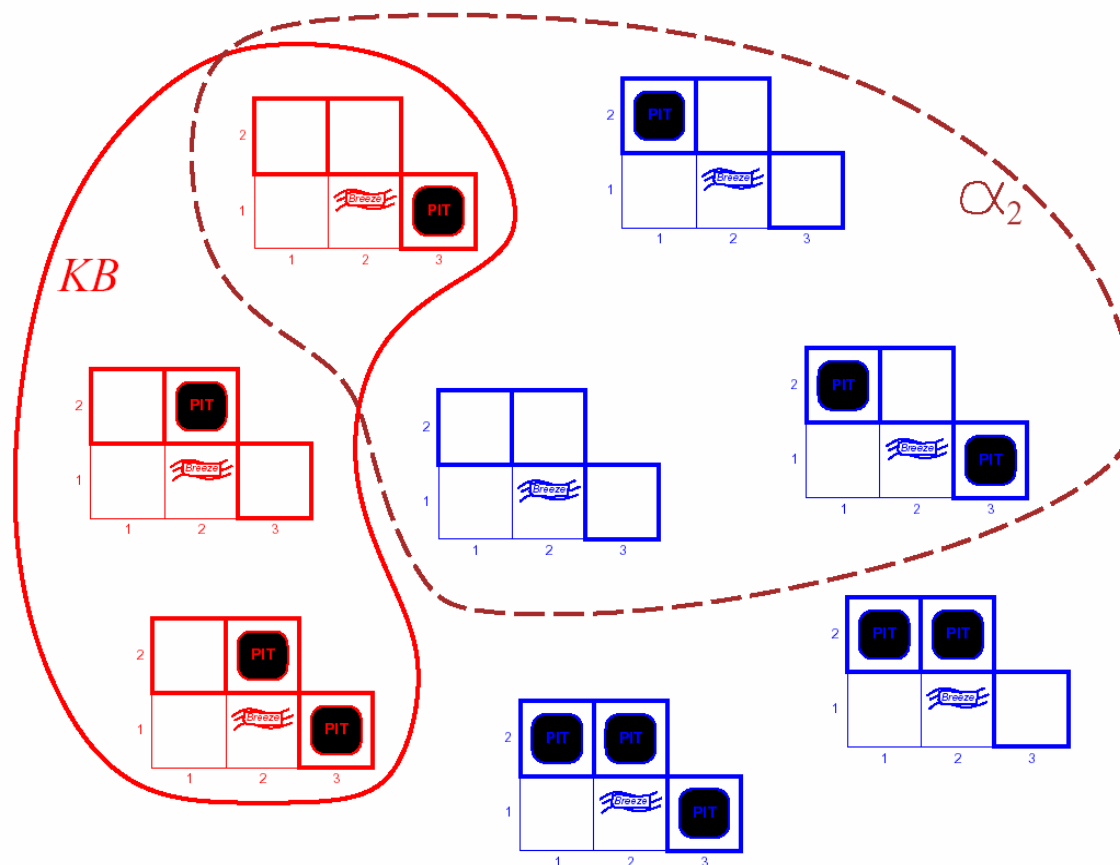
Wumpus Models (cont.)

- KB = wumpus world-rules + observations



Wumpus Models (cont.)

- KB = wumpus world-rules + observations
 - α_2 = “[2,2] is safe”
 - $KB \models \alpha_2$, proved by **model checking**



Inference

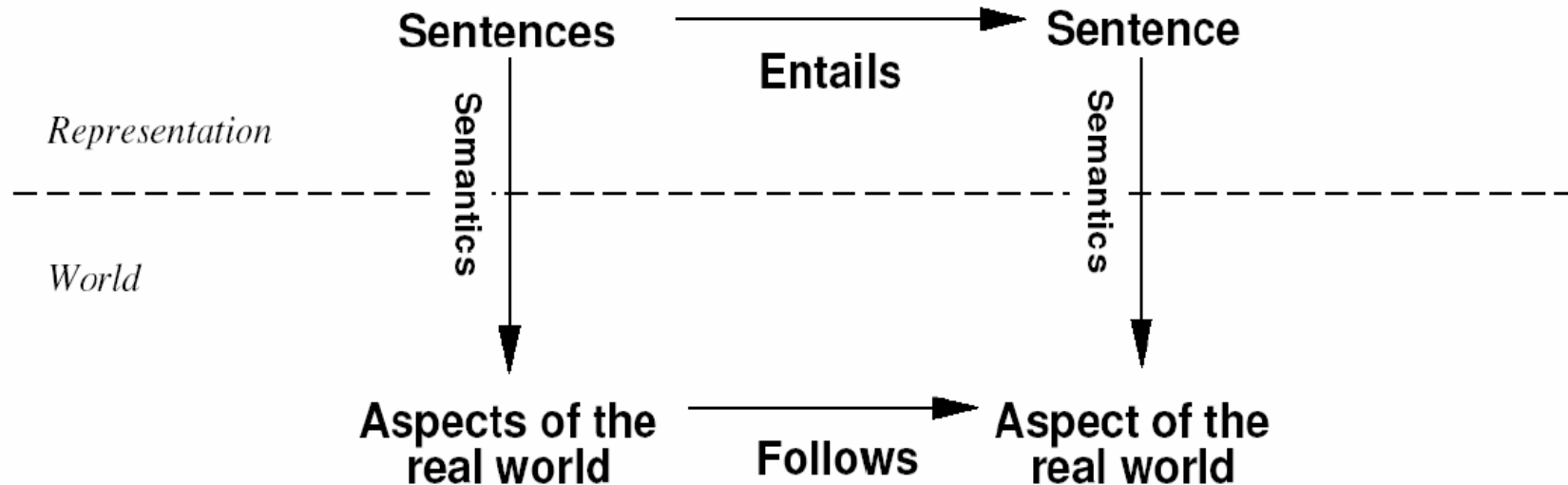
- $KB \vdash_i \alpha$
 - Sentence α can be derived from KB by inference algorithm i
 - Think of
 - the set of all consequences of KB as a **haystack**
 - α as a **needle**
 - entailment** like **the needle in the haystack**
 - inference** like **finding it**
- **Soundness** or truth-preserving inference
 - An algorithm i is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
 - **That is the algorithm derives only entailed sentences**
 - The algorithm won't announce “the discovery of nonexistent needles”

Inference (cont.)

- **Completeness**

- An algorithm i is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$
- A sentence α will be generated by an inference algorithm i if it is entailed by the KB
- Or says, the algorithm will answer any question whose answer follows from what is known by the KB

Inference (cont.)



- Sentences are physical configurations of the agent, and reasoning is a process of constructing new physical configurations from old ones
- Logical reasoning should ensure that the new configurations represent aspects of the world that actually follow from the aspects that the old configurations represent

Propositional Logic: Syntax

- Propositional logic is the simplest logic that illustrates basic ideas
- **Syntax**: defines the allowable sentences
 - Atomic sentences consist of a single propositional symbols
 - Propositional symbols: e.g., P , Q and R
 - Each stands for a proposition (fact) that can be either true or false
 - Complex sentences are constructed from simpler one using logic connectives
 - \wedge (and) conjunction
 - \vee (or) disjunction
 - \Rightarrow (implies) implication
 - \Leftrightarrow (equivalent) equivalence, or biconditional
 - \neg (not) negation

Propositional Logic: Syntax (cont.)

- **BNF** (Backus-Naur Form) grammar for propositional logic

Sentence \rightarrow *Atomic Sentence* | *Complex Sentence*

Atomic Sentence \rightarrow True | False | *Symbol*

Symbol \rightarrow *P* | *Q* | *R* ...

Complex Sentence \rightarrow \neg *Sentence*

| (*Sentence* \wedge *Sentence*)

| (*Sentence* \vee *Sentence*)

| (*Sentence* \Rightarrow *Sentence*)

| (*Sentence* \Leftrightarrow *Sentence*)

- **Order of precedence:** (from highest to lowest)

\neg , \wedge , \vee , \Rightarrow , and \Leftrightarrow

– E.g., $\neg P \vee Q \wedge R \Rightarrow S$ means $((\neg P) \vee (Q \wedge R)) \Rightarrow S$

$A \Rightarrow B \Rightarrow C$ is not allowed !

Propositional Logic: Semantics

- Define the rules for determining the truth of a sentence with respect to a particular model
 - Each model fixes the truth value (**true** or **false**) for every propositional symbol
 - E.g., $P_{1,2}$ $P_{2,2}$ $P_{3,1}$
 - 3 symbols, 8 possible models, can be enumerated automatically
 - A possible model $m_1 \{P_{1,2} = \text{false}, P_{2,2} = \text{false}, P_{3,1} = \text{true}\}$
 - Simple recursive process evaluates an arbitrary sentence, e.g.,

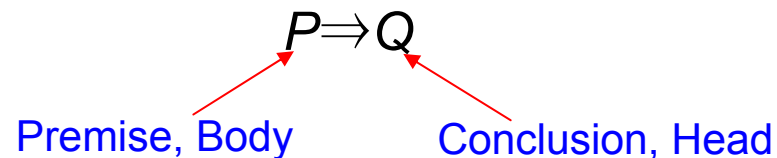
$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \text{true} \wedge (\text{false} \vee \text{true}) = \text{true} \wedge \text{true} = \text{true}$$

Models for PL are just sets of truth values for the propositional symbols

Truth Tables for Connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>
<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>
<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>

$\neg P$ is true iff P is false
 $P \wedge Q$ is true iff P is true and Q is true
 $P \vee Q$ is true iff P is true or Q is true
 $P \Rightarrow Q$ is false iff P is true and Q is false
 $P \Leftrightarrow Q$ is true iff $P \Rightarrow Q$ is true and $Q \Rightarrow P$ is true



Wumpus World Sentences

- Let $P_{i,j}$ be true if there is a pit in $[i, j]$
- Let $B_{i,j}$ be true if there is a breeze in $[i, j]$
- A square is breezy *if only if* there is an adjacent pit

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

no pit in $[1,1]$

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

pits cause breezes in adjacent squares

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

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

no breeze in $[1,1]$

$$R_5: B_{2,1}$$

breeze in $[2,1]$

- Note: there are 7 proposition symbols involved
 - $B_{1,1}, B_{2,1}, P_{1,1}, P_{1,2}, P_{2,1}, P_{2,2}, P_{3,1}$
 - There are $2^7=128$ models !
 - While only three of them satisfy the above 5 descriptions/sentences

Truth Tables for Inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>
<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<u><i>true</i></u>	<u><i>true</i></u>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<u><i>true</i></u>	<u><i>true</i></u>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<u><i>true</i></u>	<u><i>true</i></u>
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>

128 models

$$R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$$

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

- $P_{2,2}$?

Conjunction of sentences of KB

Inference by Enumeration

Test if KB is true α is also true

```
function TT-ENTAILS?(KB,  $\alpha$ ) returns true or false  
inputs: KB, the knowledge base, a sentence in propositional logic  
          $\alpha$ , the query, a sentence in propositional logic  
  
symbols  $\leftarrow$  a list of the proposition symbols in KB and  $\alpha$   
return TT-CHECK-ALL(KB,  $\alpha$ , symbols, [])
```

Implement the definition of entailment

```
function TT-CHECK-ALL(KB,  $\alpha$ , symbols, model) returns true or false  
if EMPTY?(symbols) then  
  if PL-TRUE?(KB, model) then return PL-TRUE?( $\alpha$ , model)  
  else return true (if not a model for KB  $\rightarrow$  don't care)  
else do  
  P  $\leftarrow$  FIRST(symbols); rest  $\leftarrow$  REST(symbols)  
  return TT-CHECK-ALL(KB,  $\alpha$ , rest, EXTEND(P, true, model)) and  
        TT-CHECK-ALL(KB,  $\alpha$ , rest, EXTEND(P, false, model))
```

Return a new partial model in which *P* has the value *true*

- A recursive depth-first enumeration of all models (assignments to variables)
 - Sound and complete
 - Time complexity: $O(2^n)$ exponential in the size of the input
 - Space complexity: $O(n)$

Logical Equivalences

- Two sentences are logically equivalent iff true in same set of models

$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

entailment

$$M(\alpha) \subseteq M(\beta) \text{ and} \\ M(\beta) \subseteq M(\alpha)$$

$$\therefore M(\beta) = M(\alpha)$$

Logical Equivalences (cont.)

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

Validity and Satisfiability

- A sentence is valid (or tautological) if it is true in all models

$True, A \vee \neg A, A \Rightarrow A, (A \wedge (A \Rightarrow B)) \Rightarrow B$

- Validity is connected to inference via Deduction Theorem:

$KB \models \alpha$ if only if $(KB \Rightarrow \alpha)$ is valid

- A sentence is satisfiable if it is true in some model

$A, B \wedge \neg C$

- A sentence is unsatisfiable if it is true in no models

$A \wedge \neg A$

- Satisfiability is connected to inference via **refutation** (or proof by **contradiction**)

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

Determination of satisfiability of sentences in PL is NP-complete

Patterns of Inference: Inference Rules

- Applied to derive chains of conclusions that lead to the desired goal
- Modus Ponens (Implication Elimination, *if-then* reasoning)

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

- And Elimination

$$\frac{\alpha \wedge \beta}{\alpha}$$

- Biconditional Elimination

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

Patterns of Inference: Inference Rules (cont.)

- Example

- With the KB as the following, show that $\neg P_{1,2}$

$R_1: \neg P_{1,1}$	no pit in [1,1]
$R_2: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$	pits cause breezes in adjacent squares
$R_3: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$	
$R_4: \neg B_{1,1}$	no breeze in [1,1]
$R_5: B_{2,1}$	breeze in [2,1]

1. Apply biconditional elimination to R_2

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

2. Apply And-Elimination to R_6

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

3. Logical equivalence for contrapositives

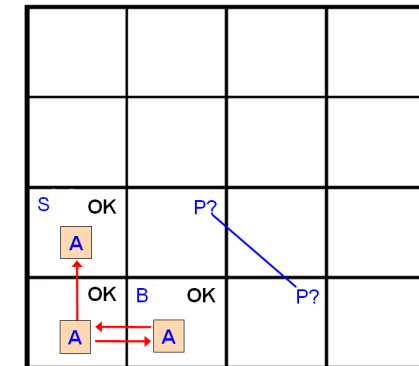
$$R_8: \neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})$$

4. Apply Modus Ponens with R_8 and the percept R_4

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

5. Apply De Morgan's rule and give the conclusion

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



6. Apply And-Elimination to R_{10}

$$R_{11}: \neg P_{1,2}$$

Patterns of Inference: Inference Rules (cont.)

- Unit Resolution

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

$$\frac{l_1 \vee l_2 \vee \dots \vee l_k, \quad m}{l_1 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k}$$

l_i and m are complementary literals



- Resolution

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

$$\frac{l_1 \vee l_2 \vee \dots \vee l_k, \quad m_1 \vee \dots \vee m_n}{l_1 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k \vee m_1 \vee \dots \vee m_{j-1} \vee m_{j+1} \vee \dots \vee m_n}$$

Resolution is used to either confirm or refute a sentence, but it can't be used to enumerate sentences

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

l_i and m_j are complementary literals



– E.g.,

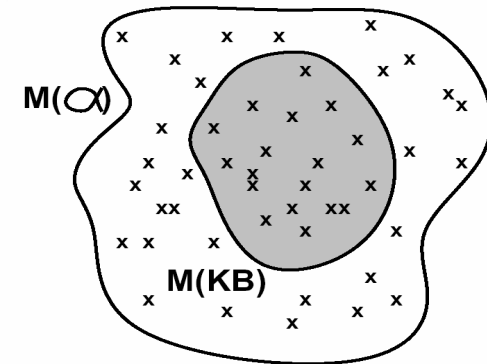
- Multiple copies of literals in the resultant clause should be removed (such a process is called **factoring**)

Monotonicity

- The set of entailed sentences can only increase as information is added to the knowledge base

If $KB \models \alpha$ then $KB \wedge \beta \models \alpha$

- The additional assertion β can't invalidate any conclusion α already inferred
- E.g., α : there is not pit in [1,2]
 β : there is eight pits in the world



Normal Forms

- Conjunctive Normal Form (*CNF*)
 - A sentence expressed as a conjunction of disjunctions of literals
 - E.g., $(P \vee Q) \wedge (\neg P \vee R) \wedge (\neg S)$
- Also, Disjunction Normal Form (*DNF*)
 - A sentence expressed as a disjunction of conjunctions of literals
 - E.g., $(P \wedge Q) \vee (\neg P \wedge R) \vee (\neg S)$
- An arbitrary propositional sentence can be expressed in *CNF* (or *DNF*)

Normal Forms (cont.)

- Example: convert $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ into *CNF*

1. Eliminate \Leftrightarrow , replace $\alpha \Leftrightarrow \beta$ 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 , replace $\alpha \Rightarrow \beta$ with $(\neg \alpha \vee \beta)$

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

3. Move \neg inwards

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

4. Apply distributivity law

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

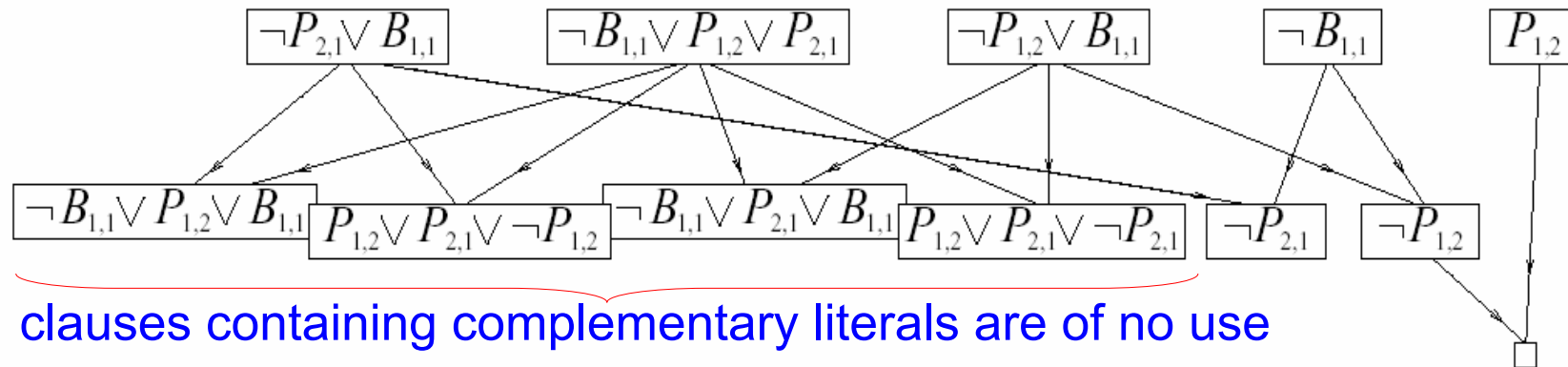
Resolution Algorithm

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

- To show that $KB \models \alpha$, we show that $(KB \wedge \neg \alpha)$ is unsatisfiable
- Each pair that contains complementary literals is resolved to produce new clause until one of the two things happens:
 - (1) No new clauses can be added $\Rightarrow KB$ does not entail α
 - (2) Empty clause is derived $\Rightarrow KB$ entails α

Resolution Example

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



- Empty clause – disjunction of no disjuncts
 - Equivalent to false
 - Represent a contradiction here

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

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

We have shown it before !

Horn Clauses

- A Horn clause is a disjunction of literals of which at most one is positive
 - E.g., $\neg P_1 \vee \neg P_2 \vee \dots \vee \neg P_n \vee Q$
- Every Horn clause can be written as an implication
 - The premise is a conjunction of positive literals
 - The conclusion is a single positive literal
 - E.g., $\neg P_1 \vee \neg P_2 \vee \dots \vee \neg P_n \vee Q$ can be converted to $(P_1 \wedge P_2 \wedge \dots \wedge P_n) \Rightarrow Q$
- Inference with Horn clauses can be done naturally through the **forward chaining** and **backward chaining**, which we will be discussed later on
 - The application of **Modus Ponens**
- Not every *PL* sentence can be represented as a conjunction of Horn clauses

Forward Chaining

- As known, if all the premises of an implication are known, then its conclusion can be added to the set of known facts
- **Forward Chaining** fires any rule whose premises are satisfied in the *KB*, add its conclusion to the *KB*, **until query is found or until no further inferences can be made**
 - Applications of Modus Ponens

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

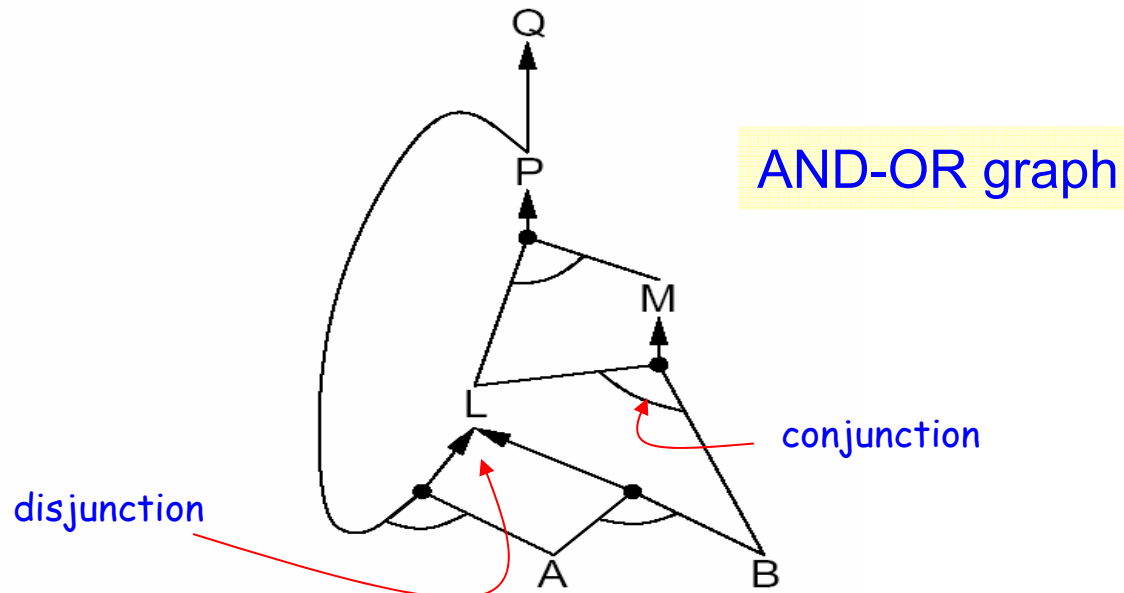
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

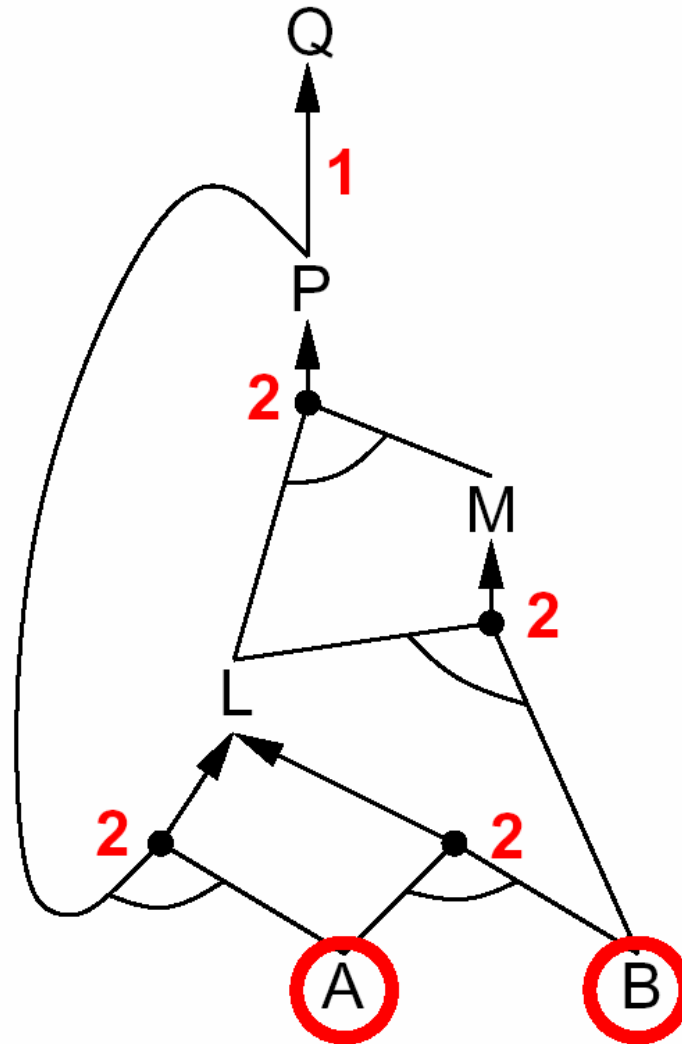
$$A \wedge B \Rightarrow L$$

A

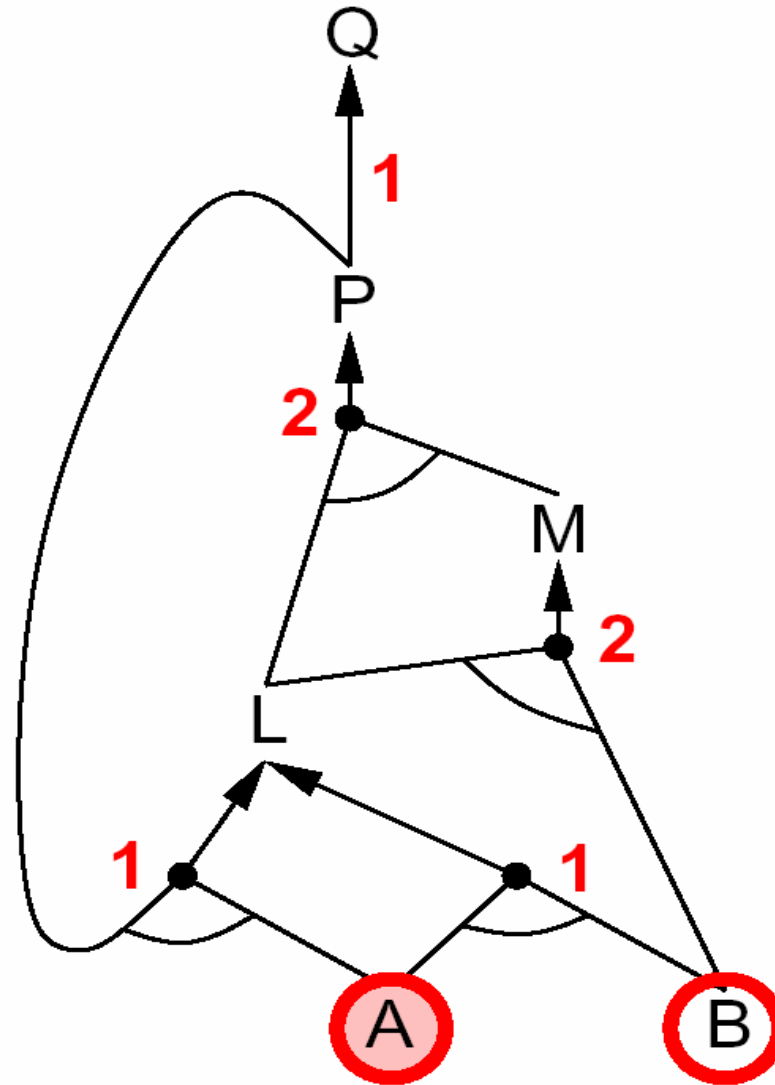
B



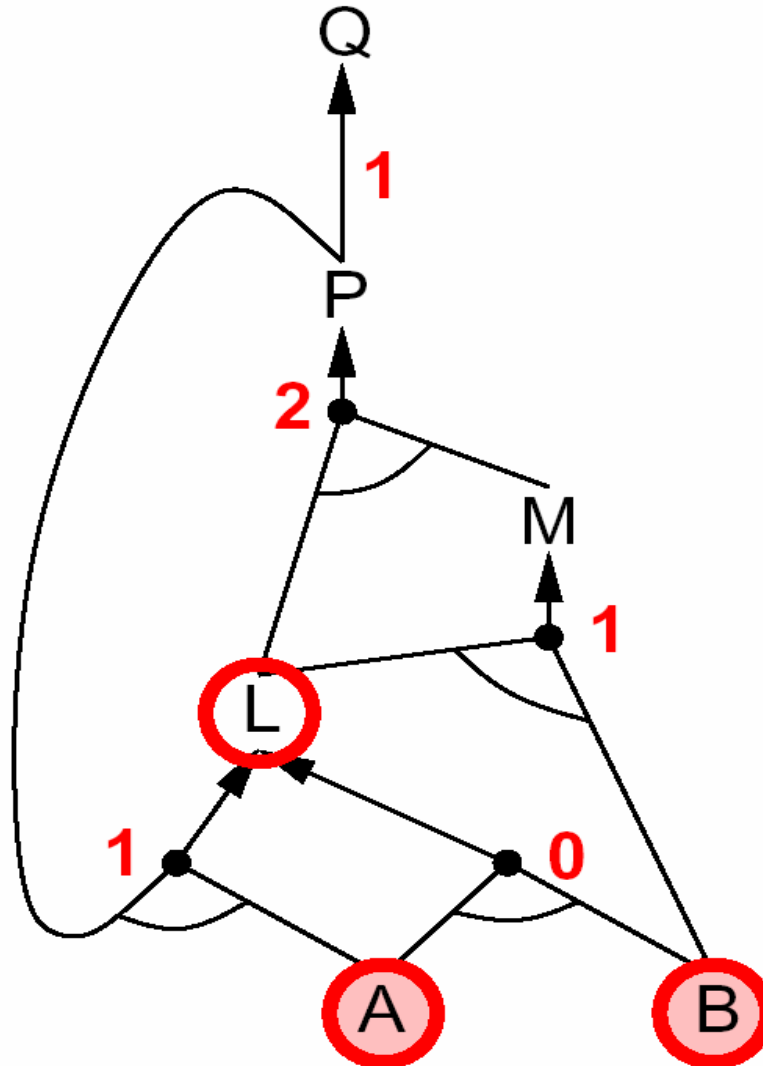
Forward Chaining: Example



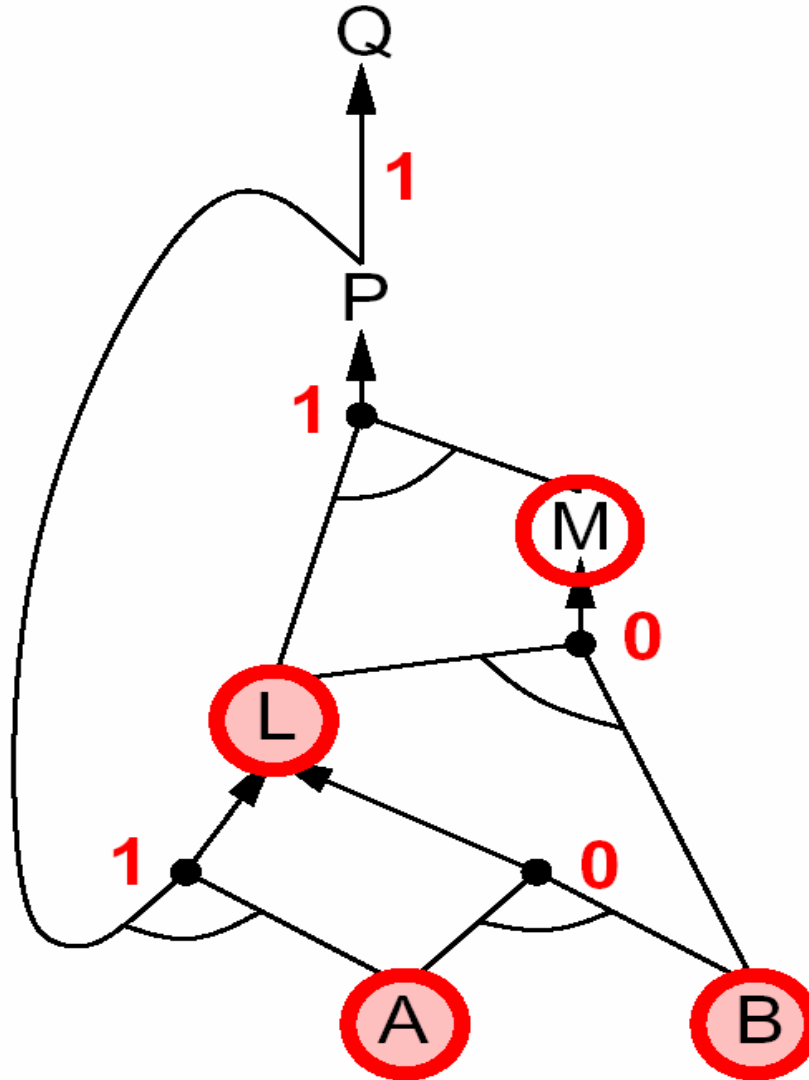
Forward Chaining: Example (cont.)



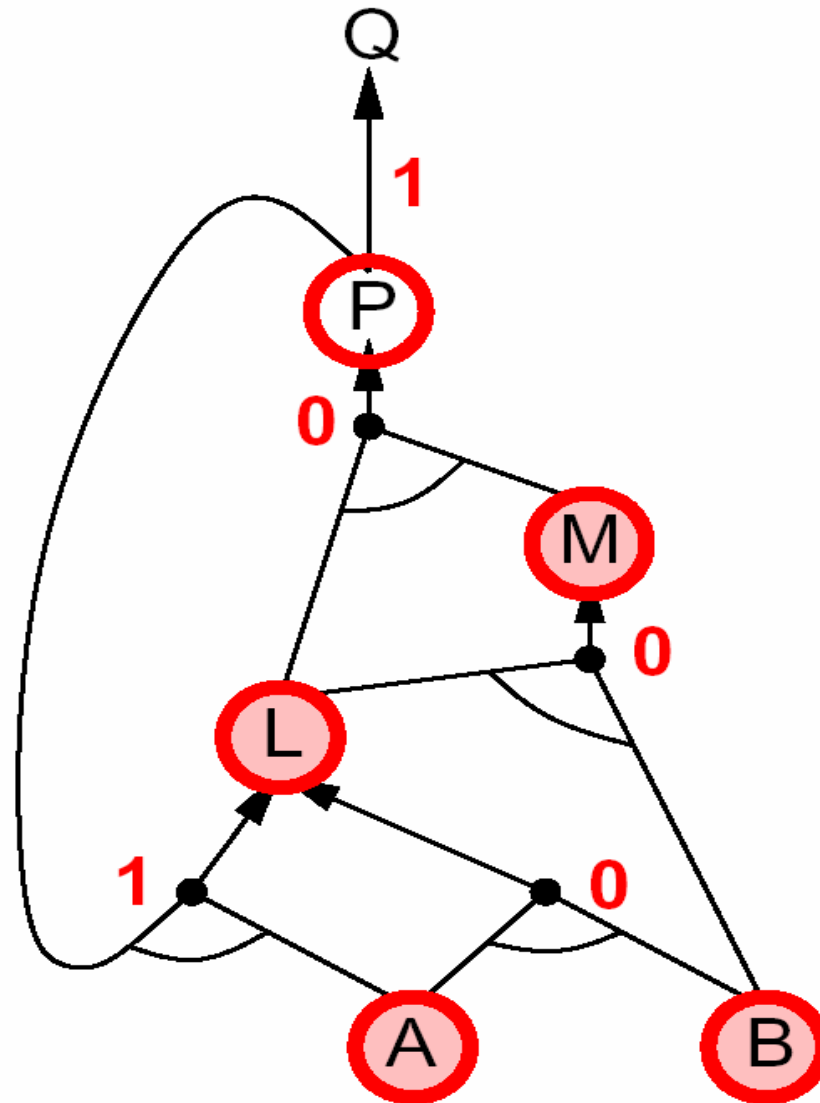
Forward Chaining: Example (cont.)



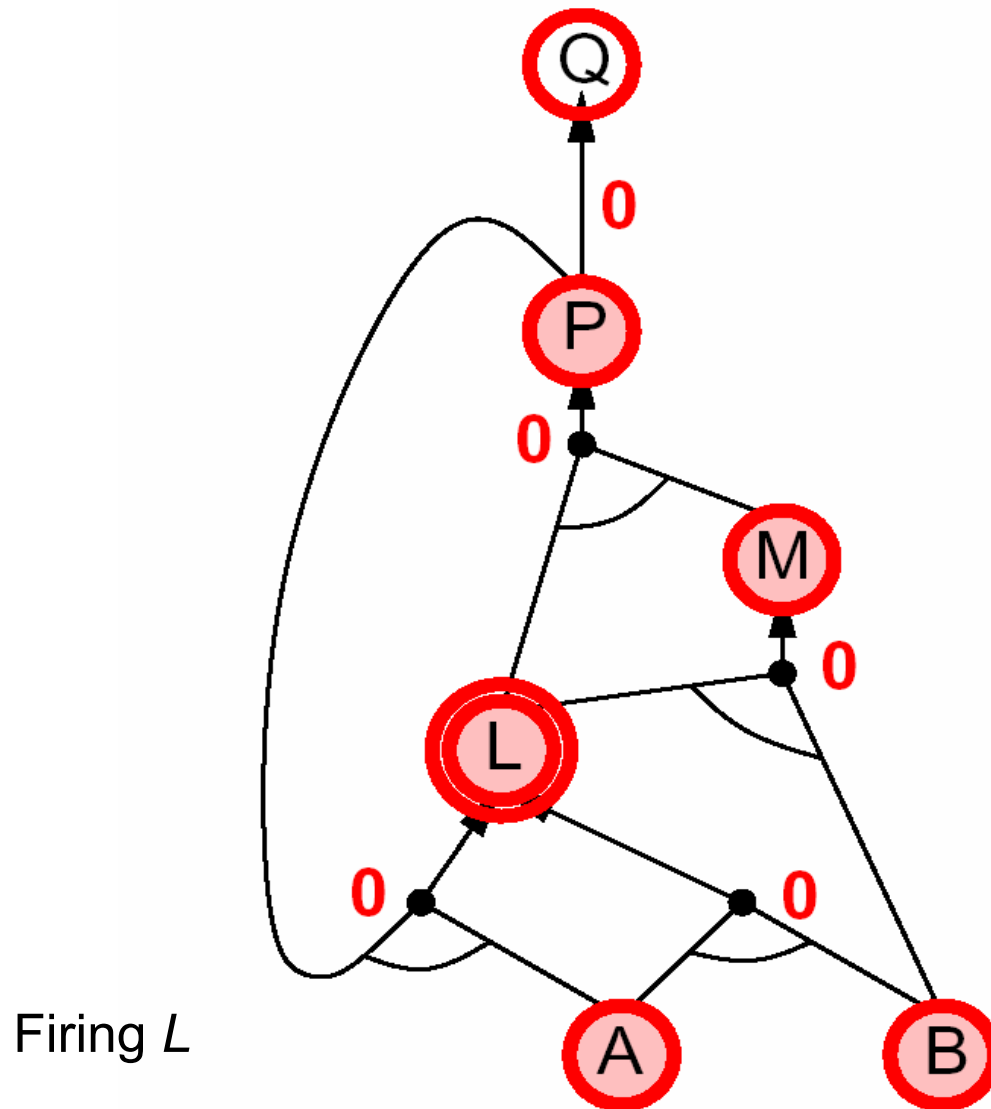
Forward Chaining: Example (cont.)



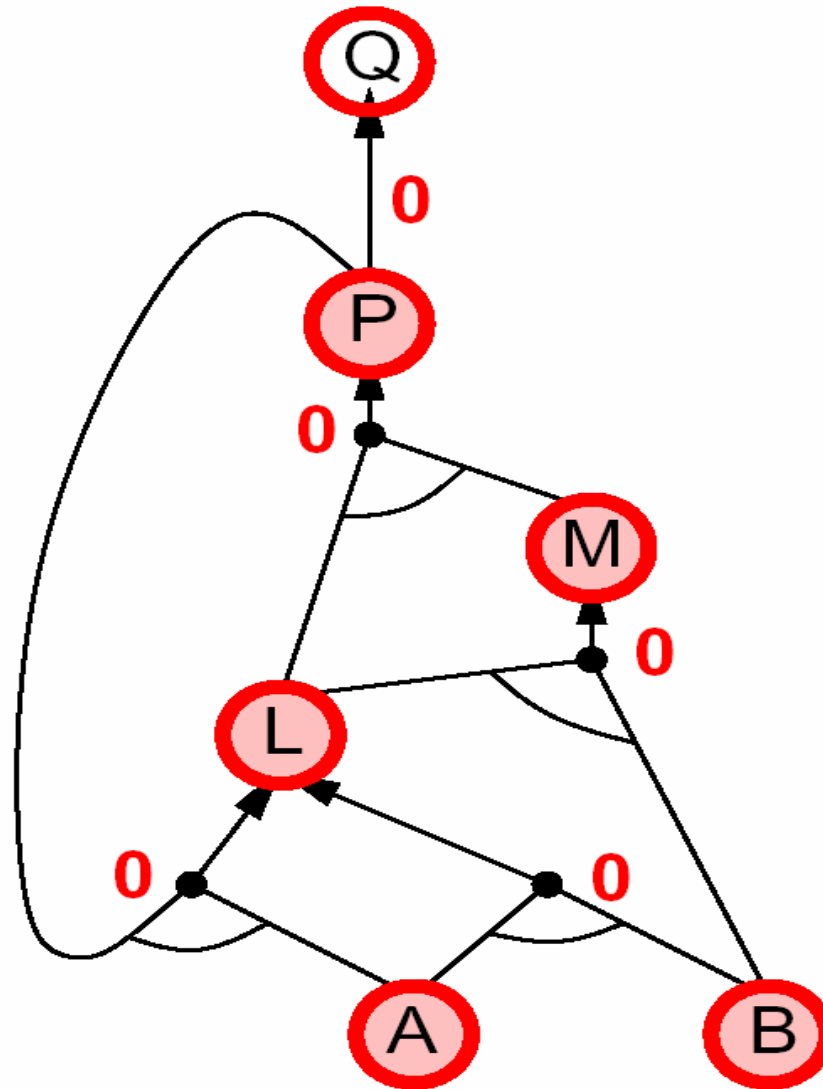
Forward Chaining: Example (cont.)



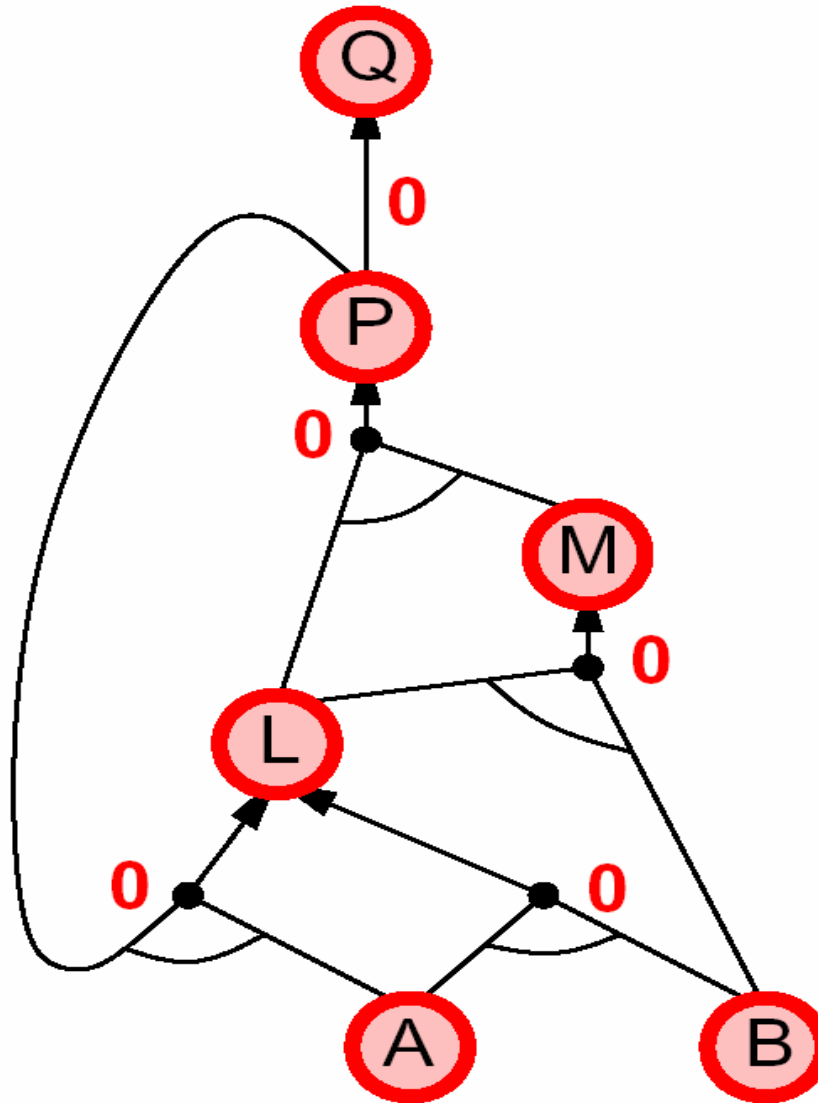
Forward Chaining: Example (cont.)



Forward Chaining: Example (cont.)



Forward Chaining: Example (cont.)



Forward Chaining: Algorithm (cont.)

```
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 to be true in KB

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

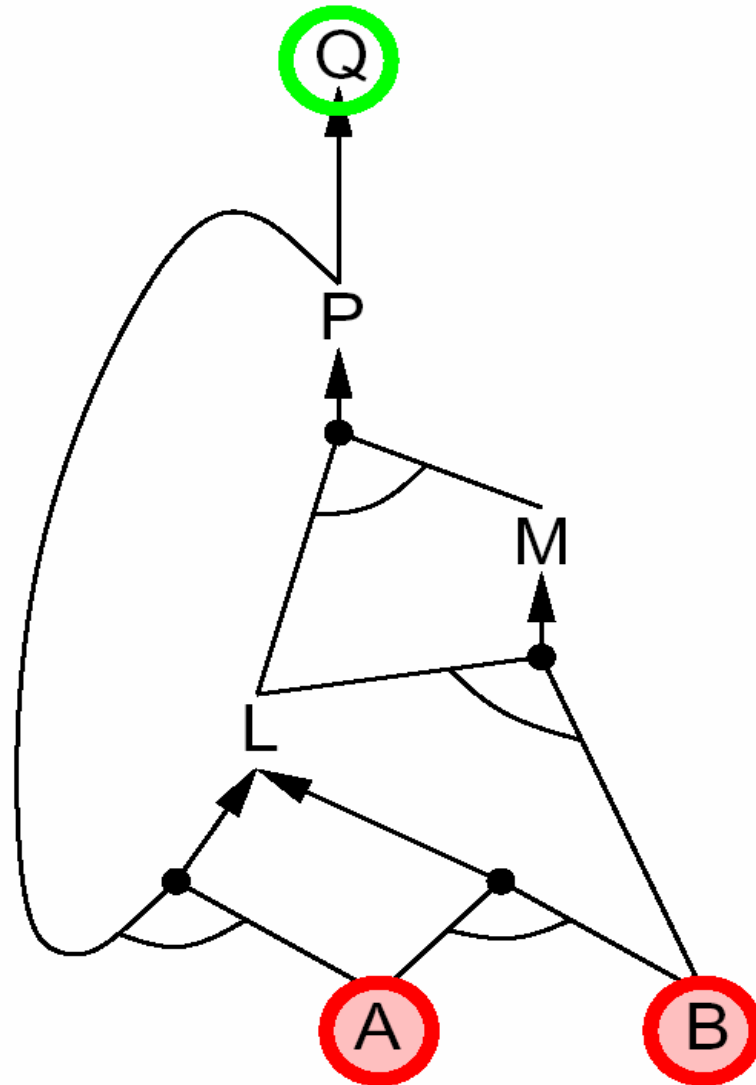
Forward Chaining: Properties

- Sound
 - Because every inference is an application of Modus Ponens
- Complete
 - Every entailed atomic sentence will be derived
 - But may do lots of work that is irrelevant to the goal
- A form of data-driven reasoning
 - Start with known data and derive conclusions from incoming percepts

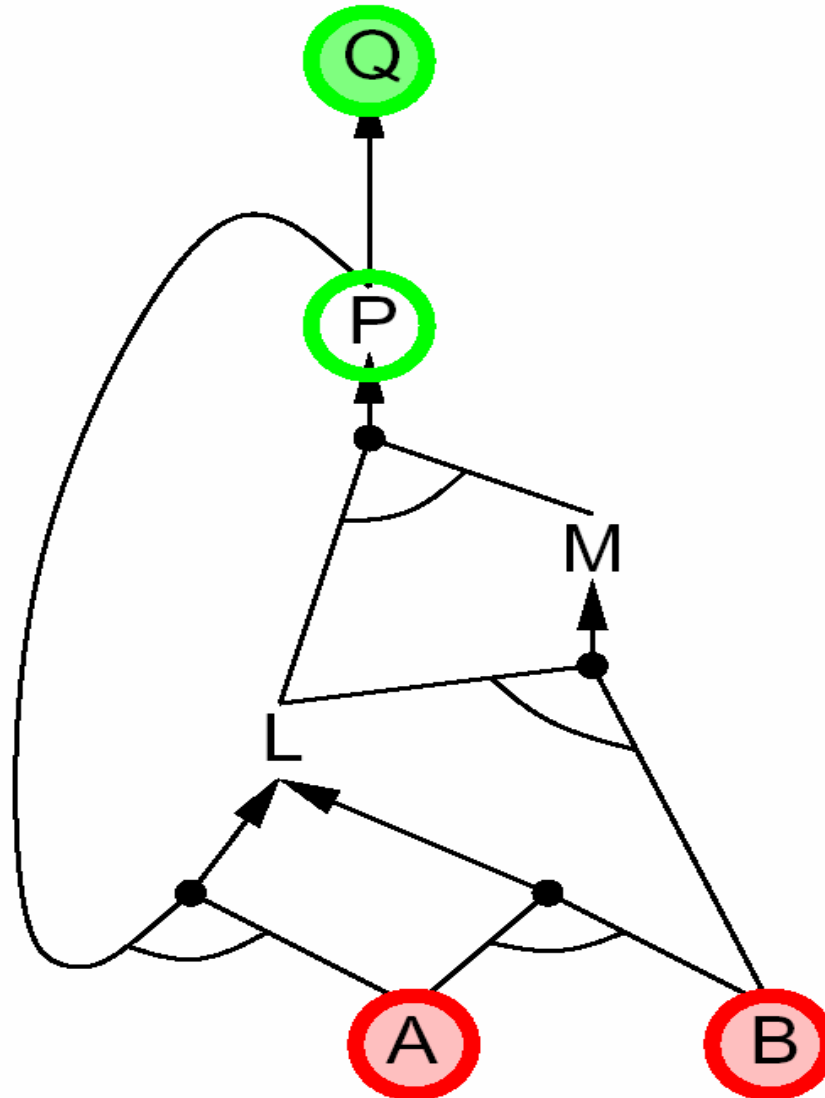
Backward Chaining

- Work backwards from the query q to prove q by backward chaining (BC)
- Check if q is known already, or prove by BC all premises of some rule concluding q
- A form of goal-directed reasoning

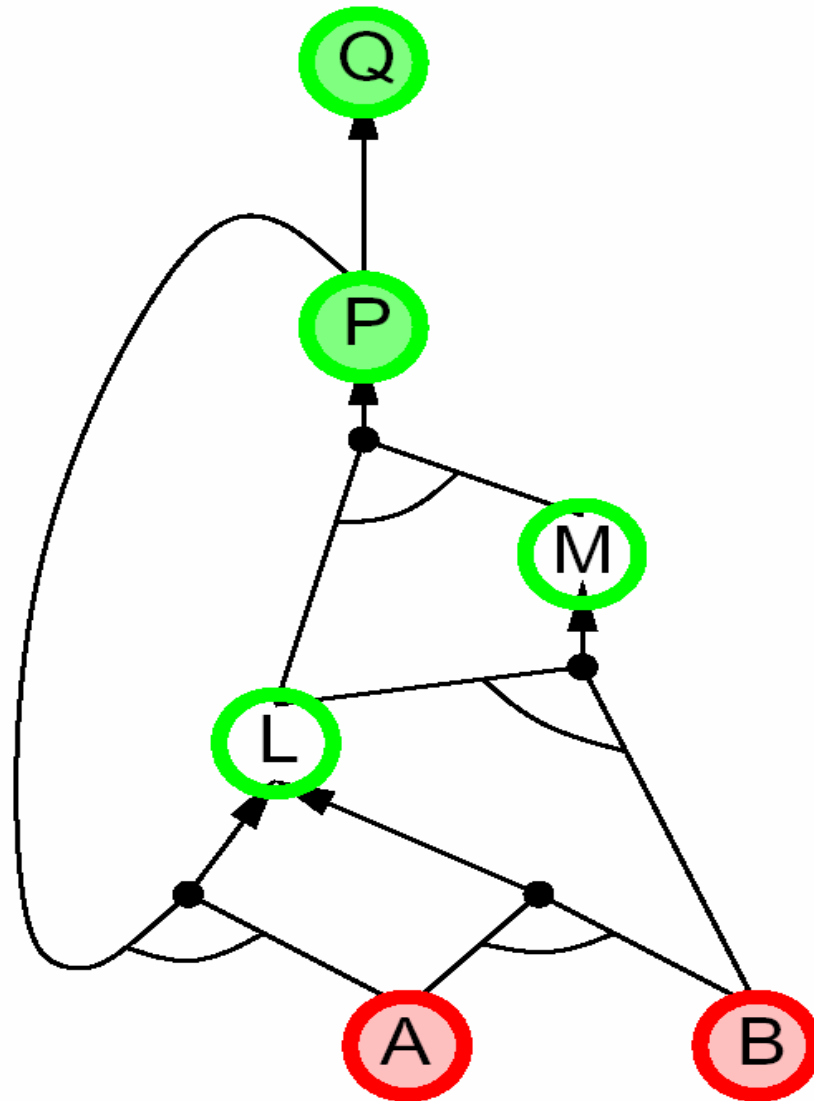
Backward Chaining: Example



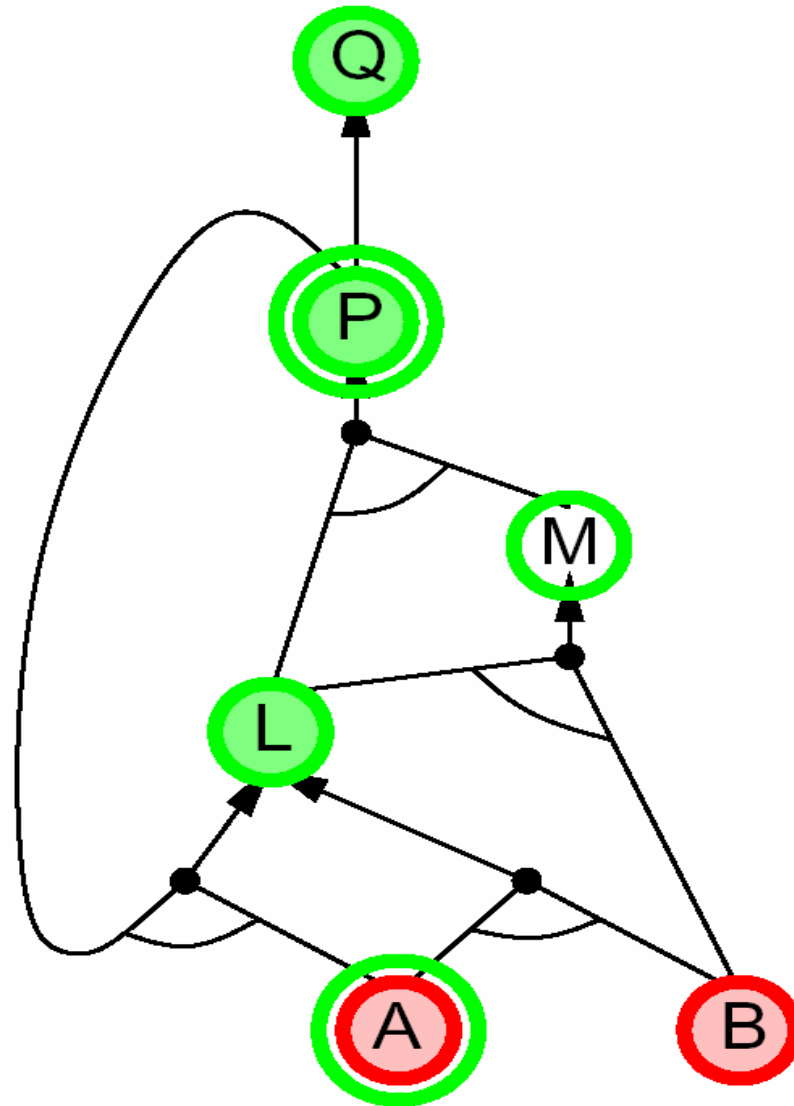
Backward Chaining: Example (cont.)



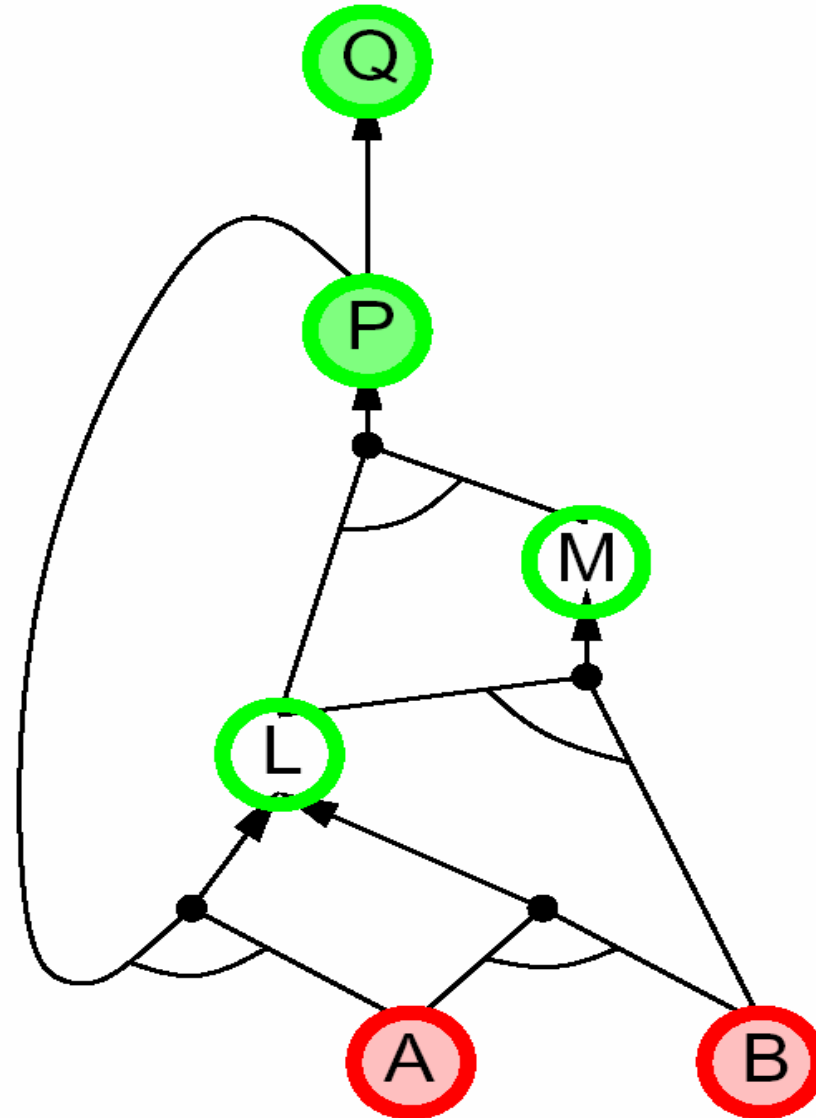
Backward Chaining: Example (cont.)



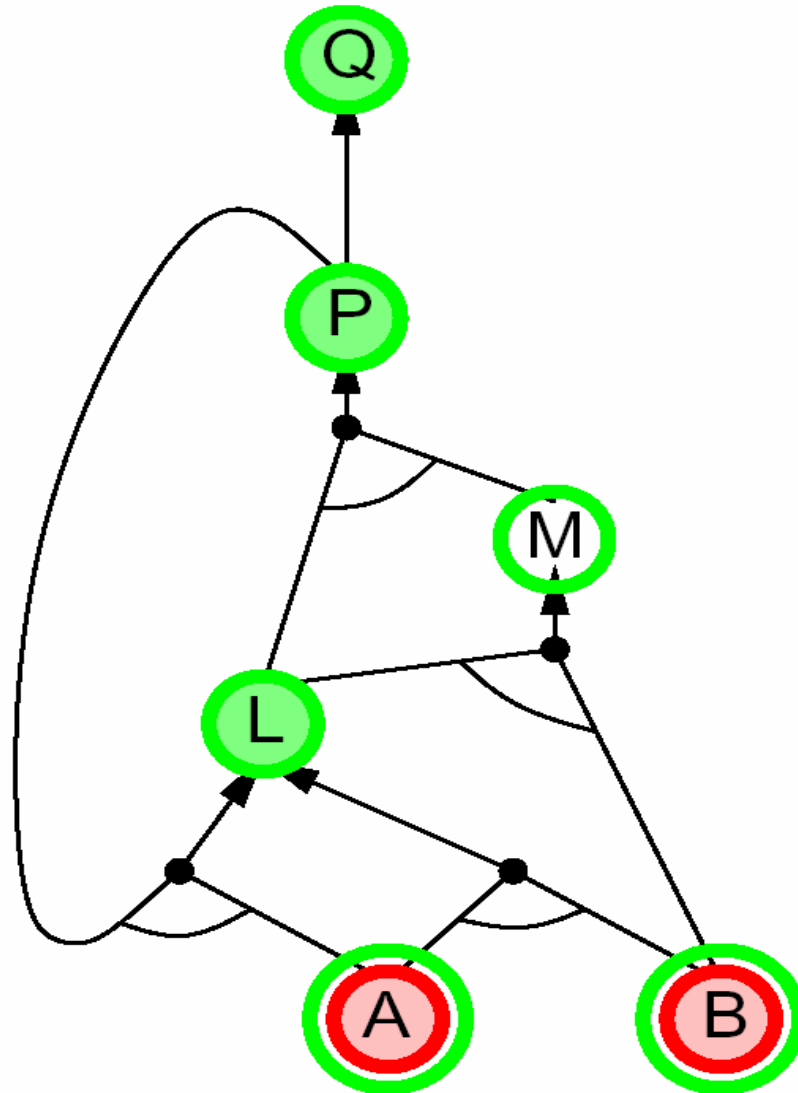
Backward Chaining: Example (cont.)



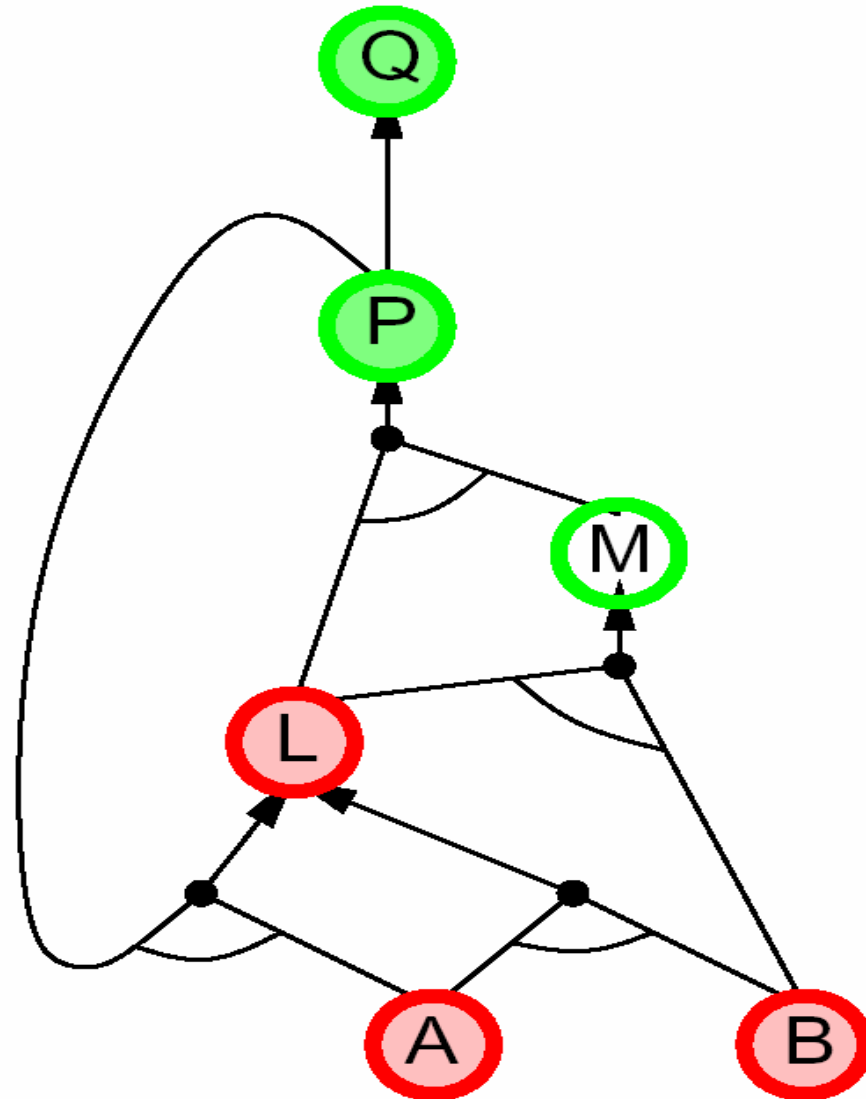
Backward Chaining: Example (cont.)



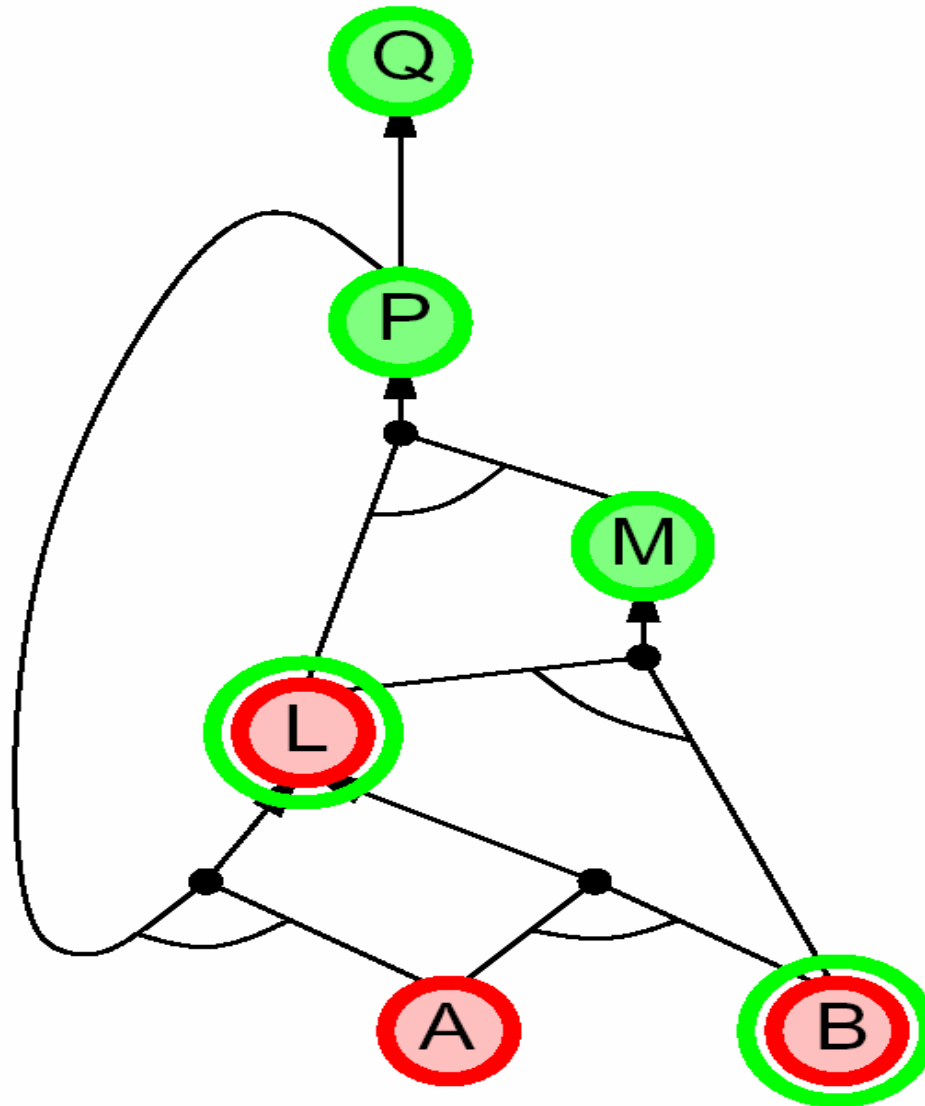
Backward Chaining: Example (cont.)



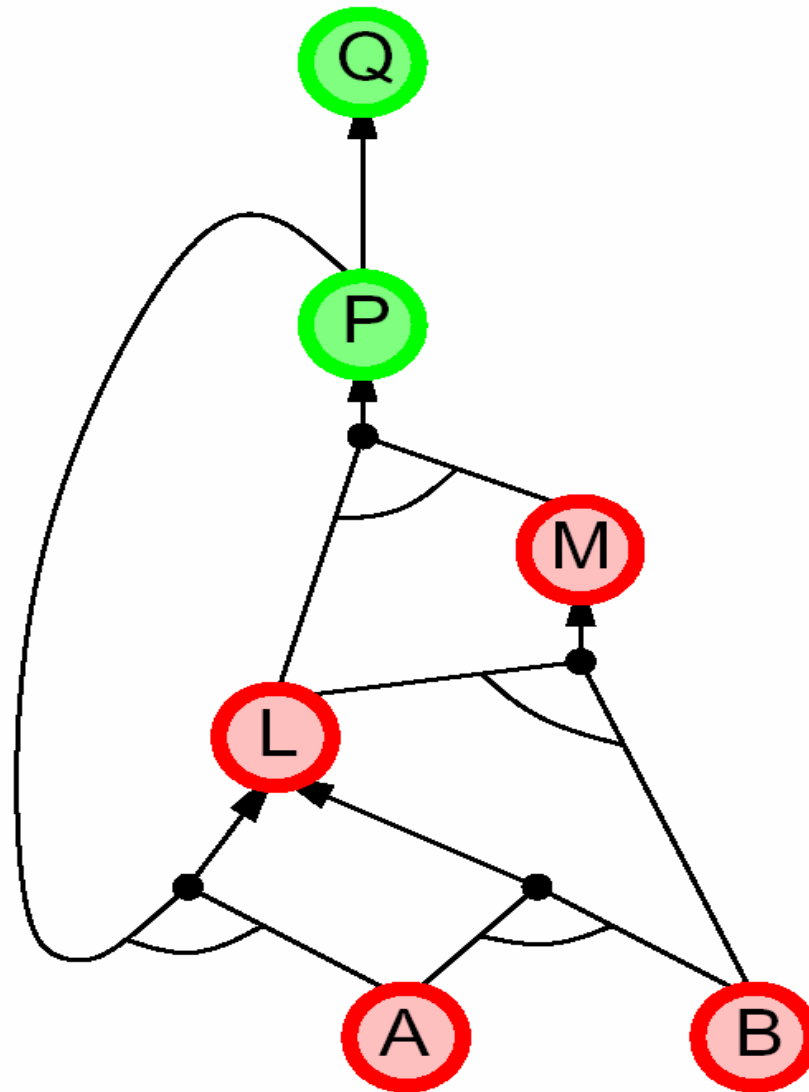
Backward Chaining: Example (cont.)



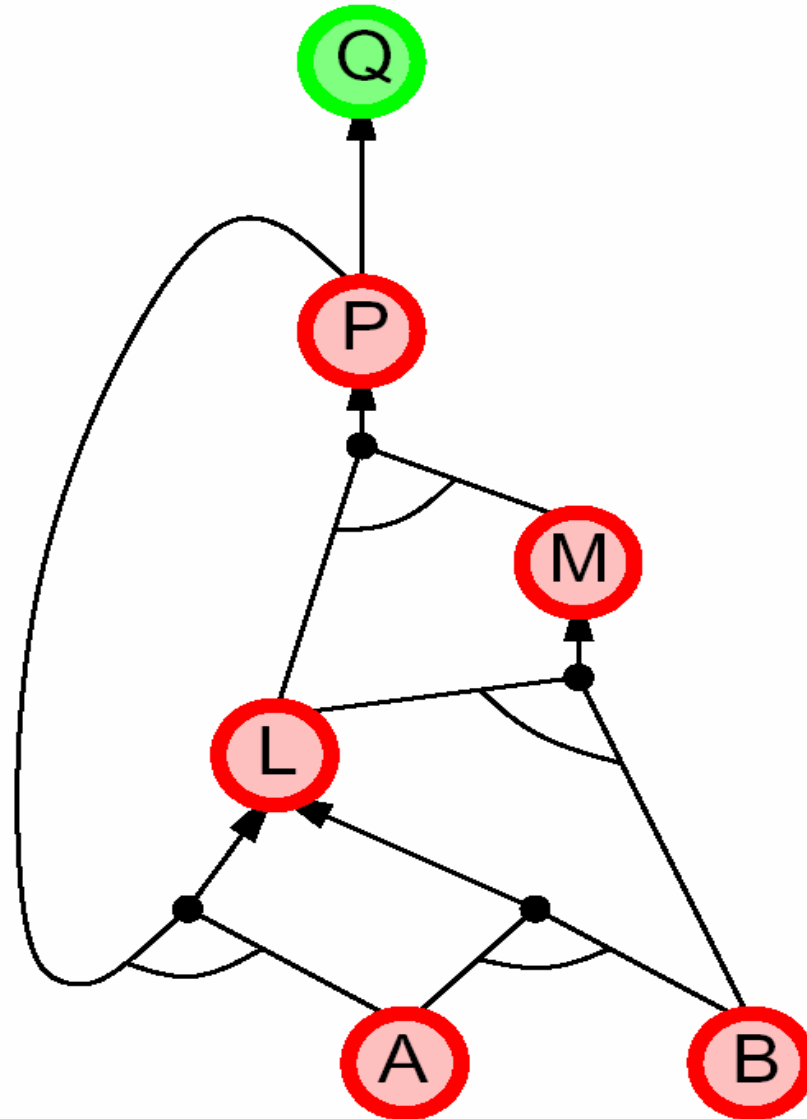
Backward Chaining: Example (cont.)



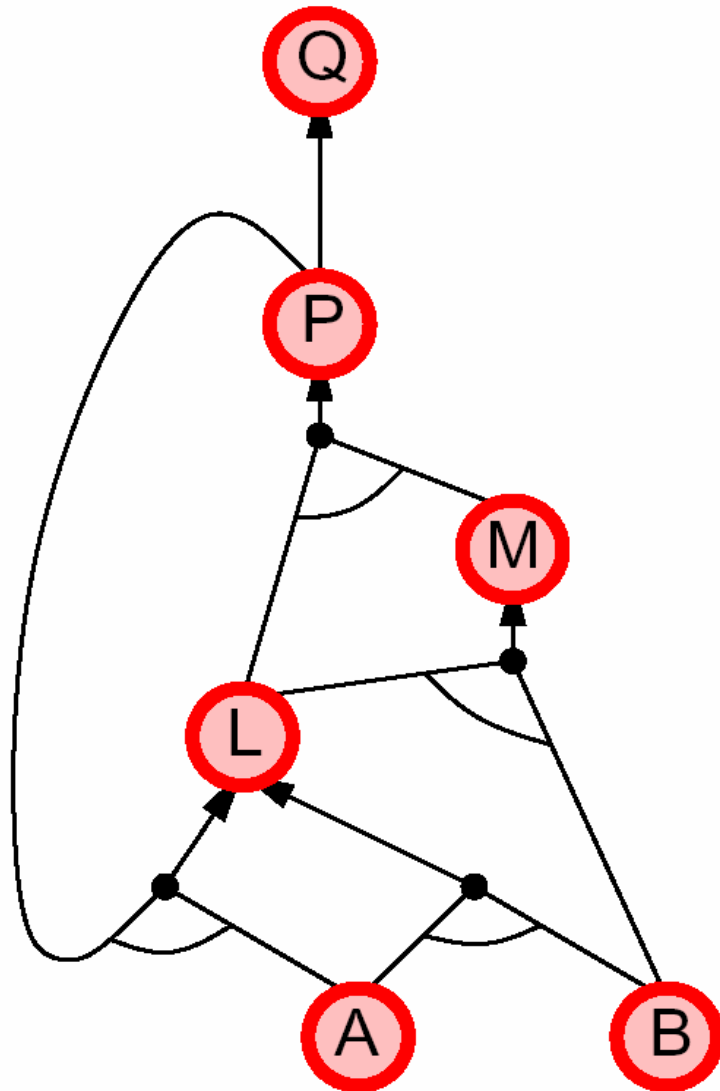
Backward Chaining: Example (cont.)



Backward Chaining: Example (cont.)



Backward Chaining: Example (cont.)



Forward vs. Backward Chaining

- FC (data-driven)
 - May do lots of work that is irrelevant to the goal
- BC (goal-driven)
 - Complexity of BC can be **much less** than linear in size of *KB*

Propositional Logic: Drawbacks

- Propositional Logic is declarative and compositional
- The lack of expressive power to describe an environment with many objects concisely
 - E.g., we have to write a separate rule about breezes and pits for each square

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