U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Artificial Intelligence

Lecture 4a: Knowledge Representation and Reasoning

Henrique Lopes Cardoso, Luís Paulo Reis

hlc@fe.up.pt, lpreis@fe.up.pt

Knowledge-based Agents

- Humans know things, which helps them do things!
  - Processes of **reasoning** that operate on internal **representations** of knowledge

- **Logic**: a general class of representations to support knowledge-based agents
  - Combine and recombine information to suit myriad purposes

- **Knowledge-based agents** can accept new tasks in the form of explicitly described goals
  - Being told or learning new knowledge about the environment
  - Adapt to changes in the environment by updating the relevant knowledge

The Knowledge Base

- **Knowledge base (KB)**
  - A set of “sentences”, each representing some assertion about the world
  - Expressed in a **knowledge representation language**
  - Initial content: **background knowledge**

- Adding new sentences to the knowledge base (assertions): **TELL**
- Querying what is known: **ASK**

- **Inference**: deriving new sentences from existing ones
  - When asking a question of the knowledge base, the answer should *follow* from what has been told to the knowledge base (previous assertions)

Knowledge-based Agent Program

```
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
```

- TELL the KB what it perceives
- ASK the KB what action to perform
  - Reasoning about the current state of the world, outcomes of possible actions, …
- TELL the KB which action was performed in the world

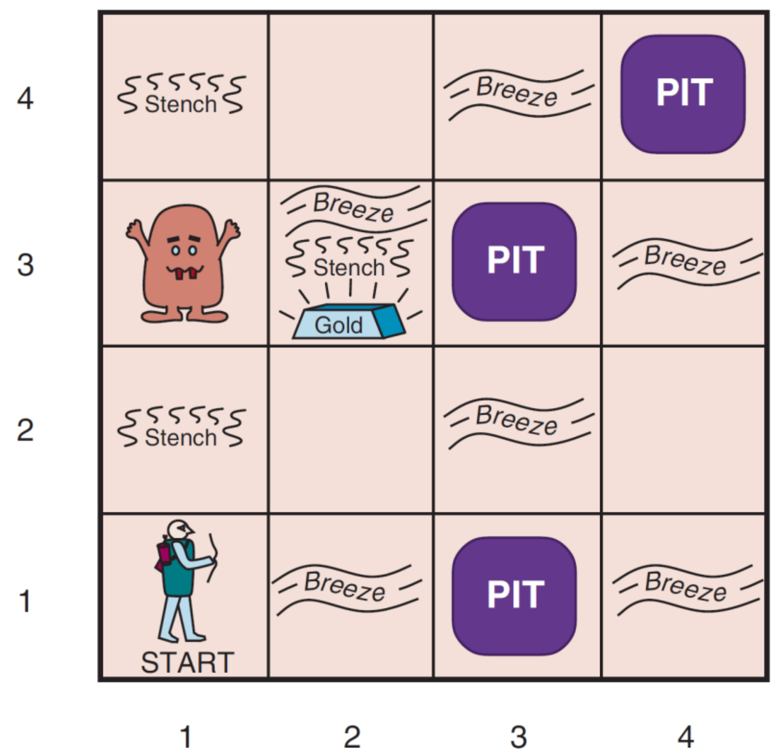
Knowledge vs. Implementation Level

- A knowledge-based agent can be described at the **knowledge level**
  - We need only to specify what the agent knows and what its goals are
  - Example:
    - An automated taxi has the goal of taking a passenger from Porto to Gaia and might know that it must cross one of the beautiful bridges on the Douro river.
    - We can expect it to cross a bridges because **it knows this will achieve its goal**!
  - **Declarative** approach to system building: TELLing the agent what it needs to know

- **Implementation level**: data structures inside the KB and algorithms that work on them
  - **Procedural** approach: encode behaviors directly as program code

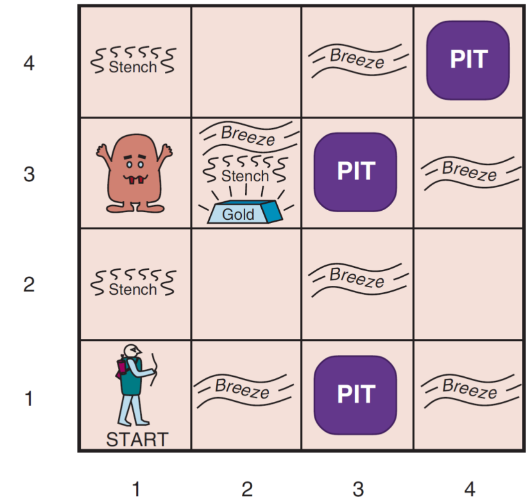
The Wumpus World

- A cave consisting of **rooms** connected by passageways
- Player must take the **gold** and return to the start position without entering any room with a bottomless **pit** or **wumpus**
- **Wumpus** can be **killed**, but the agent has only **one arrow**



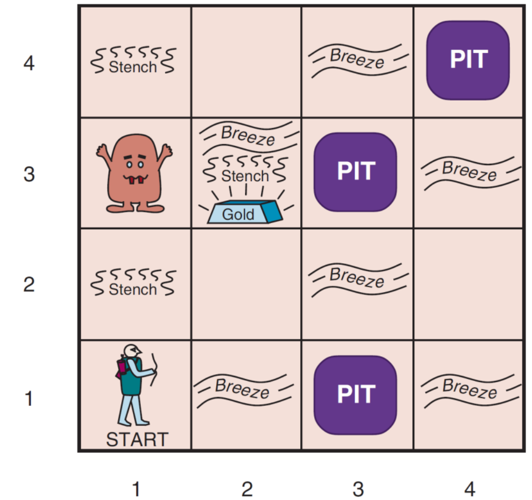
Wumpus World PEAS Description

- **P**erformance measure
  - Gold and at [1,1] +1000; death -1000
  - -1 per step; -10 for using the arrow
- **E**nvironment
  - 4x4 grid, agent starts at [1,1], gold and wumpus at random locations, pit with prob 0.2
- **A**ctuators
  - *Forward*, *Turn left 90º*, *Turn right 90º*
  - *Grab* gold (only at gold position)
  - *Shoot* (only once, kills wumpus if it is in that direction)
- **S**ensors
  - *Stench* at cells adjacent to the wumpus
  - *Breeze* at cells adjacent to a pit
  - *Glitter* at gold position
  - *Bump* when hitting a wall
  - *Scream* when wumpus is killed

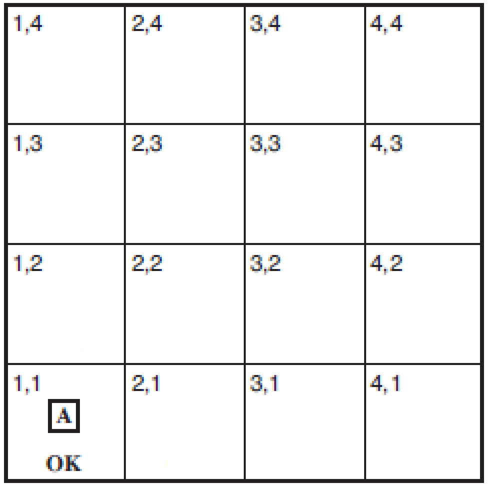


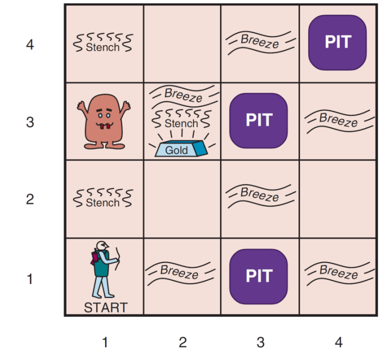
Wumpus World Environment

- Observable?
  - Partially: only local perception
- Deterministic?
  - Yes (for the actions actually available)
- Episodic?
  - Sequential: rewards may come only after many actions are taken
- Static?
  - Yes
- Discrete?
  - Yes
- Single-agent?
  - Yes (wumpus doesn't move)



Exploring a Wumpus World

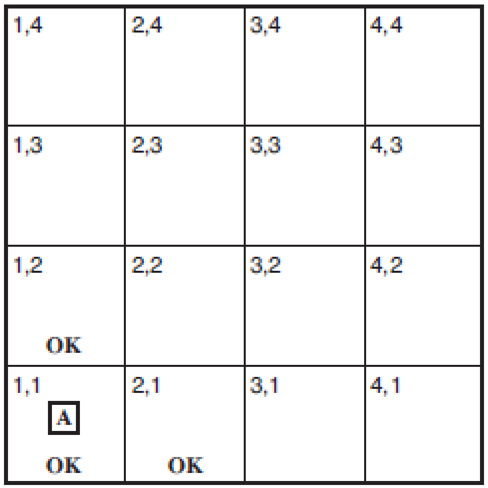


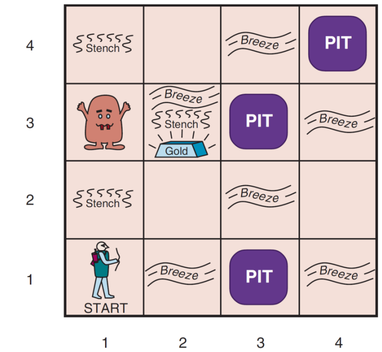


[*None*,*None*,*None*,*None*,*None*]

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

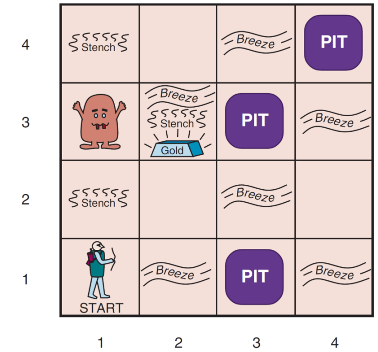
Exploring a Wumpus World

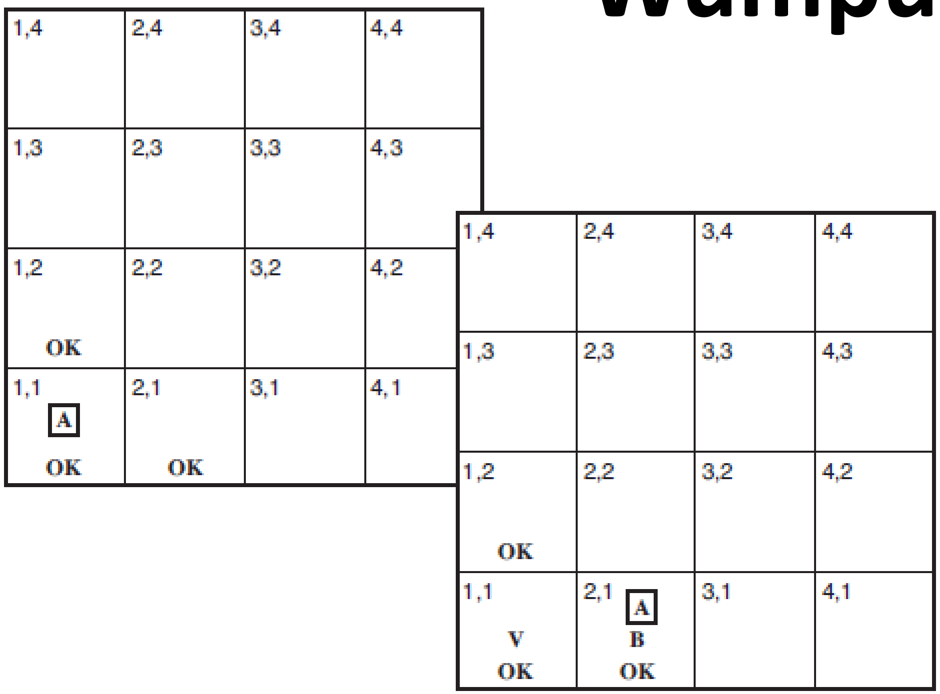




A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

Exploring a Wumpus World

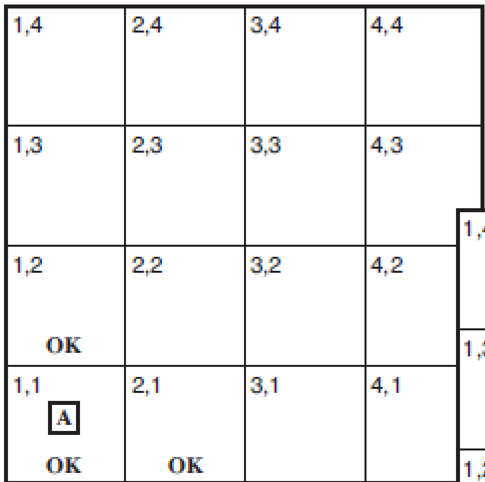


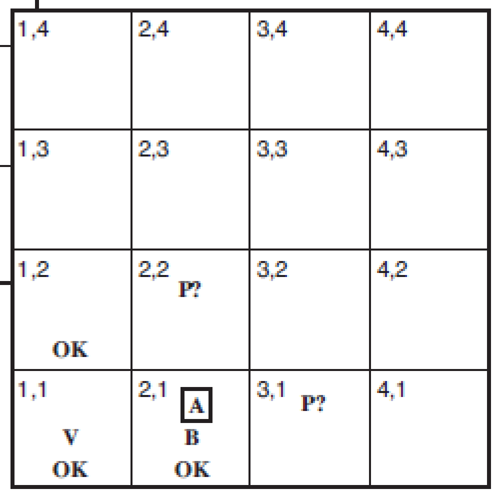


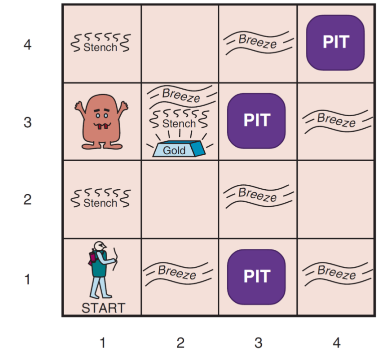
[*None*,*Breeze*,*None*,*None*,*None*]

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

Exploring a Wumpus World

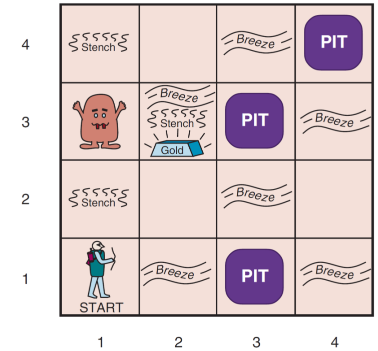


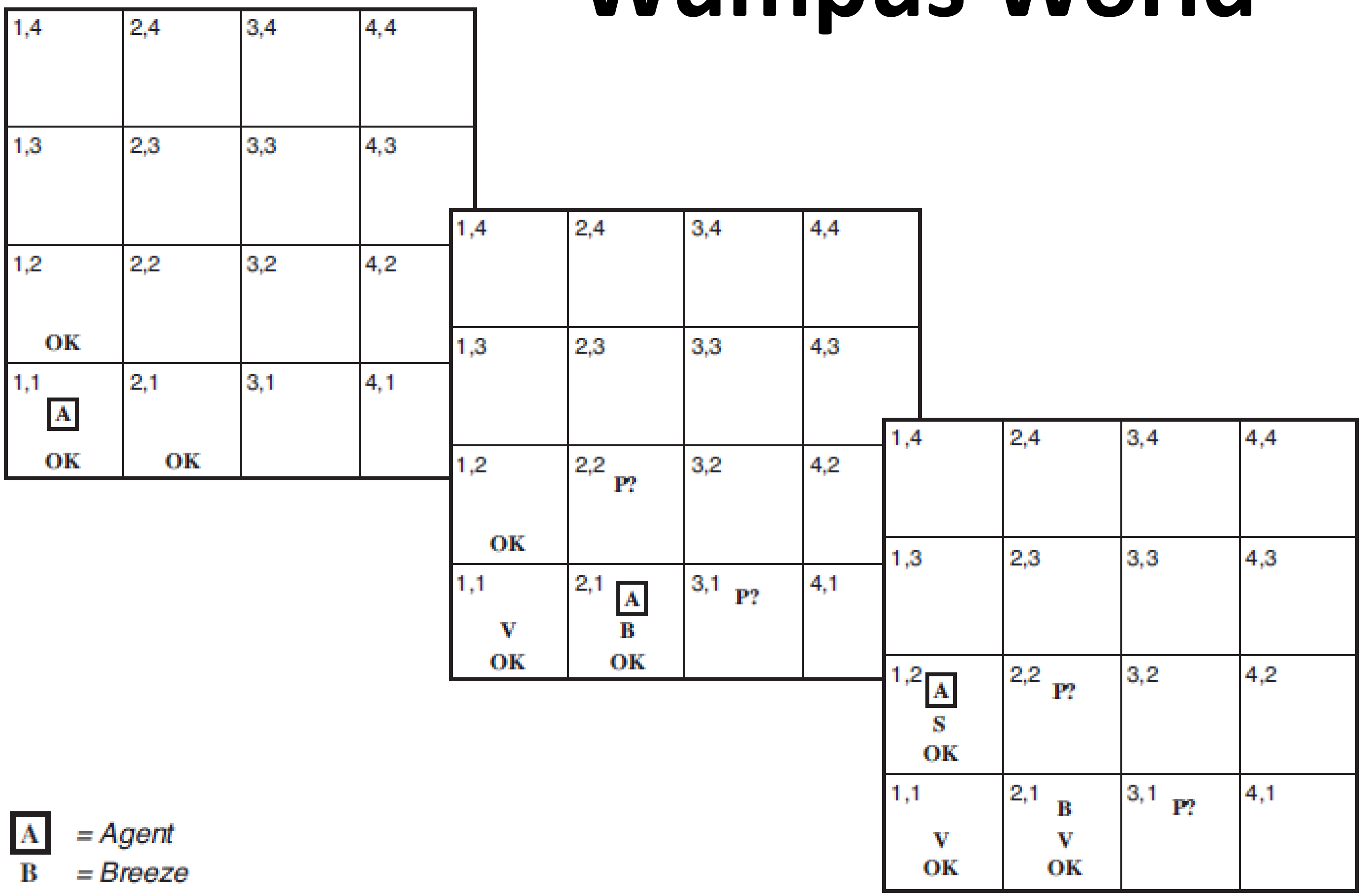




A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

Exploring a Wumpus World

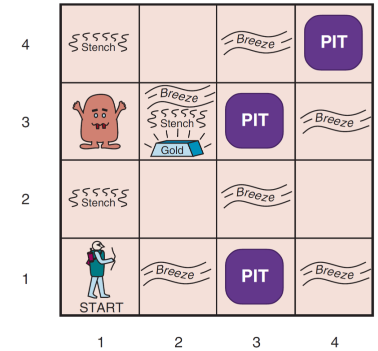


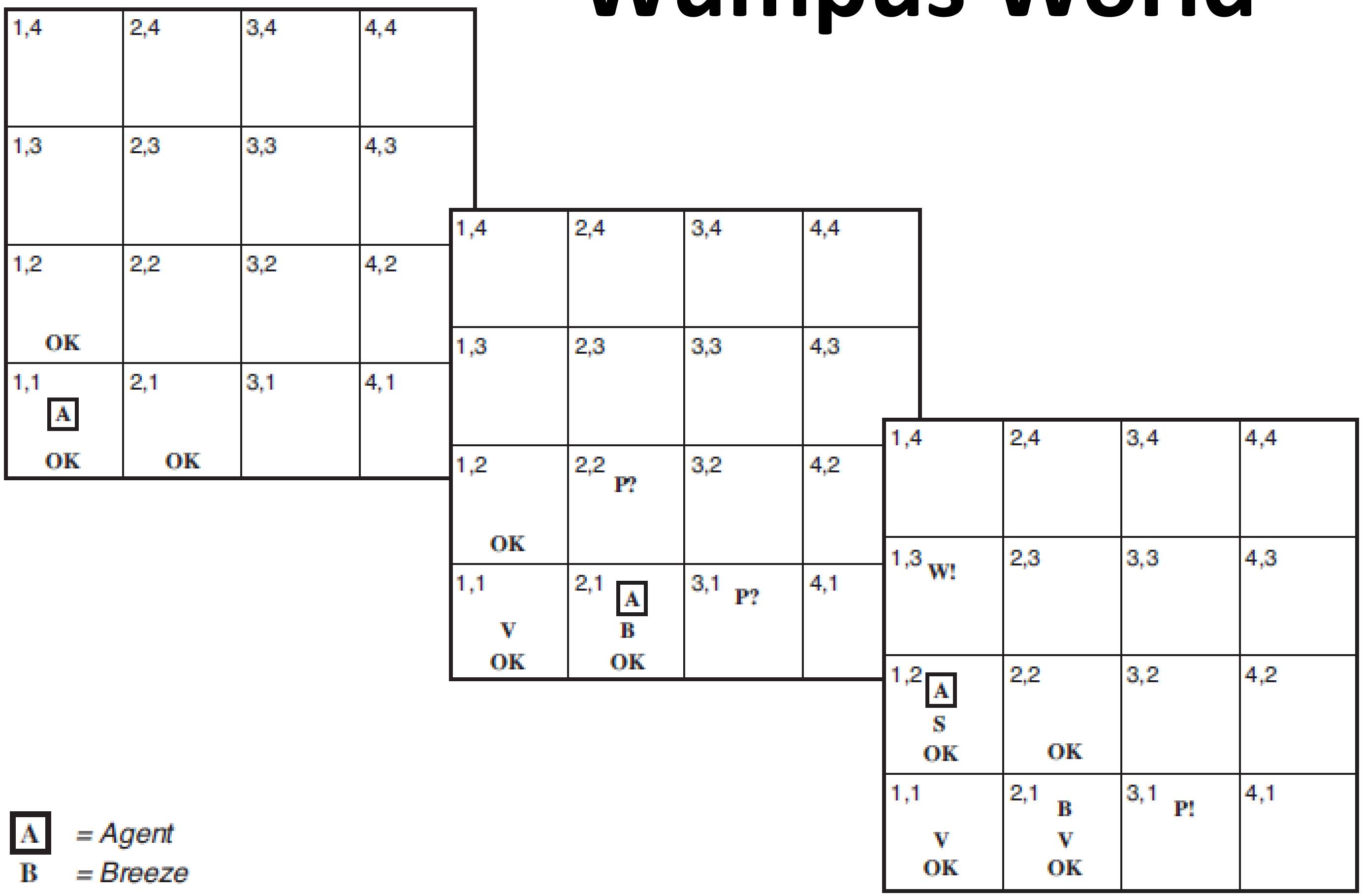


[*Stench,None,None,None,None*]

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

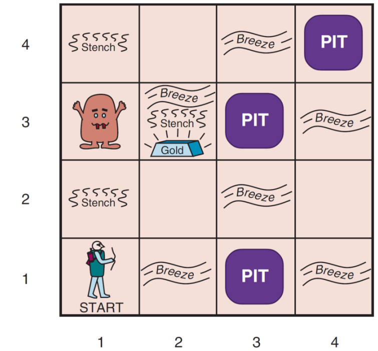
Exploring a Wumpus World

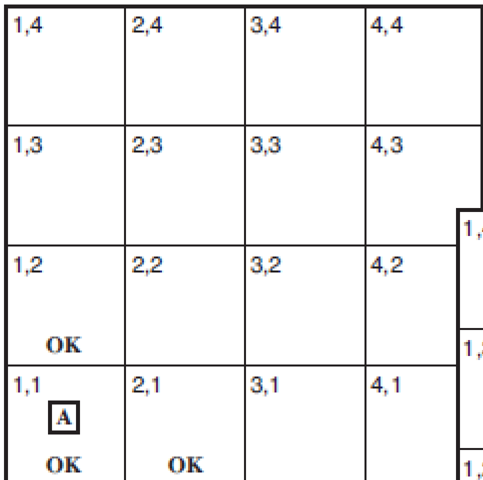


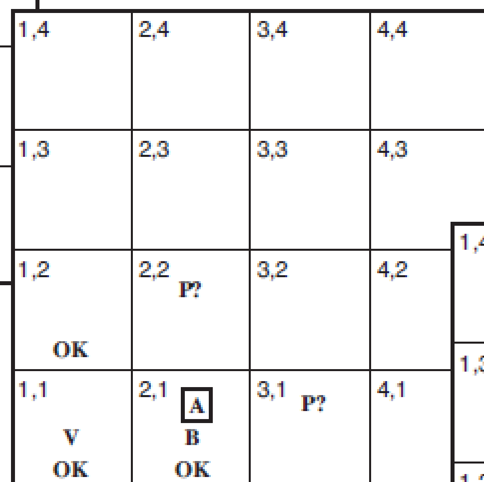


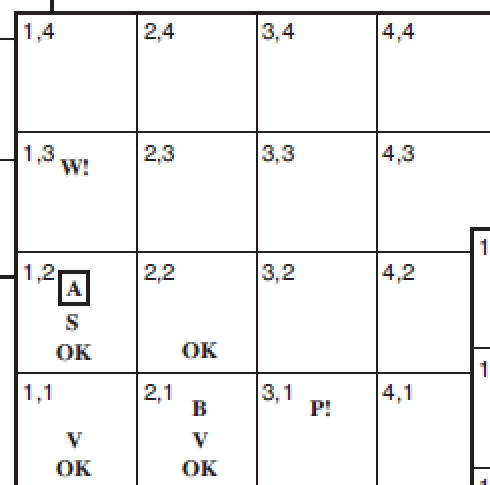
- **A** = *Agent*
- **B** = *Breeze*
- **G** = *Glitter, Gold*
- **OK** = *Safe square*
- **P** = *Pit*
- **S** = *Stench*
- **V** = *Visited*
- **W** = *Wumpus*

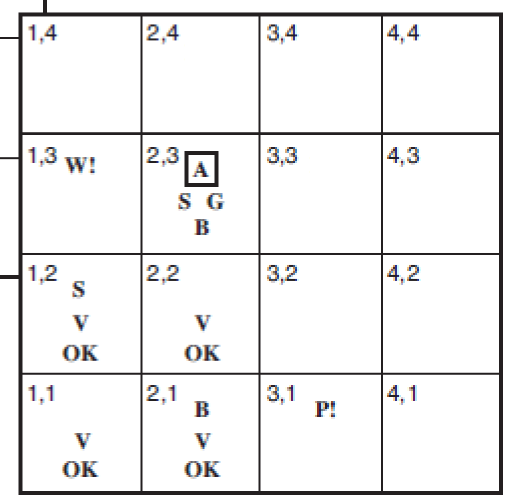
Exploring a Wumpus World







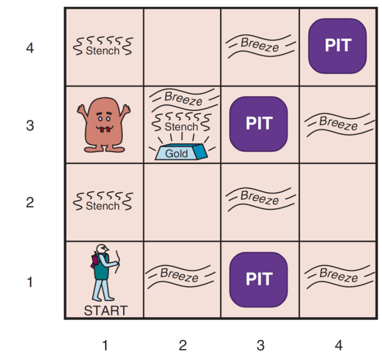


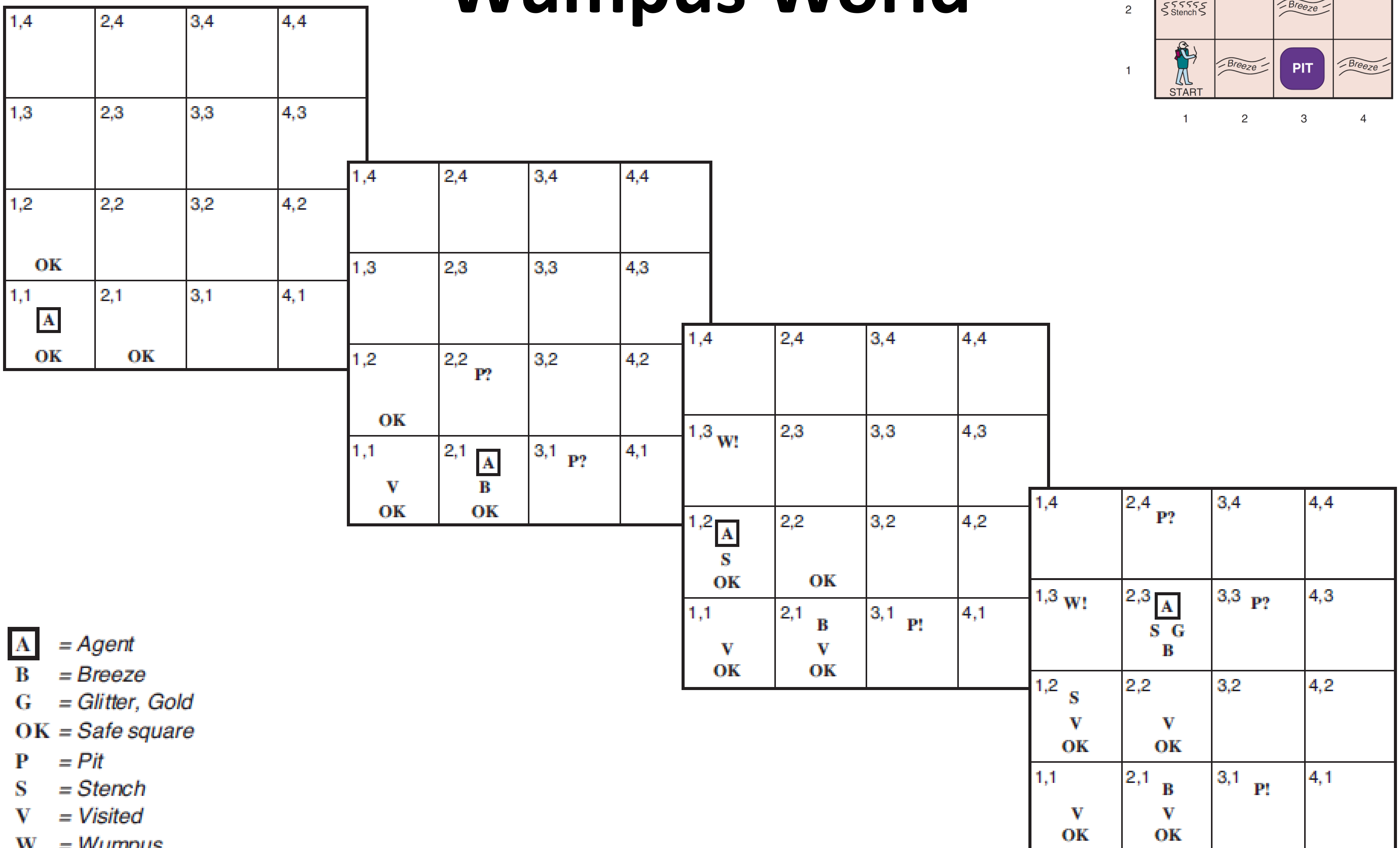


$\boxed{A}$ = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

[*Stench,Breeze,Glitter,None,None*]

Exploring a Wumpus World





$\boxed{A}$ = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

Logic

- Representing the sentences in the KB
  - Syntax: specifies the sentences that are well formed
    - e.g., "$x + y = 4$", not "$x4y +=$"
  - Semantics: assigns meaning to sentences, determining their truthfulness in respect to each **possible world**, or **model**
    - e.g., "$x + y = 4$" is true in a world in which both x and y are 2, but false in a world where they are both 1

- Sentence $\boldsymbol{\alpha}$ is true in a model $\boldsymbol{m}$
  - m **satisfies** α, or m **is a model of** α

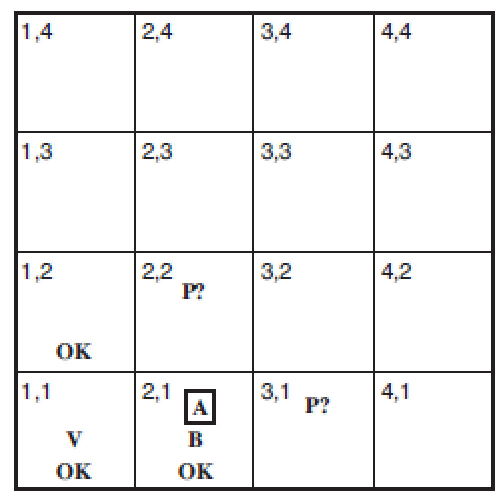
- $M(\alpha)$: the set of all models of α

Entailment

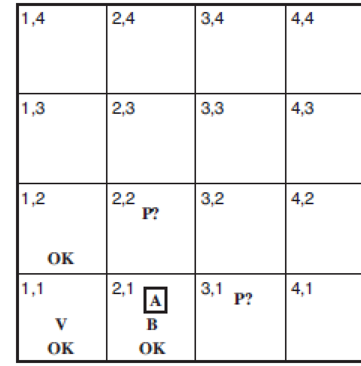
- **Entailment: $\boldsymbol{\alpha} \vDash \boldsymbol{\beta}$**
  - α entails β (or β follows logically from α)
  - $\alpha \vDash \beta$ if and only if $M(\alpha) \subseteq M(\beta)$
    - α is a stronger assertion than β

- Adding knowledge to a KB:
  - $\boldsymbol{KB} \vDash \boldsymbol{\alpha}$

- Example:
  - KB: nothing in [1,1] and a breeze in [2,1]
  - Is there a pit in [1,2], [2,2], or [3,1]?

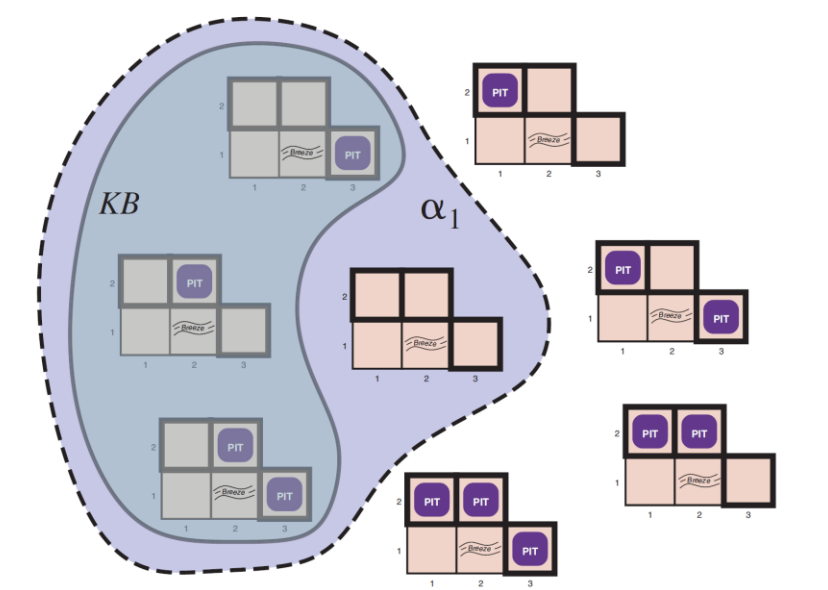


Entailment in the Wumpus World

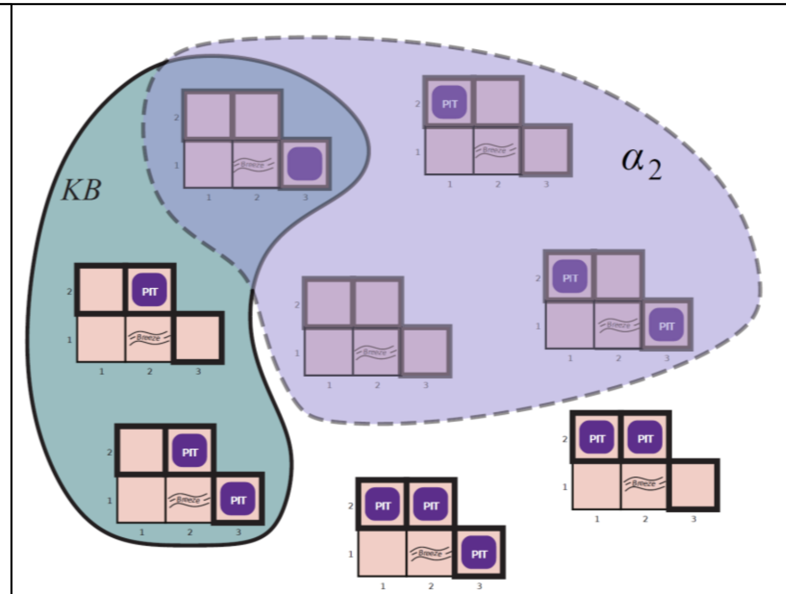


- Is there a pit in [1,2], [2,2], or [3,1]? → $= 2^3 = 8$ states

— models of KB (nothing in [1,1] and a breeze in [2,1])



---- models of α_1(no pit in [1,2])



---- models of α_2(no pit in [2,2])

- In every model in which KB is true, α_1 is also true
  - $KB \vDash \alpha_1$: there is no pit in [1,2]
- In some model in which KB is true, α_2 is false
  - $KB \nvDash \alpha_2$: cannot conclude whether there is a pit in [2,2]

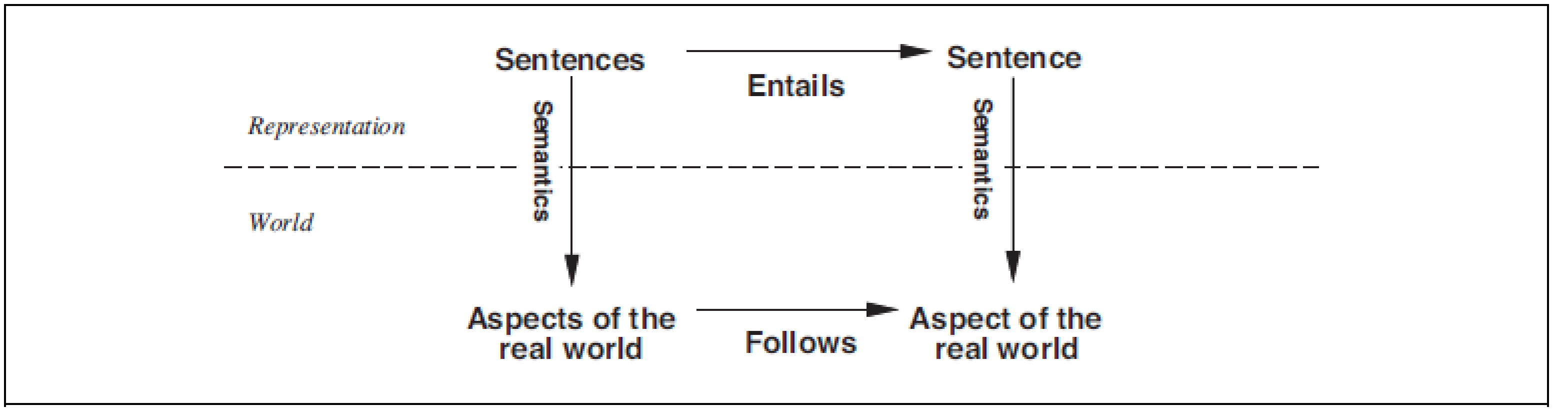
Logical Inference

- Entailment can be applied to derive conclusions: **logical inference**

- Algorithm i that can derive α from KB
  - $KB \vdash_i \alpha_1$
- Properties of inference algorithms:
  - **Soundness** (or **truth preserving**): derive *only* entailed sentences
  - **Completeness**: derive *any* sentence that is entailed

- *If KB is true in the real world, then any sentence α derived from KB by a **sound inference procedure** is also true in the real world*

Correspondence



- The inference procedure:
  - Operates on the syntactic representations (sentences), but *corresponds* to the real-world relationship
  - Constructs new sentences from existing ones
  - To be sound, should entail only sentences representing facts that follow from the facts represented by the KB

Propositional Logic: Syntax

- Symbols:
  - Logical constants $True$ and $False$
  - Propositional symbols such as P and Q
  - Logical connectives: $\wedge \quad \vee \quad \Rightarrow \quad \Leftrightarrow \quad \neg$
  - Parentheses (and)
- Sentences are sequences of symbols, such that:
  - $True$, $False$, P or Q are sentences by themselves (atomic sentences)
  - Complex sentences are constructed from simpler sentences, using parenthesis and logical connectives:

    $\wedge$ (and). A sentence whose main connective is $\wedge$ is called a **conjunction**: $P \wedge (Q \vee R)$

    $\vee$ (or). A sentence whose main connective is $\vee$ is called a **disjunction**: $A \vee (P \wedge Q)$

    $\Rightarrow$ (implies). A sentence in the form $(P \wedge Q \Rightarrow R)$ is called an **implication**

    $\Leftrightarrow$ (if and only if). A sentence in the form $(P \wedge Q) \Leftrightarrow (Q \wedge P)$ is an **equivalence**

    $\neg$ (not). A sentence in the form $\neg P$ is called a **negation** of P
  - Operator precedence: $\neg \quad \wedge \quad \vee \quad \Rightarrow \quad \Leftrightarrow$
    - Sentence $\neg P \vee Q \wedge R \Rightarrow S$ is equivalent to sentence $\big((\neg P) \vee (Q \wedge R)\big) \Rightarrow S$

Propositional Logic: Semantics

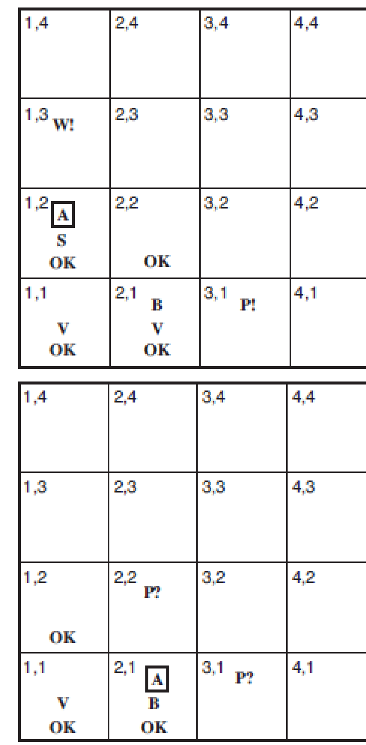
- $True$ represents a true fact; $False$ represents a false fact
- Truth table for the logical connectives:

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*

- The meaning of a complex sentence is derived from the meaning of its parts by a process of decomposition
  - $(P \vee Q) \wedge \neg S$: first determine the meaning of $(P \vee Q)$ and of $\neg S$, then combine the two using the definition of $\wedge$

Propositional Logic: Semantics

- The truth value of every other proposition symbol must be specified directly in the model
  - $m_1 = \{P_{1,2} = false, P_{2,2} = false, P_{3,1} = true\}$



- Evaluated in m_1:
  - $\neg P_{1,2} \land (P_{2,2} \lor P_{3,1})$ gives
    $true \land (false \lor true) = true \land true = true$

- Defining rules of the wumpus world:
  - $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$

Wumpus World Knowledge Base

- Symbols for each [x, y] location:

  $P_{x,y}$ 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 $[x,y]$.

- There is no pit in [1,1]:

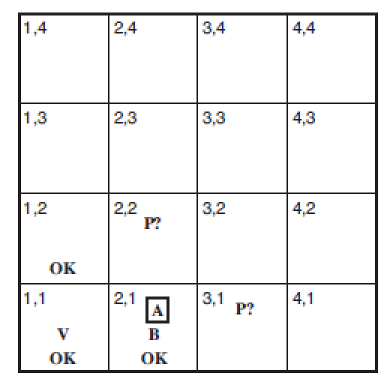
  $R_1: \quad \neg P_{1,1}\ .$

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

  $R_2: \quad B_{1,1} \quad \Leftrightarrow \quad (P_{1,2} \lor P_{2,1})\ .$
  $R_3: \quad B_{2,1} \quad \Leftrightarrow \quad (P_{1,1} \lor P_{2,2} \lor P_{3,1})\ .$

- Breeze percepts for the first two squares visited:

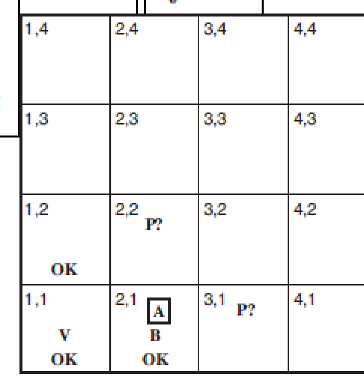
  $R_4: \quad \neg B_{1,1}\ .$
  $R_5: \quad B_{2,1}\ .$



Model Checking through Enumeration

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

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



- KB is true if R_1 through R_5 are true
  - $P_{1,2}$ is always false: there is no pit in [1,2]

Theorem Proving

- If KB and α contain n symbols, there are 2^n models
  - Time complexity: $O(2^n)$
- Can we do without model enumeration?
  - Yes!
- Logical equivalence
- Validity and satisfiability
- **Inference rules**

Logical Equivalence

- Two sentences α e β are **logically equivalent** if they are true in the same set of models: $M(\alpha) = M(\beta)$
- In other words: $\alpha \equiv \beta$ if and only if $\alpha \vDash \beta$ and $\beta \vDash \alpha$

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

Validity and Satisfiability

- A sentence is **valid** if it is true in *all* models
  - **Tautology**: a necessarily true sentence
    - $P \vee \neg P$
  - **Deduction** theorem: $\alpha \vDash \beta$ if and only if $(\alpha \Rightarrow \beta)$ is valid

- A sentence is **satisfiable** if it is true in *some* model
  - $KB = (R_1, R_2, R_3, R_4, R_5)$ is satisfiable because it is true in three models

- α is valid iff $\neg\alpha$ is **unsatisfiable**
- α is satisfiable iff $\neg\alpha$ is not valid
- $\alpha \vDash \beta$ if and only if the sentence $(\alpha \wedge \neg\beta)$ is unsatisfiable
  - Principle of the proof by contradiction

Inference Rules

- Truth tables can be used to test for valid sentences
  - If the sentence is true in every row, then it is valid
  - $\big((P \vee H) \wedge \neg H\big) \Rightarrow P$

P	H	$P \vee H$	$(P \vee H) \wedge \neg H$	$\big((P \vee H) \wedge \neg H\big) \Rightarrow P$
False	*False*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*
True	*False*	*True*	*True*	*True*
True	*True*	*True*	*False*	*True*

- **Inference rules** allow us to make inference without the need for building truth tables
  - An inference rule is sound if its conclusion is true whenever its premises are true

Inference Rules

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

- **Modus Ponens**: from an implication and its premise, we can infer the conclusion

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n}{\alpha_i}$$

- **And-Elimination**: from a conjunction, we can infer any of its conjuncts

$$\frac{\alpha_1, \alpha_2, \dots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n}$$

- **And-Introduction**: from a list of sentences, we can infer their conjunction

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

- **Or-Introduction**: from a sentence, we can infer its disjunction with anything else

$$\frac{\neg\neg\alpha}{\alpha}$$

- **Double-Negation Elimination**: from a doubly negated sentence, we can infer the sentence

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

- **Unit Resolution**: from a disjunction, if one of the disjuncts is false, we can infer that the other one is true
- **Resolution**: since β cannot be both true and false, one of the other disjuncts must be true

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

Inference and Proofs

- Searching for proofs is an alternative to enumerating models
- Finding a proof can be more efficient because the proof can ignore irrelevant propositions, no matter how many of them there are
- **Monotonicity**: the set of entailed sentences can only increase as information is added to the knowledge base
  - if $KB \vDash \alpha$, then $KB \wedge \beta \vDash \alpha$

Resolution

- Full **resolution** rule:

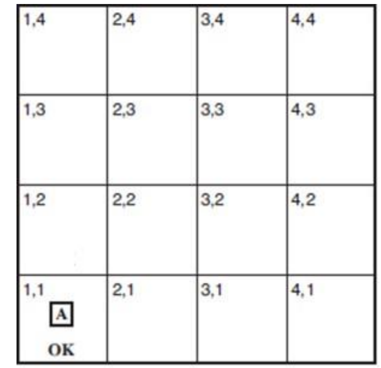
$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

  - where l_i and m_j are complementary literals

- Need all clauses in **conjunctive normal form (CNF)**
  (check the Logic Programming course)

- Resolution is **complete**
  - *If a set of clauses is unsatisfiable, then the resolution closure of those clauses contains the empty clause*

Resolution Example



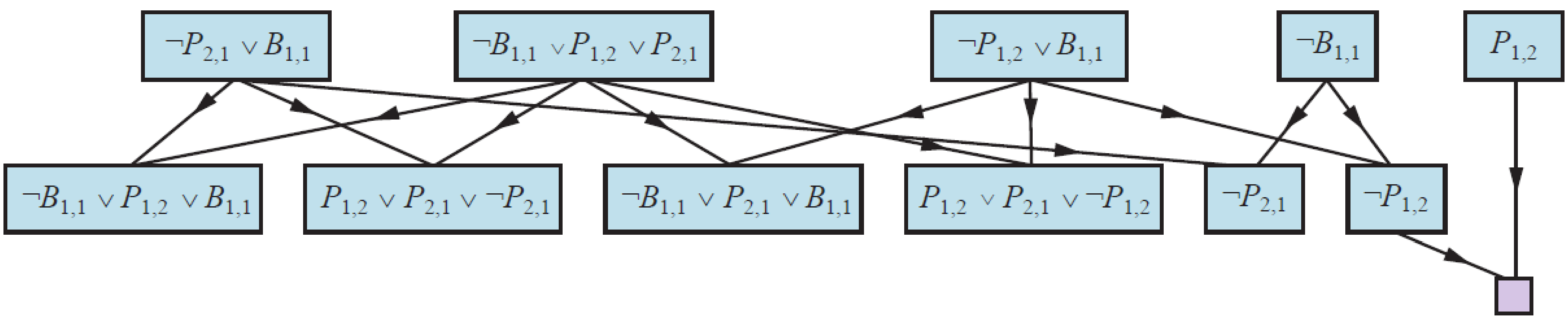
- Agent in [1,1]

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

- Prove $KB \vDash \neg P_{1,2}$
- Convert $(KB \wedge P_{1,2})$ into CNF

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

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



Horn Clauses

- In many cases, the KB can be expressed through **Horn clauses**
  - Implications in the form: $P_1 \wedge P_2 \wedge \cdots \wedge P_n \Rightarrow Q$
  - Special cases:
    - If Q is $False$, we get a sentence in the form $\neg P_1 \vee \neg P_2 \vee \cdots \vee \neg P_n$

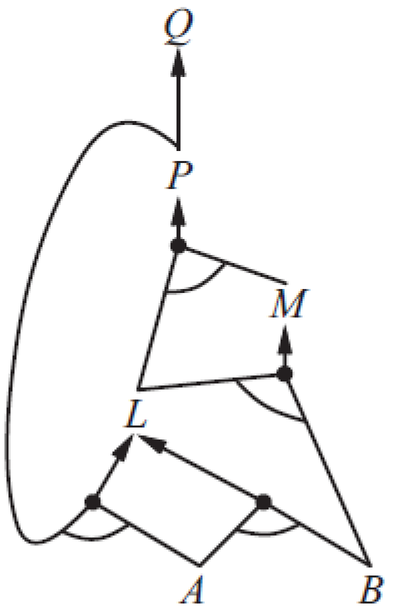
      (aka a *query*)
    - If $n = 1$ and $P_1 = True$, we get $True \Rightarrow Q$, which is the same as Q
      - (aka a *fact*)

- Inference with Horn clauses can be done through the **forward-chaining** and **backward-chaining** algorithms
  - These algorithms run in linear time

Forward Chaining

- Fire any rule whose premises are satisfied by the *KB*
- Add its conclusion to the *KB*

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



- **Data-driven** reasoning: start from the known data
  - Derive conclusions from incoming percepts, without a specific query in mind

Forward Chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

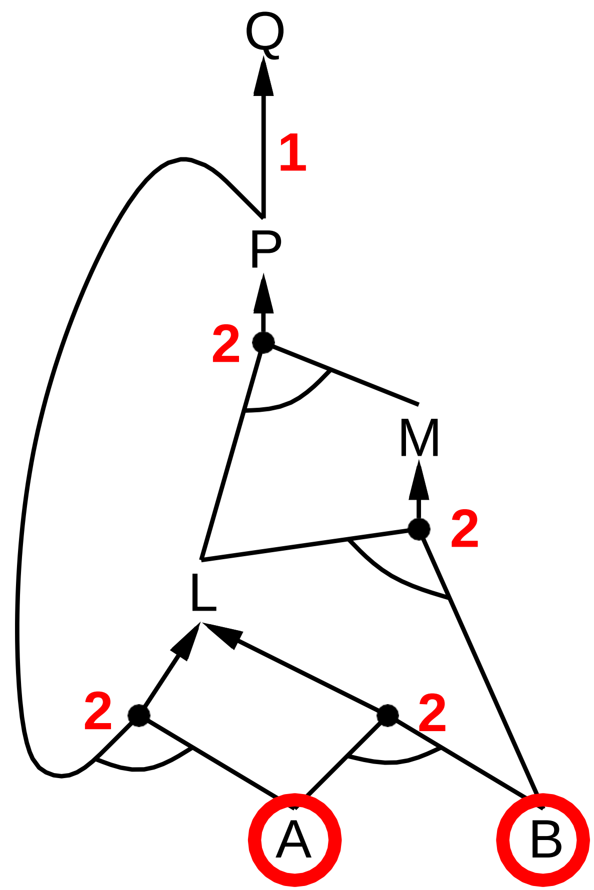
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward Chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

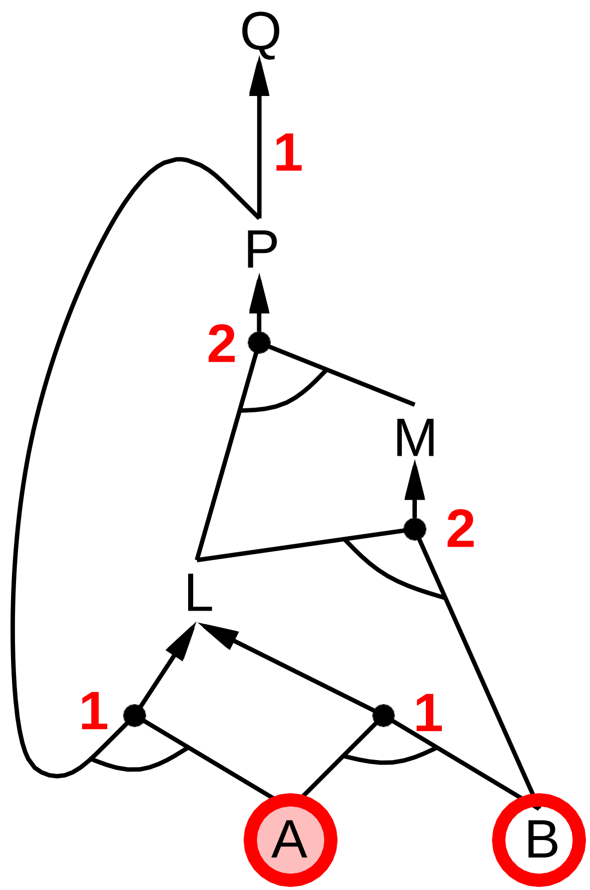
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward Chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

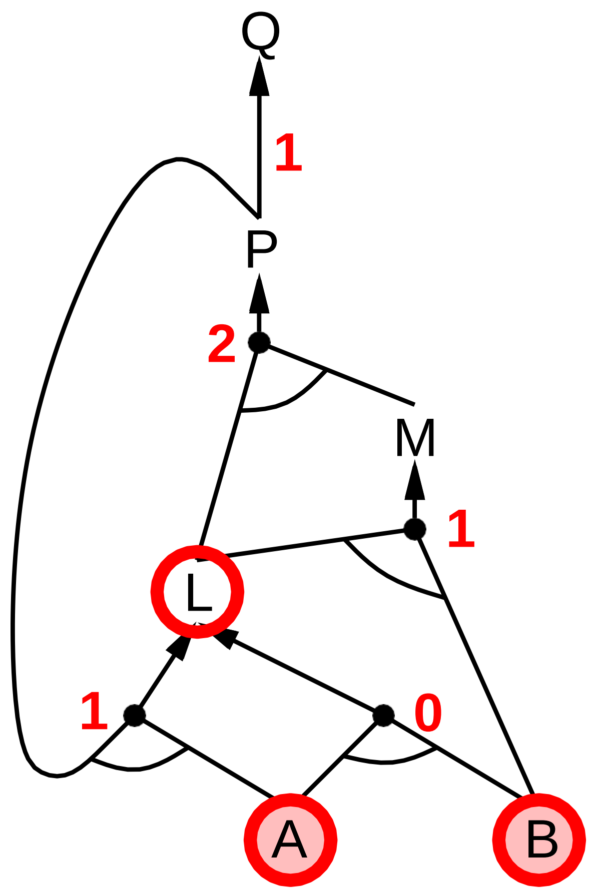
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward Chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

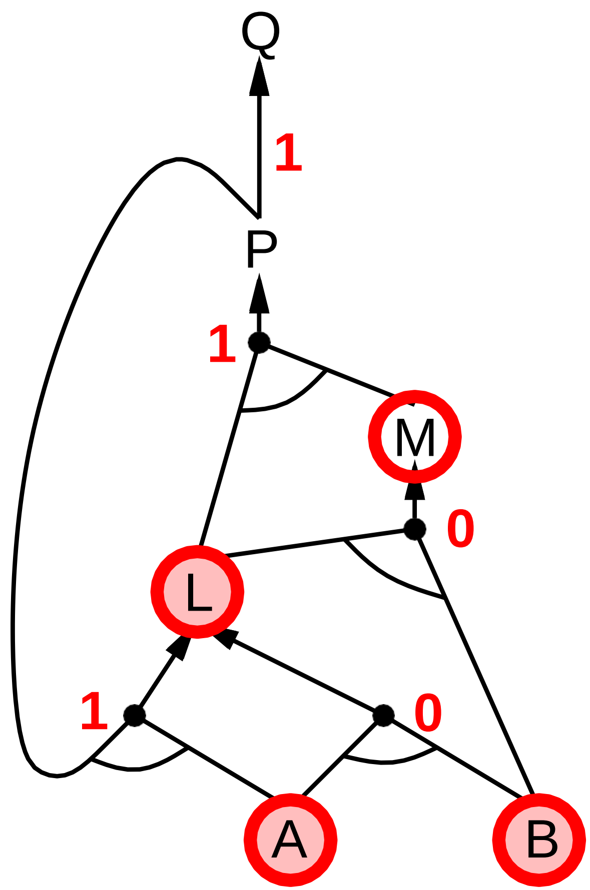
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

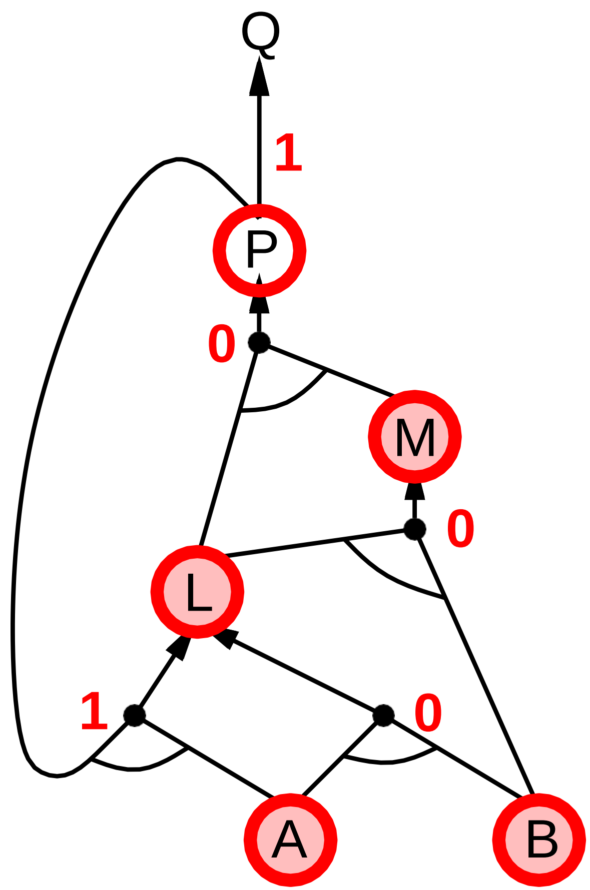
A

B



Forward Chaining

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



Forward Chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

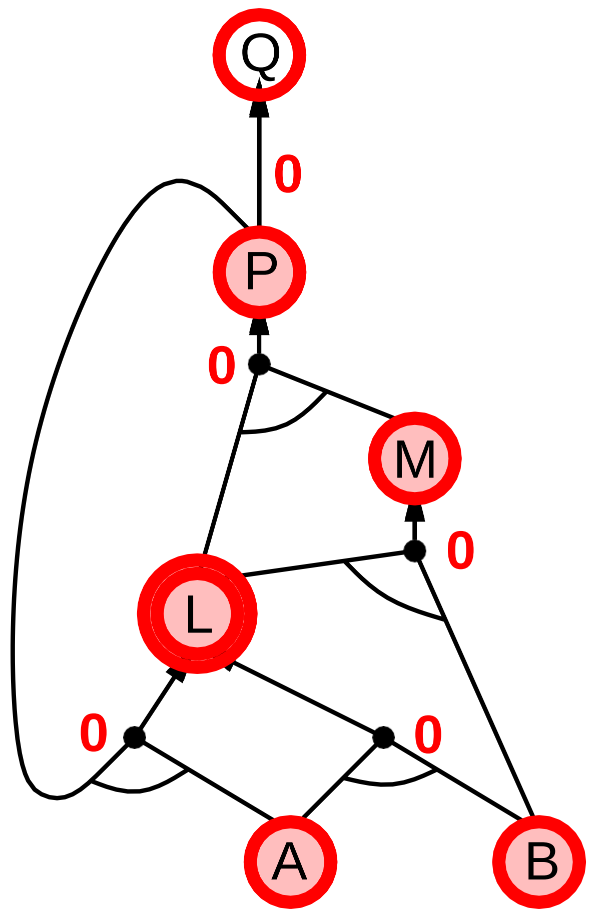
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward Chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

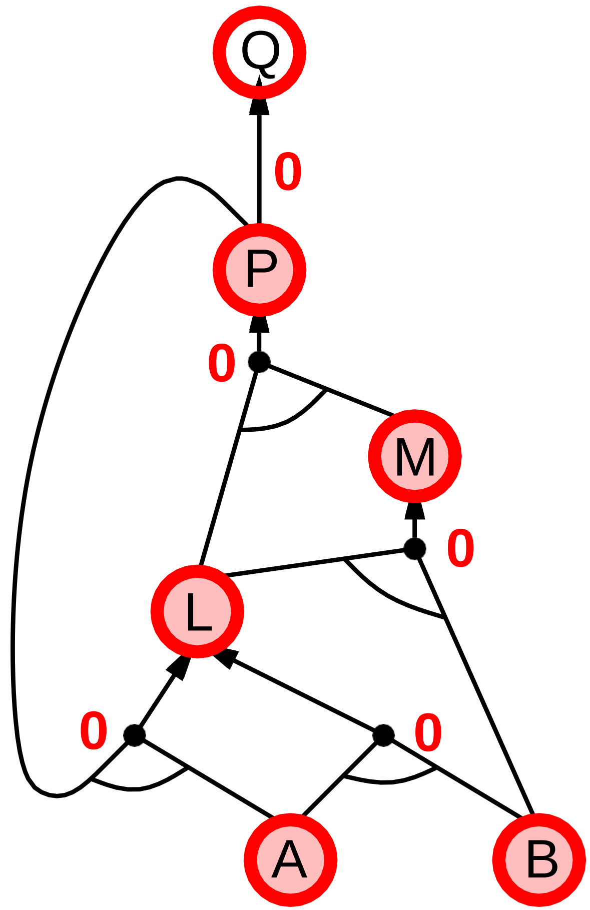
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward Chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

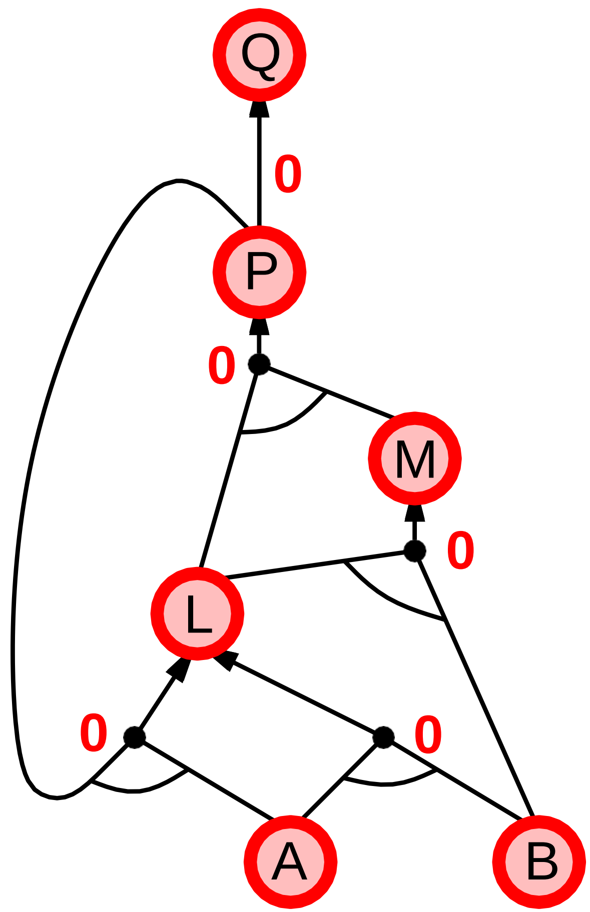
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

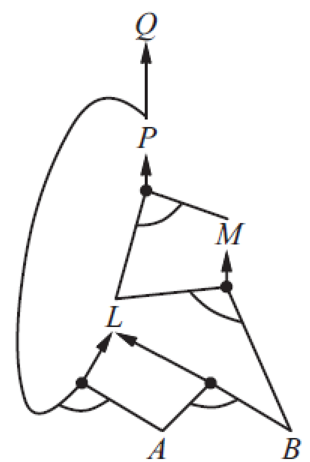
B



Backward Chaining

- Work backwards from a query q
  - If q is known to be true, no work needed
  - Otherwise find implications in the KB whose conclusion is q
    - Try to prove the premises of one of such implications (through backward chaining)

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



- **Goal-directed** reasoning: start from a query
  - Derive answers to specific goals
  - Often, the cost of backward chaining is much less than linear in the size of the KB, because the search process focuses on the query

Backward Chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

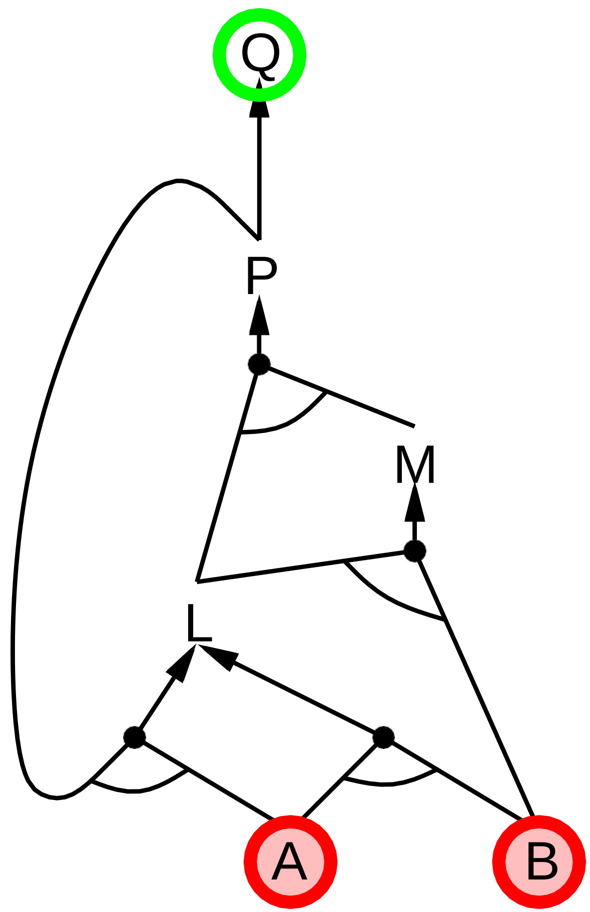
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

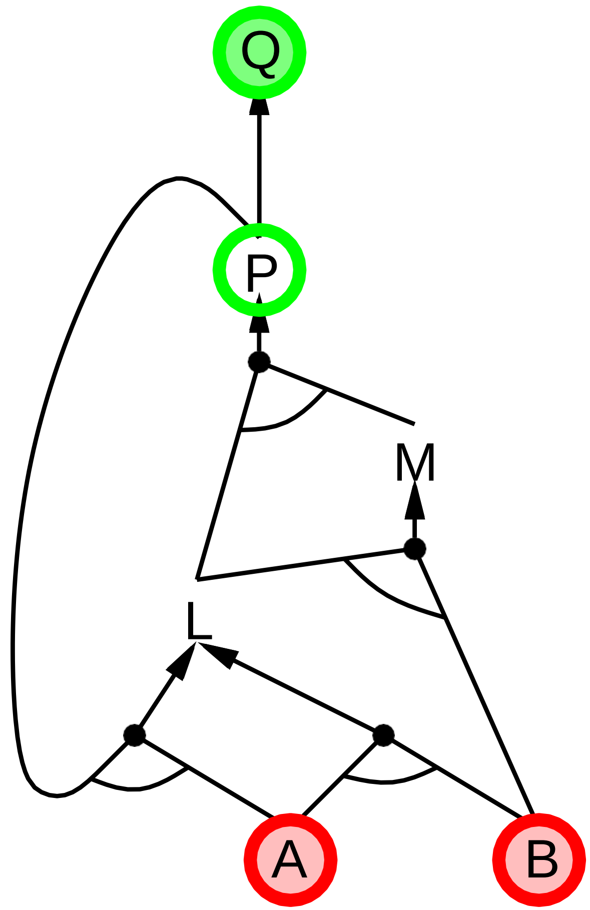
A

B



Backward Chaining

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



Backward Chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

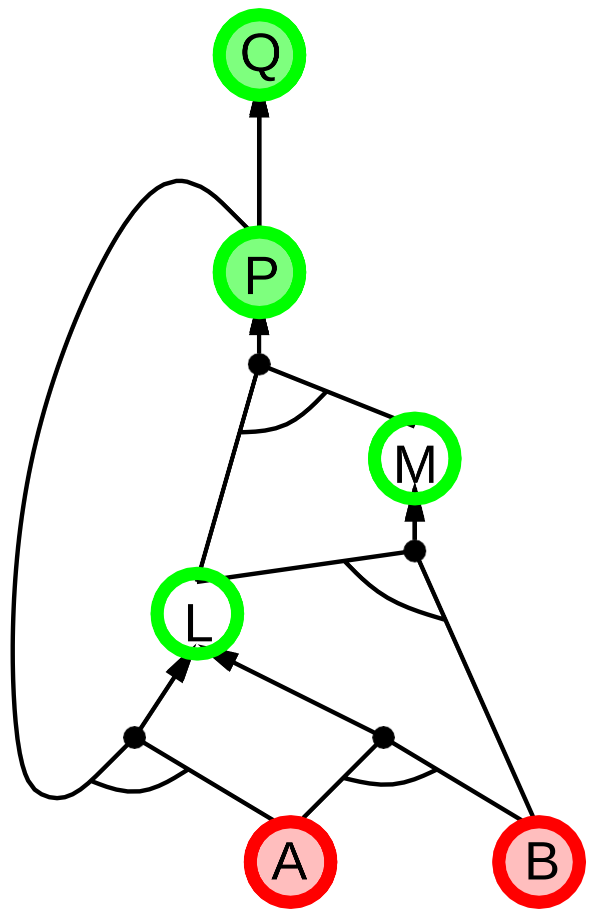
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward Chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

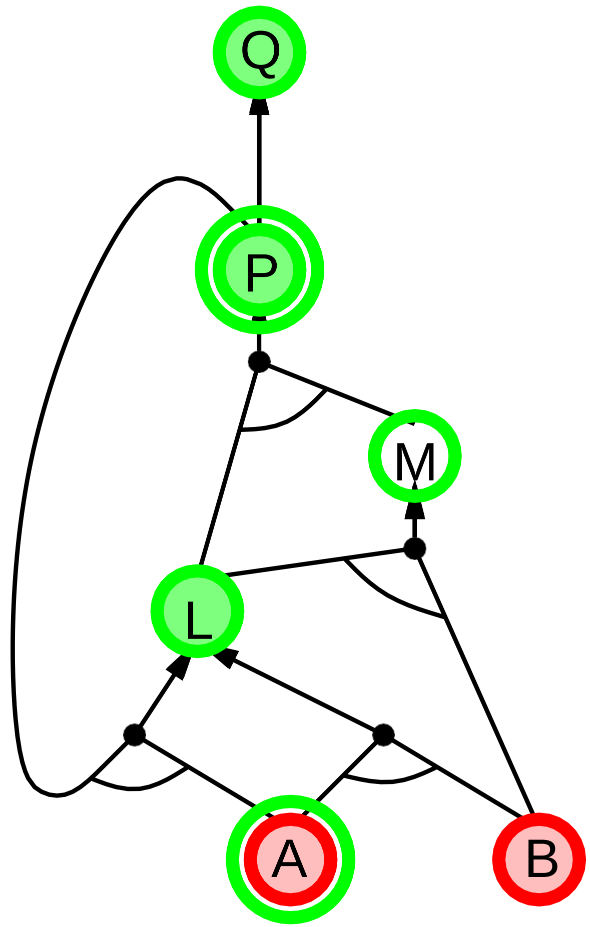
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward Chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

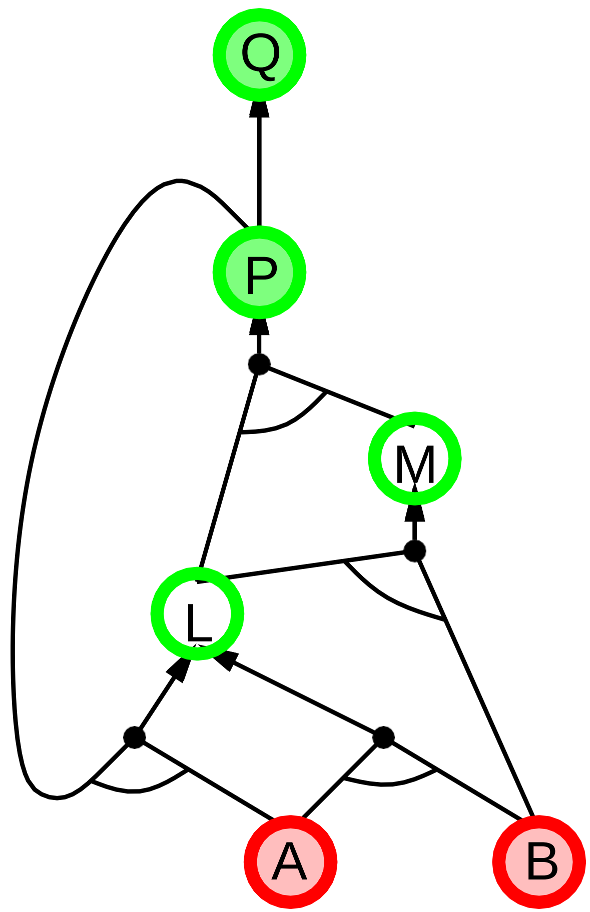
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

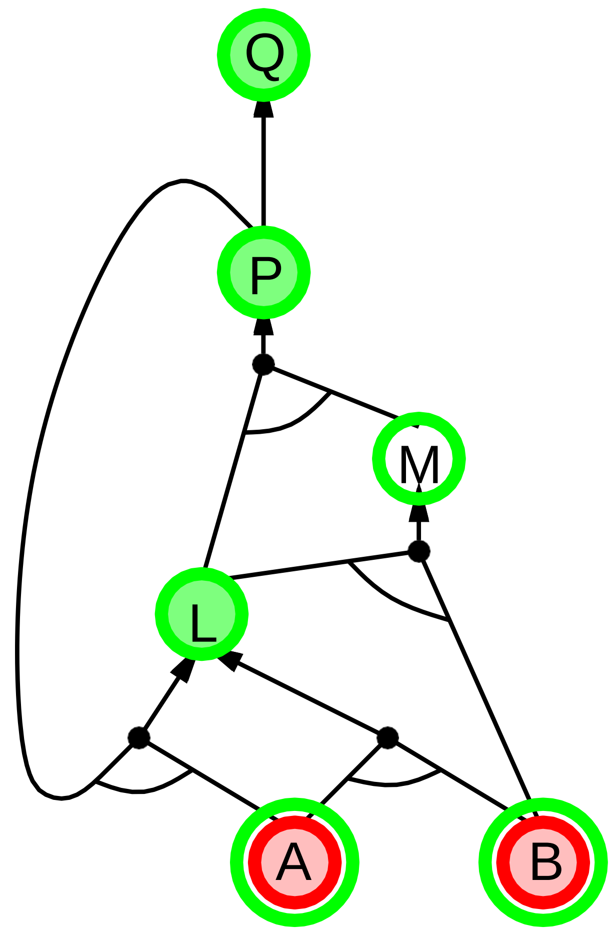
A

B



Backward Chaining

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



Backward Chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

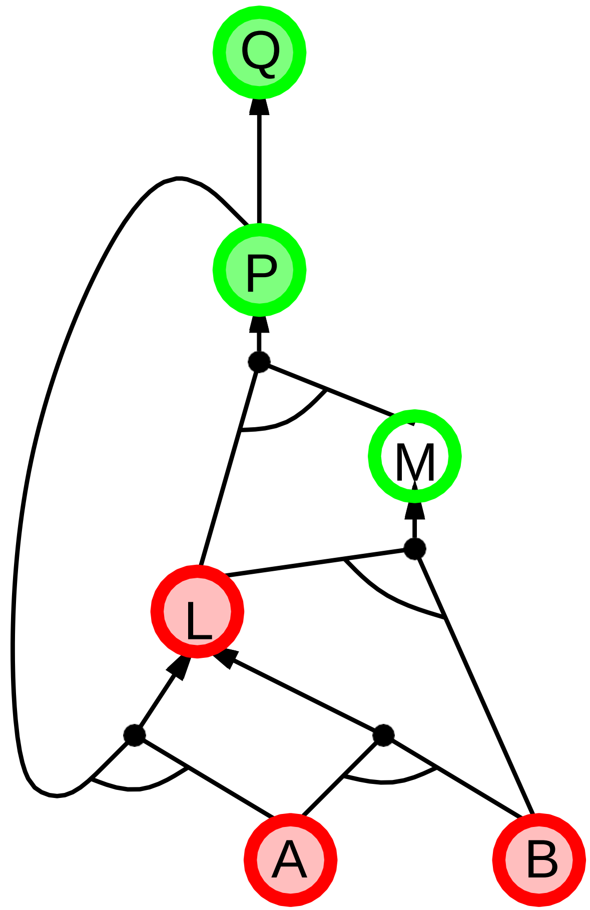
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward Chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

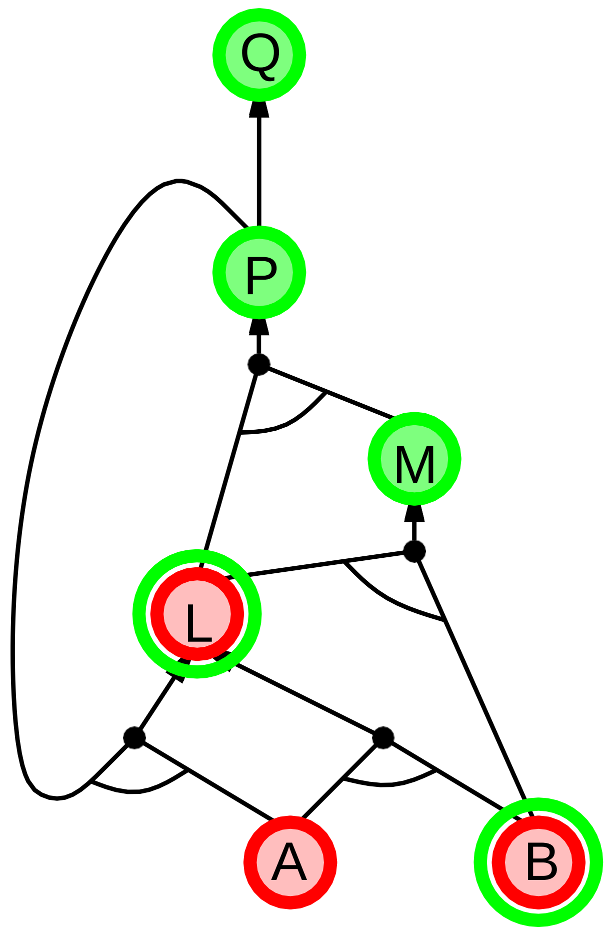
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward Chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

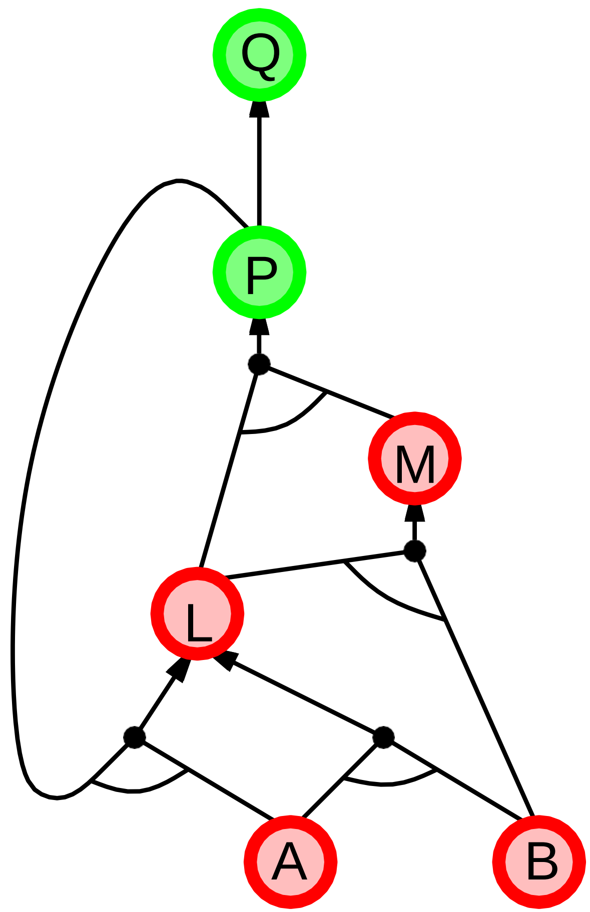
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward Chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

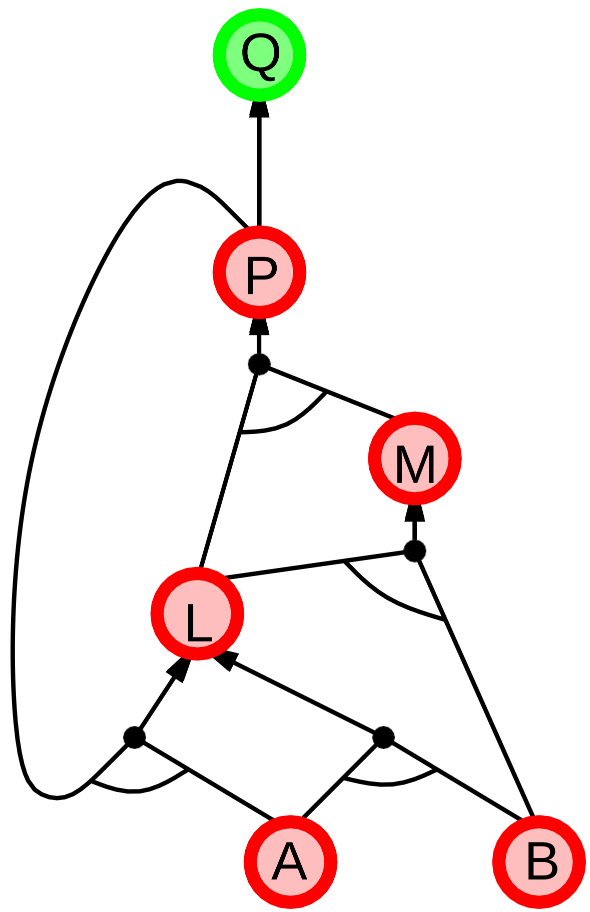
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Backward Chaining

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

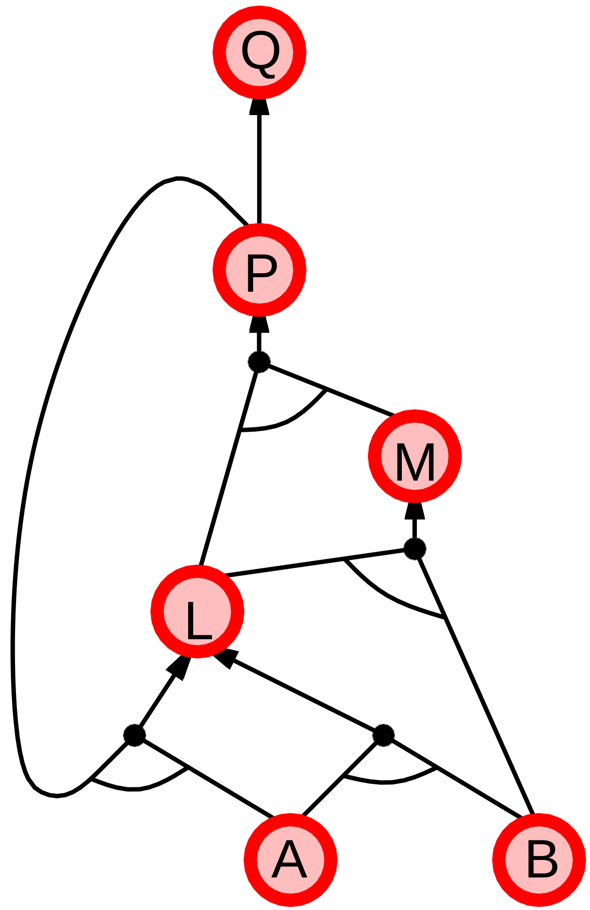
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Logics: Ontological and Epistemological Commitments

Language	Ontological Commitment (What exists in the world)	Epistemological Commitment (What an agent believes about facts)
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0, 1]$
Fuzzy logic	facts with degree of truth $\in [0, 1]$	known interval value

First Order Logic

- The world consists of objects with properties and relations

- Symbols
  - Constants (represent objects): $A, B, C, John, FatherOfJohn, \ldots$
  - Relations: $Round, Brother, LowerThan, \ldots$
  - Functions (relations with one possible value only): $Cosine, Father, LeftLeg, \ldots$
- Variables: $a, x, s, \ldots$
- Terms: made of constants, variables or functions
  - $John, x, LeftLeg(John), \ldots$

- Atomic sentences: predicate and list of terms
  - $Brother(Richard, John)$
  - $Married\big(Father(Richard), Mother(John)\big)$
- Complex sentences: use logical connectives
  - $\neg \quad \wedge \; \vee \quad \Rightarrow \quad \Leftrightarrow$

Quantifiers (∀ and ∃)

- **Universal** (∀): express properties of collections of objects
  - "Every cat is a mammal": $\forall x\ Cat(x) \Rightarrow Mammal(x)$

- **Existential** (∃): state something about some object, without naming it
  - "John has got a married sister": $\exists x\ Sister(x, John) \wedge Married(x)$

- Nested quantifiers:
  - $\forall x, y \equiv \forall x\ \forall y \equiv \forall y\ \forall x$
  - $\forall x\ \exists y \neq \exists y\ \forall x$
    - $\forall x\ \exists y\ Likes(x, y)$ "Everyone likes somebody."
    - $\exists y\ \forall x\ Likes(x, y)$ "There is someone who everybody likes."
    - $\forall y\ \exists x\ Likes(x, y)$ "Everyone has someone who likes her/him."
    - $\exists x\ \forall y\ Likes(x, y)$ "There is someone that likes everybody."

Quantifiers (∀ and ∃)

- Connections between ∀ and ∃, through negation (De Morgan laws)
  - $\forall x\ \neg Likes(x, Exams) \equiv \neg\exists x\ Likes(x, Exams)$
  - $\forall x\ Likes(x, Health) \equiv \neg\exists x\ \neg Likes(x, Health)$
  - $\exists x\ \neg Likes(x, Soup) \equiv \neg\forall x\ Likes(x, Soup)$
  - $\exists x\ Likes(x, Soup) \equiv \neg\forall x\ \neg Likes(x, Soup)$

Inference Rules for Quantifiers

- Consist of substituting variables for specific objects
  - $\boldsymbol{SUBST(\theta, \alpha)}$: apply substitution θ to sentence α
    - $SUBST(\{x/John, y/Cabbage\}, Likes(x, y)) = Likes(John, Cabbage)$

- Universal Instantiation:
  - For any sentence α, variable v and ground term g:

$$\frac{\forall v\ \alpha}{SUBST(\{v/g\}, \alpha)}$$

    - From $\forall x\ Likes(x, Icecream)$, we can use substitution $\{x/John\}$ and infer $Likes(John, Icecream)$

Inference Rules for Quantifiers

- Existential Instantiation:
  - For any sentence α, variable v and constant k not yet used in the KB:

$$\frac{\exists v\ \alpha}{SUBST(\{v/k\}, \alpha)}$$

  - We are giving a name to the object that satisfies the existential condition!
  - From $\exists x\ Killed(x, Victim)$ we may infer $Killed(Assassine, Victim)$, provided that $Assassine$ is not the name for any other object

Generalized Modus Ponens

- For atomic sentences p_i, p_i' and q, if there is a substitution θ such that $SUBST(\theta, p_i') = SUBST(\theta, p_i)$ for every i:

$$\frac{p_1', p_2', \dots, p_n', (p_1 \land p_2 \land \cdots \land p_n \Rightarrow q)}{SUBST(\theta, q)}$$

  - That is, if there is a substitution that makes the premises in the implication identical to sentences in the KB, we can infer the conclusion of the implication after applying the substitution

- Makes use of the **unification** algorithm, which takes two sentences and returns a substitution that makes them identical (if one exists)

Resolution

- For two disjunctions of any size, if one of the disjuncts in a clause unifies with the negation of a disjunct in the other clause, then we can infer the disjunction of the remaining disjuncts:

$$\left.\begin{array}{c} a \vee h \vee c \\ d \vee \neg h \vee e \end{array}\right\rangle \quad a \vee c \vee d \vee e$$

- For atomic sentences p_i and q_i, where $UNIFY(p_j, \neg q_k) = \theta$:

$$\frac{\begin{array}{c} p_1 \vee \cdots \vee p_j \vee \cdots \vee p_m \\ q_1 \vee \cdots \vee q_k \vee \cdots \vee q_n \end{array}}{SUBST(\theta, p_1 \vee \cdots \vee p_{j-1} \vee p_{j+1} \vee \cdots \vee p_m \vee q_1 \vee \cdots \vee q_{k-1} \vee q_{k+1} \vee \cdots \vee q_n)}$$

- Any sentence in first-order logic can be converted into the form of the premises in the resolution rule: **conjunctive normal form (CNF)**

Resolution Proof

- Proof by contradiction: to prove P, assume P is false (add $\neg P$ to the KB)

- Example:

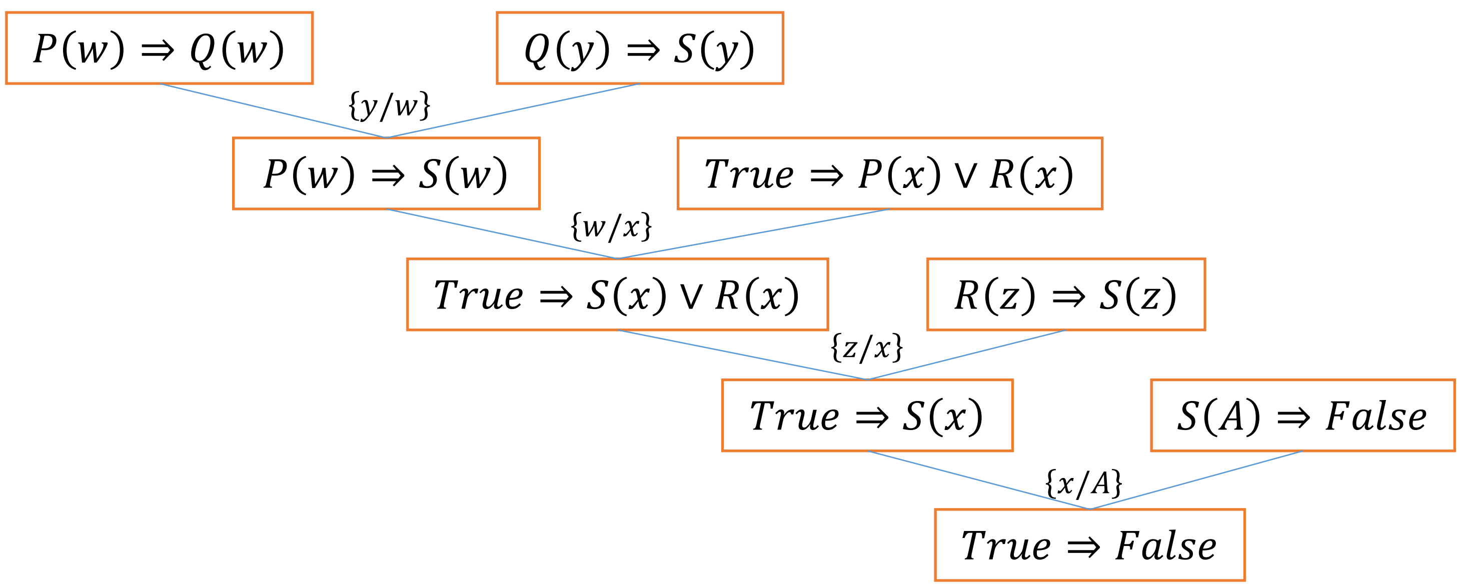
$$
\begin{array}{ll}
\text{C1: } \neg P(w) \vee Q(w) & \equiv P(w) \Rightarrow Q(w) \\
\text{C2: } P(x) \vee R(x) & \equiv True \Rightarrow P(x) \vee R(x) \\
\text{C3: } \neg Q(y) \vee S(y) & \equiv Q(y) \Rightarrow S(y) \\
\text{C4: } \neg R(z) \vee S(z) & \equiv R(z) \Rightarrow S(z)
\end{array}
$$

  - Prove $S(A)$:

$$
\begin{array}{ll}
\text{C5: } \neg S(A) & \equiv S(A) \Rightarrow False
\end{array}
$$

Resolution Proof

$P(w) \Rightarrow Q(w)$
$True \Rightarrow P(x) \vee R(x)$
$Q(y) \Rightarrow S(y)$
$R(z) \Rightarrow S(z)$
$S(A) \Rightarrow False$



Knowledge Engineering

Intelligent Systems

Exhibit intelligent behavior

Knowledge Based Systems

Use explicit domain knowledge, stored separately

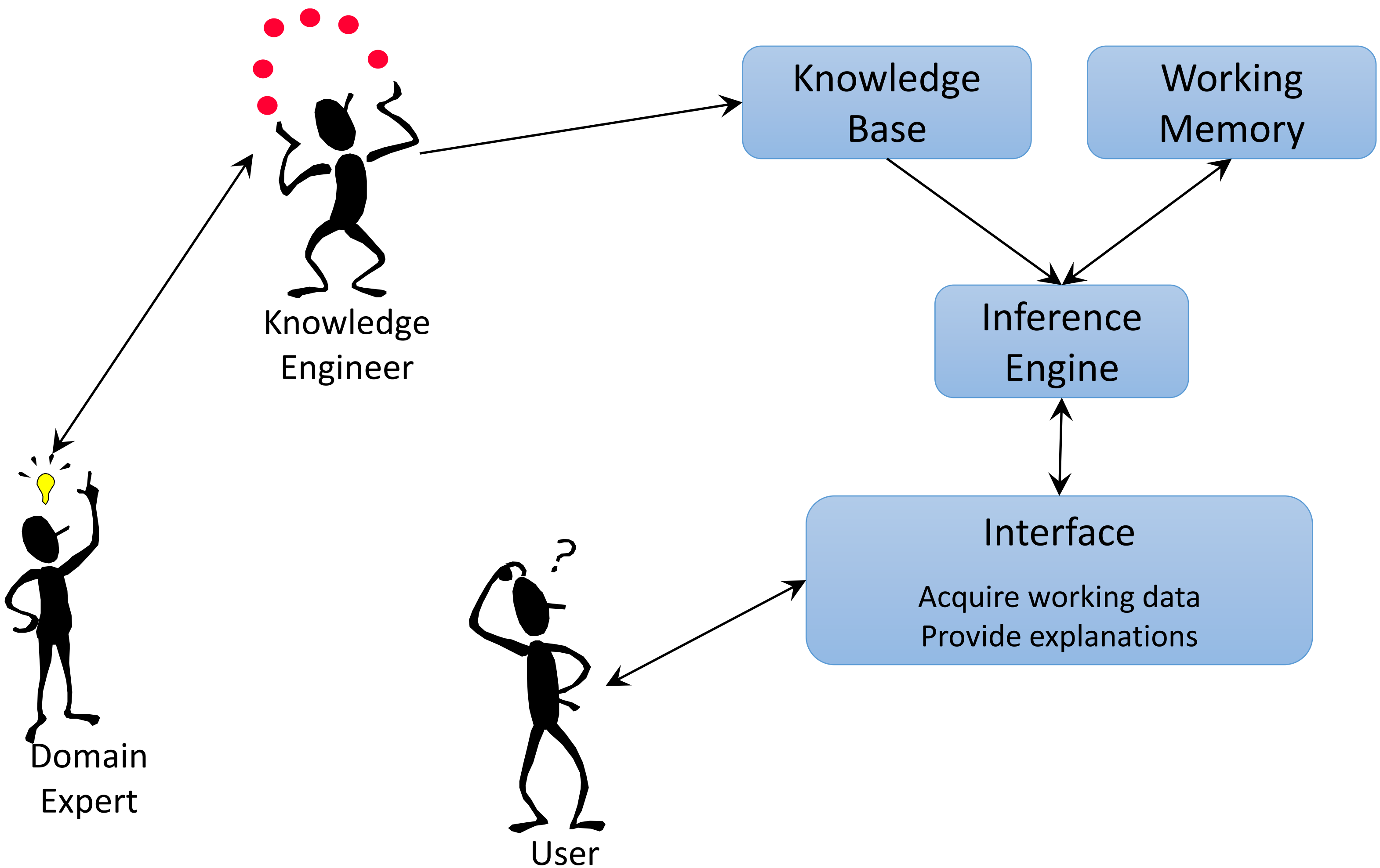
Expert Systems

Use expert knowledge to solve difficult real-world problems, replacing the human expert

Expert Systems

- Main advantages:
  - **Availability**: expertise becomes permanently and quickly available
  - Higher **reliability**: a computer will always give you the same answer
  - **Explainability**: the reasoning process can be traced to check the correctness of the decision
- Main tasks:
  - **Knowledge acquisition**: acquiring (expert) knowledge regarding problem solving in a specific domain
  - **Knowledge representation**: represent the knowledge in a computable representation language
  - **Reasoning control** and **explanation**
- Some application domains:
  - Chemistry (DENDRAL, ...), Electronics (ACE, ...), Medicine (MYCIN, ...), Engineering (REACTOR, ...), Geology (PROSPECTOR, ...), Computer systems (XCON, ...), …

Components of an Expert System



Rule Chaining in Expert Systems

- Backward chaining
  - Diagnosis (e.g., MYCIN) or identification problems
  - There is a moderate number of possible answers
  - The system will try to prove or refute each possible answer, adding the needed information during execution
  - It is easier to provide explanations, based on the chain of reasoning employed

- Forward chaining
  - Prognostics, control, or configuration problems (e.g., XCON)
  - The combinatorial explosion of the available data generates a virtually infinite number of possible answers
  - These kinds of systems are known as **production systems** (their rules *produce* new data as output)