Chapter 7: Logical Agents
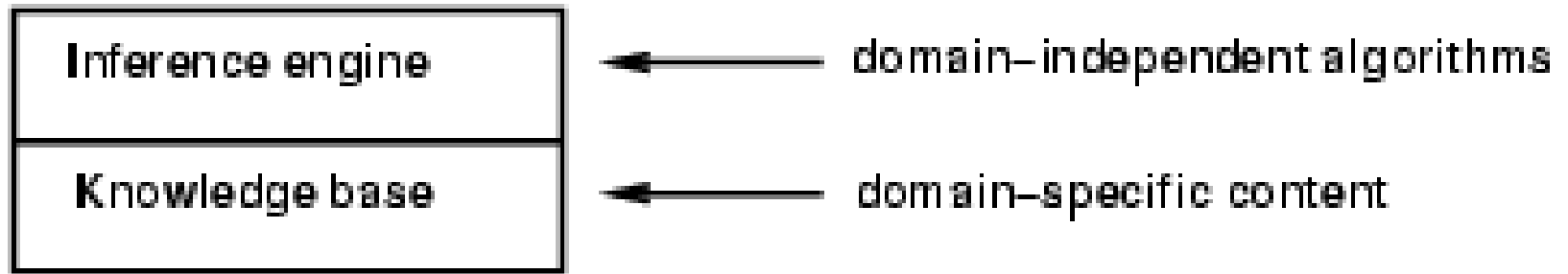
Dr. Daisy Tang

In which we design agents that can form representations of the world, use a process of inference to derive new representations about the world, and use these new representations to deduce what to do.

Motivation of Knowledge-Based Agents

- Humans know things and do reasoning, which are important for artificial agents

- Knowledge-based agents benefit from knowledge expressed in general forms, combining information to suit various purposes

- Knowledge and reasoning is important when dealing with partially observable environments

- Understanding natural language requires inferring hidden states

- Flexibility for accepting new tasks

Knowledge Bases

```
┌──────────────────────┐
│  Inference engine    │  ◄──────  domain–independent algorithms
├──────────────────────┤
│  Knowledge base      │  ◄──────  domain–specific content
└──────────────────────┘
```

- Knowledge base = set of sentences in a formal language

- Declarative approach to building an agent (or other system):
  - **Tell** it what it needs to know
  - Then it can **Ask** itself what to do - answers should follow from the KB

- Both "Tell" and "Ask" involve inference – deriving new sentences from old

A Simple Knowledge-Based Agent

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

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

- Details hidden in three functions
- Similar to agent with internal state in Chapter 2

- Agents can be viewed at the knowledge level
  i.e., what they know, regardless of how they are implemented

- Or at the implementation level
  - i.e., data structures in KB and algorithms that manipulate them

Agent's Requirements

- The agent 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

Our New Toy Problem – Wumpus World
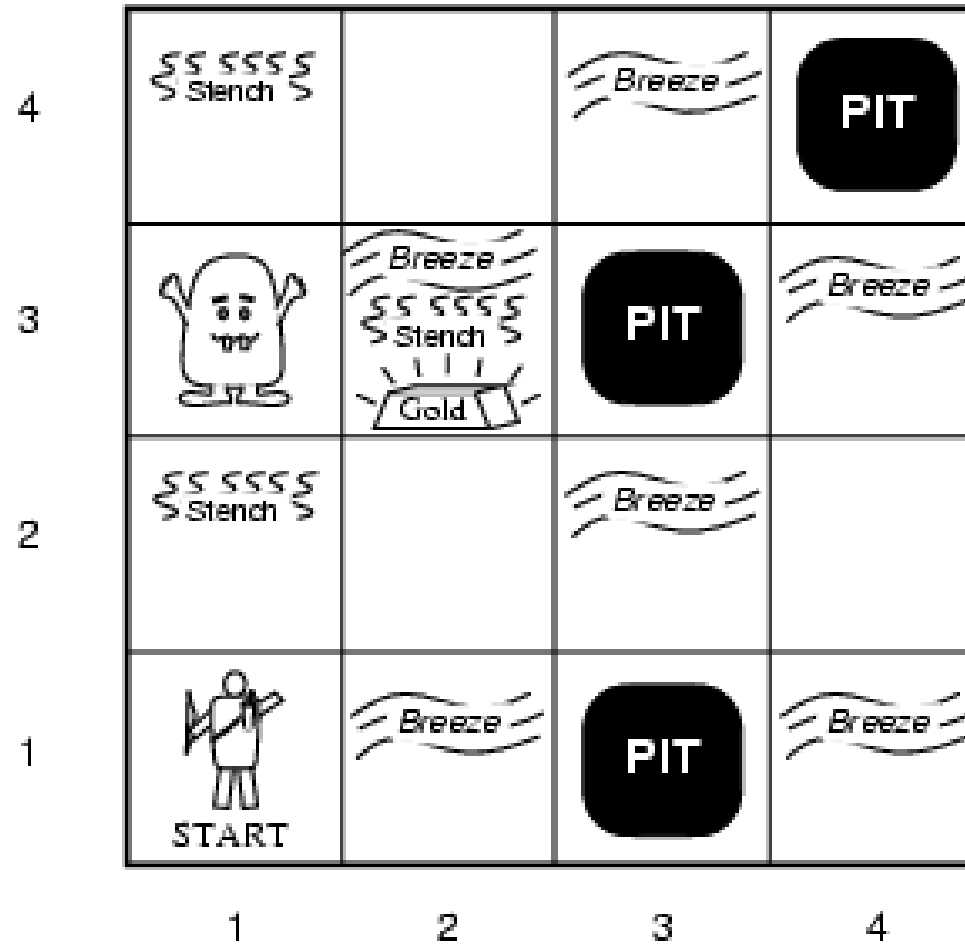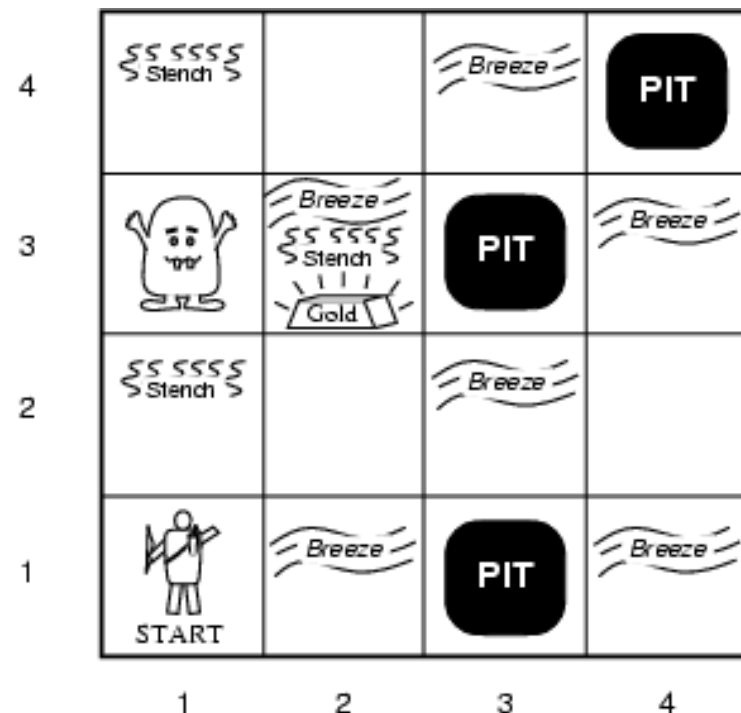
Wumpus World PEAS

- **Performance measure**
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow

- **Environment**
  - 4x4 grid of rooms
  - Agent always starts at [1,1]
  - Wumpus and gold randomly chosen
  - Each square other than start is pit with 0.2 probability
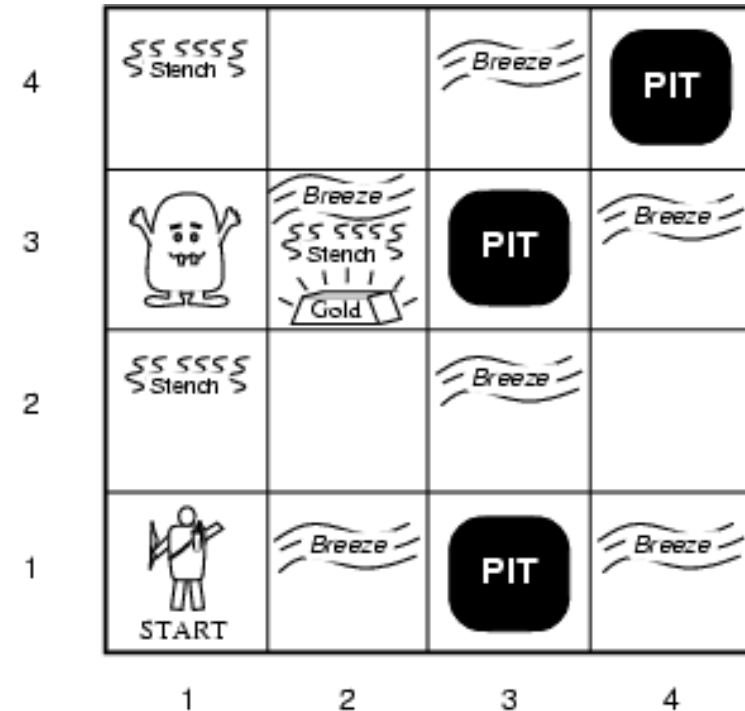
Wumpus World PEAS

- **Actuators:** Left turn, Right turn, Forward, Grab, Release, Shoot, Climb
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square
  - Climbing out of the cave from [1, 1]

- **Sensors (percepts):** Stench, Breeze, Glitter, Bump, Scream
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Bump when agent walks into a wall
  - When wumpus is killed, it screams

Wumpus World Properties

- ## Fully observable?
  - No – only local perception
- ## Deterministic?
  - Yes – outcomes exactly specified
- ## Episodic?
  - No – sequential at the level of actions
- ## Static?
  - Yes – Wumpus and Pits do not move
- ## Discrete?
  - Yes
- ## Single-agent?
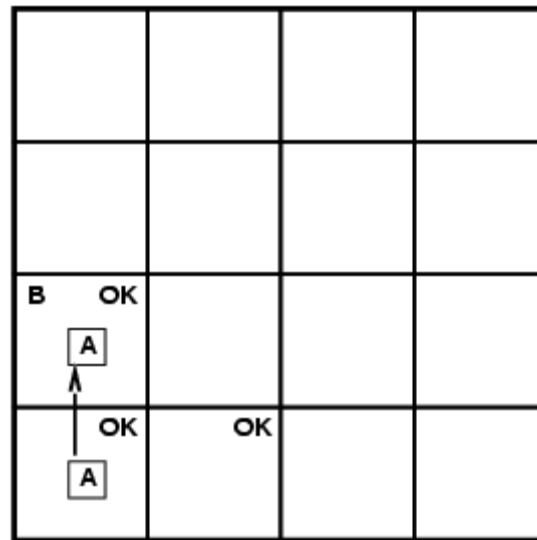 - Yes – Wumpus is essentially a natural feature
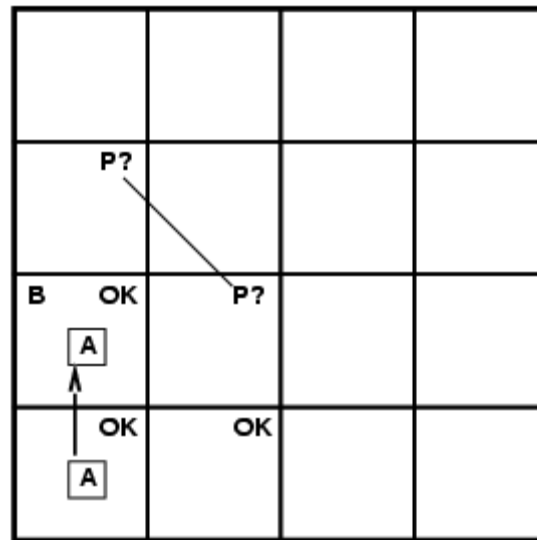
Exploring a Wumpus World

percept: [none, none, none, none, none]

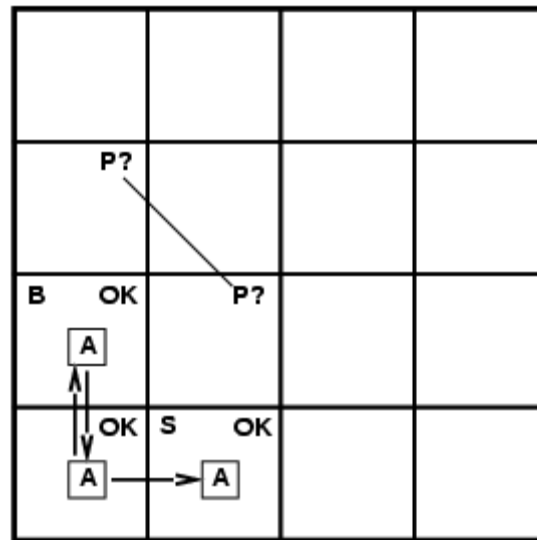# Exploring a Wumpus World

percept: [none, breeze, none, none, none]

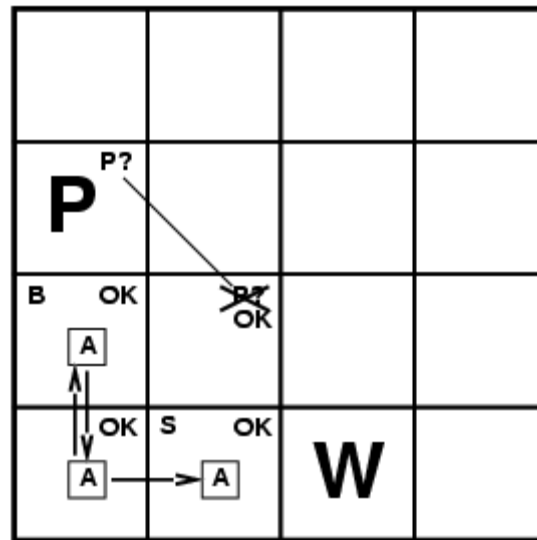# Exploring a Wumpus World

percept: [none, breeze, none, none, none]
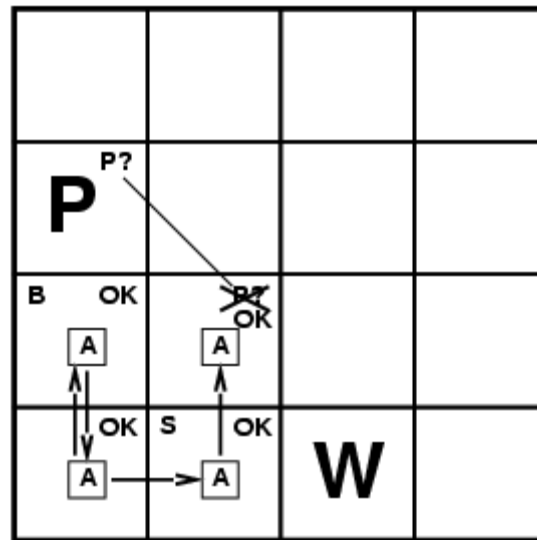
Exploring a Wumpus World

our cautious agent will go back and to [2,1]

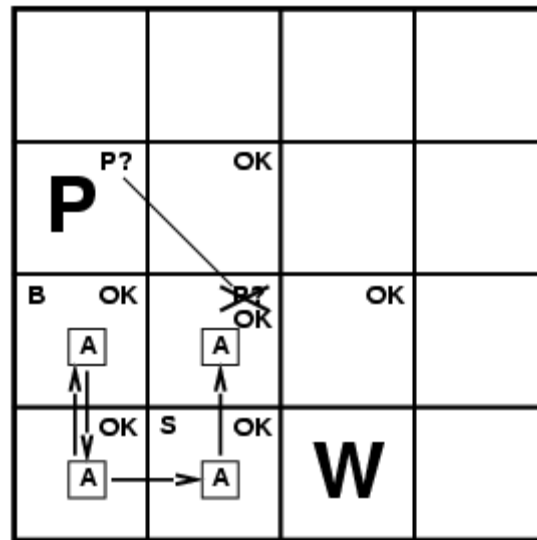# Exploring a Wumpus World

percept: [stench, none, none, none, none]

Exploring a Wumpus World
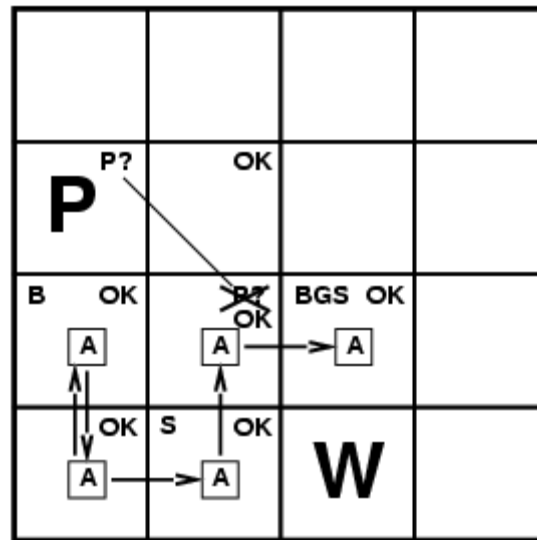

percept: [none, none, none, none, none]

Exploring a Wumpus World

Exploring a Wumpus World

percept: [stench, breeze, glitter, none, none]

Logical Reasoning

- In each case where the agent draws a conclusion from the available information, that conclusion is guaranteed to be correct if the available information is correct

- The above is the fundamental of logical reasoning

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 of a sentence in a world

- E.g., the language of arithmetic
  - $x+2 \geq y$ is a sentence; $x2+y > \{\}$ 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 $x = 0$, $y = 6$

Entailment

- Now, we have a notion of truth, we are ready to study logical reasoning, which involves logical entailment between sentences

- Entailment means that one thing follows from another:

$$KB \models \alpha$$

- Knowledge base *KB* entails sentence *a* if and only if *a* 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 won or the Reds won"
  - E.g., x+y = 4 entails 4 = x+y
  - Entailment is a relationship between sentences (i.e., syntax) that is based on semantics

Models

- We use the term "model" to represent "possible world"
- We say *m* is a model of a sentence *a* if *a* is true in model *m*

- *M(a)* is the set of all models of *a*

- Then KB $\models$ *a* iff $M(KB) \subseteq M(a)$
    - E.g. *KB* = Giants won and Reds won, *a* = Giants won

M(α)

M(KB)
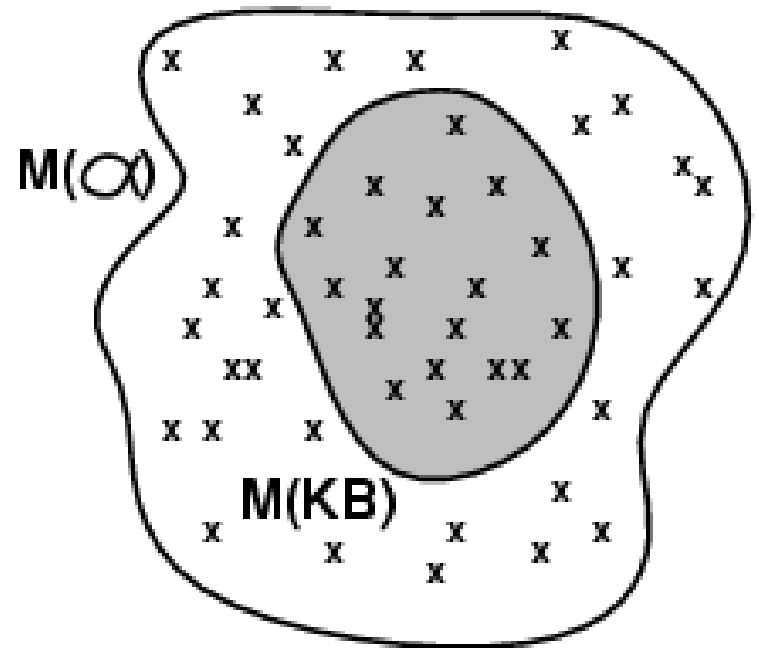
Entailment in the Wumpus World

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

- Consider possible models for *KB* assuming only pits

- 3 Boolean choices $\Rightarrow$ 8 possible models
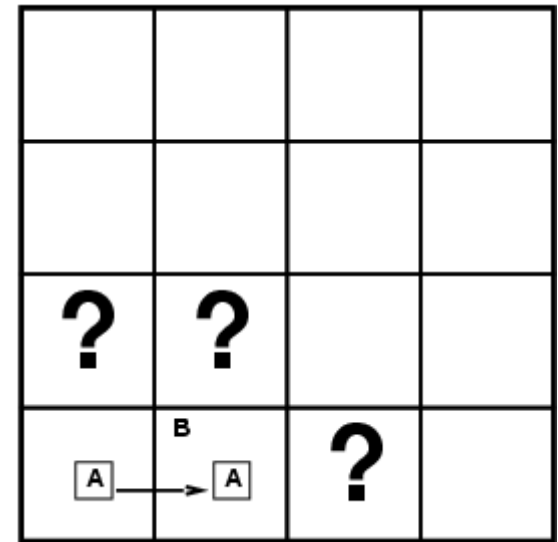
Wumpus Models
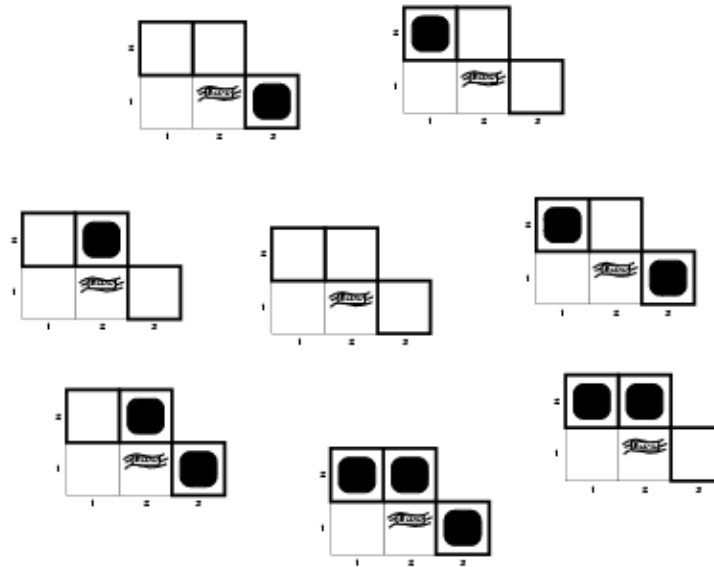
Wumpus Models
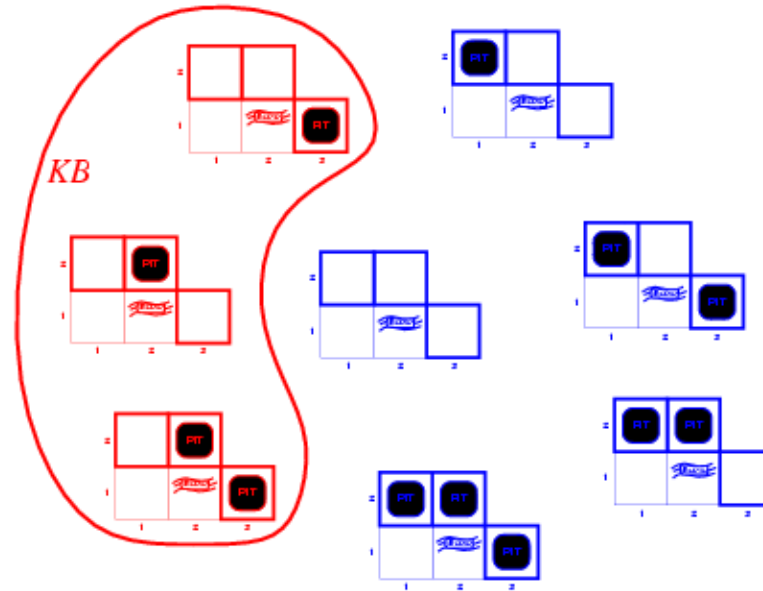
- *KB* = wumpus-world rules + observations

Wumpus Models

- *KB* = wumpus-world rules + observations
- a_1 = "[1,2] is safe", *KB* $\models$ a_1, proved by model checking
  - enumerate all possible models to check that a is true in all models in which KB is true

Wumpus Models
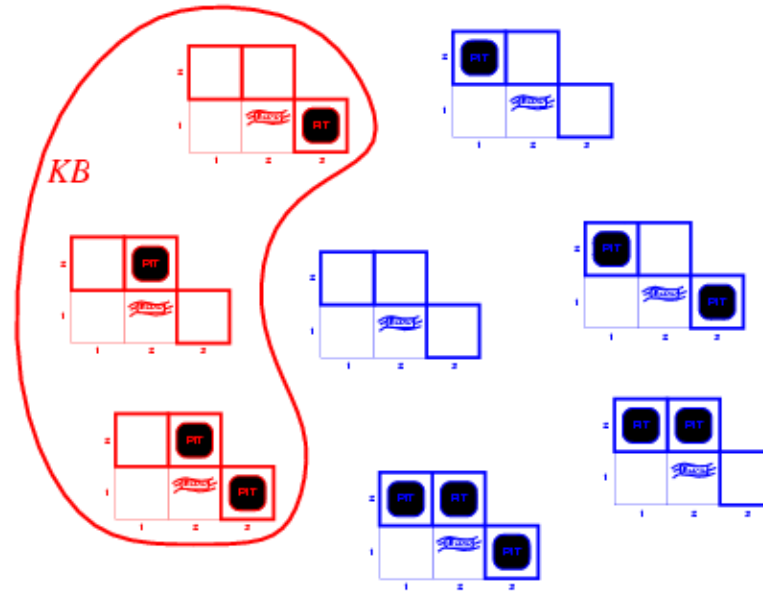
- *KB* = wumpus-world rules + observations
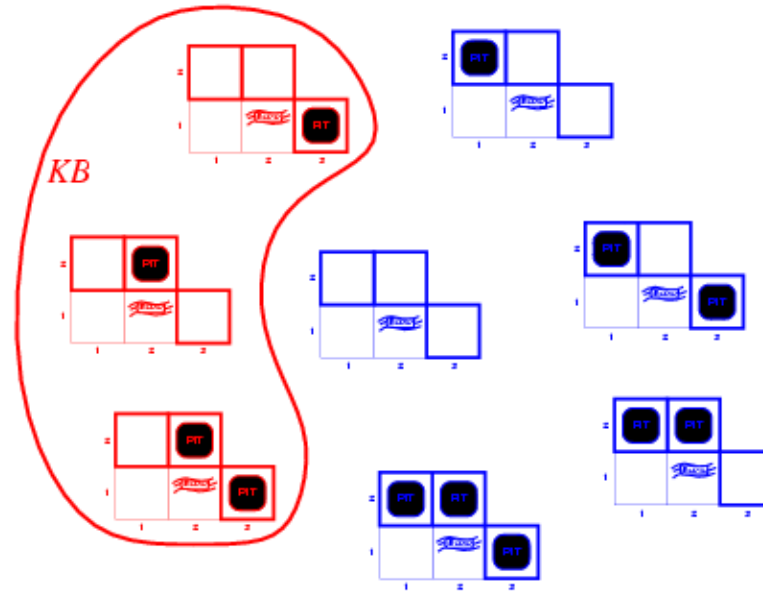
Wumpus Models

- *KB* = wumpus-world rules + observations
- α₂ = "[2,2] is safe", *KB* ⊨ α₂?

Wumpus Models

- *KB* = wumpus-world rules + observations
- α_2 = "[2,2] is safe", $KB \not\models \alpha_2$
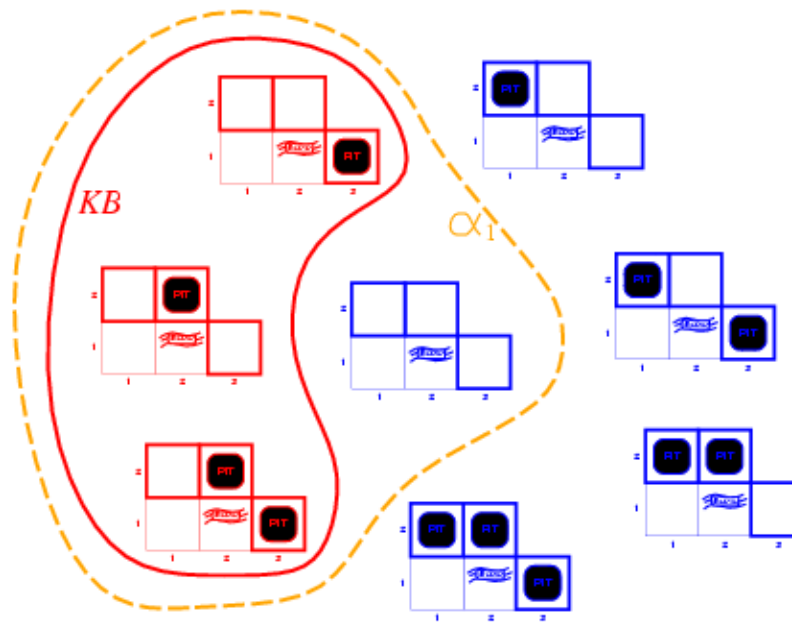
28
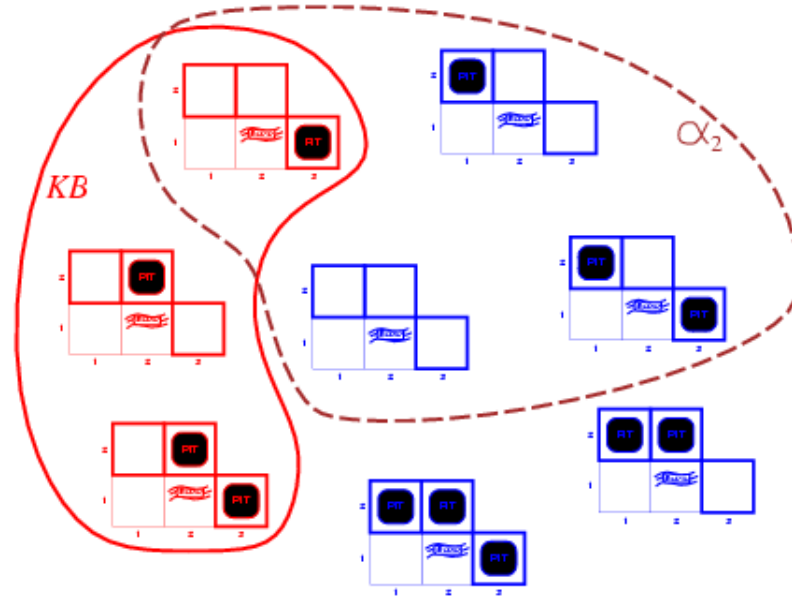
Soundness and Completeness

- An inference algorithm that derives only entailed sentences is called sound

- An inference algorithm is complete if it can derive any sentence that is entailed

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

Grounding Problem

- Grounding: how do we know that KB is true in the real world?
  - Agent's sensors create the connection
    - Meaning and truth of percept sentences through sensing and sentence construction
  - General rules
    - From perceptual experience

Propositional Logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas

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

Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

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

With these symbols, 8 possible models, can be enumerated automatically.

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

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

Simple recursive process evaluates an arbitrary sentence, e.g.,

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

Truth Tables for Connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

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]
    - R1: $\neg P_{1,1}$
    - R2: $\neg B_{1,1}$
    - R3: $B_{2,1}$

- "Pits cause breezes in adjacent squares"
    - R4: $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$
    - R5: $B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

- The KB consists of the above 5 sentences. It can also be considered as a single sentence – the conjunction R1 $\land$ R2 $\land$ R3 $\land$ R4 $\land$ R5
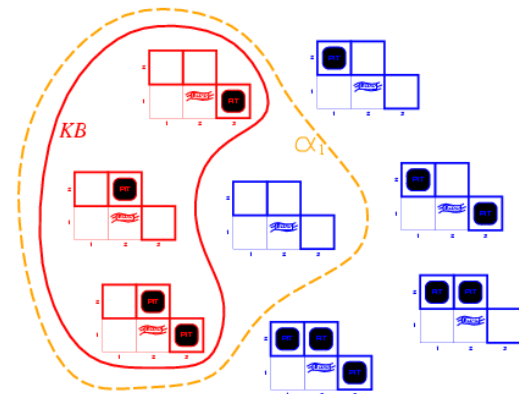
Inference

- The aim of logical inference is to decide whether $KB \models \alpha$ for some sentence α.

- Is $P_{2,2}$ entailed?

- Our first algorithm for inference will be a direct implementation of the definition of entailment:
  - Enumerate the models, and check that α is true in every model in which KB is true

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
false	false	false	false	false	false	false	false	true
false	false	false	false	false	false	true	false	true
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	true	false	false	false	false	false	false	true
false	true	false	false	false	false	true	_true_	_true_
false	true	false	false	false	true	false	_true_	_true_
false	true	false	false	false	true	true	_true_	_true_
false	true	false	false	true	false	false	false	true
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	true	true	true	true	true	true	false	false

$P_{1,2}$ is false, cannot decide on $p_{2,2}$

α_1 = "[1,2] is safe"

Inference by Enumeration

- Depth-first enumeration of all models is sound and complete

```
function TT-ENTAILS?(KB, α) returns true or false

    symbols ← a list of the proposition symbols in KB and α
    return TT-CHECK-ALL(KB, α, symbols, [])


function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
    if EMPTY?(symbols) then
        if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
        else return true
    else do
        P ← FIRST(symbols);  rest ← REST(symbols)
        return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model)) and
               TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model)
```

- For n symbols, time complexity is $O(2^n)$, space complexity is $O(n)$
  PL-TRUE? returns true if a sentence holds within a model.

Proof Methods In General

- Proof methods divide into (roughly) two kinds:
  - Model checking
    - truth table enumeration (always exponential in n)
    - improved backtracking, e.g., Davis--Putnam-Logemann-Loveland (DPLL)
    - heuristic search in model space (sound but incomplete)
      - e.g., min-conflicts-like hill-climbing algorithms
  - Application of inference rules
    - Legitimate (sound) generation of new sentences from old
    - Proof = a sequence of inference rule applications
    - Can use inference rules as operators in a standard search algorithm
    - Typically require transformation of sentences into a normal form

Validity and Satisfiability

- A sentence is valid if it is true in all models, also known as tautology
  - e.g., *True*, A $\lor$ $\neg$A,   A $\Rightarrow$ A, (A $\land$ (A $\Rightarrow$ B)) $\Rightarrow$ B

- Validity is connected to inference via the Deduction Theorem
  - *KB* $\models$ a if and only if (*KB* $\Rightarrow$ a) is valid

- A sentence is satisfiable if it is true in some model
  - e.g., A $\lor$ B, C

- A sentence is unsatisfiable if it is true in no models
  - e.g., A $\land$ $\neg$A

- Satisfiability is connected to inference via the following:
  - *KB* $\models$ a if and only if (*KB* $\land$ $\neg$a) is unsatisfiable
 - Thus proof by contradiction

Equivalence Rules

- Two sentences are logically equivalent iff they are true in the same models: α ≡ ß iff α ⊨ β and β ⊨ α

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

Inference Rules

| From	Can Derive	Abbreviation for rule
$R, R \rightarrow S$	S	Modus Ponens- **mp**
$R \rightarrow S, S'$	R'	Modus Tollens- **mt**
R, S	$R \wedge S$	Conjunction-**con**
$R \wedge S$	R, S	Simplification- **sim**
R	$R \vee S$	Addition- **add**
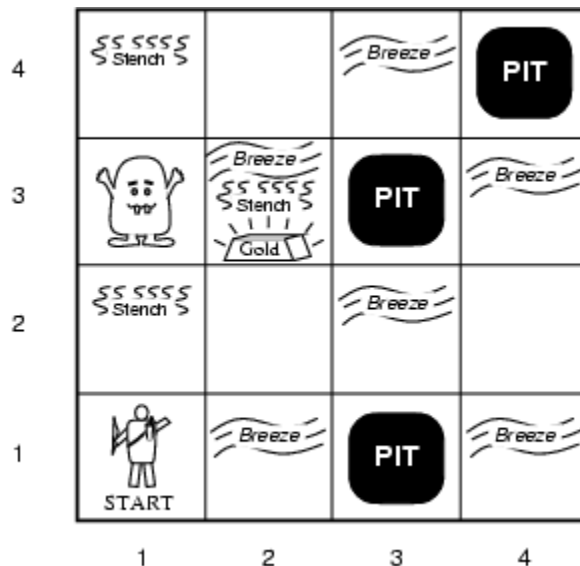
- R1: $\neg P_{1,1}$
- R2: $B_{1,1} \Leftrightarrow P_{1,2} \vee P_{2,1}$
- R3: $B_{2,1} \Leftrightarrow P_{1,1} \vee P_{2,2} \vee P_{3,1}$
- R4: $\neg B_{1,1}$
- R5: $B_{2,1}$

- Can we prove $\neg P_{1,2}$?

41

In-Class Exercises #7.1

From	Can Derive	Abbreviation for rule
R, R $\rightarrow$ S	S	Modus Ponens- **mp**
R $\rightarrow$ S, S′	R′	Modus Tollens- **mt**
R, S	R $\wedge$ S	Conjunction-**con**
R $\wedge$ S	R, S	Simplification- **sim**
R	R $\vee$ S	Addition- **add**

- Prove $\neg P_{2,1}$ and $P_{3,1}$ given:

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

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

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

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

$R5 : B_{2,1}$

$R6 : \neg P_{2,2}$
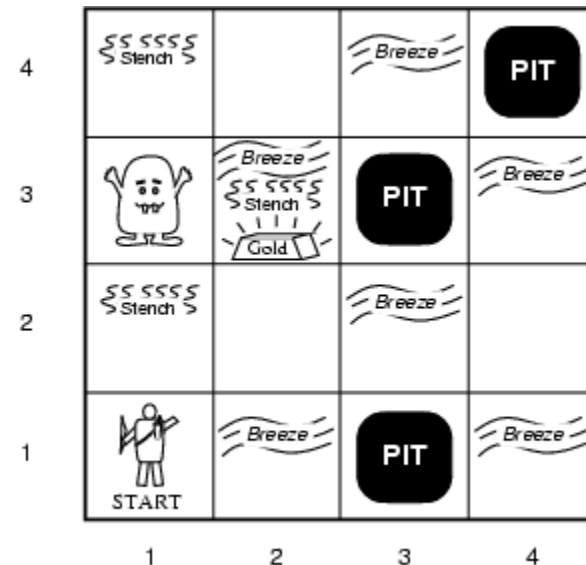
Proof Problem as Search Problem

- **Initial state**: initial knowledge base

- **Actions**:
  - set of actions consists of all the inference rules applied to all the sentences that match the left half of the inference rule

- **Result**: add the sentence in the right half of the inference rule

- **Goal**: the goal is a state that contains the sentence we are trying to prove

- Practically, finding a proof can be more efficient because the proof can ignore irrelevant propositions, no matter how many of them there are

Resolution

- The inference rules can be applied whenever suitable premises are found in the KB

- The conclusion of the rule must follow regardless of what else is in the KB

- The above rules are sound, but if the available inference rules are inadequate, then it's not complete

- Now, we introduce a single inference rule, **resolution**, that gives a complete inference algorithm when coupled with any complete search algorithm

Example

- Recall our previous proof of $P_{3,1}$ given:

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

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

$R3 : B_{2,1} \Leftrightarrow P_{1,1} \vee P_{2,2} \vee P_{3,1}$
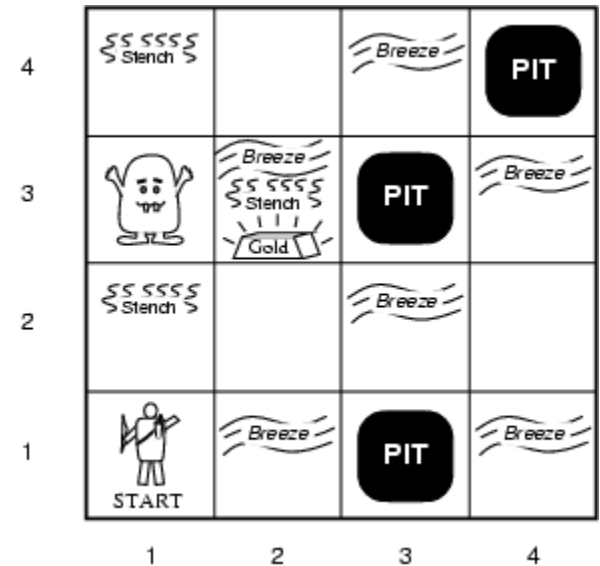
$R4 : \neg B_{1,1}$

$R5 : B_{2,1}$

$R6 : \neg P_{2,2}$

...

Unit resolution

$R9 : P_{1,1} \vee P_{2,2} \vee P_{3,1}$

$R10 : P_{2,2} \vee P_{3,1}$ Resolution rule: $\neg P_{1,1}$ resolves with $P_{1,1}$ in R9

$R11 : P_{3,1}$ Resolution rule: $\neg P_{2,2}$ resolves with $P_{2,2}$ in R10

Resolution Rules

- **Unit resolution**
  - li and m are complementary literals

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

- **Generalized resolution rule**

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

  - where li and mj are complementary literals

Soundness and Completeness

- ## Soundness of resolution inference rule:
  - if l_i is true, then m_j is false
    - hence $(m_1 \lor \dots \lor m_{j-1} \lor m_{j+1} \lor \dots \lor m_n)$ must be true
  - if l_i is false, then
    - $(l_1 \lor \dots \lor l_{i-1} \lor l_{i+1} \lor \dots \lor l_k)$ must be true
  - no matter what
    - $(l_1 \lor \dots \lor l_{i-1} \lor l_{i+1} \lor \dots \lor l_k)$ V $(m_1 \lor \dots \lor m_{j-1} \lor m_{j+1} \lor \dots \lor m_n)$ is true

- *Any complete search algorithm, applying only the resolution rule, can derive any conclusion entailed by any knowledge base in propositional logic*

- A resolution-based theorem prover can, for any sentences α and β in propositional logic, decide whether α ⊨ β

Resolution and CNF

- Resolution rule applies only to disjunctions of literals

- *Every sentence of propositional logic is logically equivalent to a conjunction of disjunctions of literals*

- Conjunctive Normal Form (CNF):
    - conjunction of disjunctions of literals clauses
    - E.g., $(A \lor \neg B) \land (B \lor \neg C \lor \neg D)$
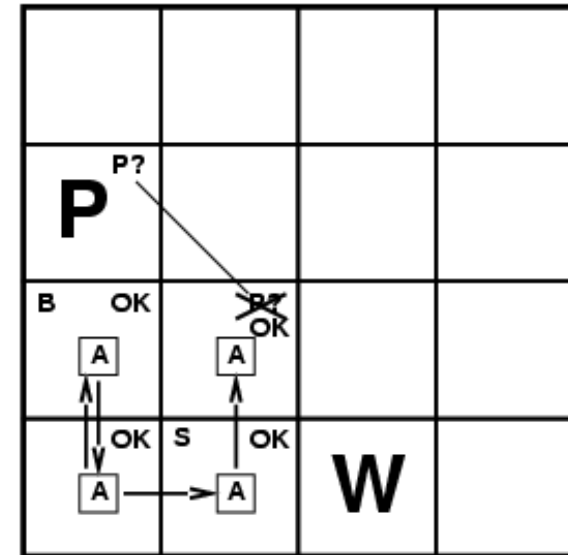
$$\frac{P_{1,3} \lor P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$$

Conversion to CNF

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

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. Move $\neg$ inwards using de Morgan's rules and double-negation:
$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Apply distributive law ($\wedge$ over $\vee$) and flatten:
$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Resolution Algorithm

- To show that KB ╞ α, we show that (KB ^ ¬α) is unsatisfiable, proof by contradiction

- First, (KB ^ ¬α) is converted into CNF
- Then the resolution rule is applied to the resulting clauses
  - Each pair that contains complementary literals is resolved to produce a new clause, which is added to the set if it's not present
- Process continues until:
  - no new clauses can be added, thus KB does not entail α
  - two clauses resolve to yield {}, thus KB entails α

- {} is a disjunction of no disjuncts is equivalent to False, thus the contradiction

Resolution Algorithm

- Proof by contradiction, i.e., show that $KB \land \neg\alpha$ is unsatisfiable

function PL-RESOLUTION(KB, α) **returns** *true* or *false*

    clauses $\leftarrow$ the set of clauses in the CNF representation of $KB \land \neg\alpha$
    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*

Resolution Example

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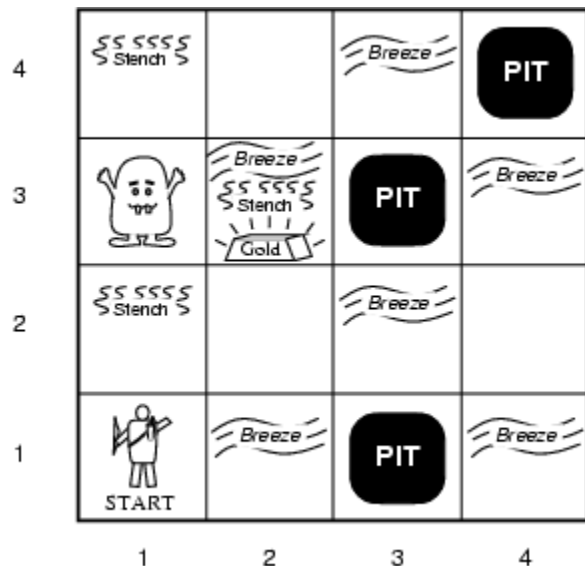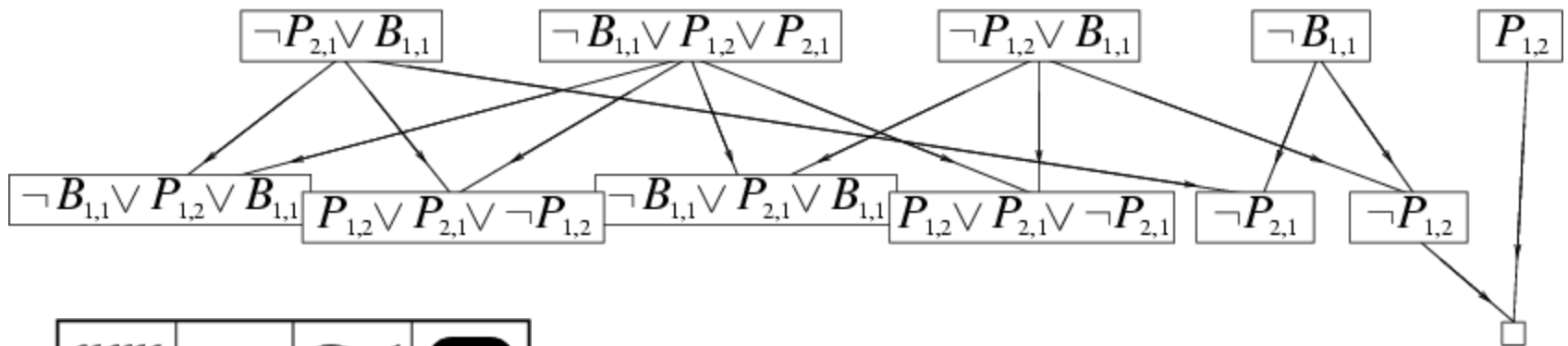
Horn Clauses

- **Why horn clauses?**
  - In many practical situations, however, the full power of resolution is not needed
  - Real-world KBs often contain some restricted clauses – Horn clauses

- **Horn Form** (restricted)
  - KB = conjunction of Horn clauses
  - Horn clause = disjunction of literals of which at most one is positive
    - E.g., $(\neg L_{1,1} \lor \neg Breeze \lor B_{1,1})$

Properties of Horn Clauses

- Every horn clause can be written as an implication whose premise is a conjunction of positive literals and whose conclusion is a single positive literal
  - E.g., $(L_{1,1} \land Breeze) \Rightarrow B_{1,1}$

- Modus Ponens (for Horn Form): complete for Horn KBs

$$\frac{\alpha_1, \dots, \alpha_n, \quad \alpha_1 \land \dots \land \alpha_n \Rightarrow \beta}{\beta}$$

- Definite clause: horn clauses with exactly one positive literal
  - The positive literal is called the **head** and the negative literals form the **body**
  - Definite clauses form the basis for logic programming

- Can be used with forward chaining or backward chaining
- These algorithms are very natural and run in linear time

Forward Chaining

- Idea:
    - fire any rule whose premises are satisfied in the *KB*
    - add its conclusion to the *KB*, until query is found
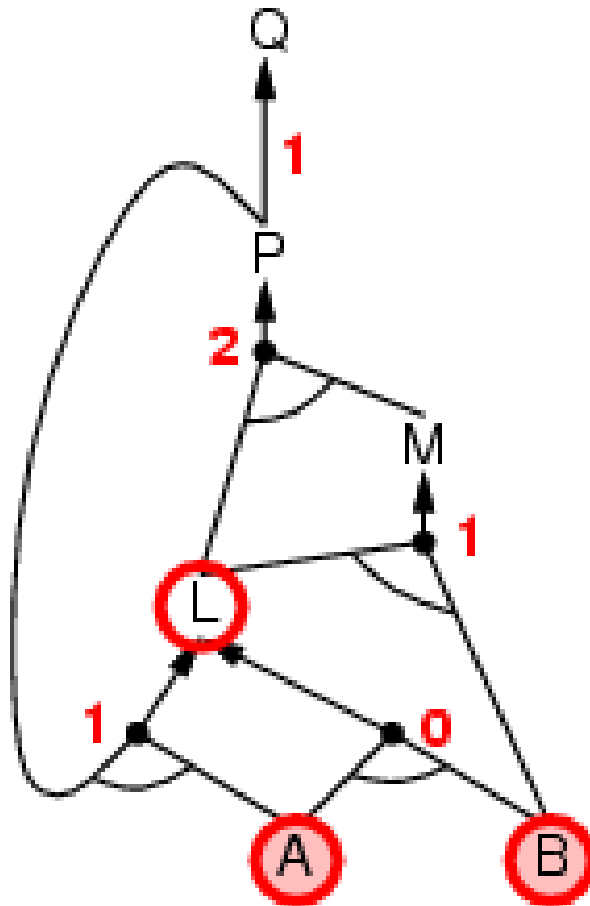
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$
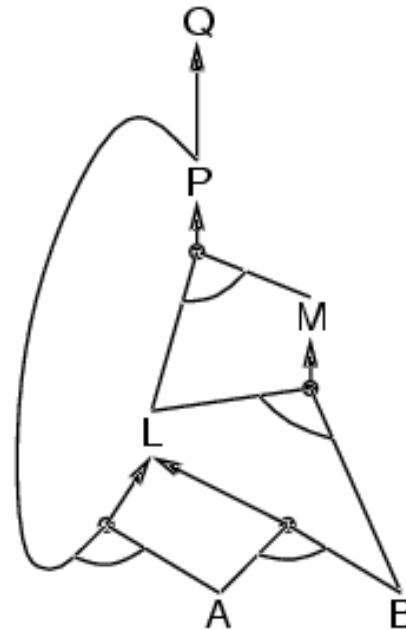
and-or graph

Forward Chaining Algorithm

function PL-FC-ENTAILS?(KB, q) **returns** *true* or *false*
   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

   while *agenda* is not empty **do**
      $p \leftarrow$ POP(*agenda*)
         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 agenda

   return *false*

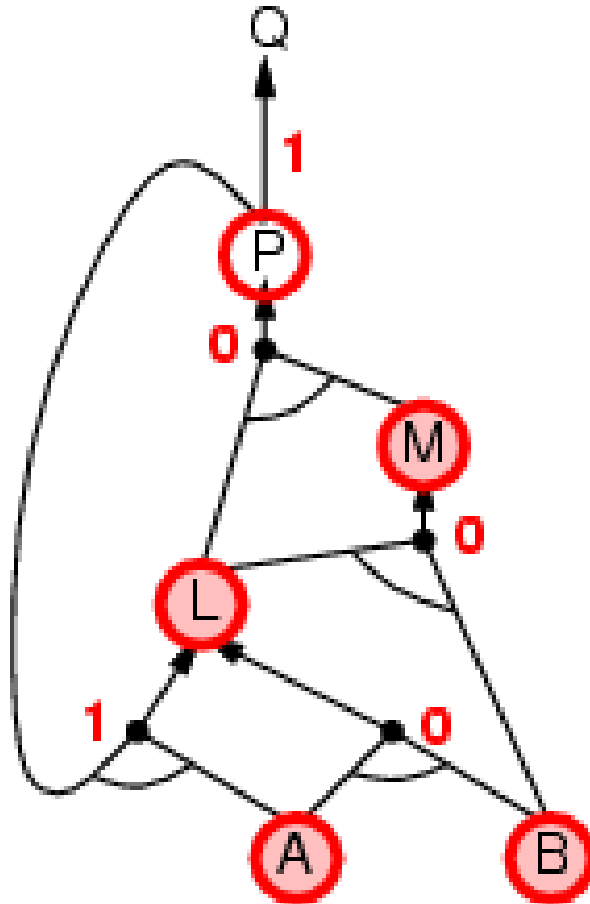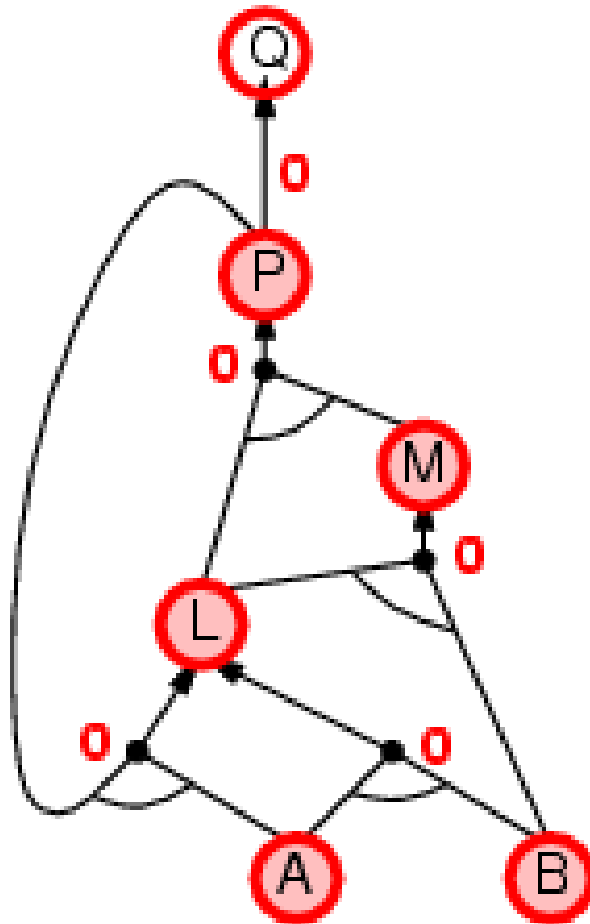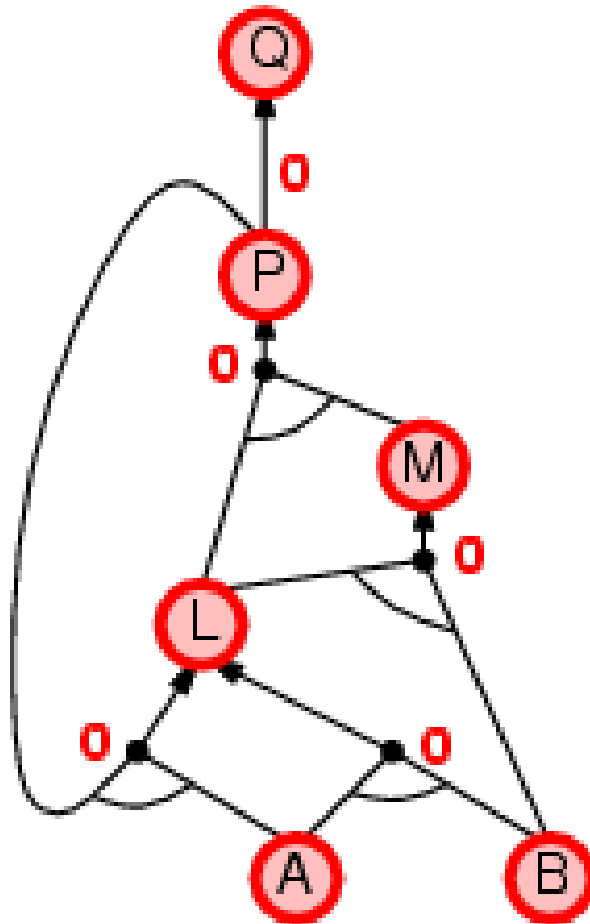- Forward chaining is sound and complete for Horn KB

Forward Chaining Example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Forward Chaining Example

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

Forward Chaining Example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B

Forward Chaining Example

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

Forward Chaining Example

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B
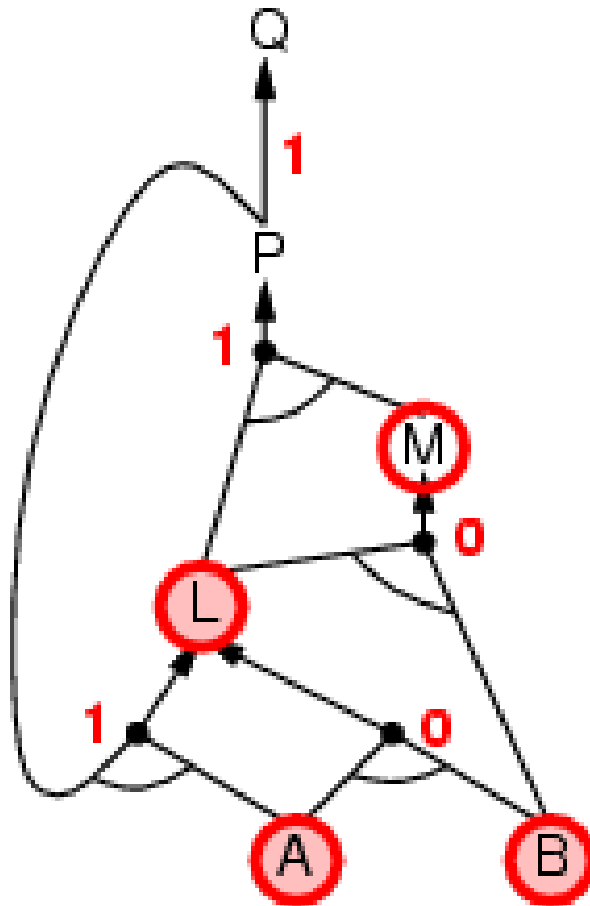
Forward Chaining Example

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

Proof of Soundness

- Every inference is an application of Modus Ponens

Proof of Completeness

- FC derives every atomic sentence that is entailed by *KB*

  1. FC reaches a fixed point where no new atomic sentences are derived

  2. Consider the final state as a model *m*, assigning true/false to symbols; true for each symbol inferred, false otherwise

  3. Every definite clause in the original *KB* is true in *m*

     $a_1 \wedge \ldots \wedge a_k \Rightarrow b$, assuming the opposite, then it contradicts "fixed point"

  4. Hence, every entailed atomic sentence will be derived

Backward Chaining

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

- Avoid loops: check if new subgoal is already on the goal stack

- Avoid repeated work: check if new subgoal
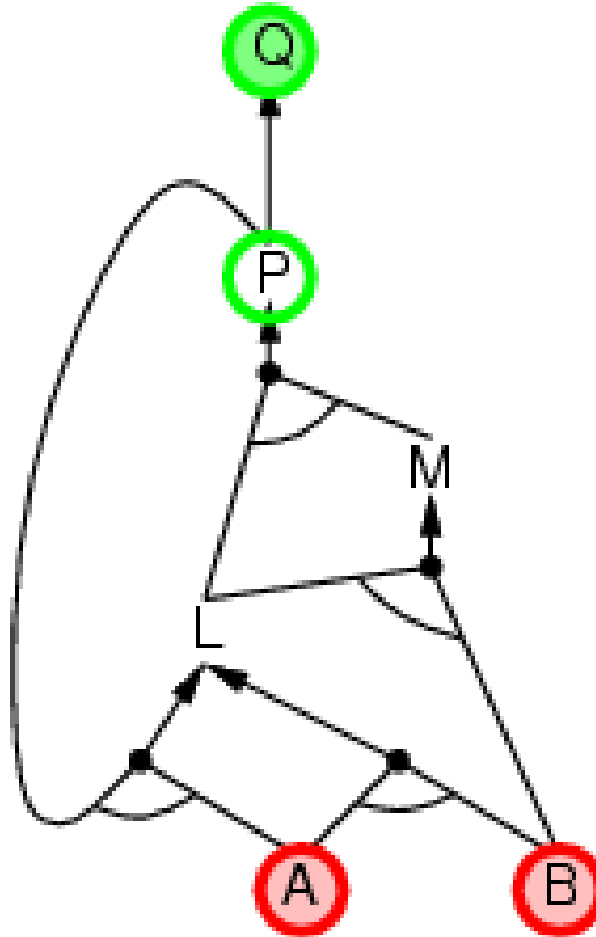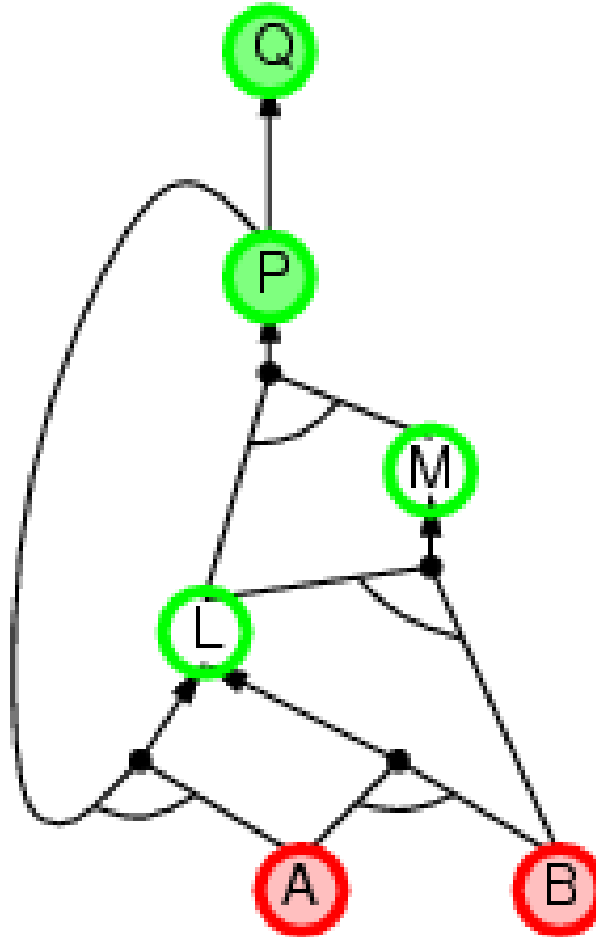 - has already been proved true, or
  - has already failed
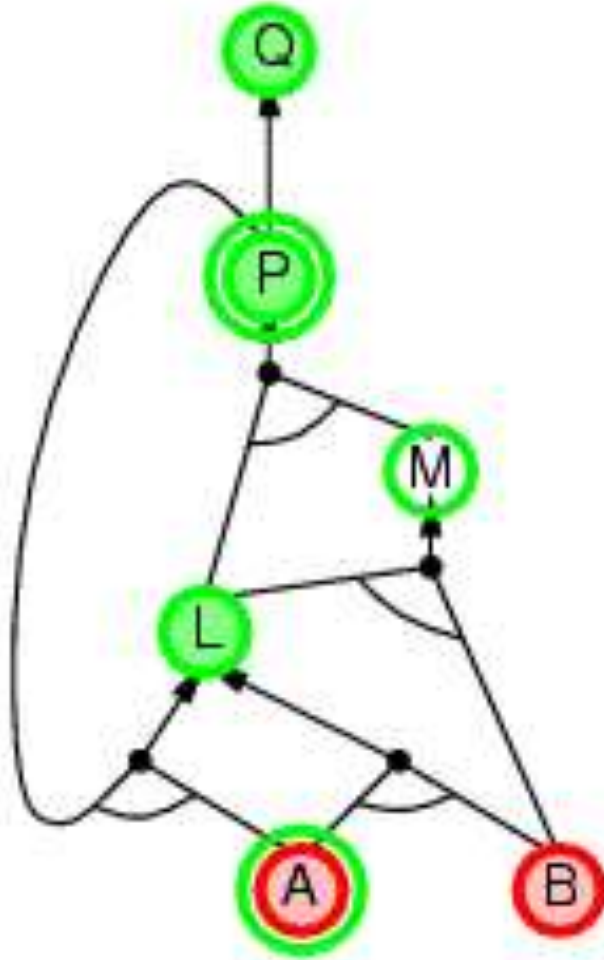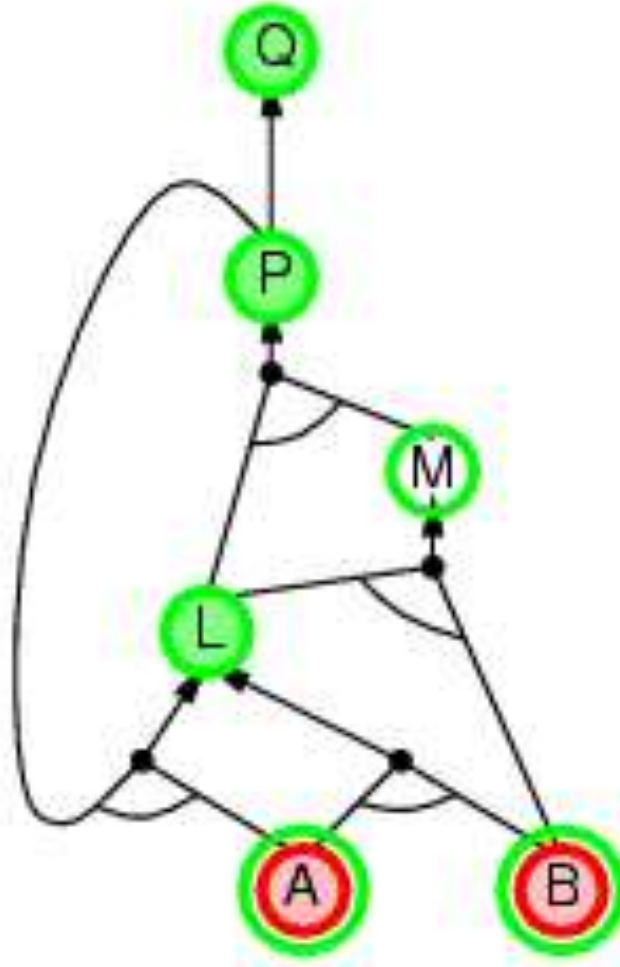
Backward Chaining Example

Backward Chaining Example

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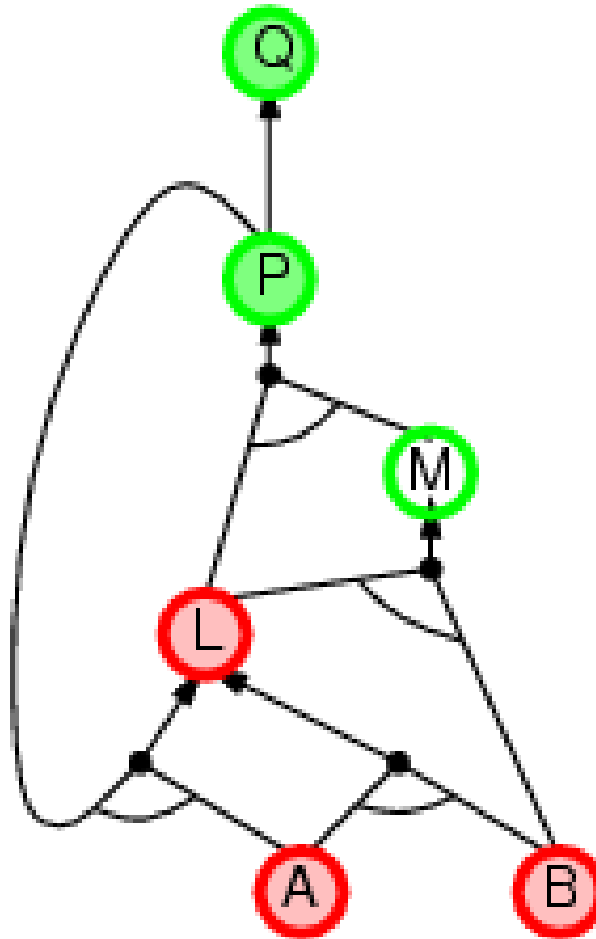A
B

Backward Chaining Example

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

Backward Chaining Example

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

Backward Chaining Example

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

Backward Chaining Example

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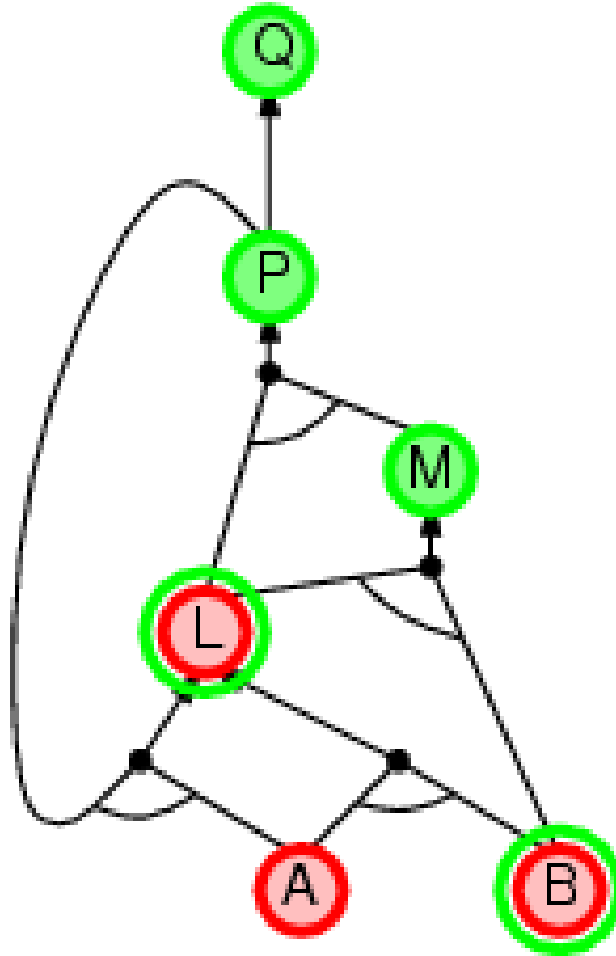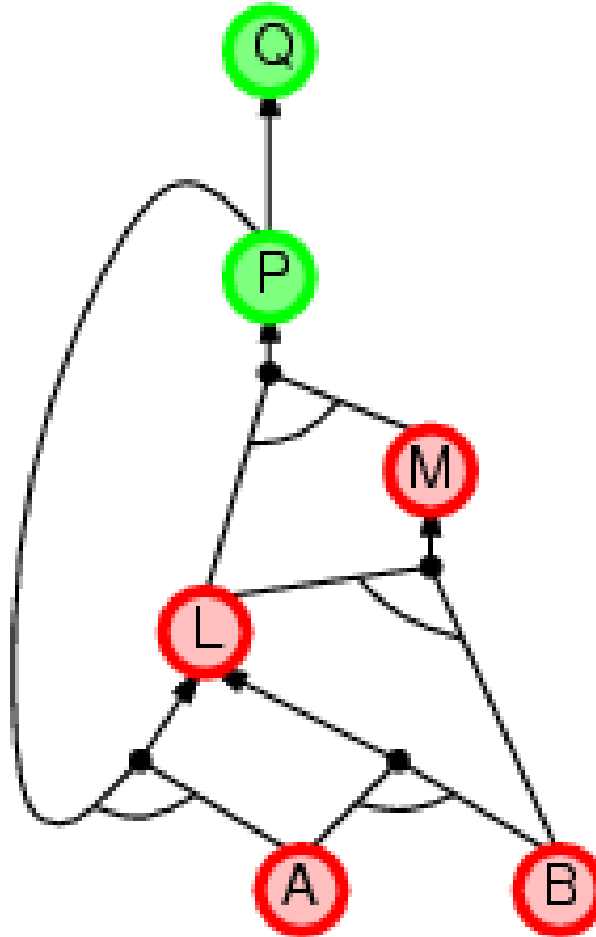$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$

Backward Chaining Example

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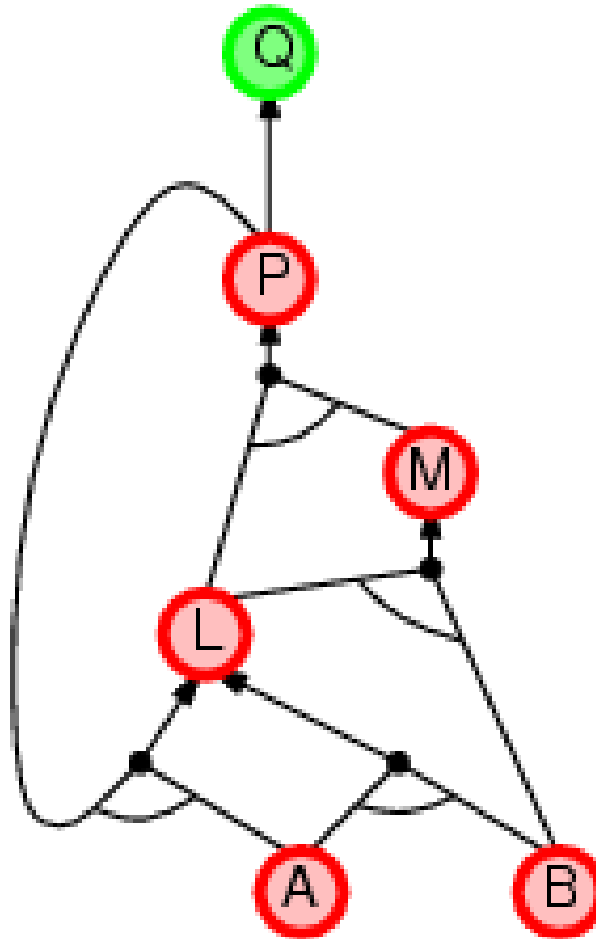A
B

Backward Chaining Example

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

Backward Chaining Example

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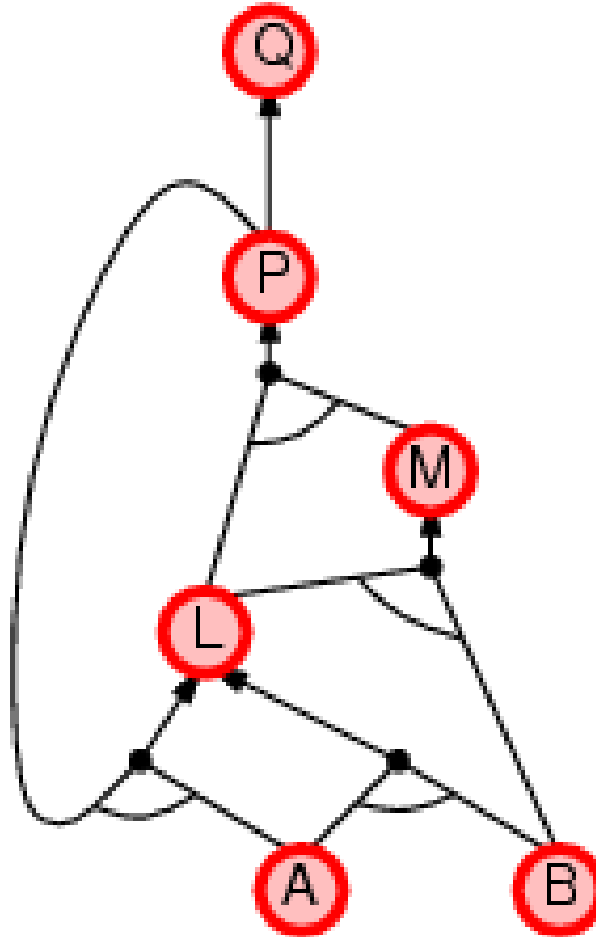$A \wedge B \Rightarrow L$
A
B

Backward Chaining Example

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

Forward vs. Backward Chaining

- FC is data-driven, automatic, unconscious processing
  - e.g., object recognition, routine decisions

- May do lots of work that is irrelevant to the goal

- BC is goal-driven, appropriate for problem-solving,
  - e.g., Where are my keys? How do I get into a PhD program?

- Complexity of BC can be much less than linear in size of KB

Inference-Based Agents in the Wumpus World

- A wumpus-world agent using propositional logic:

$\neg P_{1,1}$

$\neg W_{1,1}$

$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$

$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$

$W_{1,1} \vee W_{1,2} \vee ... \vee W_{4,4}$

$\neg W_{1,1} \vee \neg W_{1,2}$

$\neg W_{1,1} \vee \neg W_{1,3}$

...

$\Rightarrow$ 64 distinct proposition symbols, 155 sentences

Expressiveness Limitation

- KB contains "physics" sentences for every single square

- For every time t and every location $[x,y]$,

  $L_{x,y} \wedge FacingRight^{\,t} \wedge Forward^{\,t} \Rightarrow L_{x+1,y}$

- Rapid proliferation of clauses

Summary

- Logical agents apply inference to a knowledge base to derive new information and make decisions

- Basic concepts of logic:
    - syntax: formal structure of sentences
    - semantics: truth of sentences based on models
    - entailment: necessary truth of one sentence given another
    - inference: deriving sentences from other sentences
    - soundness: derivations produce only entailed sentences
    - completeness: derivations can produce all entailed sentences

- Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.

- Resolution is complete for propositional logic
Forward, backward chaining are linear-time, complete for Horn clauses

- Propositional logic lacks expressive power