# AI基础

## Lecture 7: Logical Agents

Bin Yang

School of Data Science and Engineering

byang@dase.ecnu.edu.cn

[Some slides adapted from Philipp Koehn, JHU]

# Lecture 6 ILOs

- Constraint Satisfaction Problems
  - Definition of CSPs
    - Variables, Domains, Constraints (in the form of <scope, rel>)
  - Types of constrains
    - Unary, binary, high-order, preferences
  - Solving CSPs
    - Standard search
      - $n!n^d$
    - Backtracking
      - DFS + Variable ordering + filtering (consistency checking for a single arc and for all arcs)
      - Variable ordering heuristics: Minimum remaining values (MRV):
    - Problem structures
      - Independent subproblems (connected components)
      - Tree structure, removing nodes, collapsing nodes
    - Local search
      - Min-conflicts heuristic

# Lecture 7 ILOs

- Knowledge-based agents
  - Wumpus world
- Logic in general
  - Models and entailment
- Propositional logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
  - Search
  - Resolution
  - Forward chaining, only for Horn KB
  - Backward chaining, only for Horn KB

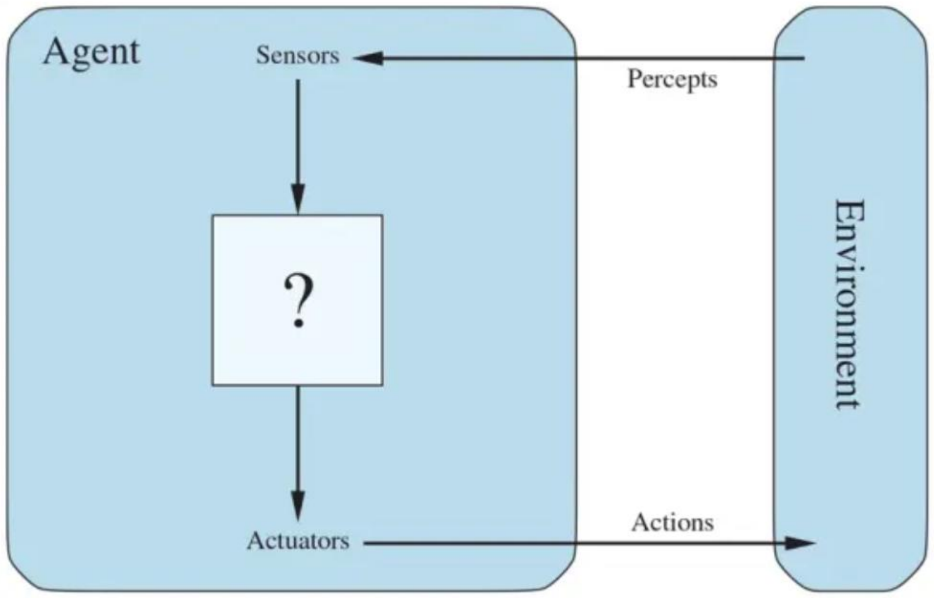
# Outline

- Knowledge-based agents
- The Wumpus world example
- Logic in general
- Propositional logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
  - Search
  - Resolution
  - Forward chaining
  - Backward chaining

# What are agents?

- An agent is anything that
  - is able to perceive its **environment** through **sensors**, and
  - is able to act upon that **environment** through **actuators**
- A human agent
  - Eyes and ears as sensors
  - Hands and legs as actuators
- A robotic agent
  - Cameras and infrared as sensors
  - Various motors as actuators

Figure 2.1




Agents interact with environments through sensors and actuators.

# Open-loop and closed-loop system

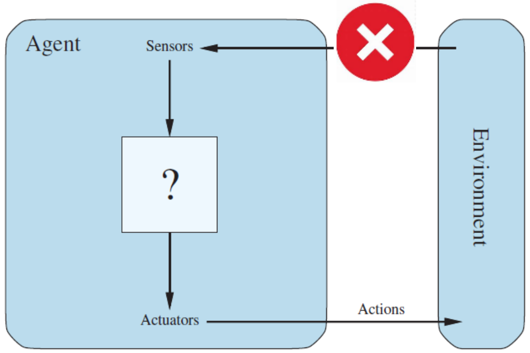
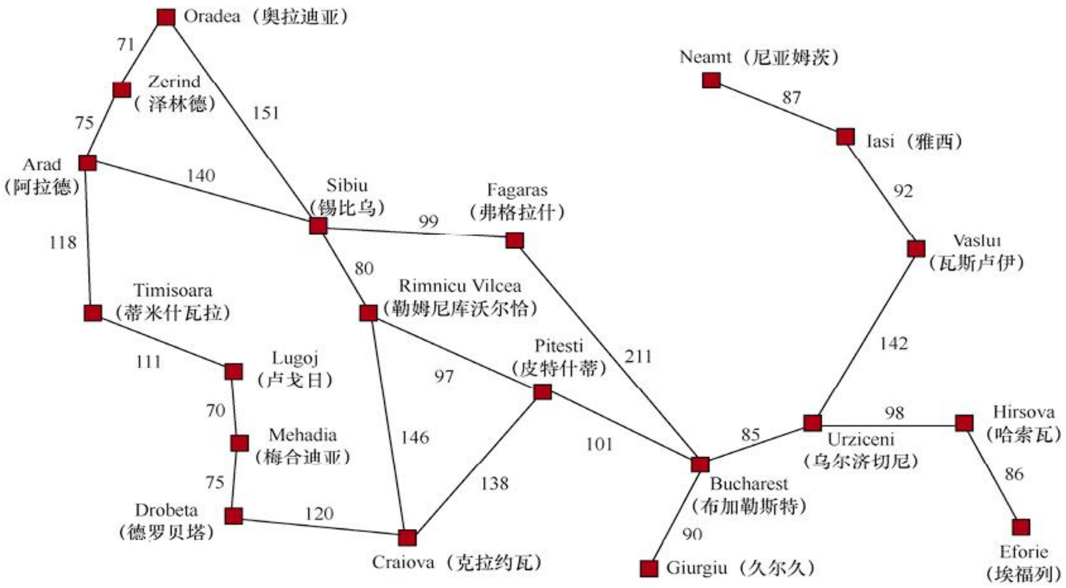


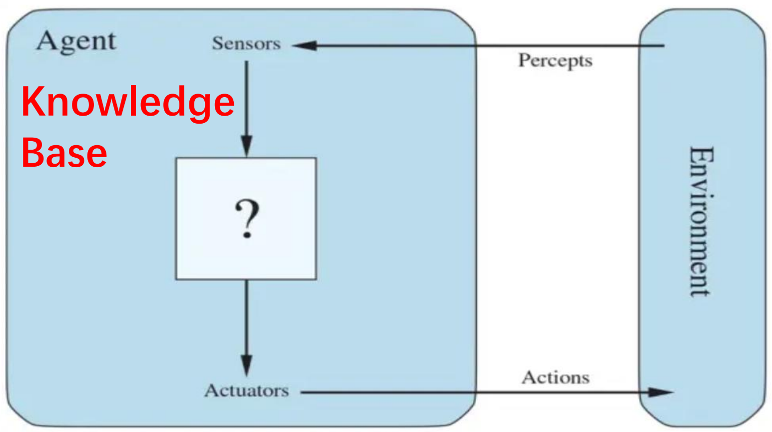

Figure 2.1 Agents interact with environments through sensors and actuators.

- Fully observable, deterministic, and known environment
  - The solution to any problem is a **fixed sequence of actions**
  - **Open-loop**, ignore percepts while executing breaks the loop between agent and environment.
  - Example: shortest path




- Partially observable, non-deterministic
  - Actions depends on what percepts arrive
  - Example: traffic based fastest path
  - The solution is a **strategy**
    - Different future actions based on percepts
  - **Closed-loop**

# Problem settings



Agents interact with environments through sensors and actuators.

- Problem-solving agents
  - Know things in a very limited, inflexible sense.
  - What actions are available and what the result of performing an action on a state.
  - Do not know general facts.
- Knowledge-based agents
  - Has a knowledge base. Use logic.
  - Knowledge-based agents can accept new tasks in the form of explicitly described goals.
  - They can achieve competence quickly by being told or learning new knowledge about the environment
  - They can adapt to changes in the environment by updating the relevant knowledge.

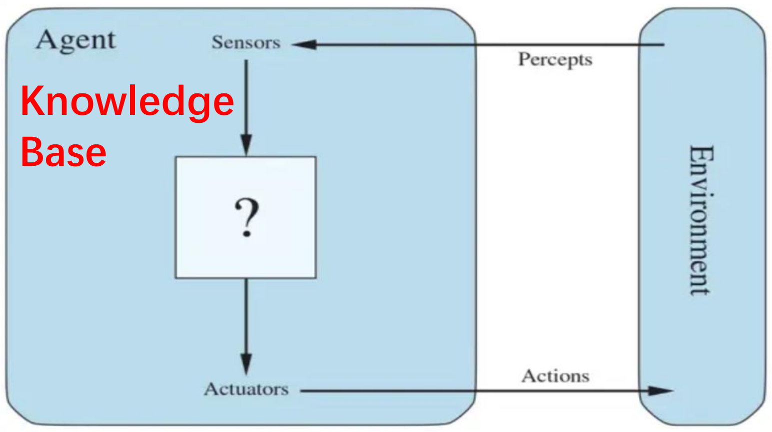
# Knowledge-based agents

- Central component: a knowledge base, KB
- A knowledge base
  - A set of sentences, expressed in knowledge representation language
    - Axiom
      - the sentence is taken as being given without being derived from other sentences
- Tell
  - Add new sentences to the KB
- Ask
  - Query what is known in the KB
- Inference
  - Deriving new sentences from old sentences.

# A generic knowledge-based agent

- It TELLs the knowledge base what it perceives.
- It ASKs the knowledge base what action it should perform.
  - In the process of answering this query, extensive reasoning may be done about the current state of the world, about the outcomes of possible action sequences, and so on.
- TELLs the knowledge base which action was chosen
- Returns the action so that it can be executed.

Figure 2.1




Agents interact with environments through sensors and actuators.

Figure 7.1

**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*

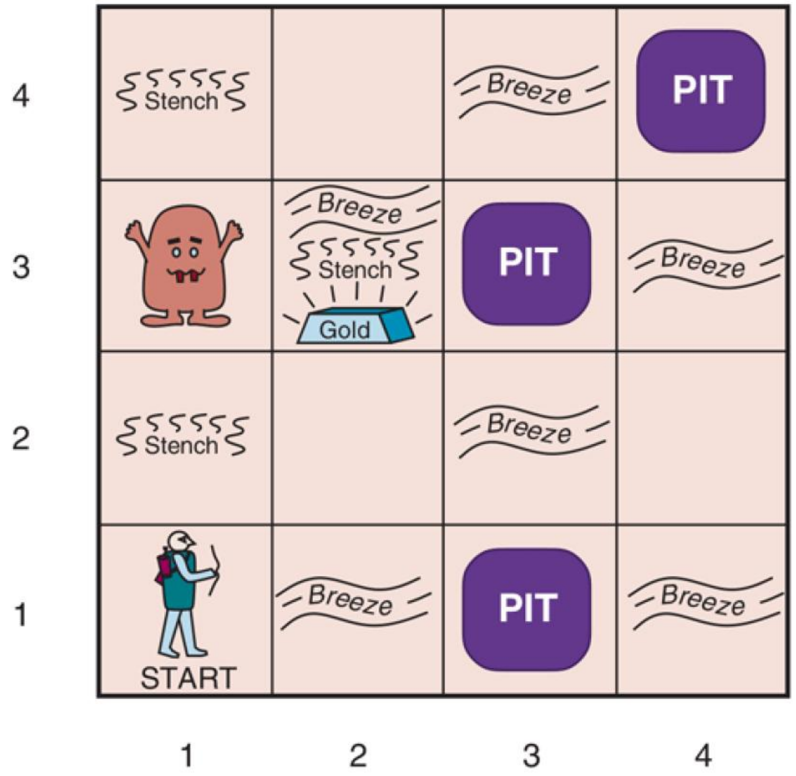
A generic knowledge-based agent. Given a percept, the agent adds the percept to its knowledge base, asks the knowledge base for the best action, and tells the knowledge base that it has in fact taken that action.

# Outline

- Knowledge-based agents
- The Wumpus world example
- Logic in general
- Propositional logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
  - Search
  - Resolution
  - Forward chaining
  - Backward chaining

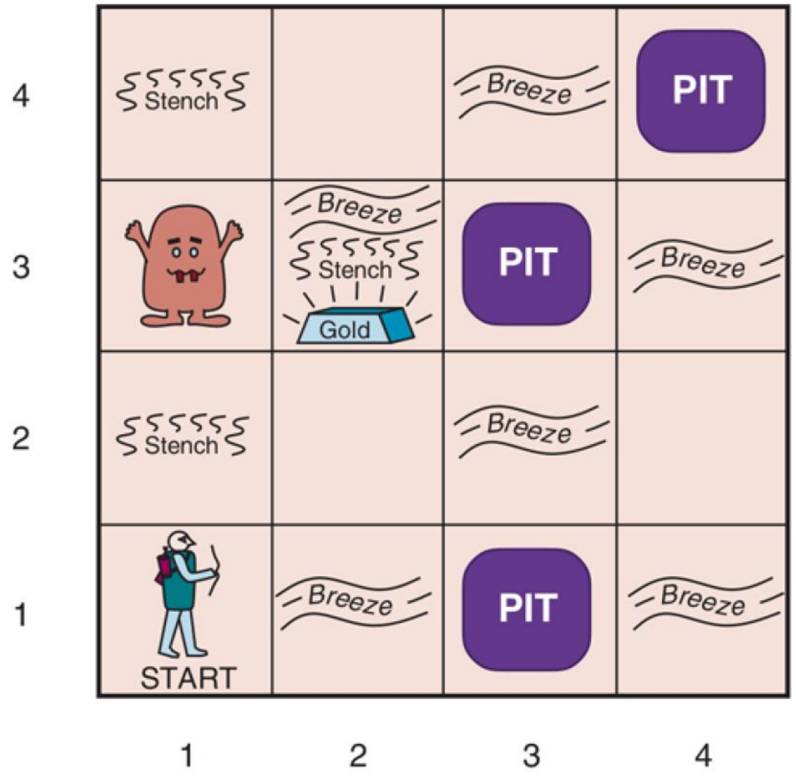
# Wumpus World

- The wumpus world is a cave consisting of rooms connected by passageways.
- The terrible Wumpus is a beast that eats anyone who enters its room.
- The wumpus can be shot by an agent, but the agent has only one arrow.
- Some rooms contain bottomless pits that will trap anyone who wanders into these rooms
- The only redeeming feature of this bleak environment is the possibility of finding a heap of gold.



# Wumpus World

- PEAS description
- **P**erformance, **E**nvironment, **A**ctuators, **S**ensors
- Performance measure
  - gold +1000, death (eaten by the Wumpus, falling into a pit) -1000
  - -1 per step, -10 for using the arrow
- Environment
  - squares directly adjacent (not diagonally adjacent) to wumpus are smelly
  - squares directly adjacent to pit are breezy
  - glitter iff gold is in the same square
  - 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
- Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot
- Sensors: Stench, Breeze, Glitter

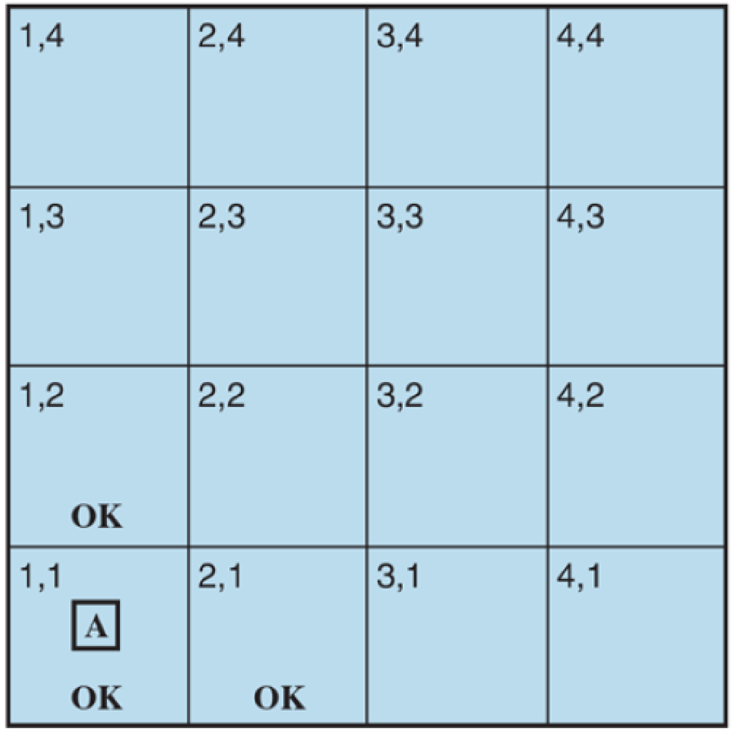


# Wumpus World characterization

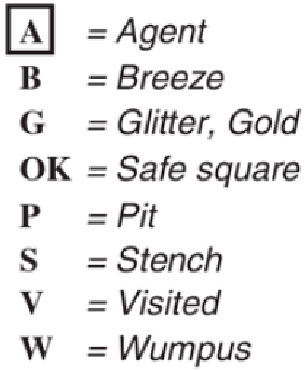
- Observable? Partially—only local perception
- Deterministic? Yes—outcomes exactly specified
- Episodic? Sequential: rewards may come after many actions are taken
- Static? Yes—Wumpus and Pits do not move
- Discrete? Yes
- Single-agent? Yes—Wumpus does not move

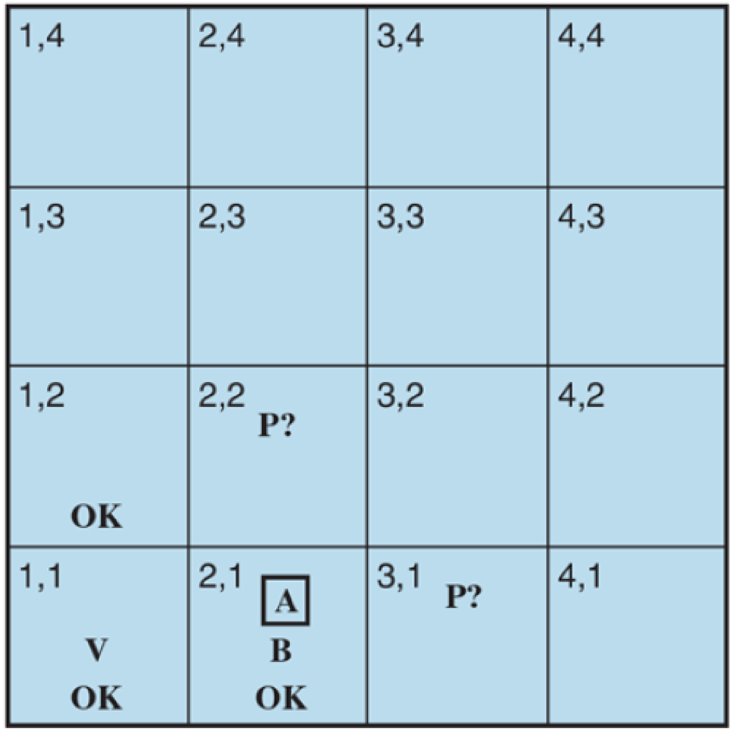
KB: The rules of the environment.
<Stench, Breeze, Glitter>

# Exploring a Wumpus World

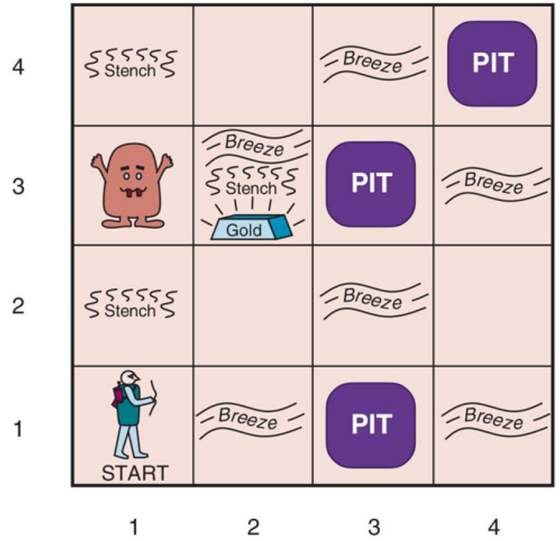



(a)



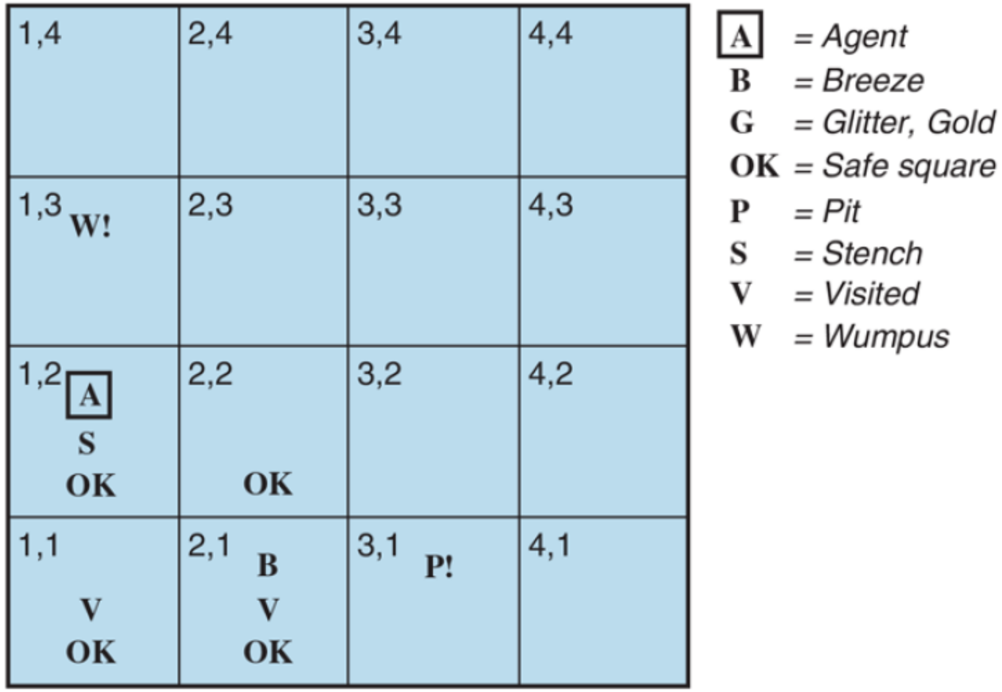

(b)



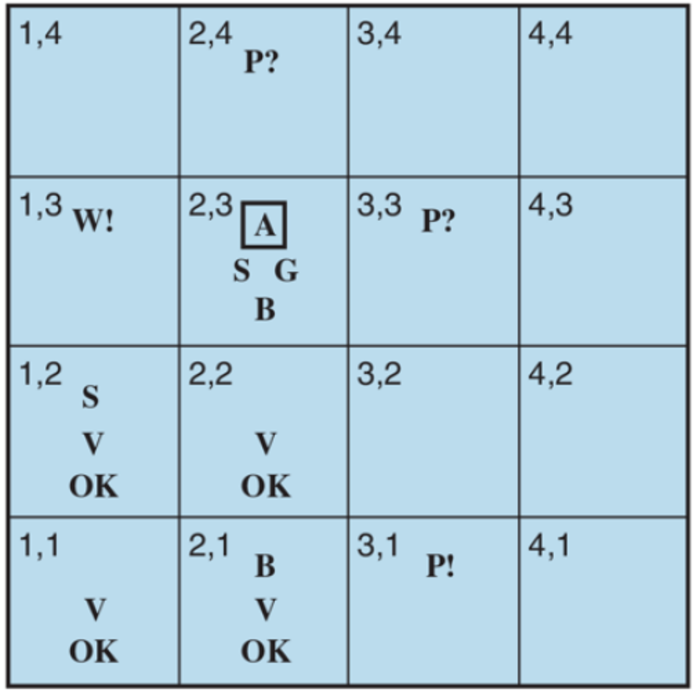

Perceive: <none, none, none>
(1, 2) and (2, 1) are safe.

Move to (2, 1)
Perceive: <none, B, none>: (3, 1) or (2, 2) must have a pit
Risky to go to these two cells. Go back to start then go to (1, 2)

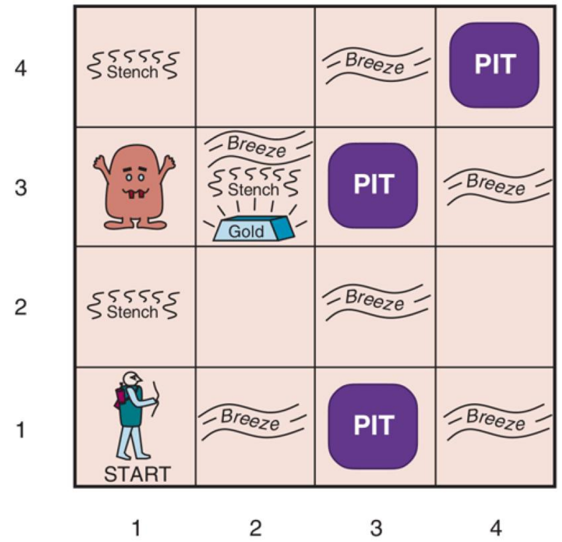
# Exploring a Wumpus World



(a)

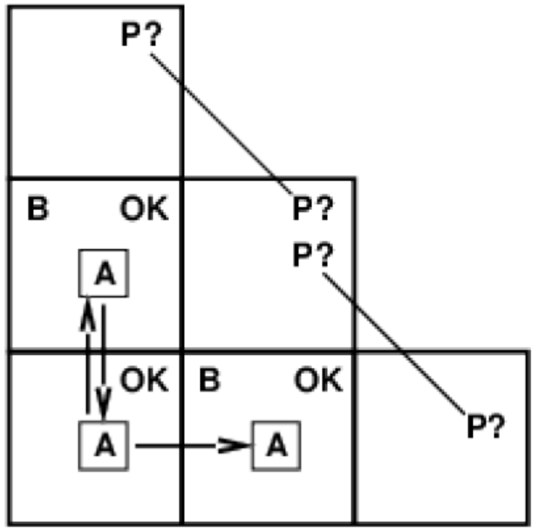


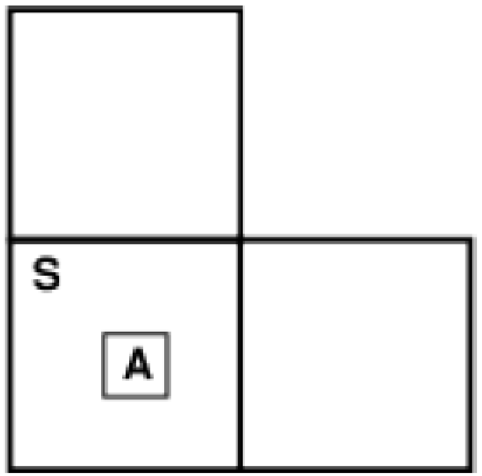
(b)




Perceive: <S, N, N>
The stench in [1,2] means a wumpus nearby.
The Wumpus cannot be in [1,1] (start) and [2,2] (or the agent would have detected a stench when it was in [2,1]). Then, must be in [1, 3]
No breeze in [1, 2] indicates [2, 2] has no pit.

Go to (2, 2) then (2, 3)
Detect gold, and return back to start.




- Breeze in both (1, 2) and (2, 1)
  - No safe actions
- Assuming pits uniformly distributed
  - Pit in (2, 2) with a higher probability

- Stench in start position (1, 1)
  - No safe actions
- Shoot straight
  - wumpus was there->dead->safe
  - wumpus wasn’t there -> safe

# Knowledge-based agents

- The agent draws a conclusion from the available information. That conclusion is guaranteed to be correct if the available information is correct.

- This is a fundamental property of logical reasoning.
- We describe how to build logical agents that can represent information and draw conclusions such as those described in the Wumpus world example.

# Outline

- Knowledge-based agents
- The Wumpus world example
- Logic in general
  - Models and Entailment
- Propositional logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
  - Search
  - Resolution
  - Forward chaining
  - Backward chaining

# Logic

- Logics are formal languages for representing information such that conclusions can be drawn
- Syntax
  - Defines the sentences in the language that are well formed
- Semantics
  - Defines the meaning of sentences
  - Defines the truth of each sentence w.r.t. each possible world
- Example: the language of arithmetic
  - Syntax:
    - x+y=4 is a sentence; x4y+= is not a sentence
  - Semantics:
    - x+y=4 is true in a world where x=2 and y=2
    - x+y=4 is false in a world where x=1 and y=1
    - In standard logics, every sentence must be either true or false in each possible world—there is no "in between."

# Model

- Use “model” to replace “possible world”
  - Sentence: x+y=4
  - Possible models are all possible assignments of nonnegative integers to the variables x and y.
- If a sentence α is true in model m, we say that m **satisfies** α or sometimes m is a **model** of α.
- We use the notation M(α) to mean the set of all models of α.
  - M(α)={(0, 4), (1, 3), (2, 2), (3, 1), (4, 0)}

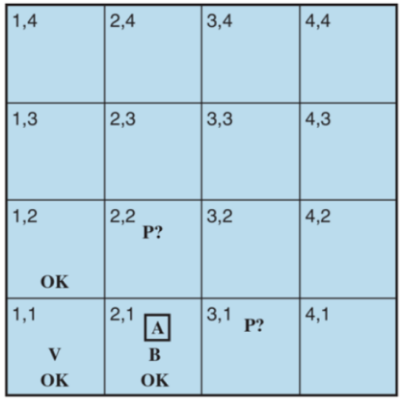
# Entailment

- Logical reasoning
- Logical entailment between sentences $\alpha \models \beta$
- The formal definition of entailment is: $\alpha \models \beta$ if and only if, in every model in which $\alpha$ is true, $\beta$ is also true.

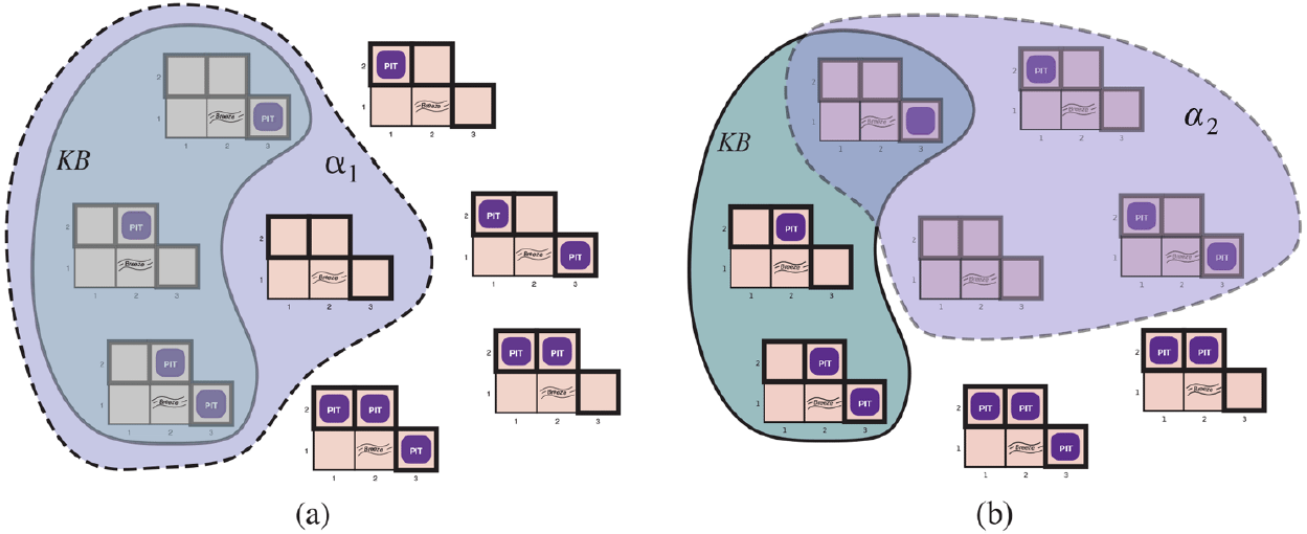
$$\alpha \models \beta \text{ if and only if } M(\alpha) \subseteq M(\beta)\,.$$

  - $\alpha$ is a stronger assertion than $\beta$
  - x=0 entails xy=0

# Models and Entailment
# Analysis of Wumpus

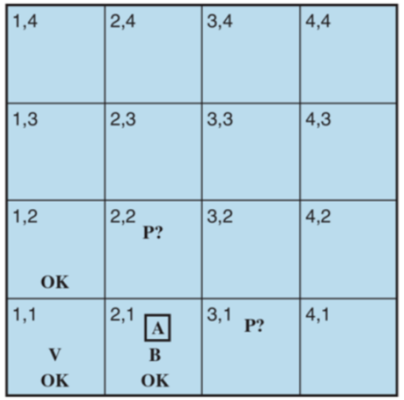



- Percepts: nothing in [1,1] and a breeze in [2,1]
- KB:
  - The above percepts
  - The rules of Wumpus
- Whether (1, 2), (2, 2), and (3, 1) contain pits.
- $2^3$ possible models.




Possible models for the presence of pits in squares [1,2], [2,2], and [3,1]. The KB corresponding to the observations of nothing in [1,1] and a breeze in [2,1] is shown by the solid line. (a) Dotted line shows models of $\alpha_1$ (no pit in [1,2]). (b) Dotted line shows models of $\alpha_2$ (no pit in [2,2]).

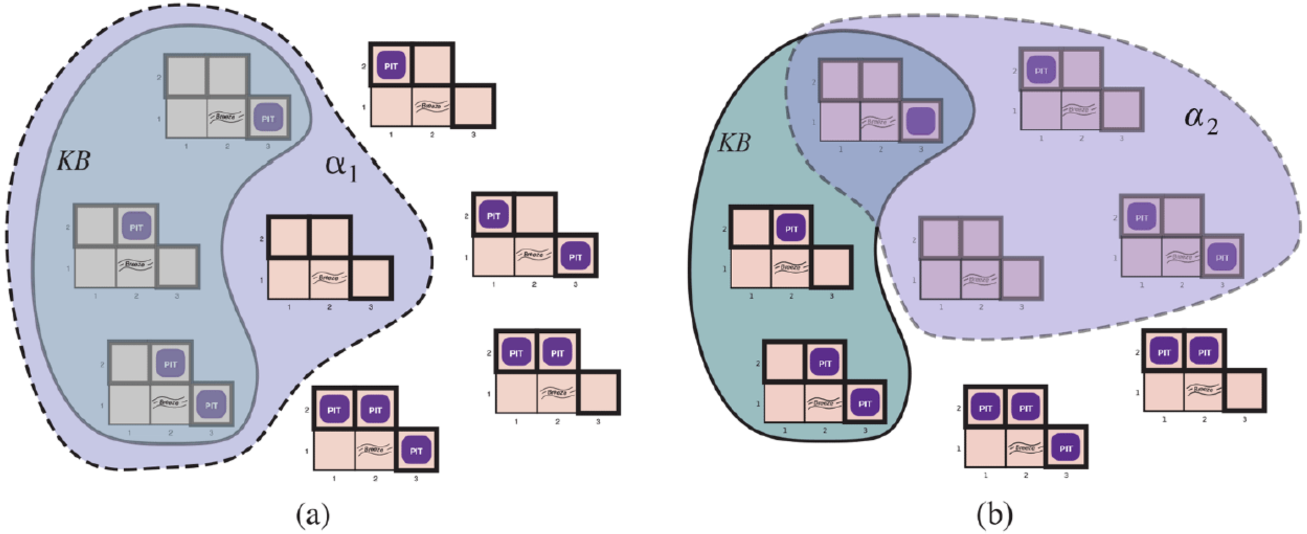
# Models and Entailment Analysis of Wumpus




- KB has a set of sentences
- KB is false in models that contradict what the agent knows
  - Percepts nothing in [1,1] and a breeze in [2.1]
- KB is true for three models
  - Surrounded by a solid line
- Sentences

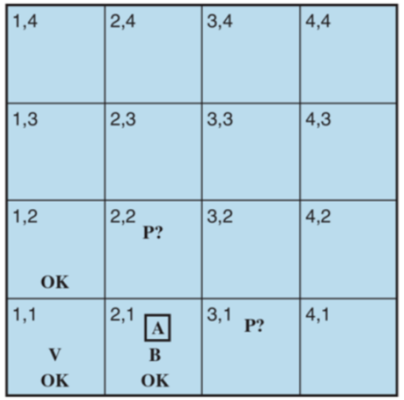
$\alpha_1 =$" There is no pit in [1,2]."

$\alpha_2 =$" There is no pit in [2,2]."




Possible models for the presence of pits in squares [1,2], [2,2], and [3,1]. The KB corresponding to the observations of nothing in [1,1] and a breeze in [2,1] is shown by the solid line. (a) Dotted line shows models of $\alpha_1$ (no pit in [1,2]). (b) Dotted line shows models of $\alpha_2$ (no pit in [2,2]).

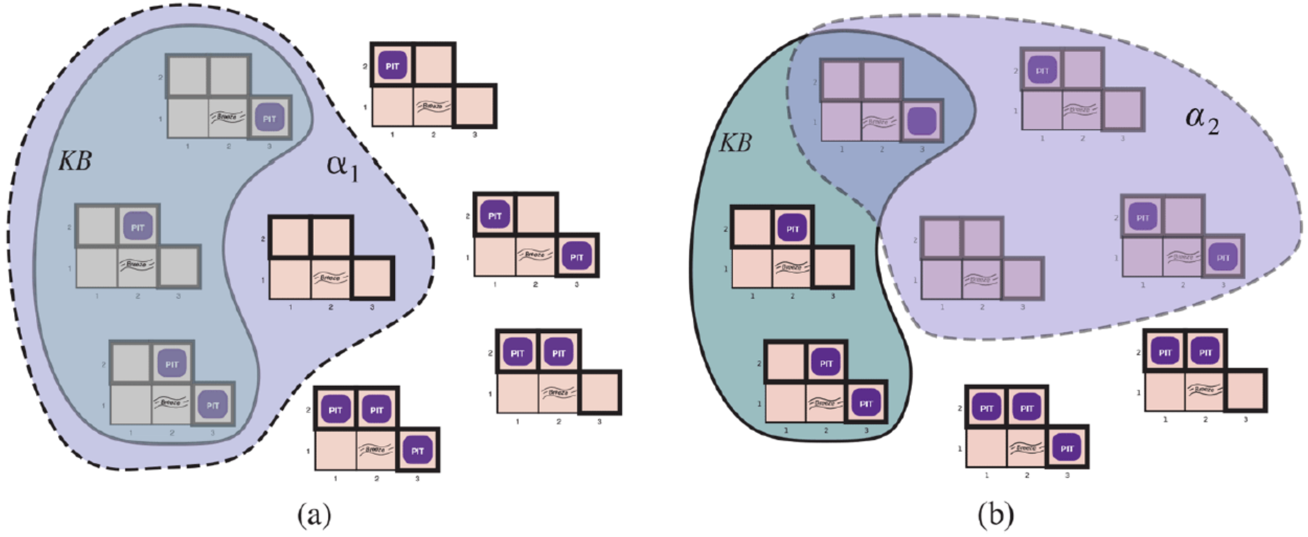
# Models and Entailment Analysis of Wumpus




- KB $\models \alpha_1$
  - In every model where KB is true, $\alpha_1$ is also true.
- KB does not entail $\alpha_2$
  - In some models where KB is true, $\alpha_2$ is not true

$\alpha_1 =$" There is no pit in [1,2]."

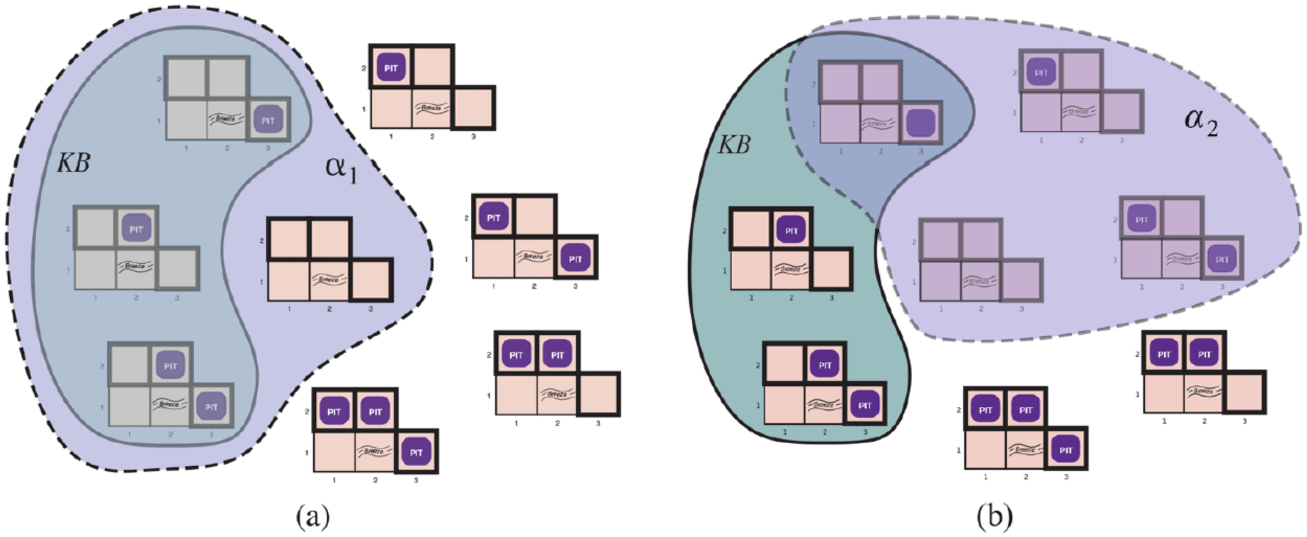
$\alpha_2 =$" There is no pit in [2,2]."




Possible models for the presence of pits in squares [1,2], [2,2], and [3,1]. The KB corresponding to the observations of nothing in [1,1] and a breeze in [2,1] is shown by the solid line. (a) Dotted line shows models of $\alpha_1$ (no pit in [1,2]). (b) Dotted line shows models of $\alpha_2$ (no pit in [2,2]).

# Model checking

- It enumerates all possible models to check that α is true in all models in which KB is true, that M(KB) ⊆ M(α).




Possible models for the presence of pits in squares [1,2], [2,2], and [3,1]. The KB corresponding to the observations of nothing in [1,1] and a breeze in [2,1] is shown by the solid line. (a) Dotted line shows models of $\alpha_1$ (no pit in [1,2]). (b) Dotted line shows models of $\alpha_2$ (no pit in [2,2]).

# Inference

- The set of all consequences of KB as a haystack; $\alpha$ as a needle.
- Entailment is like the needle in haystack
- Inference is like finding it
- An inference algorithm i can derive $\alpha$ from KB.
  - $KB \vdash_i \alpha$
  - $\alpha$ is derived from KB by i
  - i derives $\alpha$ from KB

# Inference

- Soundness: an inference algorithm i that derives only entailed sentences is called sound or truth-preserving

$$\text{whenever } KB \vdash_i \alpha \text{, it is also true that } KB \models \alpha$$

  - An unsound inference makes things up
- Completeness: an inference algorithm i that can derive any sentence that is entailed.

$$\text{whenever } KB \models \alpha \text{, it is also true that } KB \vdash_i \alpha$$

- 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

# Outline

- Knowledge-based agents
- The Wumpus world example
- Logic in general
- Propositional logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
  - Search
  - Resolution
  - Forward chaining
  - Backward chaining

# Propositional logic: a very simple logic

- Syntax defines the allowable sentences.
- Atomic sentences
  - Single proposition symbol
  - TRUE, FALSE
  - P, Q, R, $W_{1,3}$
- Complex sentences are constructed from simpler sentences using
  - parentheses, and
  - 5 logical connectives: not, and, or, implies, biconditional

# Syntax

- parentheses, and
- 5 logical connectives: not, and, or, implies, biconditional

The proposition symbols $P_1$, $P_2$ etc are sentences

If $P$ is a sentence, $\neg P$ is a sentence (negation)

If $P_1$ and $P_2$ are sentences, $P_1 \wedge P_2$ is a sentence (conjunction)

If $P_1$ and $P_2$ are sentences, $P_1 \vee P_2$ is a sentence (disjunction)

If $P_1$ and $P_2$ are sentences, $P_1 \implies P_2$ is a sentence (implication)

If $P_1$ and $P_2$ are sentences, $P_1 \Leftrightarrow P_2$ is a sentence (biconditional)

Figure 7.7

$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} & \rightarrow & \mathit{True} \mid \mathit{False} \mid P \mid Q \mid R \mid \ldots \\
\mathit{ComplexSentence} & \rightarrow & (\,\mathit{Sentence}\,) \\
& \mid & \neg\,\mathit{Sentence} \\
& \mid & \mathit{Sentence} \wedge \mathit{Sentence} \\
& \mid & \mathit{Sentence} \vee \mathit{Sentence} \\
& \mid & \mathit{Sentence} \;\Rightarrow\; \mathit{Sentence} \\
& \mid & \mathit{Sentence} \;\Leftrightarrow\; \mathit{Sentence} \\
\\
\textsc{Operator Precedence} & : & \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{array}
$$

A BNF (Backus–Naur Form) grammar of sentences in propositional logic, along with operator precedences, from highest to lowest.

# Semantics

- A model sets true or false for every proposition symbol.

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

- With three proposition symbols $P_{1,2}$, $P_{2,2}$, and $P_{3,1}$, we have $2^3$ models.
- Given a model, the semantics for propositional logic must specify how to compute the truth value of any sentence
  - Atomic sentences
  - Complex sentences

# Rules for evaluating truth with respect to a model m

- Five rules
- Truth table
- Recursive evaluation for an arbitrary sentence

- $\neg P$ is true iff $P$ is false in $m$.
- $P \wedge Q$ is true iff both $P$ and $Q$ are true in $m$.
- $P \vee Q$ is true iff either $P$ or $Q$ is true in $m$.
- $P \Rightarrow Q$ is true unless $P$ is true and $Q$ is false in $m$.
- $P \Leftrightarrow Q$ is true iff $P$ and $Q$ are both true or both false in $m$.

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

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1})$$

$$= true \wedge (false \vee true) = true \wedge true = true$$

| $P$ | $Q$ | $\neg P$ | $P \wedge Q$ | $P \vee Q$ | $P \Rightarrow Q$ | $P \Leftrightarrow Q$ |
|---|---|---|---|---|---|---|
| *false* | *false* | *true* | *false* | *false* | *true* | *true* |
| *false* | *true* | *true* | *false* | *true* | *true* | *false* |
| *true* | *false* | *false* | *false* | *true* | *false* | *false* |
| *true* | *true* | *false* | *true* | *true* | *true* | *true* |

Truth tables for the five logical connectives. To use the table to compute, for example, the value of $P \vee Q$ when $P$ is true and $Q$ is false, first look on the left for the row where $P$ is *true* and $Q$ is *false* (the third row). Then look in that row under the $P \vee Q$ column to see the result: *true*.

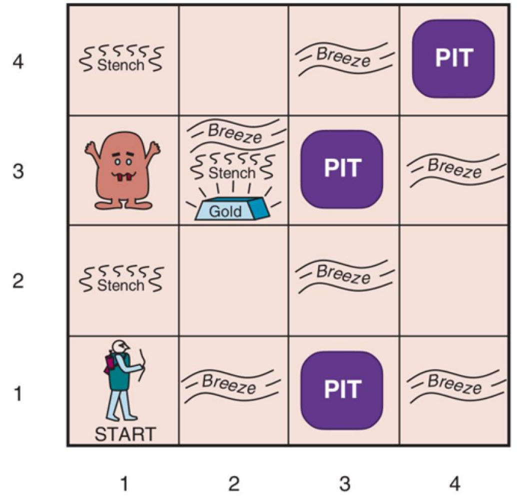
# A simple knowledge base

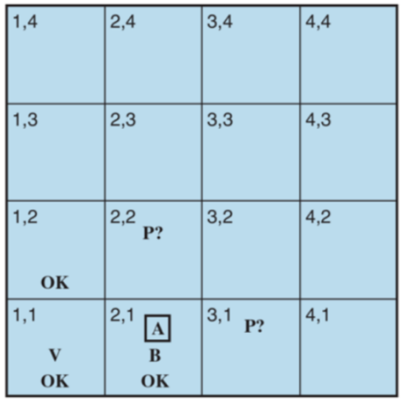
- Wumpus world
- There is no pit at start (1, 1). We use sentence $R_1$

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

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

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




$P_{x,y}$ is true if there is a pit in $[x,y]$.

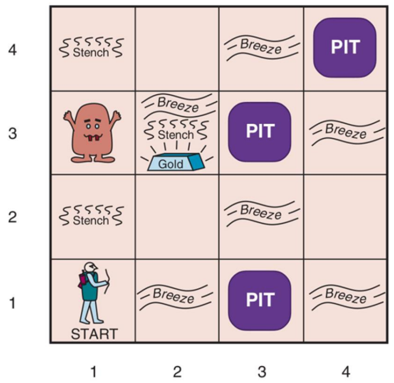
$W_{x,y}$ is true if there is a wumpus in $[x,y]$, dead or alive.

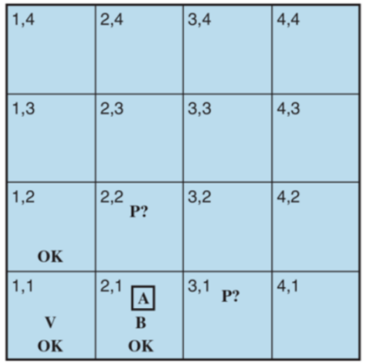
$B_{x,y}$ is true if there is a breeze in $[x,y]$.

$S_{x,y}$ is true if there is a stench in $[x,y]$.

$L_{x,y}$ is true if the agent is in location $[x,y]$.

# A simple knowledge base




- Previous sentences are true for all Wumpus worlds
  - Rules
- If we include the breeze percepts for the first two squares visited
  - Percepts
  - Did not perceive breeze in (1, 1)
  - Perceive breeze in (2, 1)

Rules of the Wumpus world

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

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

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

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

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

Based on the percept of a specific agent

# Inference on the KB

**Our KB now**

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

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

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

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

$R_5: B_{2,1}$.

- Our goal now is to decide whether KB $\models \alpha$
  - For example, is $\neg P_{1,2}$ entailed by our KB?
- Model checking
  - Enumerate the models and check whether $\alpha$ is true in every model in which KB is true.
- KB $\models \neg P_{1,2}$
- KB does not entail $P_{2,2}$

Figure 7.9

| $B_{1,1}$ | $B_{2,1}$ | $P_{1,1}$ | $P_{1,2}$ | $P_{2,1}$ | $P_{2,2}$ | $P_{3,1}$ | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R_5$ | $KB$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *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* | $\boxed{\textit{false}}$ | *false* | $\boxed{\textit{false}}$ | *true* | *true* | *true* | *true* | *true* | *true* | $\boxed{\underline{\textit{true}}}$ |
| *false* | *true* | *false* | $\boxed{\textit{false}}$ | *false* | $\boxed{\textit{true}}$ | *false* | *true* | *true* | *true* | *true* | *true* | $\boxed{\underline{\textit{true}}}$ |
| *false* | *true* | *false* | $\boxed{\textit{false}}$ | *false* | $\boxed{\textit{true}}$ | *true* | *true* | *true* | *true* | *true* | *true* | $\boxed{\underline{\textit{true}}}$ |
| *false* | *true* | *false* | *false* | *true* | *false* | *false* | *true* | *false* | *false* | *true* | *true* | *false* |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| *true* | *true* | *true* | *true* | *true* | *true* | *true* | *false* | *true* | *true* | *false* | *true* | *false* |

A truth table constructed for the knowledge base given in the text. $KB$ is true if $R_1$ through $R_5$ are true, which occurs in just 3 of the 128 rows (the ones underlined in the right-hand column). In all 3 rows, $P_{1,2}$ is false, so there is no pit in [1,2]. On the other hand, there might (or might not) be a pit in [2,2].

Mini quiz: How many models do we need to enumerate?

# Inference algorithm

- DFS
- When KB and α have n proposition symbols, worst case time complexity $O(2^n)$
- Space complexity O(n)

Figure 7.10

```
function TT-ENTAILS?(KB, α) returns true or false
  inputs: KB, the knowledge base, a sentence in propositional logic
          α, the query, a sentence in propositional logic

  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        // when KB is false, always return true
  else
     P ← FIRST(symbols)
     rest ← REST(symbols)
     return (TT-CHECK-ALL(KB, α, rest, model ∪ {P = true})
             and
             TT-CHECK-ALL(KB, α, rest, model ∪ {P = false }))
```

A truth-table enumeration algorithm for deciding propositional entailment. (TT stands for truth table.) PL-TRUE? returns *true* if a sentence holds within a model. The variable *model* represents a partial model—an assignment to some of the symbols. The keyword **and** here is an infix function symbol in the pseudocode programming language, not an operator in proposition logic; it takes two arguments and returns *true* or *false*.

# Outline

- Knowledge-based agents
- The Wumpus world example
- Logic in general
- Propositional logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
    - Search
    - Resolution
    - Forward chaining
    - Backward chaining

# Propositional Theorem Proving

- Theorem proving: applying rules of inference directly to the sentences in our knowledge base to construct a proof of the desired sentence without consulting models.
  - If the number of models is large but the length of the proof is short, then theorem proving can be more efficient than model checking.
- Logical equivalence

$$\alpha \equiv \beta \quad \text{if and only if} \quad \alpha \models \beta \text{ and } \beta \models \alpha \,.$$

# Logical eqivalence

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

# Validity and satisfiability

- A sentence is valid if it is true in all models

$$\text{e.g.}, True, \quad A \lor \neg A, \quad A \implies A,$$

- A sentence is satisfiable if it is true in some model
  - $R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$ is satisfiable as there are 3 models in which it is true
  - $A \vee B$
- A sentence is unsatisfiable if it is not true for any model
  - $A \wedge \neg A$
- Validity and satisfiability are connected

$$\alpha \text{ is valid iff } \neg\alpha \text{ is unsatisfiable}$$

$$\alpha \models \beta \textit{ if and only if the sentence } (\alpha \wedge \neg\beta) \textit{ is unsatisfiable.}$$

# Outline

- Knowledge-based agents
- The Wumpus world example
- Logic in general
- Propositional logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
  - Search with inference rules
  - Resolution
  - Forward chaining
  - Backward chaining

# Proof by inference rules

- Inference rules that can derive proof.
  - A chain of conclusions that leads to the desired goal.
- Given: some sentences on the top of the line
- We have: some sentences in the bottom of the line

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

- Modus Ponens
- And-Elimination
- Logical equivalences

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

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

# Example

- Given the KB containing $R_1$ to $R_5$, how to prove $\neg P_{1,2}$

**KB**

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

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

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

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

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

**1.** Apply biconditional elimination to $R_2$ to obtain

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

**2.** Apply And-Elimination to $R_6$ to obtain

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

**3.** Logical equivalence for contrapositives gives

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

**4.** Apply Modus Ponens with $R_8$ and the percept $R_4$ (i.e., $\neg B_{1,1}$), to obtain

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

**5.** Apply De Morgan's rule, giving the conclusion

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

That is, neither [1,2] nor [2,1] contains a pit.

# Proof by Search

- **INITIAL STATE:** the initial knowledge base.
- **ACTIONS:** the set of actions consists of all the inference rules applied to all the sentences that match the top half of the inference rule.
- **RESULT:** the result of an action is to add the sentence in the bottom half of the inference rule.
- **GOAL:** the goal is a state that contains the sentence we are trying to prove.

- Search vs enumerating models
  - Search can be more efficient
  - $R_1$, $R_3$ and $R_5$ have no bearing on the proof.

# Outline

- Knowledge-based agents
- The Wumpus world example
- Logic in general
- Propositional logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
    - Search
    - Resolution
    - Forward chaining
    - Backward chaining

# Proof by resolution

- Resolution inference

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

  - $l_i$ and $m_j$ are complementary literals
- Example

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

- Resolution is sound and complete for propositional logic

# Conversion to CNF (Conjunctive Normal Form)

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

"If breeze, then a pit adjacent."

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

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \implies \beta) \wedge (\beta \implies \alpha)$.

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

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 distributivity law ($\vee$ over $\wedge$) and flatten:

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

$\alpha \models \beta$ *if and only if the sentence* $(\alpha \land \neg\beta)$ *is unsatisfiable.*

# A Resolution Algorithm

- To prove KB ⊨ α, we show (KB ∧ ¬ α) is unsatisfiable
- Covert (KB ∧ ¬ α) into CNF, resolution rule is applied to the resulting clauses, until one of the following two things happens
  - there are no new clauses that can be added, in which case $KB$ does not entail $\alpha$; or,
  - two clauses resolve to yield the *empty* clause, in which case $KB$ entails $\alpha$.

# Resolution Example

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

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

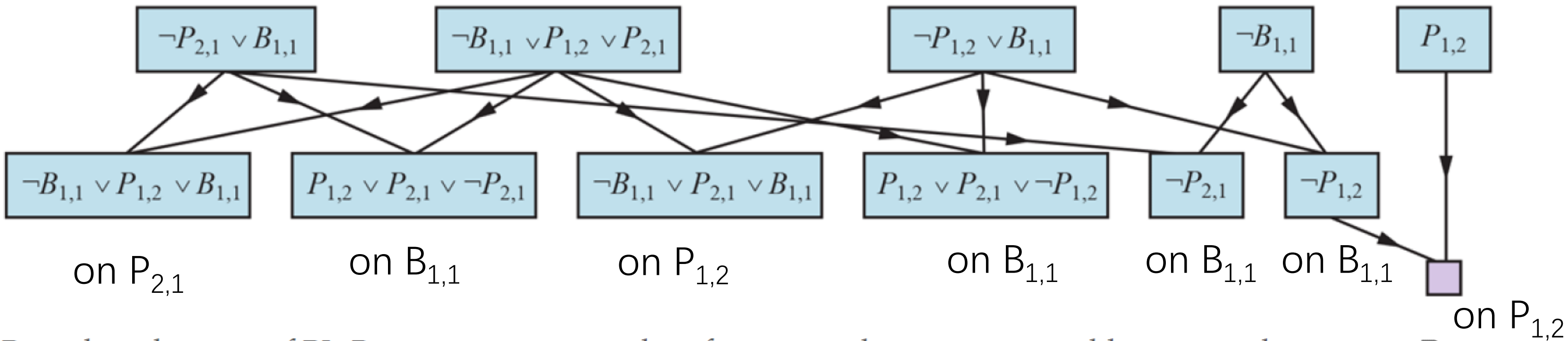
$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$$

- To prove KB $\models \alpha$, we show (KB $\wedge \neg \alpha$) is unsatisfiable
- Covert (KB $\wedge \neg \alpha$) into CNF
- Empty clause
- KB $\models \alpha$ is proven

Figure 7.14

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

Resolution Inference




Partial application of PL-RESOLUTION to a simple inference in the wumpus world to prove the query $\neg P_{1,2}$. Each of the leftmost four clauses in the top row is paired with each of the other three, and the resolution rule is applied to yield the clauses on the bottom row. We see that the third and fourth clauses on the top row combine to yield the clause $\neg P_{1,2}$, which is then resolved with $P_{1,2}$ to yield the empty clause, meaning that the query is proven.

# Outline

- Knowledge-based agents
- The Wumpus world example
- Logic in general
- Propositional logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
    - Search
    - Resolution
    - Forward chaining
    - Backward chaining

$$(\neg L_{1,1} \vee \neg Breeze \vee B_{1,1})$$

# Horn clauses

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

- A horn clause is a disjunction of literals of which at most one is positive.

- Or it is in the form of
  - proposition symbol; or
  - (conjunction of symbols) => symbol

$$C, \quad B \implies A, \quad C \wedge D \implies B$$

$$(\alpha \implies \beta) \equiv (\neg\alpha \vee \beta) \quad \text{implication elimination}$$

- Horn KB
  - Conjunction of Horn clauses

# Forward chaining

- Determine if a single proposition symbol q, the query, is entailed by a KB of Horn clauses.
- Start with given proposition symbols (atomic sentence)
  - e.g., A and B
- Iteratively try to infer truth of additional proposition symbols
  - e.g., A $\wedge$ B => C, therefor, we establish C is true
- Continue until
  - No more inference can be carried out, or
  - Goal is reached

# Forward chaining

- AND-OR graph
- q=Q
- Whether the Horn KB entails q.

**Horn KB**

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

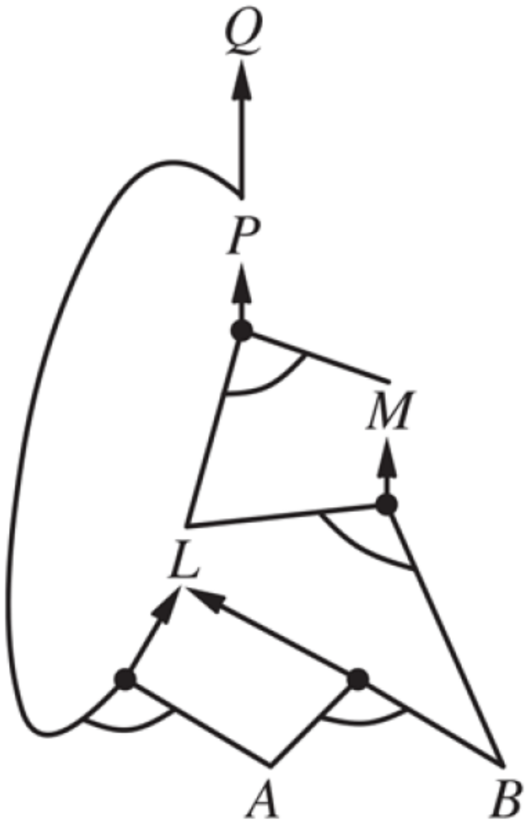
$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

$A$

$B$

(a)




(b)

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

AND node: conjunction—every edge must be proved
OR node: disjunction—any edge can be proved.

# Example

- Each AND node has Count[c]
  - Count[c]: Number of symbols in clause c's premise
- Queue
  - Tracks of symbols to be true
  - A and B in the beginning

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

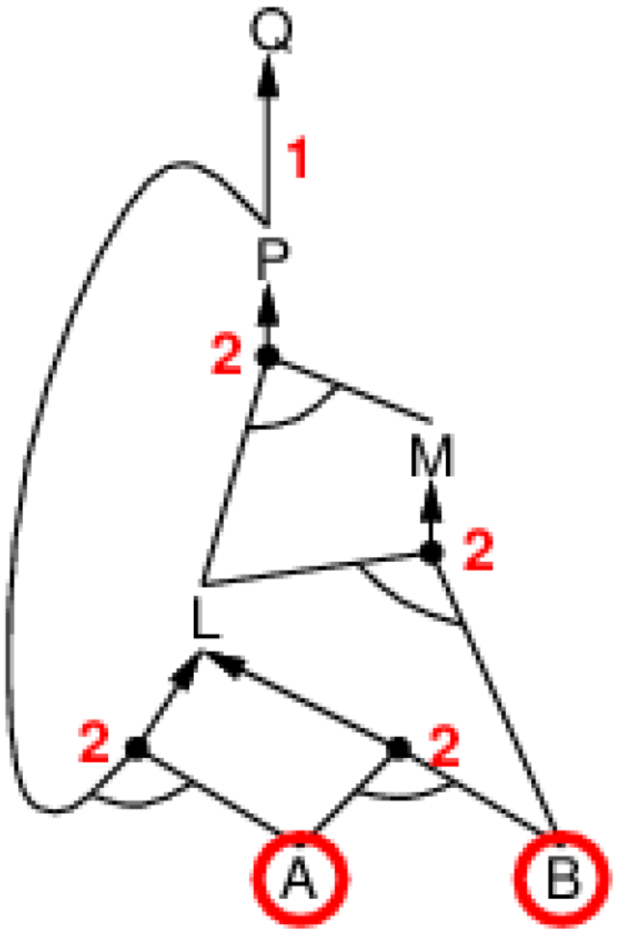
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

$A$

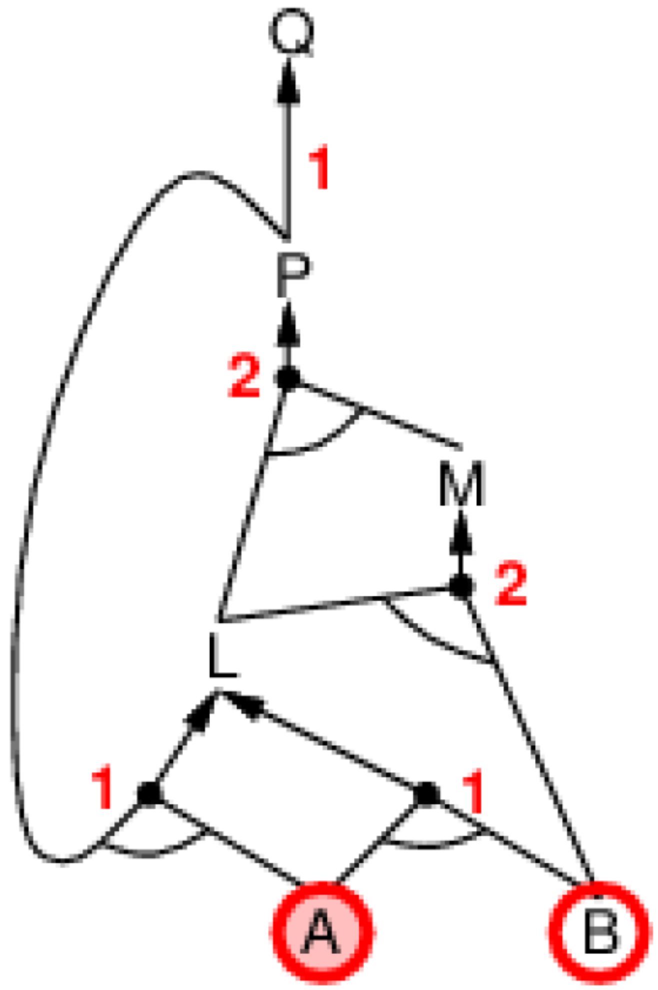
$B$




Queue: A, B

- Process A
- Decrease count for horn clauses in which A is premise

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




Queue: B

- Process B
- Decrease count for horn clauses in which B is premise
- L is added into the queue as A ∧ B =>L clause count is 0.

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

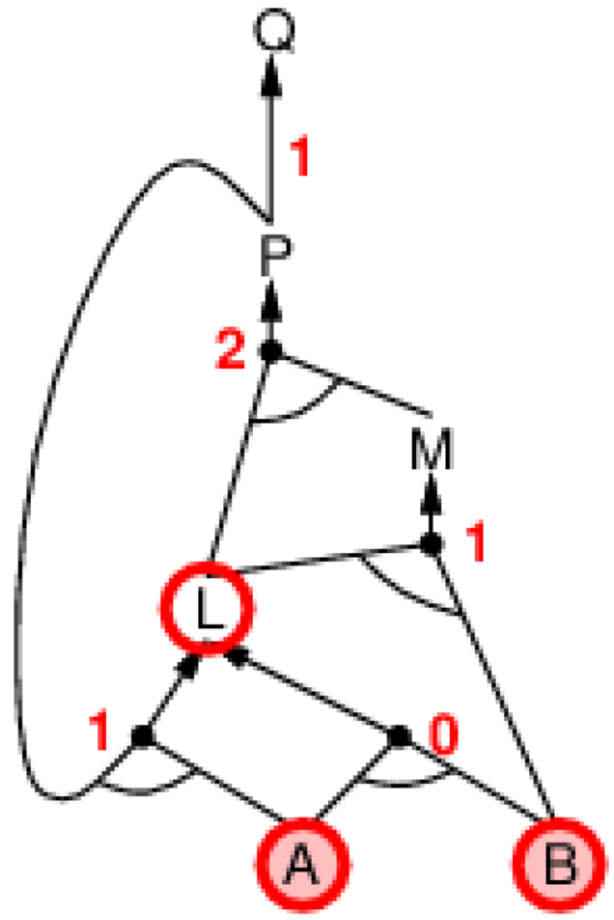
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

$A$

$B$




B is dequeued from the Queue, and then enqueue L
Queue: L

- Process L
- Decrease count for horn clauses in which L is premise
- M is added into the queue as L ∧ B =>M clause count is 0.

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

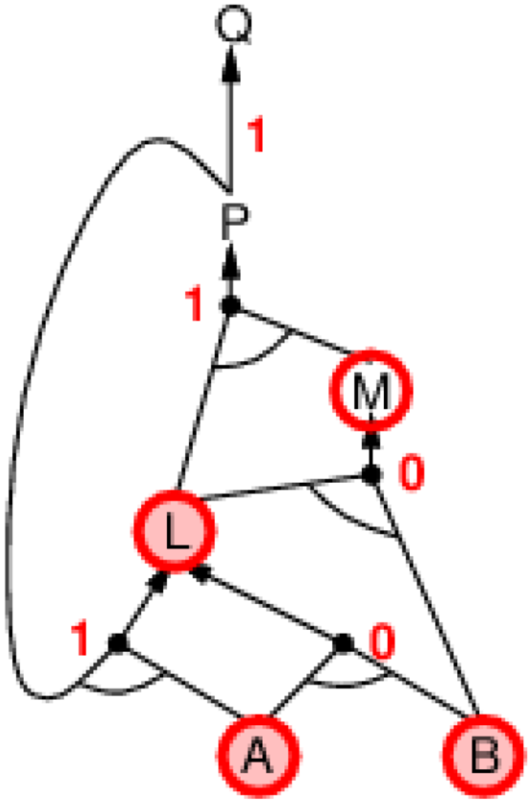
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

$A$

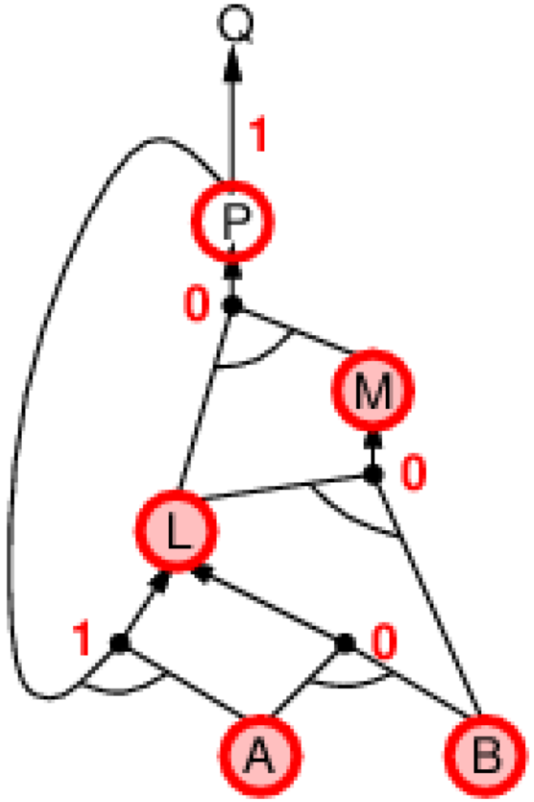
$B$




L is dequeued from the Queue, and then enqueue M
Queue: M

- Process M
- Decrease count for horn clauses in which M is premise
- P is added into the queue as $L \wedge M \Rightarrow P$ clause count is 0.

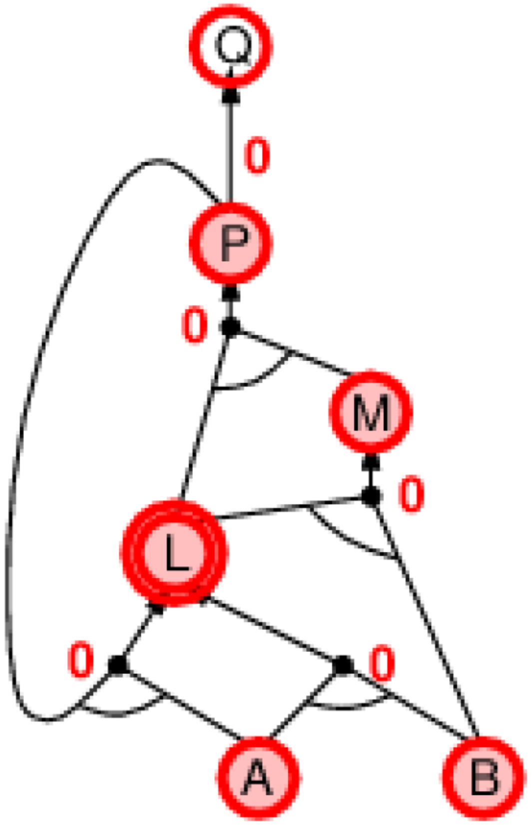
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$




M is dequeued from the Queue, and then enqueue P
Queue: P

- Process P
- Decrease count for horn clauses in which P is premise
- Q is added into the queue as P=>Q clause count is 0.
- Q is entailed by the Horn KB

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




P is dequeued from the Queue, and then enqueue Q
Queue: Q

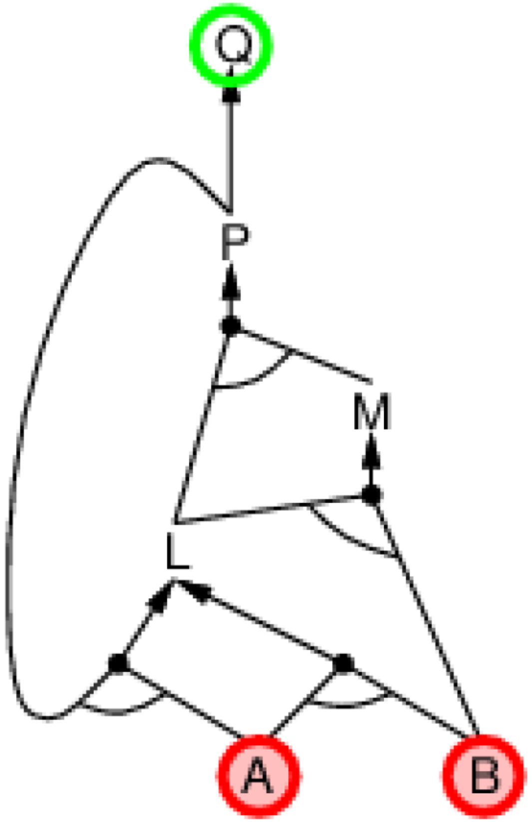
# Outline

- Knowledge-based agents
- The Wumpus world example
- Logic in general
- Propositional logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
  - Search
  - Resolution
  - Forward chaining
  - Backward chaining

# Backward chaining

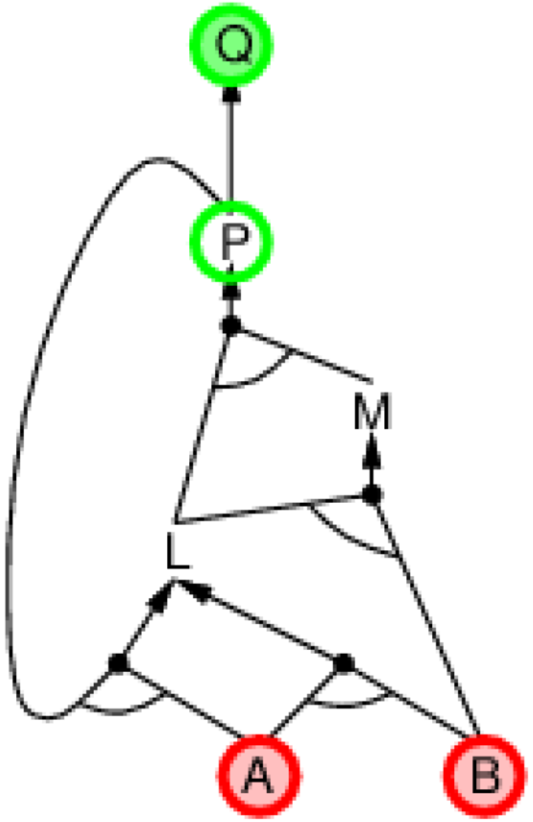
- Work backwards from the query q
- To prove q by BC,
  - check if q is known already, or
  - prove by BC all premises of some rule concluding q
- Avoid loops: check if new subgoal is already on the goal stack
- Avoid repeated work: check if new subgoal
  - has already been proved true, or
  - has already failed

- A and B are true
- Q needs to be proven



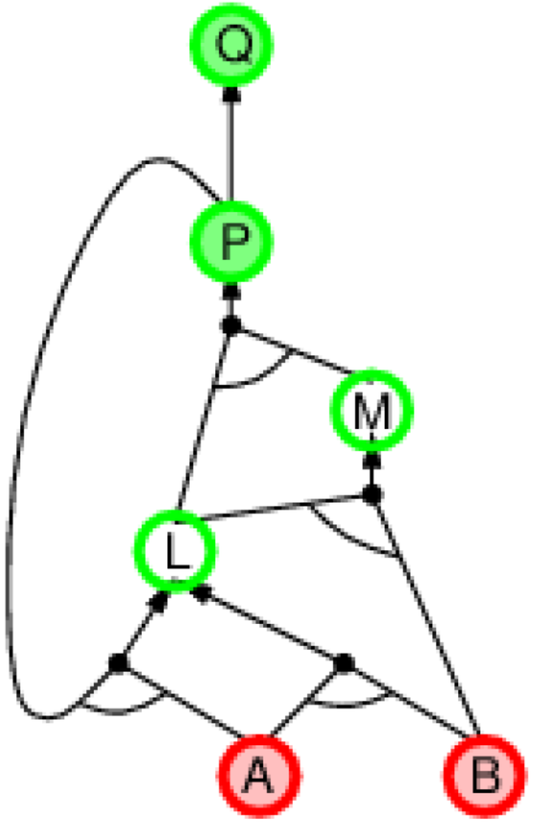

- Q

- Process Q
- P needs to be proven



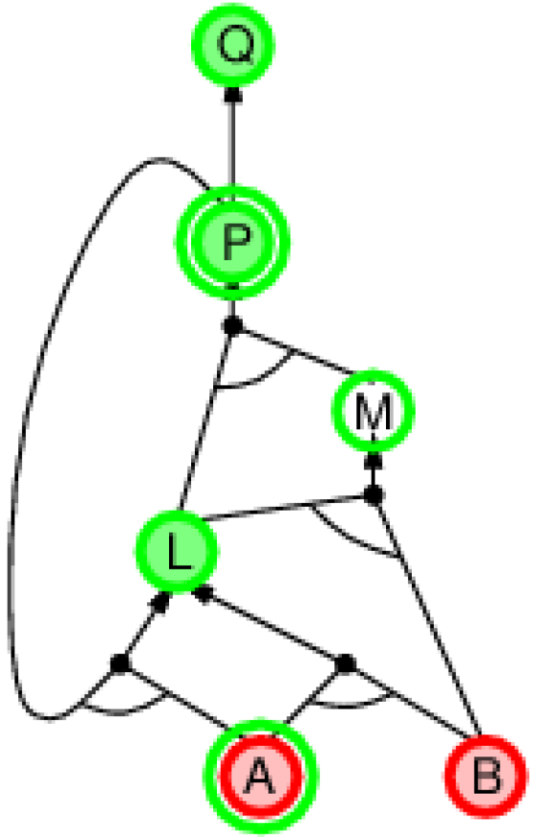

- Q, P

- Process P
- L and M need to be proven



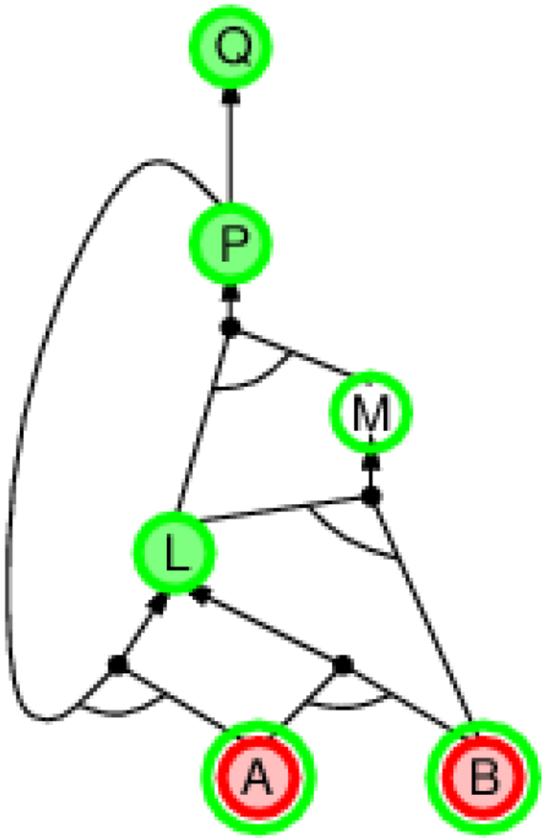

- Q, P, M, L

- Process L
- P and A need to be proven
- A is already true
- P is already a goal, repeated subgoal



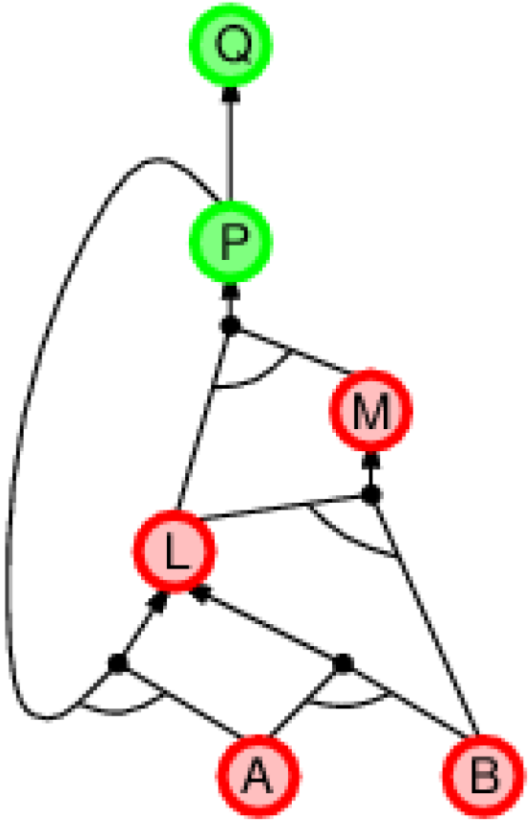

- Q, P, M, L

- Process L
- A and B need to be proven
- Both are true
- So L is true



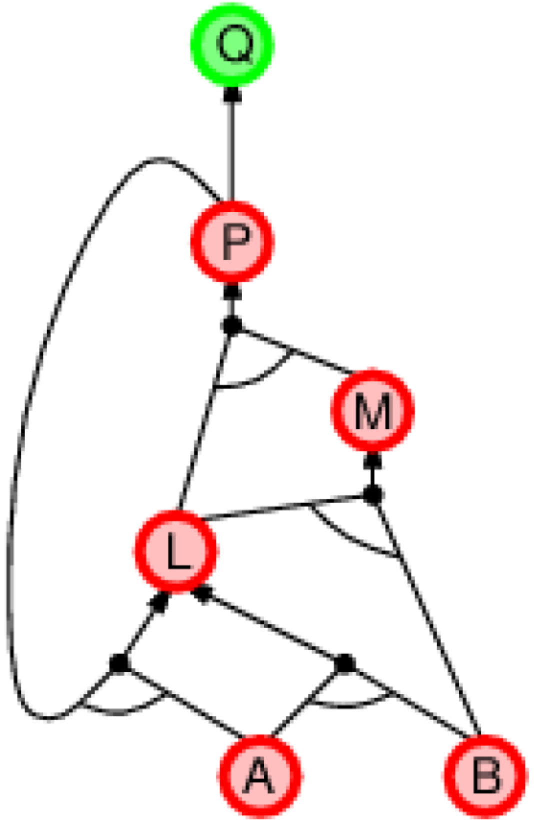

- Q, P, M

- Process M
- L and B need to be proven
- Both are true
- So M is true

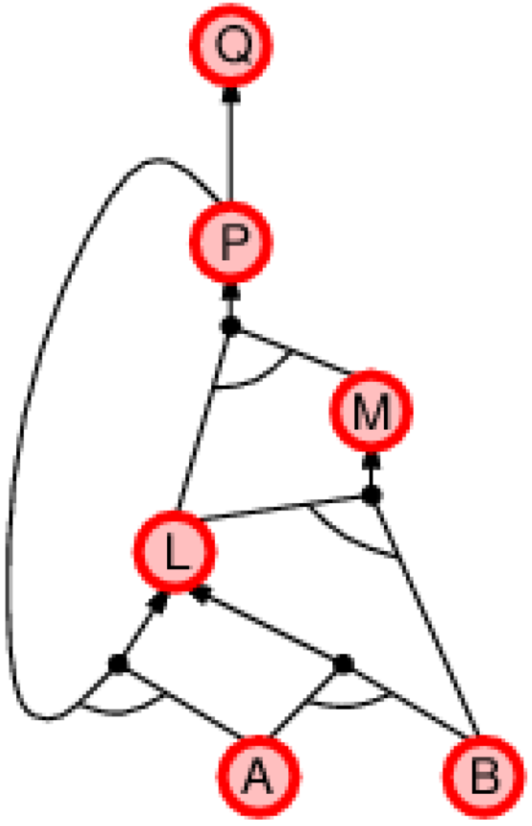



- Q, P

- Process P
- L and M need to be proven
- Both are true
- So P is true




- Process Q
- P needs to be proven
- P is true
- So Q is true




- Q

# Forward vs. backward chaining

- FC is data driven,
  - May do lots of work that is irrelevant to the goal

- BC is goal driven
  - Complexity of BC can be much less than linear in size of KB
  - Only touches relevant facts

# Logical agents

- Logical agent for Wumpus world explores actions
  - observe glitter, done
  - unexplored safe spot, plan route to it
  - If Wampus in possible spot, shoot arrow
  - take a risk to go possibly risky spot

- Propositional logic to infer state of the world
- Heuristic search to decide which action to take

# Lecture 7 ILOs

- Knowledge-based agents
  - Wumpus world
- Logic in general
  - Models and entailment
- Propositional logic
  - Symbols, parentheses, and 5 logical connectives: not, and, or, implies, biconditional
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
  - Search with inference rules
  - Resolution
  - forward chaining, only for Horn KB
  - backward chaining, only for Horn KB