

# Artificial Intelligence (CS303)

## Lecture 10: Logical Agents

# Hints for this lecture

- Human not only act based on instinct (gene? Program?), but also act based on knowledge.
- Represent, store, and exploit knowledge should also be important (or at least useful) for AI.

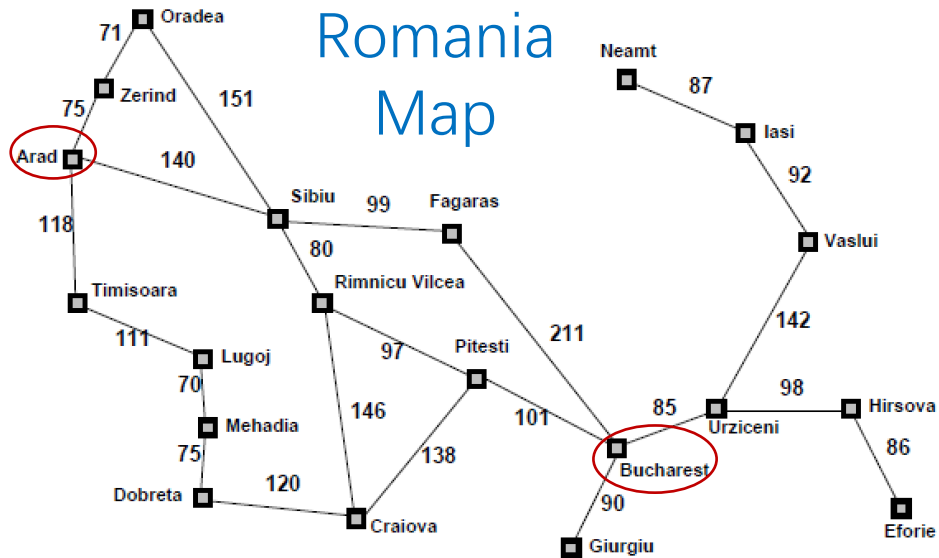
# Outline of this lecture

- **Knowledge-based Agents**
- **Represent Knowledge with Logic**
- **(Propositional) Logic**
- **Inference with Propositional Logic**

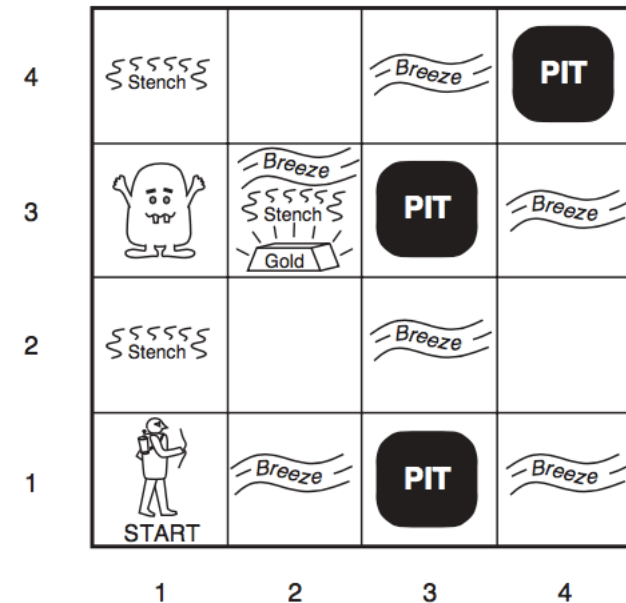
# **I. Knowledge-based Agents**

# Knowledge is important

- We (human) perceive the world, accumulate our knowledge, and act based on our perception **and knowledge**.
- In some cases, knowledge is not merely useful, but **crucial**.



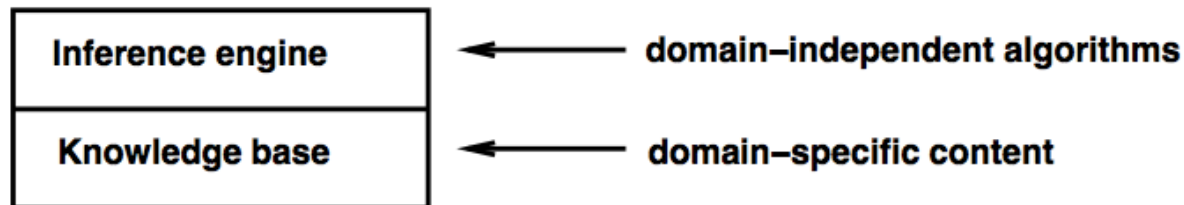
Difference?



# But what is knowledge?

- Human gain knowledge from **experience**.
- Knowledge is something **abstract**.
- A **neural network** trained with data fulfills the above two conditions, but does not fit our intuition about knowledge.
  - Knowledge is represented/stored/transferred/...through **language**.
- A “language” to **represent** knowledge should be defined first.
  - The language should be understandable by human.

# Knowledge-based Agents



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

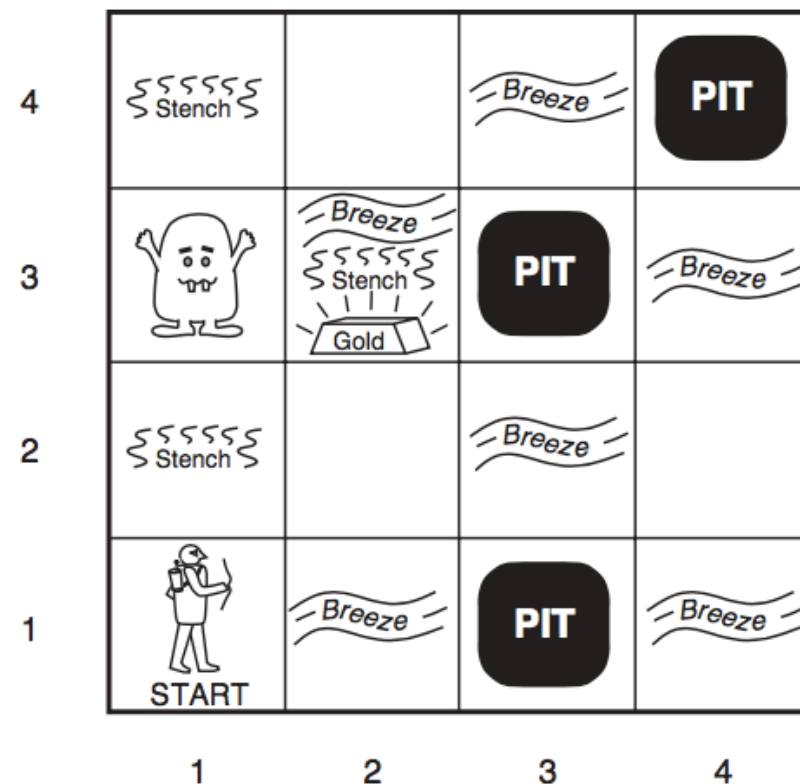
Then it can **ASK** itself what to do—answers should follow from the KB

Agents can be viewed at the **knowledge level**

i.e., **what they know**, regardless of how implemented

Or at the **implementation level**

i.e., data structures in KB and algorithms that manipulate them



## **II. Represent Knowledge with Logic**



# Sentences

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;  $x^2 + y >$  is not a sentence

$x + 2 \geq y$  is true iff the number  $x + 2$  is no less than the number  $y$

Logic is a **formal** language for representing knowledge

# Relationship Between Sentences

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

$$KB \models \alpha$$

Knowledge base  $KB$  entails sentence  $\alpha$

if and only if

$\alpha$  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”

# “Worlds”

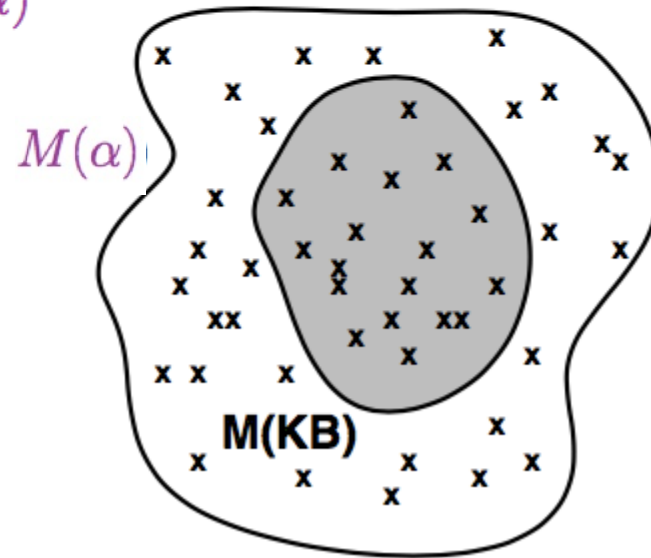
Logicians typically think in terms of **models**, which are formally structured worlds with respect to which truth can be evaluated

We say  $m$  is a **model** of a sentence  $\alpha$  if  $\alpha$  is true in  $m$

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

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

E.g.  $KB =$  Giants won and Reds won  
 $\alpha =$  Giants won



# Inference

Inference: the **procedure** of deriving a sentence from another sentence

**Model Checking**: A basic (and general) idea to inference

$KB \vdash_i \alpha$  = sentence  $\alpha$  can be derived from  $KB$  by procedure  $i$

Consequences of  $KB$  are a haystack;  $\alpha$  is a needle.

Entailment = needle in haystack; inference = finding it

**Soundness**:  $i$  is sound if

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

**Completeness**:  $i$  is complete if

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

### **III. (Propositional) Logic**

# Propositional Logic: Syntax

Propositional logic is the simplest logic—illustrates basic ideas

The proposition symbols  $P_1, P_2$  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

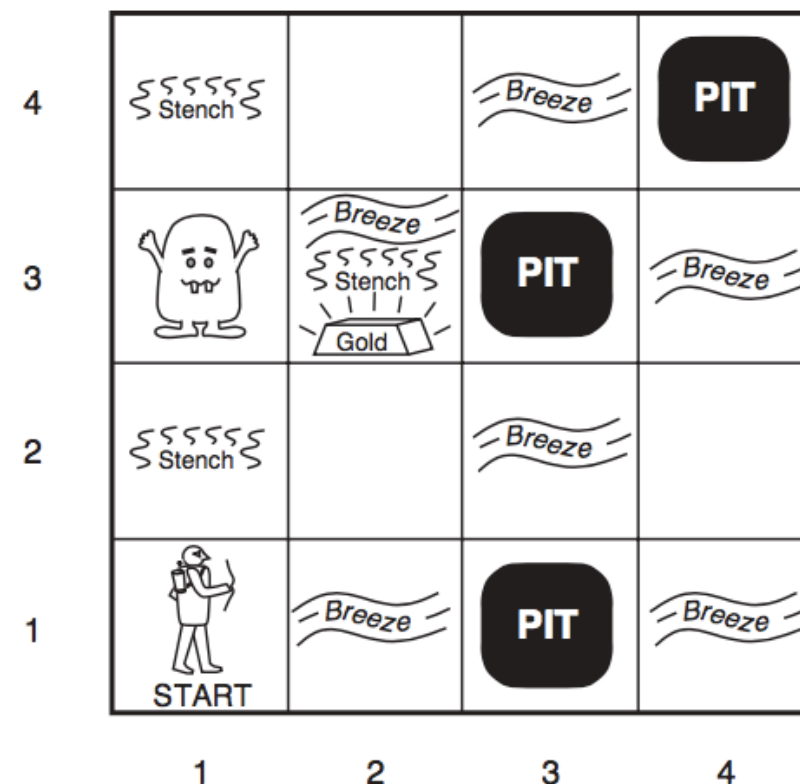
Each model specifies true/false for each proposition symbol

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

(With these symbols, 8 possible models, can be enumerated automat

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 <b>and</b> $S_2$ is true
$S_1 \vee S_2$ is true iff	$S_1$ is true <b>or</b> $S_2$ is true
$S_1 \Rightarrow S_2$ is true iff	$S_1$ is false <b>or</b> $S_2$ is true
i.e., is false iff	$S_1$ is true <b>and</b> $S_2$ is false
$S_1 \Leftrightarrow S_2$ is true iff	$S_1 \Rightarrow S_2$ is true <b>and</b> $S_2 \Rightarrow S_1$ is true



## **IV. Inference with Propositional Logic**



# Example for the Wumpus World

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

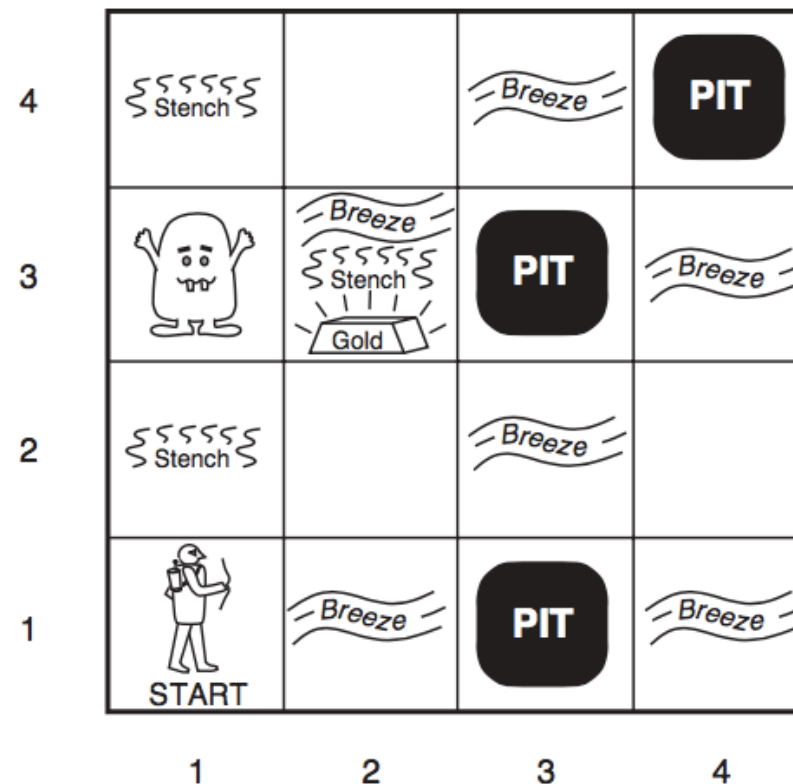
$W_{x,y}$  is true if there is a wumpus in  $[x, y]$ , dead or alive.

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

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

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

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



- Can we infer that “**position [3, 1] is safe ( $\alpha$ )**” in the Wumpus world given a **KB**?

# Inference by Enumeration

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	$R_1$	$R_2$	$R_3$	$R_4$	$R_5$	$KB$
false	false	false	false	false	false	false	true	true	true	true	false	false
false	false	false	false	false	false	true	true	true	false	true	false	false
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	true	false	false	false	false	false	true	true	false	true	true	false
false	true	false	false	false	false	true	true	true	true	true	true	true
false	true	false	false	false	true	false	true	true	true	true	true	true
false	true	false	false	false	true	true	true	true	true	true	true	true
false	true	false	false	true	false	false	true	false	false	true	true	false
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	true	true	true	true	true	true	false	true	true	false	true	false

Enumerate rows (different assignments to symbols),  
if  $KB$  is true in row, check that  $\alpha$  is too

# Inference by Enumeration

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	$R_1$	$R_2$	$R_3$	$R_4$	$R_5$	$KB$
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
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$O(2^n)$ for $n$ symbols; problem is <b>co-NP-complete</b>												
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
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
true	true	true	true	true	true	true	false	true	true	false	true	false

Enumerate rows (different assignments to symbols),  
if **KB** is true in row, check that  $\alpha$  is too

# Conjunctive Normal Form (CNF)

- To make inference more efficient/effective, we need more
- Any sentence of propositional logic is equivalent to a conjunction of clauses.

$$CNFSentence \rightarrow Clause_1 \wedge \dots \wedge Clause_n$$

$$Clause \rightarrow Literal_1 \vee \dots \vee Literal_m$$

$$Literal \rightarrow Symbol \mid \neg Symbol$$

$$Symbol \rightarrow P \mid Q \mid R \mid \dots$$

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

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

What is the relationship between CSP and SAT?

# Satisfiability Problem

- Propositional model checking could be done by solving a **satisfiability problem**.
  - Put the KB and ASK into CNF, and solve the SAT problem.

A sentence is **valid** if it is true in **all** models,

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

Validity is connected to inference via the **Deduction Theorem**:

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

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

e.g.,  $A \vee B$ ,  $C$

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

e.g.,  $A \wedge \neg A$

Satisfiability is connected to inference via the following:

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

**SAT**: (Boolean) Satisfiability Problem

# Backtracking Algorithm for SAT (DPLL)

- A search algorithm for SAT.
- Similar to Backtracking for CSP, but using different problem-dependent information/heuristics, such as
  - Early Termination
  - Pure symbol heuristic
  - Unit clause heuristic

# Inference by Theorem Proving

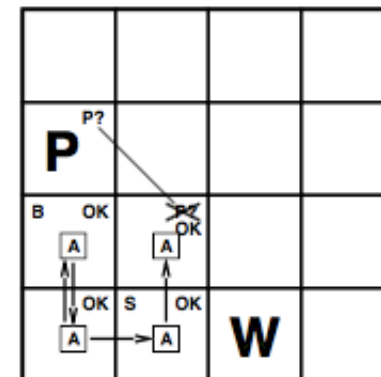
**Resolution** inference rule (for CNF): complete for propositional logic

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

where  $\ell_i$  and  $m_j$  are complementary literals. E.g.,

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

## Resolution is sound and complete for propositional logic





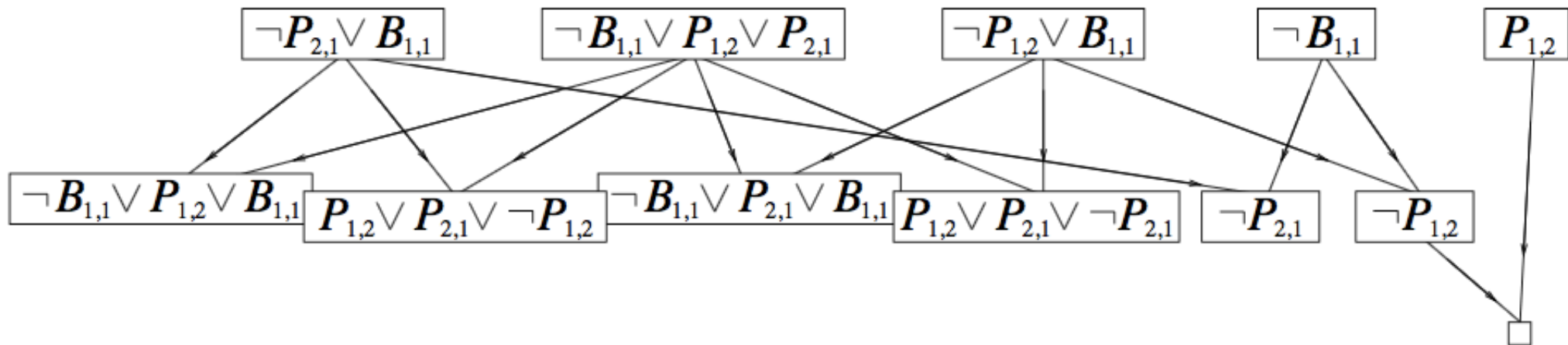
# Inference by Theorem Proving

- Apply inference rules to generate new sentences based on old ones (i.e., without searching in the model space).
- Inference rules works like search operators.
- Example: Resolution algorithm
  - Convert the sentence to “ASK” (or to proof) into CNF.
  - Use Resolution rules as the inference rule.
- 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  $\alpha$ .

# Inference by Theorem Proving

An example of resolution algorithm

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



# Wumpus Again

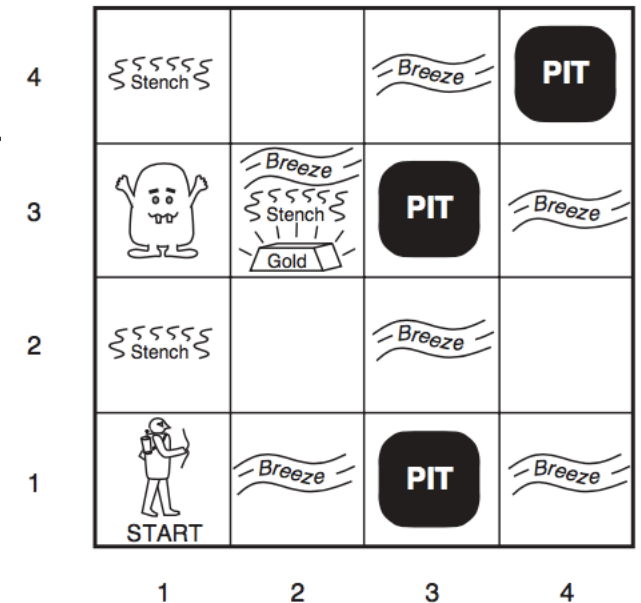
## Before Playing

- Write KB in the form of propositional logic
- Transform the prior knowledge into CNF

## During Playing

- Inference based on the KB
  - Formulate a query (e.g.,  $P_{1,2}$  is true) in CNF
  - Combine the query and the KB as a sentence in CNF, leading to a SAT problem.
  - Use a heuristic search algorithm (e.g., PDLL) to solve the Boolean SAT problem.
  - If the SAT is satisfiable, then the query is false.
  - Otherwise, it is true.
  - Repeat
- Make decisions based on the results of inference
- Action (move to a new place)
- Update the KB with the newly acquired knowledge (e.g.,  $P_{2,1}$  is false)
- Repeat

$P_{x,y}$  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 the agent perceives a breeze in  $[x, y]$ .  
 $S_{x,y}$  is true if the agent perceives a stench in  $[x, y]$ .



To be continued