

# Artificial Intelligence

## 7 Logical Agents

Russell & Norvig, AI: A Modern Approach, 3rd Ed

Lec Zinia Sultana  
CSE Dept, MIST

# Logical Agents

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

# Outline

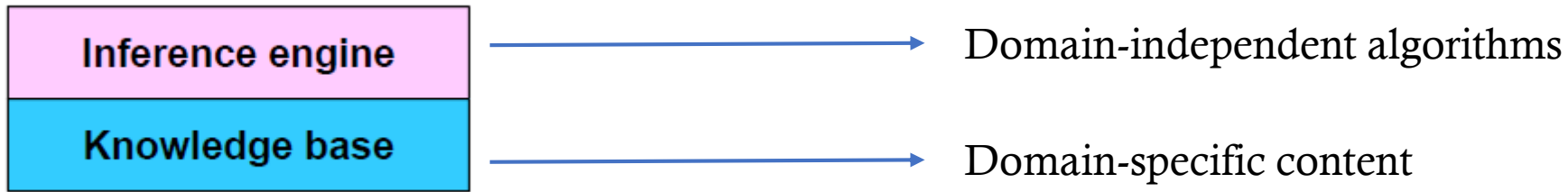
- ❖ Knowledge-based agents
- ❖ The Wumpus World
- ❖ Logic in general—models and entailment
- ❖ Propositional (Boolean) logic
- ❖ Equivalence, validity, satisfiability
- ❖ Inference rules and theorem proving, CNF
- ❖ Horn clauses and definite clauses
  - ❖ Forward chaining and Backward chaining
- ❖ Effective propositional model checking: DPLL and WalkSat

# Knowledge-based agents

Knowledge-based agents can

- ✓ 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; and
- ✓ they can adapt to changes in the environment by updating the relevant knowledge.

# Knowledge-based agents



- ❑ Knowledge base (KB) = set of sentences in a **formal** language
- ❑ Each sentence is expressed in a language called a **knowledge representation language**
- ❑ The agent maintains a knowledge base, KB, which may initially contain some **background knowledge**.

# Knowledge-based agents

The agent does **three** things.

1. It **TELLs** the knowledge base what it perceives.
2. 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.
3. The agent program TELLs the knowledge base which action was chosen, and the agent executes the action

# A generic knowledge-based agent

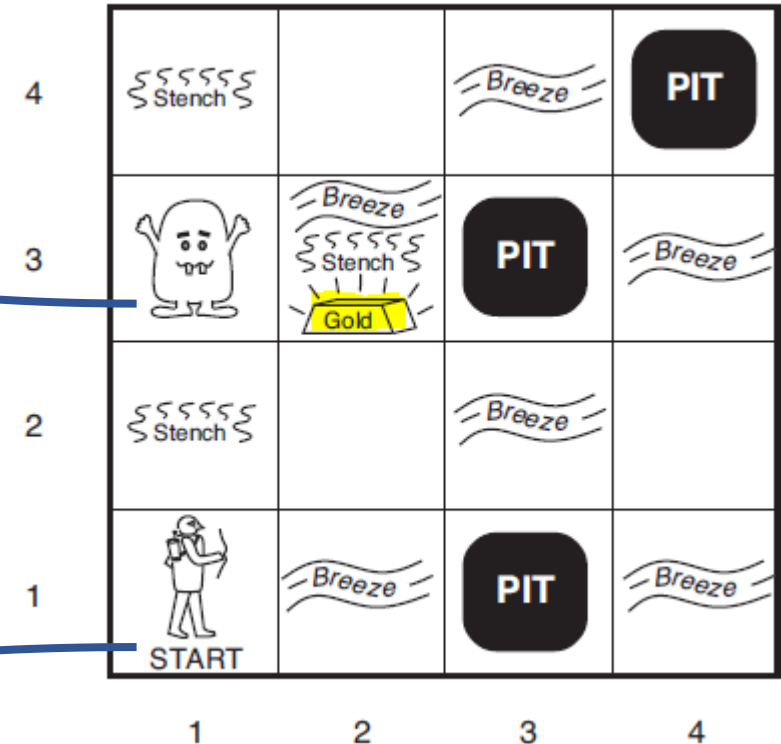
```
function KB-AGENT( percept) returns an action
  static: KB, a knowledge base
  t, a counter, initially 0, indicating time
  TELL(KB, MAKE-PERCEPT-SENTENCE( percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

# The Wumpus World- Example

A Computer game from 1972



Agent



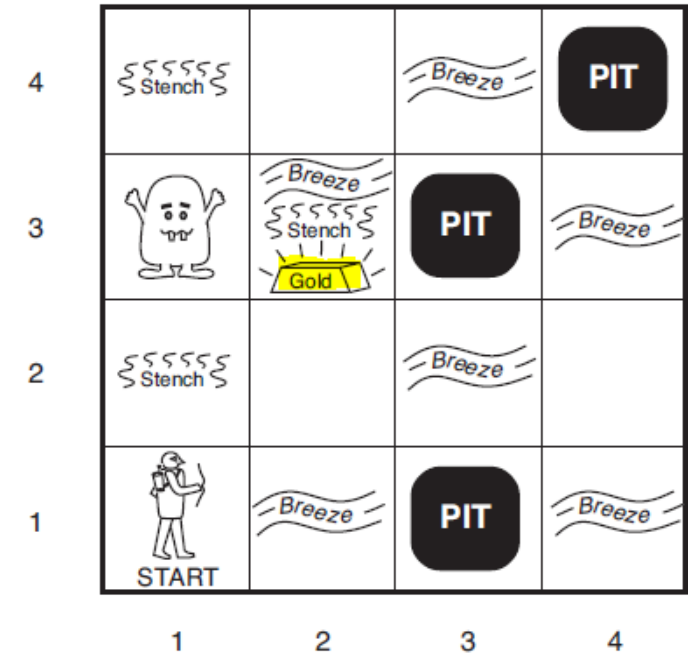
Online Simulator:

<https://thiagodnf.github.io/wumpus-world-simulator/>



# The Wumpus World- Description

The **wumpus world** is a cave consisting of rooms connected by passageways. Lurking somewhere in the cave is the terrible wumpus, 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 (except for the wumpus, which is too big to fall in). The only mitigating feature of this bleak environment is the possibility of finding a heap of gold.

# The Wumpus World- PEAS Description

## ❖ Performance measure

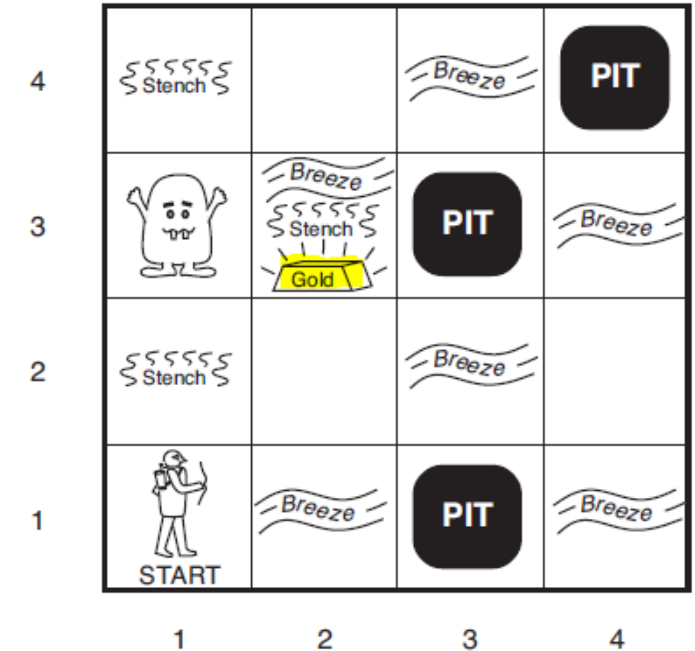
- gold +1000, death -1000
- (-1) per step, (-10) for using the arrow

## ❖ Environment

- squares adjacent to wumpus are smelly
- squares adjacent to 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

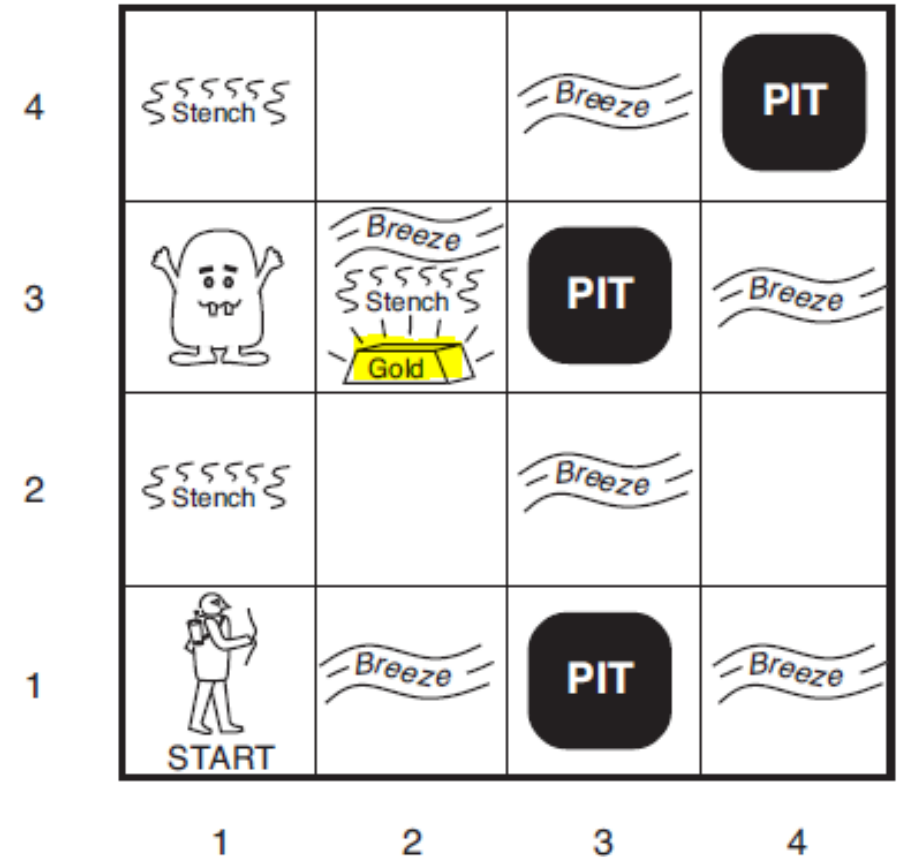
## ❖ Actuators Left turn, Right turn, Forward, Grab, Release, Shoot

## ❖ Sensors Breeze, Glitter, Smell/Stench, Scream, Bump



# The Wumpus World- Characterization

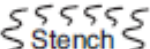
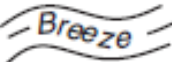


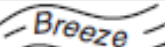
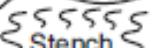
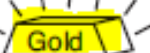

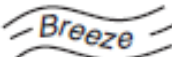
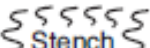
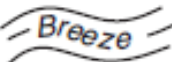

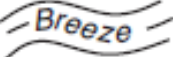

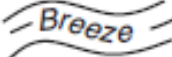
- ❖ Observable? **No** — only local perception
- ❖ Deterministic? **Yes** — outcomes exactly specified
- ❖ Episodic? **No** — sequential at the level of actions
- ❖ Static? **Yes** — Wumpus and Pits do not move
- ❖ Discrete? **Yes**
- ❖ Single-agent? **Yes** — Wumpus is essentially a natural feature



# The Wumpus World- Exploring

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2	2,2	3,2	4,2
OK			
1,1	2,1	3,1	4,1
<b>A</b> OK	OK		

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

 Stench		 Breeze	 PIT
	 Breeze  Stench  Gold	 PIT	 Breeze
 Stench		 Breeze	
 START	 Breeze	 PIT	 Breeze
1	2	3	4

# The Wumpus World- Exploring

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2 P?	3,2	4,2
1,1 V OK	2,1 <b>A</b> B OK	3,1 P?	4,1

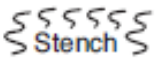
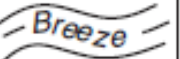


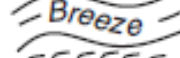
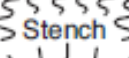


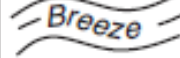
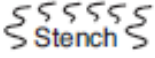
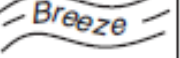

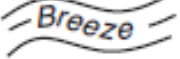

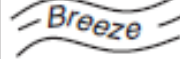
**A** = Agent  
**B** = Breeze  
**G** = Glitter, Gold  
**OK** = Safe square  
**P** = Pit  
**S** = Stench  
**V** = Visited  
**W** = Wumpus

4

3

2

1

 Stench		 Breeze	 PIT
	   Gold	 PIT	 Breeze
 Stench		 Breeze	
 START	 Breeze	 PIT	 Breeze
1	2	3	4

# The Wumpus World- Exploring

1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 <b>A</b> S OK	2,2  OK	3,2	4,2
1,1  V OK	2,1 B  V OK	3,1 P!	4,1

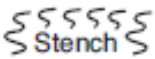
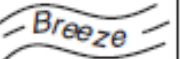


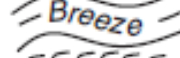
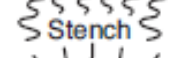


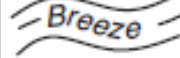
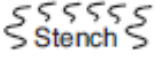
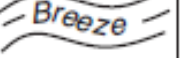

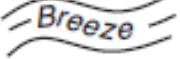

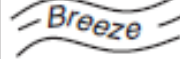
**A** = Agent  
**B** = Breeze  
**G** = Glitter, Gold  
**OK** = Safe square  
**P** = Pit  
**S** = Stench  
**V** = Visited  
**W** = Wumpus

4

3

2

1

 Stench		 Breeze	
	  		 Breeze
 Stench		 Breeze	
 START	 Breeze		 Breeze
1	2	3	4

# The Wumpus World- Exploring

1,4	2,4 P?	3,4	4,4
1,3 W!	2,3 <b>A</b> S G B	3,3 P?	4,3
1,2 S V OK	2,2 V OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

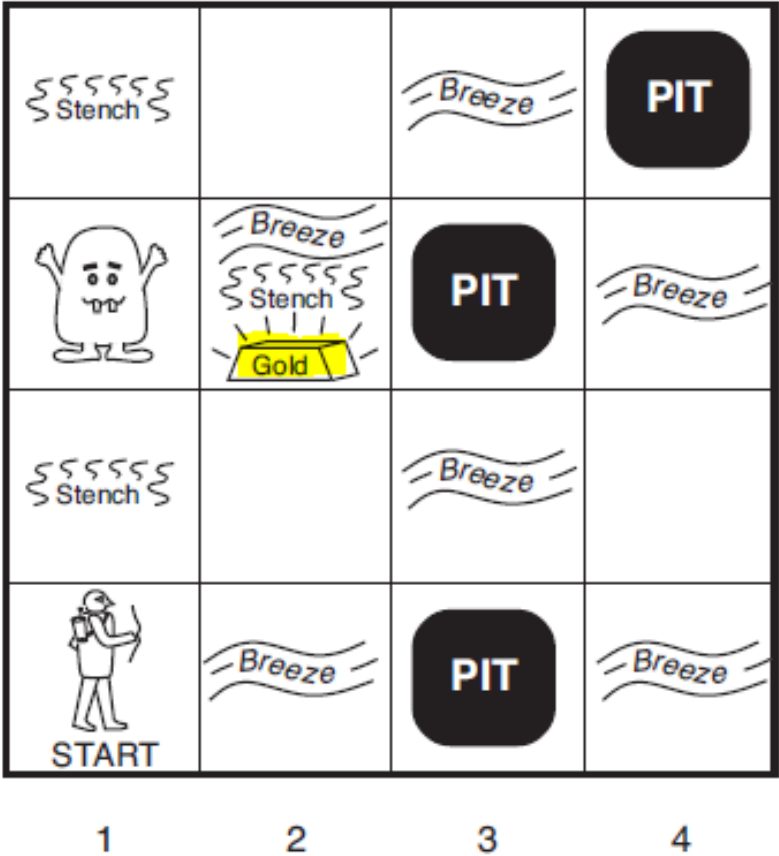
- A** = Agent
- B = Breeze
- G = Glitter, Gold
- OK = Safe square
- P = Pit
- S = Stench
- V = Visited
- W = Wumpus

4

3

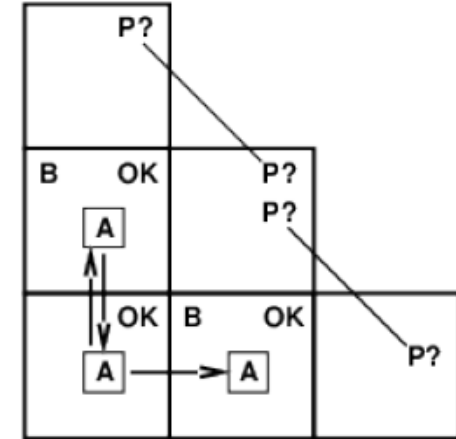
2

1

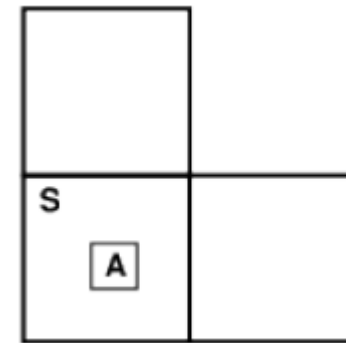


# The Wumpus World- Critical Spot

- Breeze in (1,2) and (2,1) → no safe actions  
Assuming pits uniformly distributed, (2,2) has pit  
w/ prob 0.86, vs. 0.31



- Smell in (1,1) → cannot move  
Can use a strategy of coercion: shoot straight ahead
- wumpus was there => dead => safe
  - wumpus wasn't there => safe





# Logic – Knowledge Representation Language

Logic is the primary concern for representing and reasoning about knowledge.

- ❖ Propositional Logic
- ❖ First Order Logic
- ❖ Second Order Logic
- ❖ Horn Clause Logic
- ❖ Fuzzy Logic

# Logic

**Logic** is a formal system for manipulating facts so that true conclusions may be drawn – “The tool for distinguishing between the true and the false” (Averroes)

- ❖ **Syntax:** rules for constructing valid sentences
  - E.g.,  $x + 2 \geq y$  is a valid arithmetic sentence,  $\geq x 2 y +$  is not
- ❖ **Semantics:** “meaning” of sentences, or relationship between logical sentences and the real world
  - Specifically, semantics defines truth of sentences
  - E.g.,  $x + 2 \geq y$  is true in a world where  $x = 5$  and  $y = 7$
- ❖ **Syntactic Inference Method:** refers to a mechanism for deriving new (true) sentences from existing sentences.

# Propositional Logic - Syntax

Atomic sentence	– A proposition symbol representing a true or false statement
Negation	– If $P$ is a sentence, $\neg P$ is a sentence
Conjunction	– If $P$ and $Q$ are sentences, $P \wedge Q$ is a sentence
Disjunction	– If $P$ and $Q$ are sentences, $P \vee Q$ is a sentence
Implication	– If $P$ and $Q$ are sentences, $P \Rightarrow Q$ is a sentence
Biconditional	– If $P$ and $Q$ are sentences, $P \Leftrightarrow Q$ is a sentence

$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$  are called *logical connectives*

# Propositional Logic - Syntax

How to define a sentence in propositional Logic?

1. Any statement is a sentence and it must be represented by a symbol. For example,  $P$ ,  $Q$ ,  $R$
2. If  $P$  and  $Q$  are two sentences, then  $\neg P$ ,  $P \wedge Q$ ,  $P \vee Q$ ,  $P \Rightarrow Q$ ,  $P \Leftrightarrow Q$  are sentences.
3. A finite number of application of (1) and (2) is a sentence.

# Propositional Logic - Syntax

## Grammar:

$$\begin{aligned} \textit{Sentence} &\rightarrow \textit{AtomicSentence} \mid \textit{ComplexSentence} \\ \textit{AtomicSentence} &\rightarrow \textit{True} \mid \textit{False} \mid P \mid Q \mid R \mid \dots \\ \textit{ComplexSentence} &\rightarrow (\textit{Sentence}) \mid [\textit{Sentence}] \\ &\mid \neg \textit{Sentence} \\ &\mid \textit{Sentence} \wedge \textit{Sentence} \\ &\mid \textit{Sentence} \vee \textit{Sentence} \\ &\mid \textit{Sentence} \Rightarrow \textit{Sentence} \\ &\mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \end{aligned}$$

OPERATOR PRECEDENCE :  $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$

Precedence from highest to lowest.

# Propositional Logic - Semantics

A **model** specifies the true/false status of each proposition symbol in the knowledge base

-E.g.,  $P$  is true,  $Q$  is true,  $R$  is false

-With three symbols, there are 8 possible models, and they can be enumerated exhaustively

Rules for evaluating truth with respect to a model:

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

# Propositional Logic - Semantics

A **truth table** specifies the truth value of a composite sentence for each possible assignments of truth values to its atoms.

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

# Propositional Logic - Example

## Sentences:

1. The car is either at John's house or at Fred's house.
2. If the car is not at John's house, then it must be at Fred's house



# Propositional Logic - Example

## Sentences:

1. The car is either at John's house or at Fred's house.
2. If the car is not at John's house, then it must be at Fred's house

## Atomic Sentences:

**X**: car is at John's house

**Y**: car is at Fred's house

# Propositional Logic - Example

## Sentences:

1. The car is either at John's house or at Fred's house.
2. If the car is not at John's house, then it must be at Fred's house

## Atomic Sentences:

**X**: car is at John's house

**Y**: car is at Fred's house

## Propositional Sentence:

1.  $(X \wedge \neg Y) \vee (\neg X \wedge Y)$
2.  $\neg X \Rightarrow Y$

# Propositional Logic - Example

## Try yourself:

1. If it is hot, then it is humid.
2. If if it is hot and humid, then it is raining.

# Propositional Logic – Wumpus World Sentences

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

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

# Propositional Logic – Wumpus World Sentences

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

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

- A square is breezy if and only if there is a pit in a neighboring square. This has to be stated for each square; for now, we include just the relevant squares:

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

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

- The preceding sentences are true in all wumpus worlds. Now we include the breeze percepts for the first two squares visited in the specific world the agent is in, leading up to the situation in Figure 7.3(b).

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

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

# Propositional Logic - Entailment

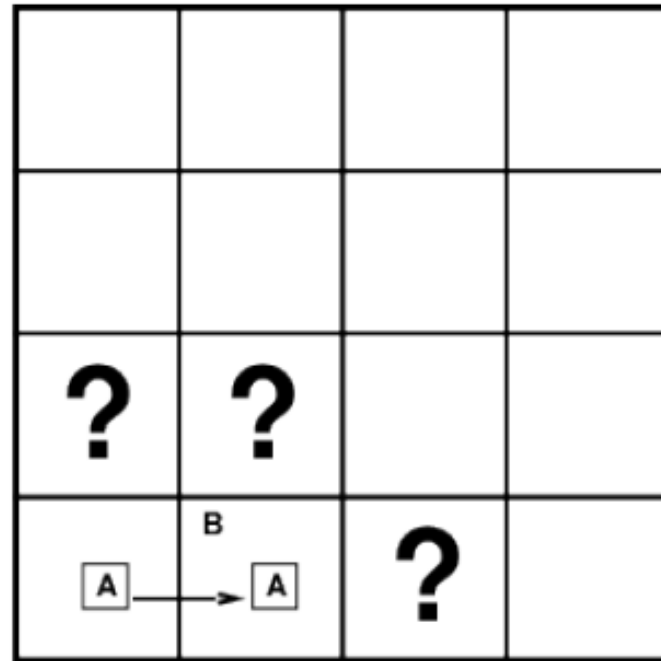
- **Entailment** means that a sentence follows from the premises contained in the knowledge base:

$$KB \models \alpha$$

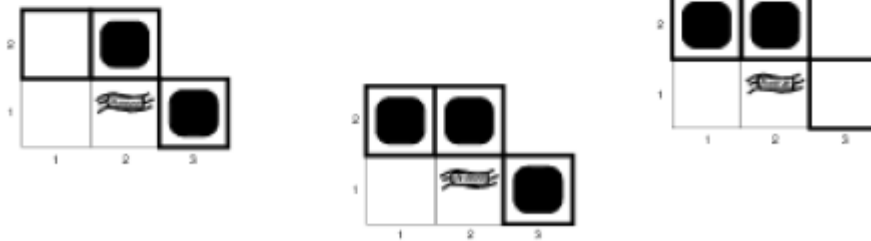
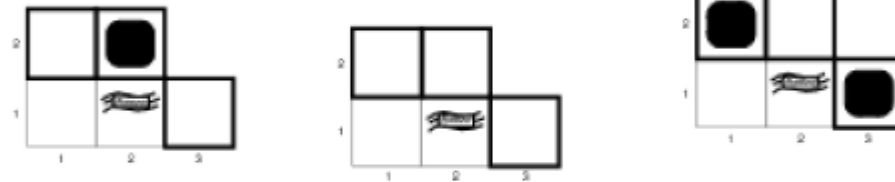
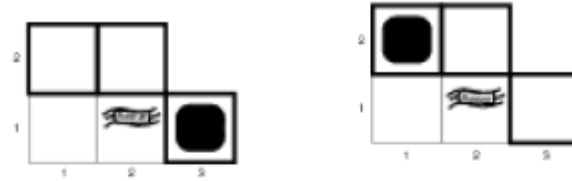
- Knowledge base  $KB$  entails sentence  $\alpha$  if and only if  $\alpha$  is true in all models where  $KB$  is true.
  - E.g.,  $x + y = 4$  entails  $4 = x + y$
- Entailment is a relationship between sentences (i.e., syntax) that is based on semantics

# Entailment in the Wumpus World

- ✓ Situation after detecting nothing in [1,1], moving right, breeze in [2,1]
- ✓ Consider possible models for all ?, assuming only pits
- ✓ 3 Boolean choices => 8 possible models

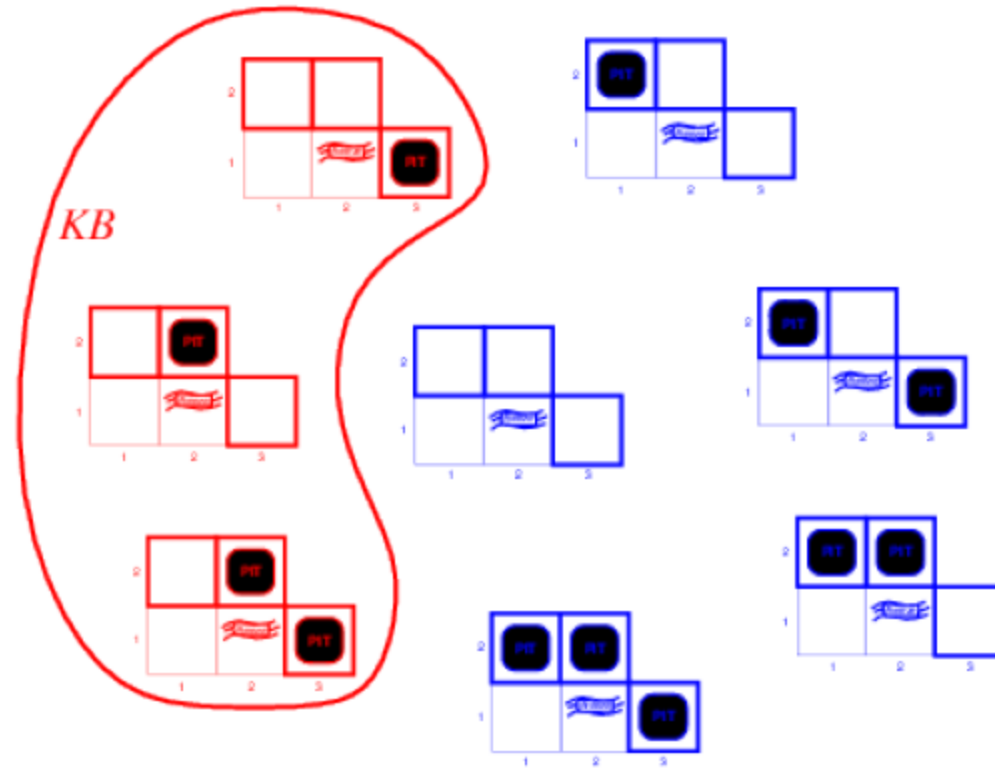


# Possible Wumpus World



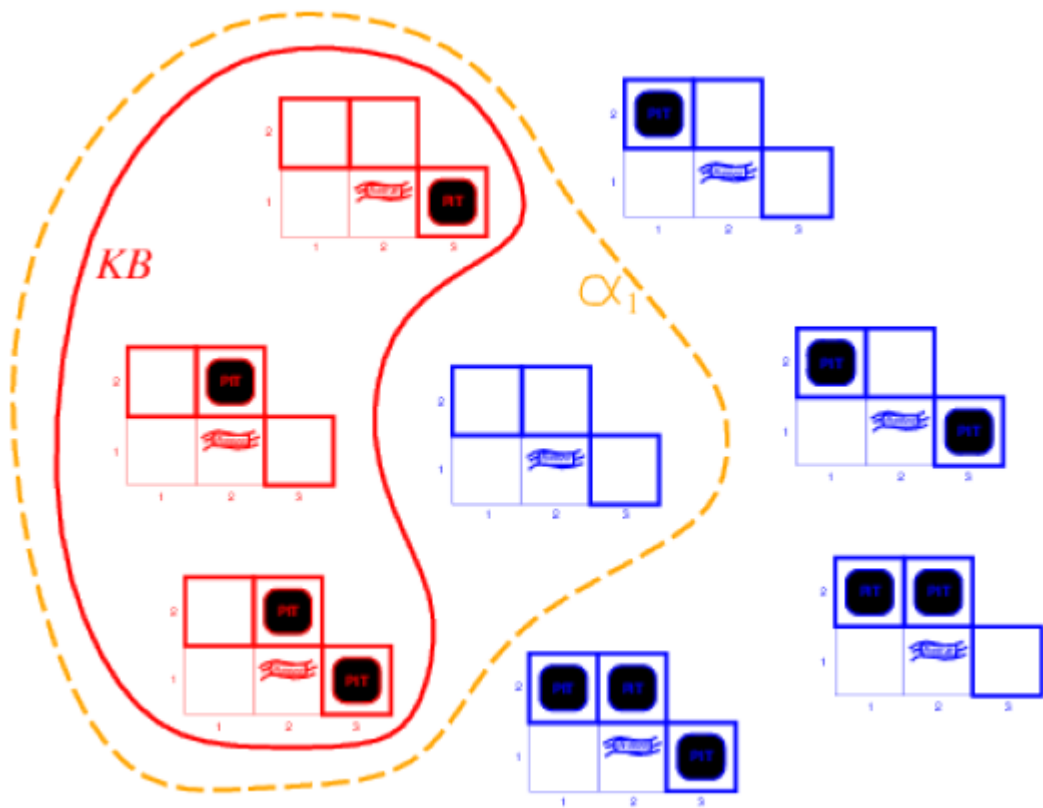


# Valid Wumpus World

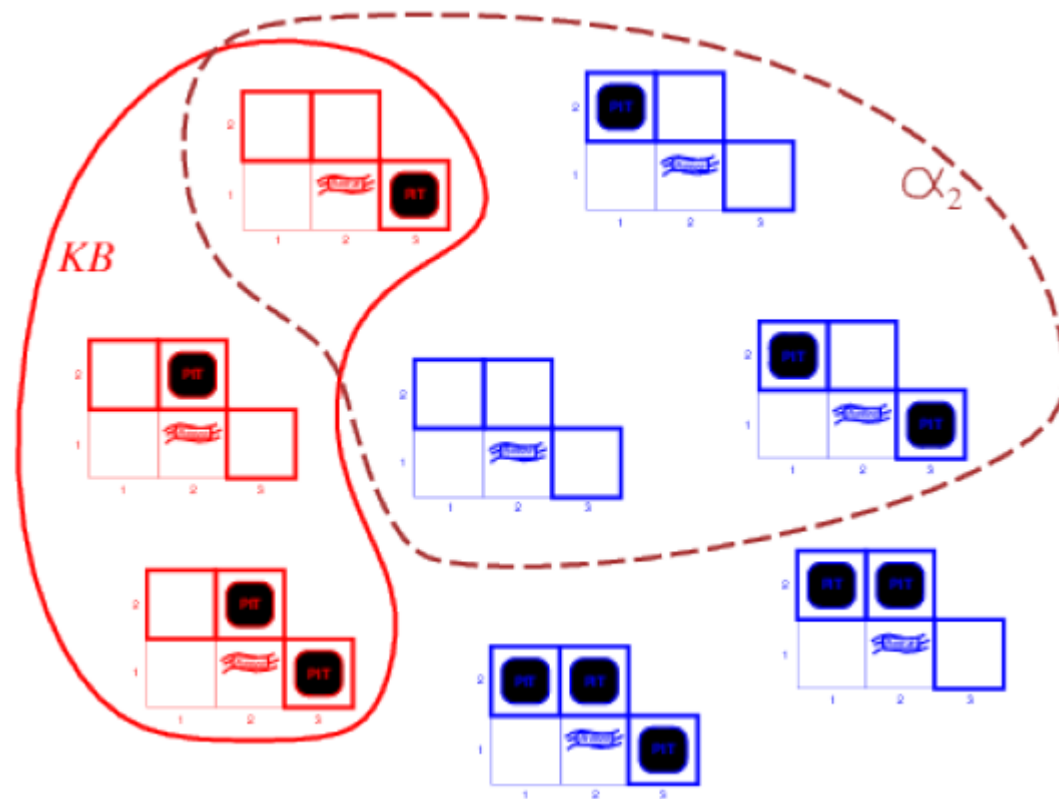


# Entailed?

$$KB \models \alpha_1$$



$$KB \not\models \alpha_2$$



# Propositional Theorem Proving

# Propositional Logic - Inference

**Logical inference:** a procedure for generating sentences that follow from a knowledge base KB

## □ Soundness and Completeness of Inference:

- An inference procedure is **sound** if whenever it derives a sentence  $\alpha$ ,  $KB \models \alpha$ 
  - A sound inference procedure can derive only true sentences
- An inference procedure is **complete** if whenever  $KB \models \alpha$ ,  $\alpha$  can be derived by the procedure
  - A complete inference procedure can derive every entailed sentence

**A simple inference procedure** - model-checking

# Inference

- How can we check whether a sentence  $\alpha$  is entailed by KB?  
How about we enumerate all possible models of the KB (truth assignments of all its symbols), and check that  $\alpha$  is true in every model in which KB is true?
  - Is this sound?
  - Is this complete?
- Problem: if KB contains  $n$  symbols, the truth table will be of size  $2^n$
- Better idea: use *inference rules*, or sound procedures to generate new sentences or *conclusions* given the *premises* in the KB

# Propositional Logic – Inference Rules

Name	Premises	Conclusion
Modus Ponens	$A, A \Rightarrow B$	$B$
AND Introduction	$A, B$	$A \wedge B$
AND Elimination	$A \wedge B$	$A, B$
Double negation	$\neg(\neg A)$	$A$
Unit Resolution	$A \vee B, \neg B$	$A$
Resolution	$A \vee B, \neg B \vee C$	$A \vee C$

# Logical equivalence

- Two sentences are **logically equivalent** iff true in same models:

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

$(\alpha \wedge \beta)$	$\equiv$	$(\beta \wedge \alpha)$	commutativity of $\wedge$
$(\alpha \vee \beta)$	$\equiv$	$(\beta \vee \alpha)$	commutativity of $\vee$
$((\alpha \wedge \beta) \wedge \gamma)$	$\equiv$	$(\alpha \wedge (\beta \wedge \gamma))$	associativity of $\wedge$
$((\alpha \vee \beta) \vee \gamma)$	$\equiv$	$(\alpha \vee (\beta \vee \gamma))$	associativity of $\vee$
$\neg(\neg\alpha)$	$\equiv$	$\alpha$	double-negation elimination
$(\alpha \implies \beta)$	$\equiv$	$(\neg\beta \implies \neg\alpha)$	contraposition
$(\alpha \implies \beta)$	$\equiv$	$(\neg\alpha \vee \beta)$	implication elimination
$(\alpha \iff \beta)$	$\equiv$	$((\alpha \implies \beta) \wedge (\beta \implies \alpha))$	biconditional elimination
$\neg(\alpha \wedge \beta)$	$\equiv$	$(\neg\alpha \vee \neg\beta)$	De Morgan
$\neg(\alpha \vee \beta)$	$\equiv$	$(\neg\alpha \wedge \neg\beta)$	De Morgan
$(\alpha \wedge (\beta \vee \gamma))$	$\equiv$	$((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	distributivity of $\wedge$ over $\vee$
$(\alpha \vee (\beta \wedge \gamma))$	$\equiv$	$((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	distributivity of $\vee$ over $\wedge$

# Validity and satisfiability

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



# Example- Inference (1)

If it is hot, then it is humid. If it is hot and humid, then it is raining. It is hot today.

Prove that, “It is raining today.” – using inferencing rules.

# Example- Inference (1)

If it is hot, then it is humid. If it is hot and humid, then it is raining. It is hot today.

Prove that, “It is raining today.” – using inferencing rules.

## Atomic Sentences:

- ✓ It is hot  $\equiv H$
- ✓ It is humid  $\equiv T$
- ✓ It is raining  $\equiv R$

# Example- Inference (1)

## Steps of Inference:

1.  $H$  ; premise
2.  $H \Rightarrow T$  ; premise
3.  $H \wedge T \Rightarrow R$  ; premise
4.  $T$  ; modus ponens (1, 2)
5.  $H \wedge T$  ; AND introduction (1, 4)
6.  $R$  ; Modus Ponens (3, 5)
7. True  
[Proved]

Name	Premises	Conclusion
Modus Ponens	$A, A \Rightarrow B$	$B$
AND Introduction	$A, B$	$A \wedge B$
AND Elimination	$A \wedge B$	$A, B$
Double negation	$\neg(\neg A)$	$A$
Unit Resolution	$A \vee B, \neg B$	$A$
Resolution	$A \vee B, \neg B \vee C$	$A \vee C$

# Example- Inference (2): Wumpus World

Given,

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

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

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

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

$$R5 : B_{2,1}$$

Prove, there is no pit in  $[1,2] \equiv \neg P_{1,2}$

# Example- Inference (2) by enumerating

## Wumpus World

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	$R_1$	$R_2$	$R_3$	$R_4$	$R_5$	$KB$
<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>false</i>
<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<u><i>true</i></u>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<u><i>true</i></u>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<u><i>true</i></u>
<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>false</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>true</i>	<i>false</i>	<i>true</i>	<i>false</i>

# Example- Inference (2) by inference rules

## Steps of Inference:

1.  $\neg P_{1,1}$
2.  $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
3.  $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
4.  $\neg B_{1,1}$
5.  $B_{2,1}$
6.  $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
7.  $(P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$
8.  $(\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1}))$
9.  $\neg(P_{1,2} \vee P_{2,1})$
10.  $\neg P_{1,2} \wedge \neg P_{2,1}$
11.  $\neg P_{1,2}$

**12. True**

**[Proved]**

Name	Premises	Conclusion
Modus Ponens	$A, A \Rightarrow B$	$B$
AND Introduction	$A, B$	$A \wedge B$
AND Elimination	$A \wedge B$	$A, B$
Double negation	$\neg(\neg A)$	$A$
Unit Resolution	$A \vee B, \neg B$	$A$
Resolution	$A \vee B, \neg B \vee C$	$A \vee C$

; premise

; premise

; premise

; premise

; premise

; bi-conditional elimination 2

; and-elimination 6

; logical contrapositive of 7

; modus ponens (4, 8)

; De-Morgan on 9

# Proof by resolution (1)

Unit Resolution	$A \vee B, \neg B$	$A$
Resolution	$A \vee B, \neg B \vee C$	$A \vee C$

Assume for some propositional symbol  $P, Q$ . If  $P \Rightarrow Q$  and  $Q \Rightarrow R$  are true.  
Prove that,  $P \Rightarrow R$  using resolution.

# Proof by resolution (1)

Proof: proving by contradiction.

Let,  $P \Rightarrow R$  is not true

i.e.  $\neg(P \Rightarrow R)$  true

Or,  $\neg(\neg P \vee R)$  true

Or,  $P, \neg R$  true.

Unit Resolution	$A \vee B, \neg B$	$A$
Resolution	$A \vee B, \neg B \vee C$	$A \vee C$



# Proof by resolution (1)

Unit Resolution	$A \vee B, \neg B$	$A$
Resolution	$A \vee B, \neg B \vee C$	$A \vee C$

## Steps:

1.  $P \Rightarrow Q$  ; premise
2.  $Q \Rightarrow R$  ; premise
3.  $P$  ; assumed
4.  $\neg R$  ; assumed
5.  $\neg P \vee Q$  ; equivalence of 1
6.  $\neg Q \vee R$  ; equivalence of 2
7.  $Q$  ; unit resolution (3, 5)
8.  $R$  ; unit resolution (6, 7)
9. **False** ; contradicts (4, 8)

# Proof by resolution (1)

Unit Resolution	$A \vee B, \neg B$	$A$
Resolution	$A \vee B, \neg B \vee C$	$A \vee C$

## Steps:

1.  $P \Rightarrow Q$  ; premise
2.  $Q \Rightarrow R$  ; premise
3.  $P$  ; assumed
4.  $\neg R$  ; assumed
5.  $\neg P \vee Q$  ; equivalence of 1
6.  $\neg Q \vee R$  ; equivalence of 2
7.  $Q$  ; unit resolution (3, 5)
8.  $R$  ; unit resolution (6, 7)
9. **False** ; contradicts (4, 8)

## Steps:

1.  $P \Rightarrow Q$  ; premise
2.  $Q \Rightarrow R$  ; premise
3.  $\neg P \vee Q$  ; equivalence of 1
4.  $\neg Q \vee R$  ; equivalence of 2
5.  $\neg P \vee R$  ; unit resolution (3, 4)
6.  $P \Rightarrow R$  ; equivalence of 5

**7. True**

[Proved]

# Proof by resolution (2)

There is pit in  $[3, 1] \equiv P_{3,1}$

11.  $\neg B_{1,2}$

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

13.  $\neg P_{2,2}$

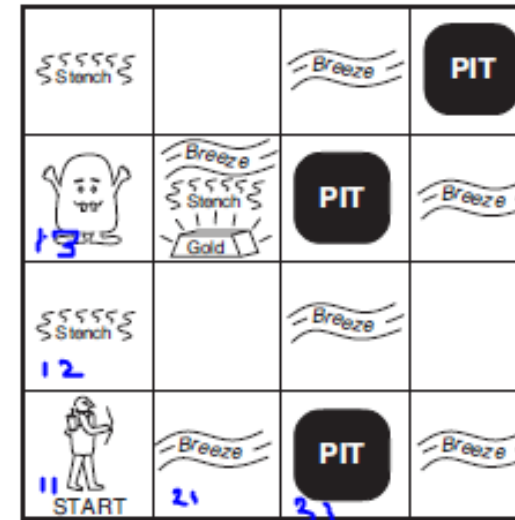
14.  $\neg P_{1,3}$

15.  $P_{1,1} \vee P_{2,2} \vee P_{3,1}$

16.  $P_{1,1} \vee P_{3,1}$

17.  $P_{3,1}$

Unit Resolution	$A \vee B, \neg B$	$A$
Resolution	$A \vee B, \neg B \vee C$	$A \vee C$



1.  $\neg P_{1,1}$
2.  $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
3.  $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
4.  $\neg B_{1,1}$
5.  $B_{2,1}$

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

; resolution (13, 15)

; resolution (1, 16)

# Conjunctive normal form

The resolution rule applies only to clauses (that is, disjunctions of literals), so it would seem to be relevant only to knowledge bases and queries consisting of clauses.

Procedure for converting to CNF:

1. Eliminate  $\Leftrightarrow$ , replacing  $\alpha \Leftrightarrow \beta$  with  $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$ .
2. Eliminate  $\Rightarrow$ , replacing  $\alpha \Rightarrow \beta$  with  $\neg\alpha \vee \beta$ :
3. CNF requires  $\neg$  to appear only in literals, “move  $\neg$  inwards”
4. Apply distributive law for distributing  $\vee$  over  $\wedge$  wherever possible

# Conjunctive normal form

Converting the sentence  $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$  into CNF.

## Steps:

1.  $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
2.  $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
3.  $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$
4.  $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1}) .$

# Home Task

$$[(A \Rightarrow B) \vee (C \Rightarrow B)] \Rightarrow [(A \wedge C) \Rightarrow B]$$

1. Using enumeration, find whether this sentence is valid, satisfiable (but not valid), or unsatisfiable.
2. Find CNF of the sentence,  $[(A \wedge C) \Rightarrow B]$
3. Find CNF of the sentence,  $[(A \Rightarrow B) \vee (C \Rightarrow B)]$

# Horn clauses and definite clauses – Restricted

## Definite clause:

- A disjunction of literals with **exactly one** is **positive literals**.
- For example, the clause  $(\neg L_{1,1} \vee \neg \text{Breeze} \vee B_{1,1})$  is a definite clause,
- whereas  $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1})$  is not.

## Horn clause:

- A disjunction of literals of with **at most one** is **positive literals**.
- clauses with no positive literals – **goal clauses**.

**Inference** with Horn clauses can be done through the **forward-chaining** and **backward-chaining** algorithms.

# Forward and backward chaining



**Figure 7.16** (a) A set of Horn clauses. (b) The corresponding AND–OR graph.



# Forward-chaining

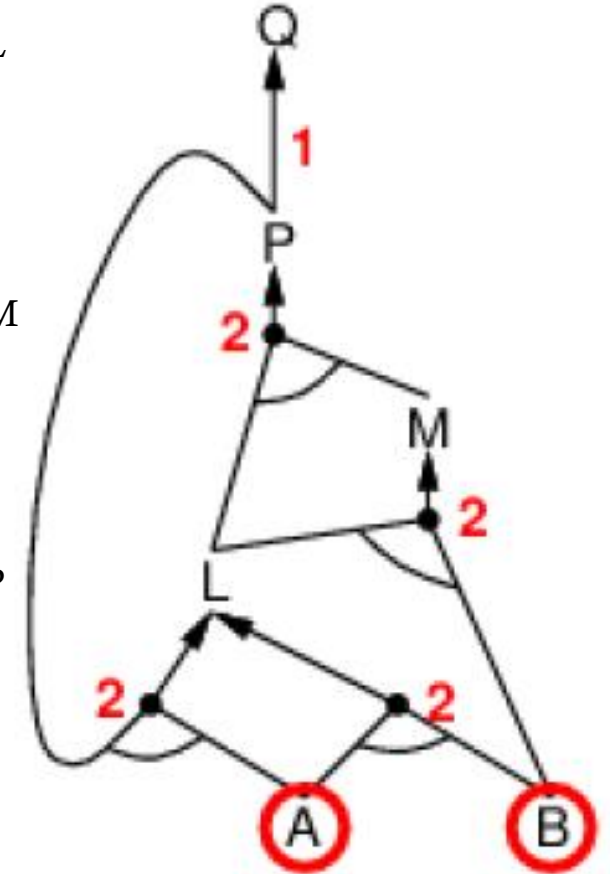
- ❖ Start with given proposition symbols (atomic sentence)
- ❖ Iteratively try to infer truth of additional proposition symbols
- ❖ **Continue** until
  - no more inference can be carried out, or
  - goal is reached

# Forward-chaining

1. Given,
  - $P \Rightarrow Q$
  - $L \wedge M \Rightarrow P$
  - $B \wedge L \Rightarrow M$
  - $A \wedge P \Rightarrow L$
  - $A \wedge B \Rightarrow L$
  - A
  - B
2. Agenda: A, B
3. Annotate horn clauses with number of premises
4. Process agenda item A
  - ✓ Decrease count for horn clauses in which A is premise
5. Process agenda item B
  - ✓ Decrease count for horn clauses in which B is premise
  - ✓  $A \wedge B \Rightarrow L$  has now fulfilled premise
  - ✓ Add L to agenda

7. Process agenda item L
  - ✓ Decrease count for horn clauses in which L is premise
  - ✓  $B \wedge L \Rightarrow M$  has now fulfilled premise
  - ✓ Add M to agenda
8. Process agenda item M
  - ✓ Decrease count for horn clauses in which M is premise
  - ✓  $L \wedge M \Rightarrow P$  has now fulfilled premise
  - ✓ Add P to agenda
9. Process agenda item P
  - ✓ Decrease count for horn clauses in which P is premise
  - ✓  $P \Rightarrow Q$  has now fulfilled premise
  - ✓ Add Q to agenda
  - ✓  $A \wedge P \Rightarrow L$  has now fulfilled premise
  - ✓ But L is already inferred
10. Process agenda item Q
11. Q is inferred [Done]

Agenda:



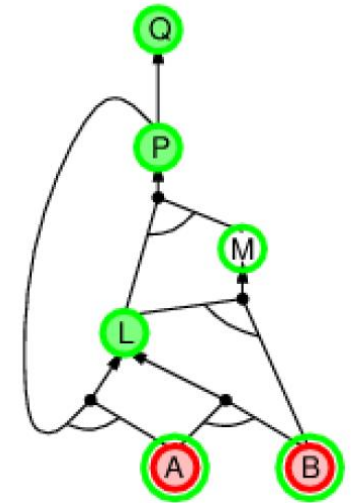
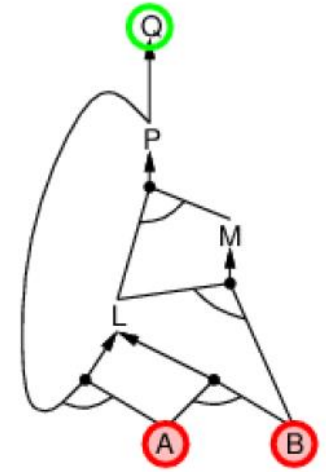
# Backward-chaining

- ❖ 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
  1. has already been proved true, or
  2. has already failed

# Backward-chaining

1. A and B are known to be true
2. Q needs to be proven
3. Current goal: Q
  - ✓ Q can be inferred by  $P \Rightarrow Q$
  - ✓ P needs to be proven
4. Current goal: P
  - ✓ P can be inferred by  $L \wedge M \Rightarrow P$
  - ✓ L and M need to be proven
5. Current goal: L
  - ✓ L can be inferred by  $A \wedge P \Rightarrow L$
  - ✓ A is already true
  - ✓ P is already a goal
  - ✓  $\Rightarrow$  repeated sub-goal

6. Current goal: L
  - ✓ L can be inferred by  $A \wedge B \Rightarrow L$
  - ✓ Both are true
  - ✓ L is true
7. Current goal: M
  - ✓ M can be inferred by  $B \wedge L \Rightarrow M$
  - ✓ Both are true
  - ✓ M is true
8. Current goal: P
  - ✓ P can be inferred by  $L \wedge M \Rightarrow P$
  - ✓ Both are true
  - ✓ P is true
9. Current goal: Q
  - ✓ Q can be inferred by  $P \Rightarrow Q$
  - ✓ P is true
  - ✓ Q is true



# Forward vs. Backward Chaining

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

# Propositional model checking algorithm

- ❑ Enumeration
- ❑ DPLL – A complete backtracking algorithm
- ❑ WalkSAT – Local Search Algorithm

# Propositional model checking algorithm: DPLL

**function** DPLL-SATISFIABLE?(*s*) **returns** *true* or *false*

**inputs:** *s*, a sentence in propositional logic

*clauses*  $\leftarrow$  the set of clauses in the CNF representation of *s*

*symbols*  $\leftarrow$  a list of the proposition symbols in *s*

**return** DPLL(*clauses*, *symbols*, { })

---

**function** DPLL(*clauses*, *symbols*, *model*) **returns** *true* or *false*

**if** every clause in *clauses* is true in *model* **then return** *true*

**if** some clause in *clauses* is false in *model* **then return** *false*

*P*, *value*  $\leftarrow$  FIND-PURE-SYMBOL(*symbols*, *clauses*, *model*)

**if** *P* is non-null **then return** DPLL(*clauses*, *symbols* – *P*, *model*  $\cup$  { *P*=*value* })

*P*, *value*  $\leftarrow$  FIND-UNIT-CLAUSE(*clauses*, *model*)

**if** *P* is non-null **then return** DPLL(*clauses*, *symbols* – *P*, *model*  $\cup$  { *P*=*value* })

*P*  $\leftarrow$  FIRST(*symbols*); *rest*  $\leftarrow$  REST(*symbols*)

**return** DPLL(*clauses*, *rest*, *model*  $\cup$  { *P*=*true* }) **or**

DPLL(*clauses*, *rest*, *model*  $\cup$  { *P*=*false* })

# Propositional model checking algorithm: WalkSAT

```
function WALKSAT(clauses, p, max_flips) returns a satisfying model or failure  
  inputs: clauses, a set of clauses in propositional logic  
           p, the probability of choosing to do a “random walk” move, typically around 0.5  
           max_flips, number of flips allowed before giving up  
  
  model  $\leftarrow$  a random assignment of true/false to the symbols in clauses  
  for i = 1 to max_flips do  
    if model satisfies clauses then return model  
    clause  $\leftarrow$  a randomly selected clause from clauses that is false in model  
    with probability p flip the value in model of a randomly selected symbol from clause  
    else flip whichever symbol in clause maximizes the number of satisfied clauses  
  return failure
```

**Figure 7.18** The WALKSAT algorithm for checking satisfiability by randomly flipping the values of variables. Many versions of the algorithm exist.



# Graded Assignment

# Assignment: Logical Puzzle

Use Propositional Logic to solve the following logical puzzle:

## Stem:

Four Friends - Artur, Betty, Charles and Dorothy - are suspected of murder. They testify as follows:

Arthur: if Betty is guilty, so is Dorothy.

Betty: Arthur is guilty, but Dorothy is not.

Charles: I am not guilty, but either Arthur or Dorothy is guilty.

Dorothy: if Arthur is not guilty, then Charles is guilty.

## Questions:

1. Are the four statements satisfiable? – use enumeration and DPLL
2. If everyone is telling the truth, who is guilty?
3. Is the statement- if Dorothy is not guilty then Betty is not guilty- a logical consequence of Artur's statement?

# 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 wrt **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
- ❖ Structure of Propositional Logic
- ❖ Inference rules
- ❖ Conjunctive normal form for resolution
- ❖ Forward chaining and Backward chaining for Horn clauses
- ❖ Efficient model-checking algorithms: DPLL and WalkSAT

Thank you 😊