Logical Agents

Up until now:

- Problem Solving Agents
  - Search, search, search,…
  - Game Playing
  - CSP

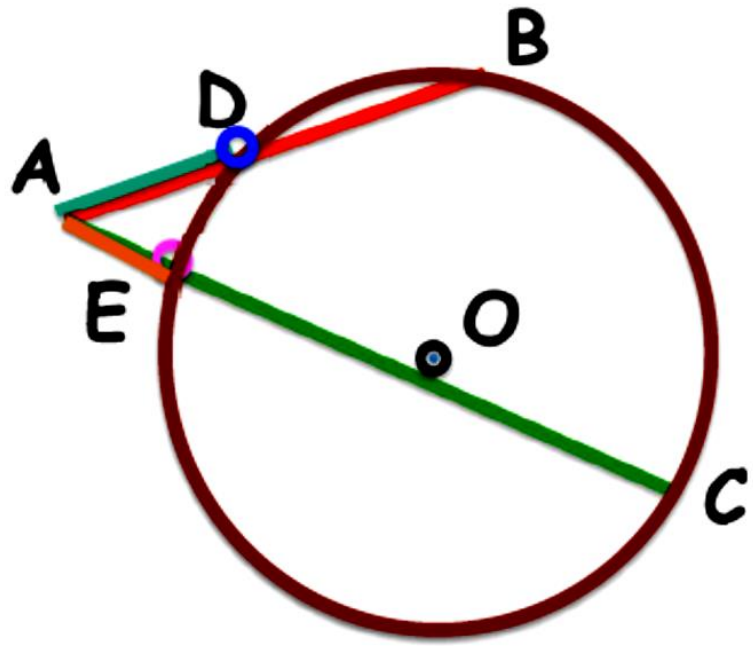
- Problem Solving Agents have very limited knowledge
  - States, Actions, Effect of Actions, Goal
  - e.g. Chess programs don't know that no piece can be on two different squares at the same time

Knowledge-Based Logical Agents

- Knowledge-based agents use general knowledge
- Knowledge-based agents combine general knowledge about the world with current precepts to *infer* new (hidden) aspects of their state
  - Crucial in partially observable environments

Solving Geometry Questions

In the diagram, secant AB intersects circle O at D, secant AC intersects circle O at E, AE = 4, AC = 24, and AB = 16. Find AD.



Knowledge about geometry axioms and theorems

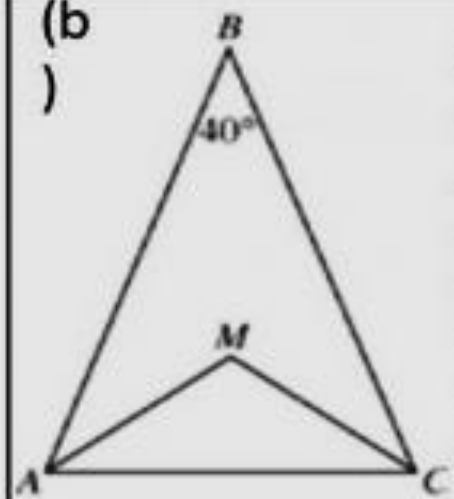
Speaking of Science

AI can now muddle its way through the math SAT about as well as you can

A computer system that can solve SAT geometry questions

A combination of computer vision, natural language and a geometric solver is used by GeoS

By **Anjali Singh Deswal** - 2015-09-22

	perpendicular to chord BD. What is the length of BD?	*Equals(what, Length(BD))* correct a) 12 b) 10 c) 8 d) 6 e) 4
(b) 	In isosceles triangle ABC at the left, lines AM and CM are the angle bisectors of angles BAC and BCA. What is the measure of angle AMC?	*IsIsoscelesTriangle(ABC)* *BisectsAngle(AM, BAC)* *IsLine(AM)* *CC(AM, CM)* *CC(BAC, BCA)* *IsAngle(BAC)* *IsAngle(AMC)* *Equals(what, MeasureOf(AMC))* correct a) 110 b) 115 c) 120 d) 125 e) 130
(c)	In the figure at left, The bisector of	*IsAngle(BAC)* *BisectsAngle(line, BAC)*

Knowledge-Based Agents

A simple model for reasoning:

- Agent is told or perceives new evidence -
  E.g., A is true

- Agent then infers new facts to add to the KB -

  E.g., KB = { A -> (B OR C) }, then given A and not C we can infer that B is true

  B is now added to the KB even though it was not explicitly asserted, i.e., the agent inferred B

KR Hypothesis

Any ***intelligent process*** will have ingredients that

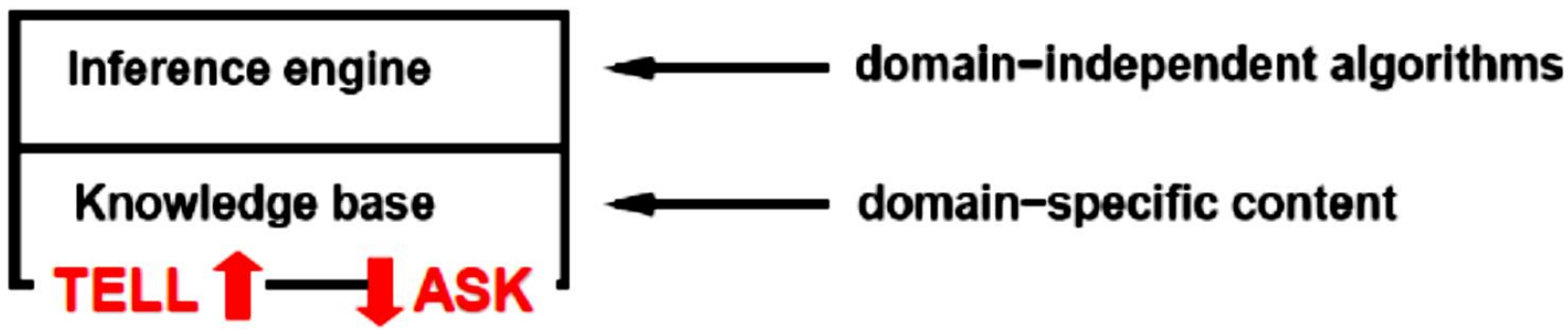
1) We as external observers interpret as knowledge
2) This knowledge plays a formal, causal & essential role in guiding the behavior

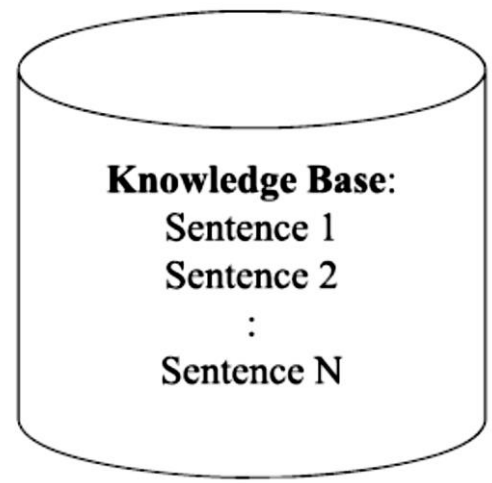
Knowledge Base and Inference

Knowledge Base : set of sentences represented in a knowledge representation language

- stores assertions about the world

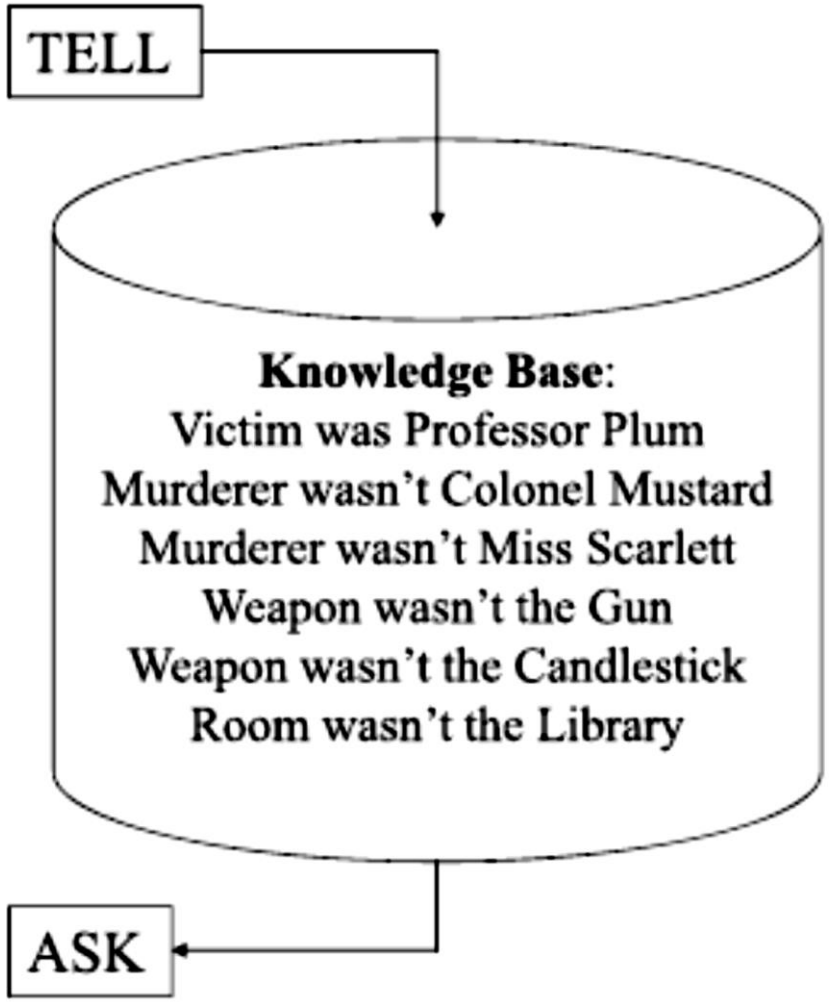
Inference: when you ASK the KB a question, answer should *follow* from what has been TELLed to the KB previously





KB Example

The CLUE Game



When you discover a new fact like "The murder room wasn't the study", you would TELL the KB

You can then ASK the KB what to ask next

Inference

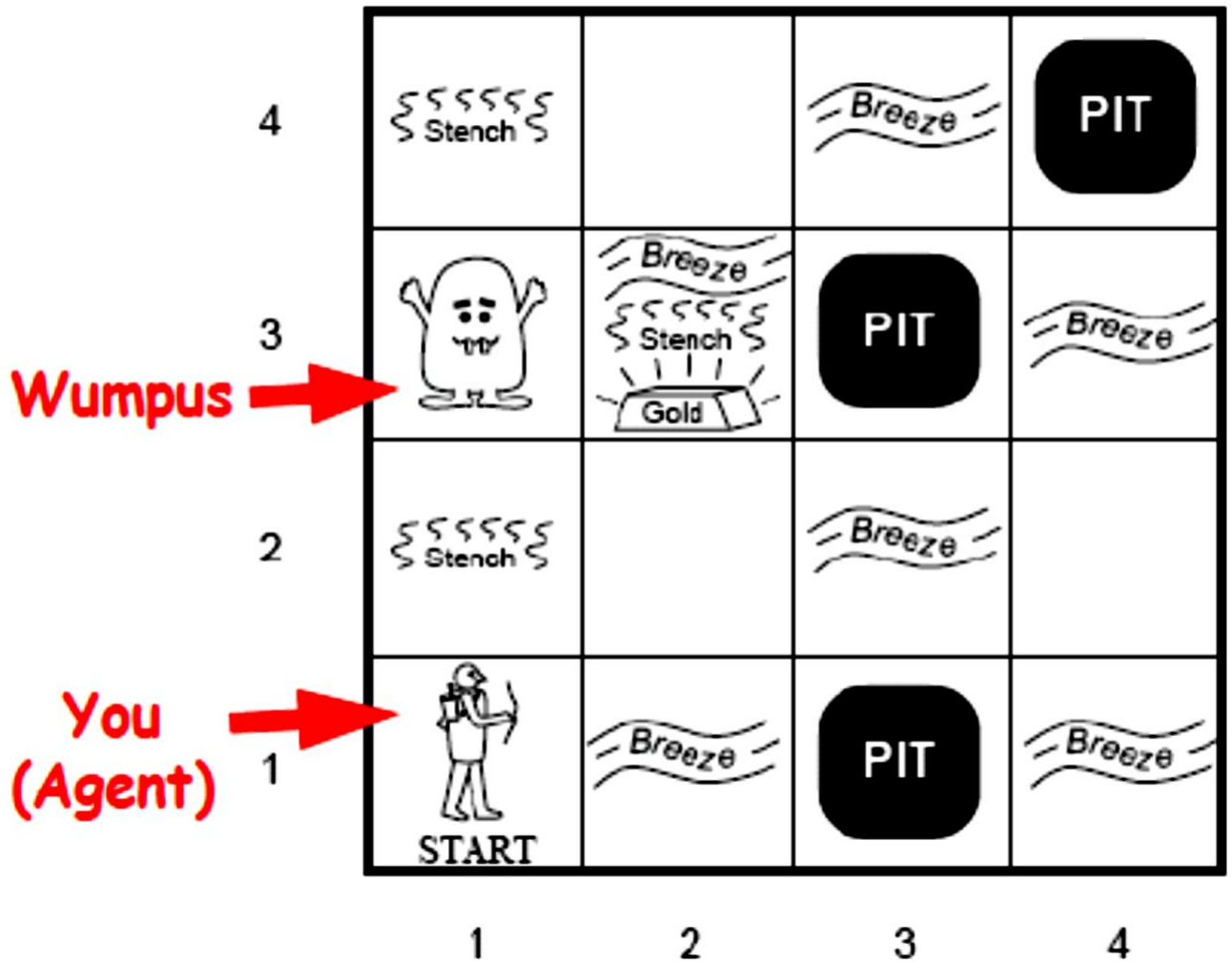
- Inference: deriving new sentences from old ones
- Must obey fundamental requirement: when one ASKs a question of the knowledge base, answer should follow from what has been TELLed to the KB previously

Abilities of a KB agent

Agent must be able to:

- Represent states and actions
- Incorporate new percepts
- Update internal representation of the world
- Deduce hidden properties of the world
- Deduce appropriate actions

A Typical Wumpus World

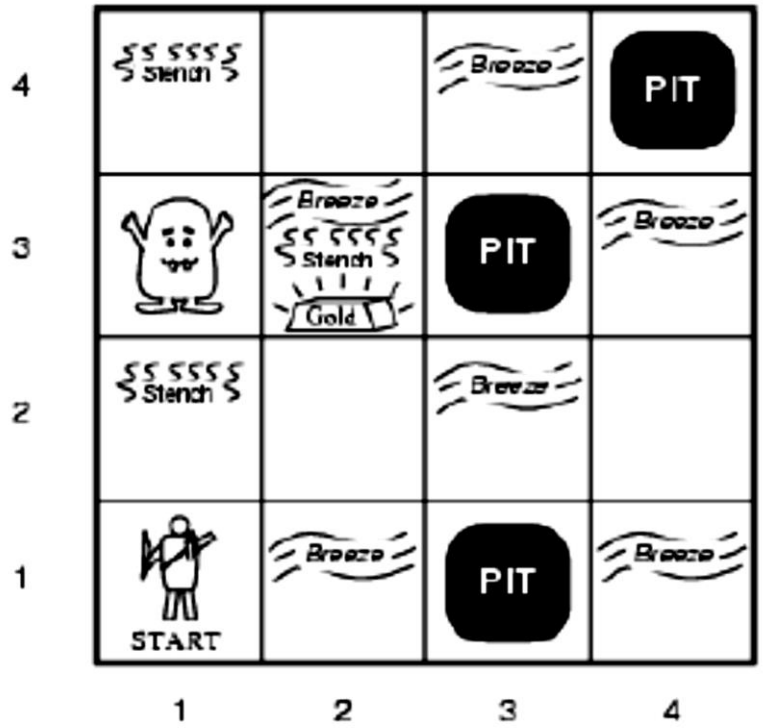


Wumpus World

Environment

Rules

- 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
- Climbing in [1,1] gets agent out of the cave



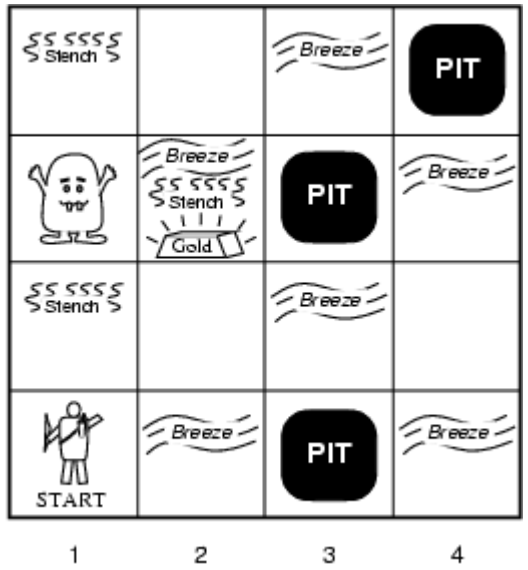
Sensors Stench, Breeze, Glitter, Bump, Scream

Actuators TurnLeft, TurnRight, Forward, Grab, Shoot, Climb

Wumpus World Characteristics

- The world is static: positions of the pits, gold, and monster do not change during the course of a game.
- The actions have deterministic effects.
- This domain is partially observable.
  - Agent does not know the map beforehand, must figure it out based on local perception.

Exploring the Wumpus World



1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2	3,2	4,2
1,1 A OK	2,1 OK	3,1	4,1

(a)

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

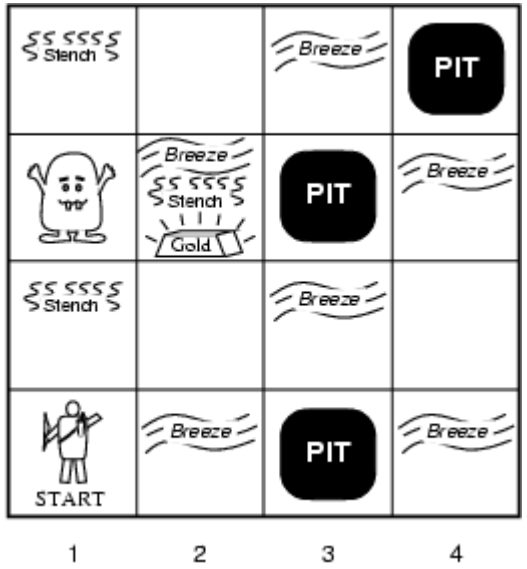
1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2 P?	3,2	4,2
1,1 V OK	2,1 A B OK	3,1 P?	4,1

(b)

[1,1] KB initially contains the rules of the environment.. First percept is [*none,none,none,none,none*], move to safe cell e.g. 2,1

[2,1] Breeze which indicates that there is a pit in [2,2] or [3,1], return to [1,1] to try next safe cell

Exploring the Wumpus World

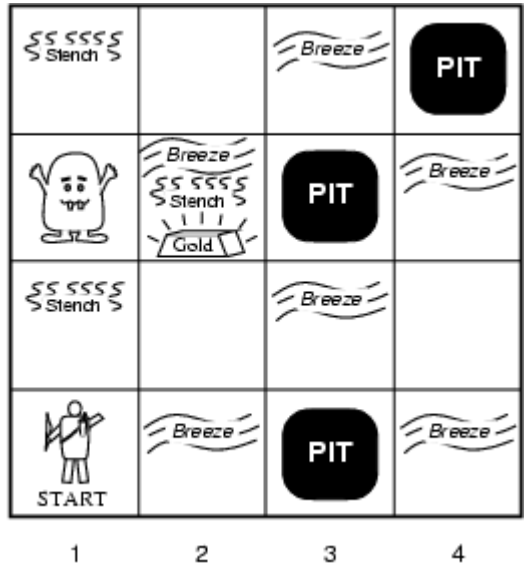


1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A S OK	2,2 OK	3,2	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

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

[1,2] Stench in cell which means that wumpus is in [1,3] or [2,2]
but not in [1,1]
YET … wumpus not in [2,2] or stench would have been
detected in [2,1]
THUS … wumpus must be in [1,3]
ALSO [2,2] is safe because of lack of breeze in [1,2]
THEREFORE pit must be in [3,1]
move to next safe cell [2,2]

Exploring the Wumpus World



1,4	2,4	3,4	4,4
1,3 W!	2,3	3,3	4,3
1,2 A 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

1,4	2,4 P?	3,4	4,4
1,3 W!	2,3 A 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

[2,2] Move to [2,3]
[2,3] Detect glitter, smell, breeze
Grab gold
Breeze implies pit in [3,3] or [2,4]

How do we represent rules of the world and percepts encountered ?

Need Knowledge Representation

Represent knowledge in a manner that facilitates inferencing (i.e. drawing conclusions) from knowledge

Logic for Knowledge Representation & Reasoning

Basic idea in Logic: By starting with true assumptions, you can deduce true conclusions

What is a logic?

A formal language

- **Syntax** – what expressions are legal (well-formed)
- **Semantics** – what legal expressions mean
  - In logic the truth of each sentence evaluated with respect to *each possible world*

E.g. the language of arithmetic

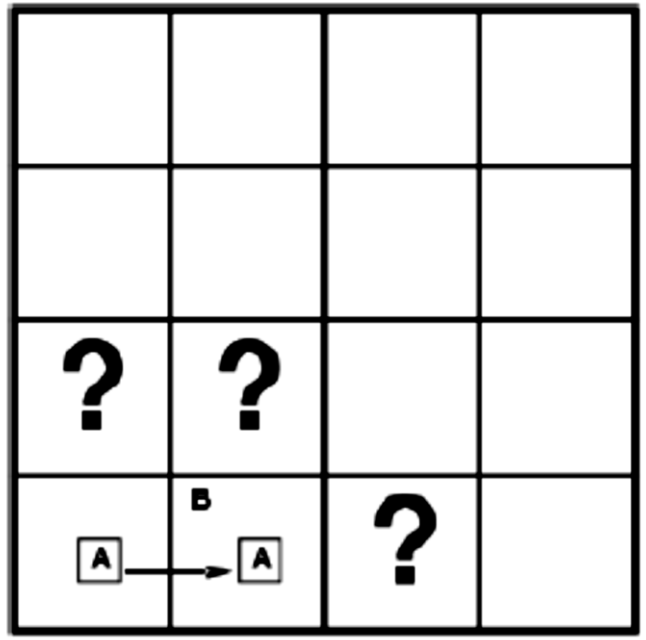
- Syntax: x+2 >= y is a sentence, x2y+= is not
- Semantics:
  - x+2 >= y is true in a world where x=7 and y=1
  - x+2 >= y is false in a world where x=0 and y=6

possible worlds:
{<x,y>, x,y ∈ integers }

How do we draw conclusions and deduce new facts about the world using logic?

Possible Worlds

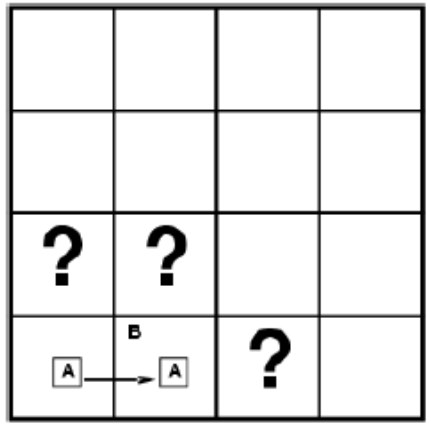
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]



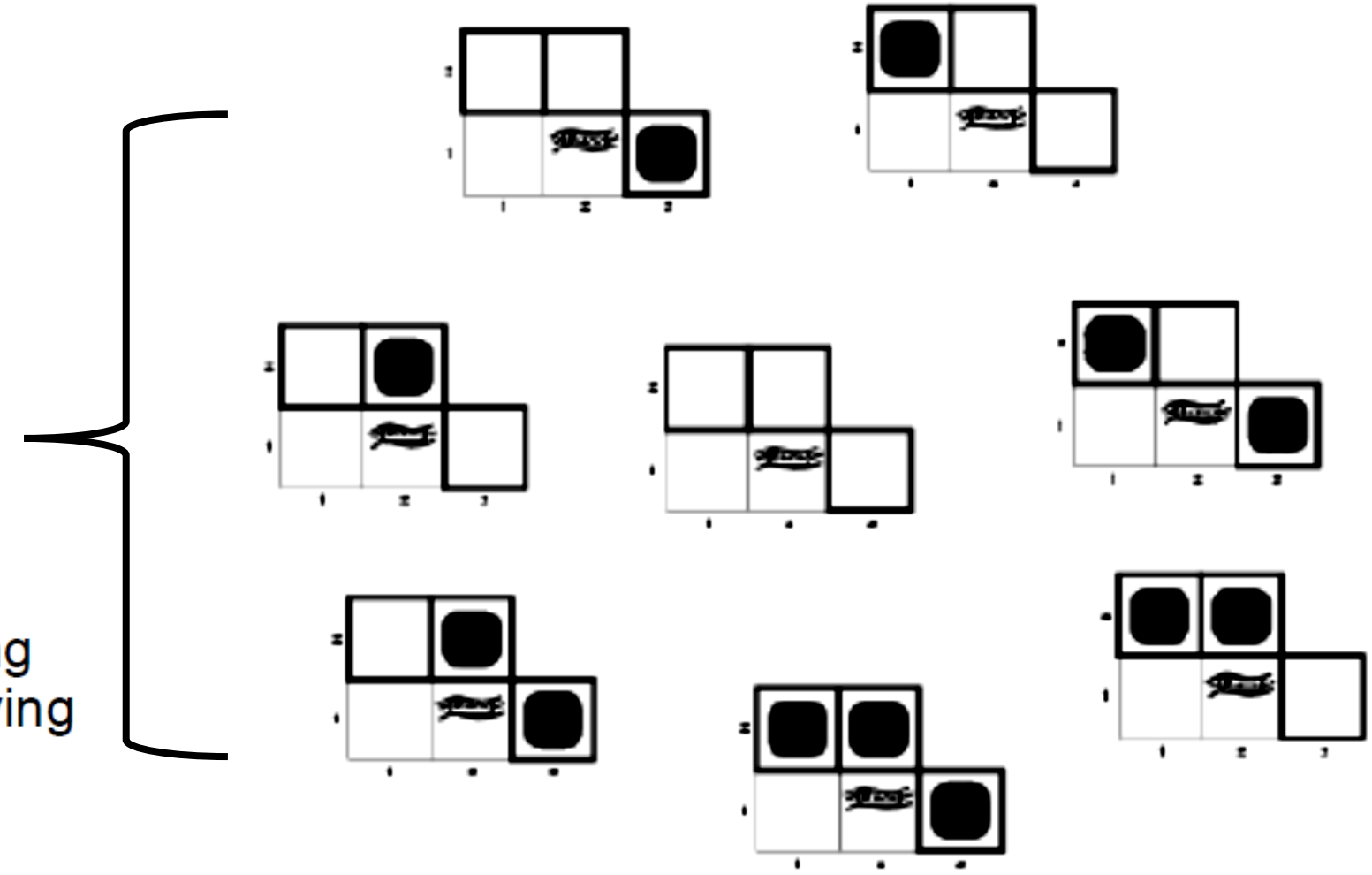
Consider Possible Worlds for KB assuming only Pits (only examine "fringe cells", namely, adjacent to cells explored)

3 Boolean choices $\Rightarrow$ 8 Possible Worlds

Possible Worlds for Wumpus Problem



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



Interpretations

Interpretations ↔ Possible Worlds

Entailment

- Entailment means that one thing follows from another:

$$KB \models \alpha$$

  Also known as Logical Consequence
- Knowledge base *KB* entails sentence α if and only if α is true in all worlds where *KB* is true

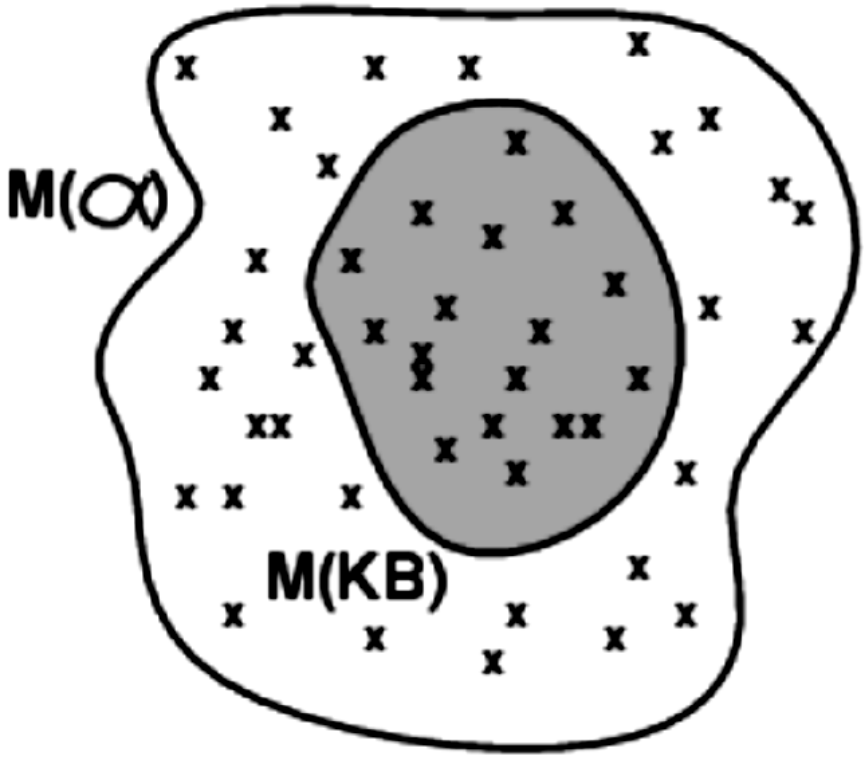
E.g. x>4 entails x>0

(because x>0 is true for all values of x for which x>4 is true)

But not vice versa! (x>0 does not entail x>4)

Models and Entailment

- We say Interpretation *m* is a model of a sentence α if α is true in *m*
- *M(α)* is the set of all models of α
- Then KB $\models$ α iff $M(KB) \subseteq M(\alpha)$
  - E.g. *KB* = Giants won and Reds won α = Giants won

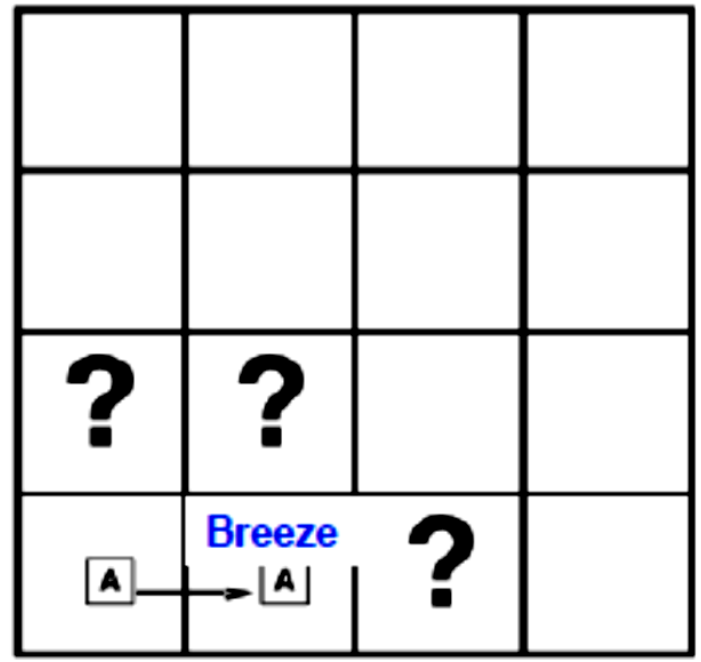


All models for KB are also models for α

E.g. *KB* = x>4
α = x>0

Wumpus world model

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

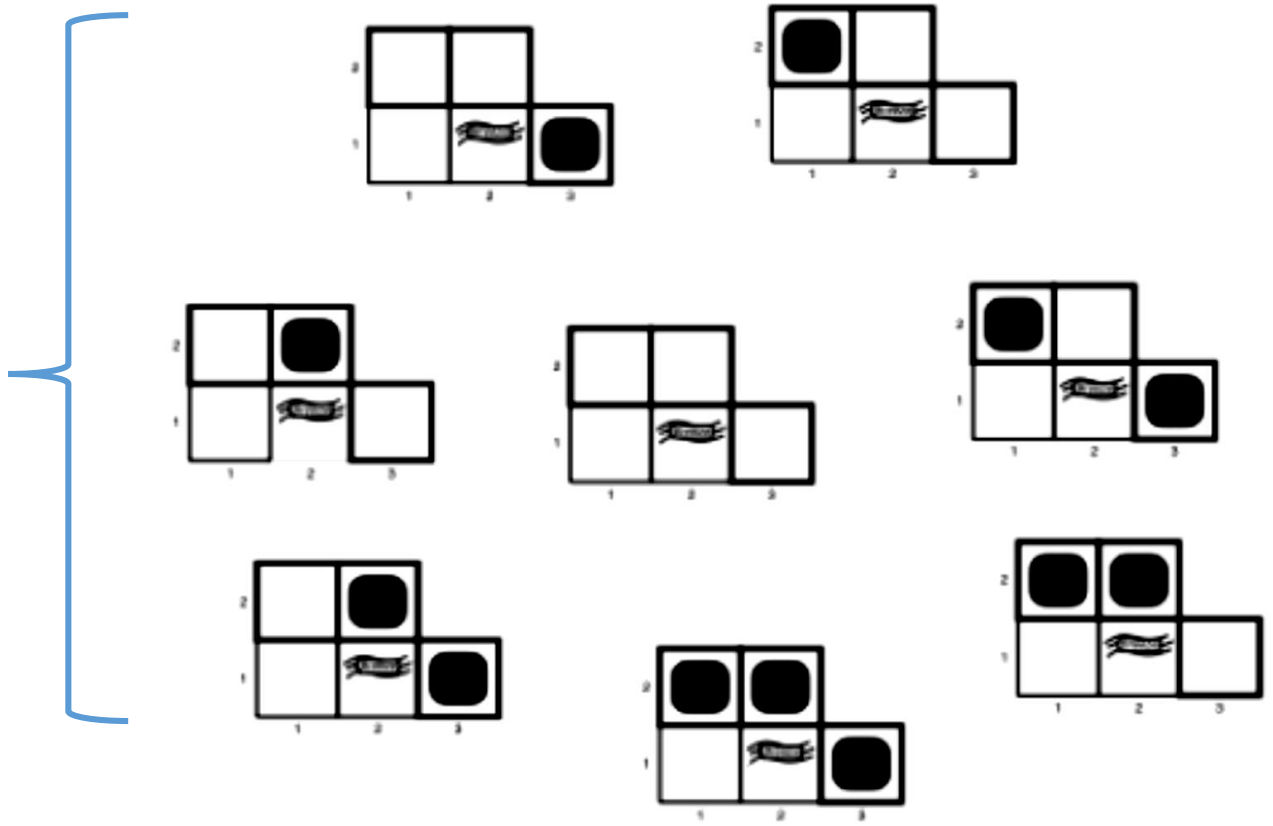


Consider possible models for ?s assuming only pits

3 Boolean choices $\Rightarrow$ 8 possible models

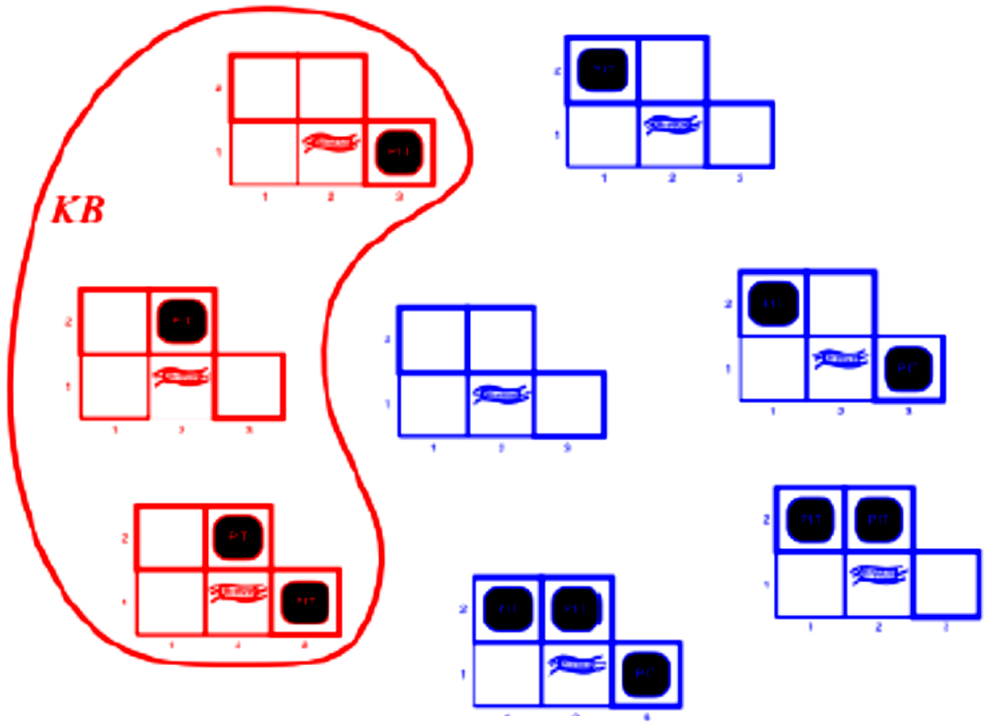
8 possible Interpretations

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



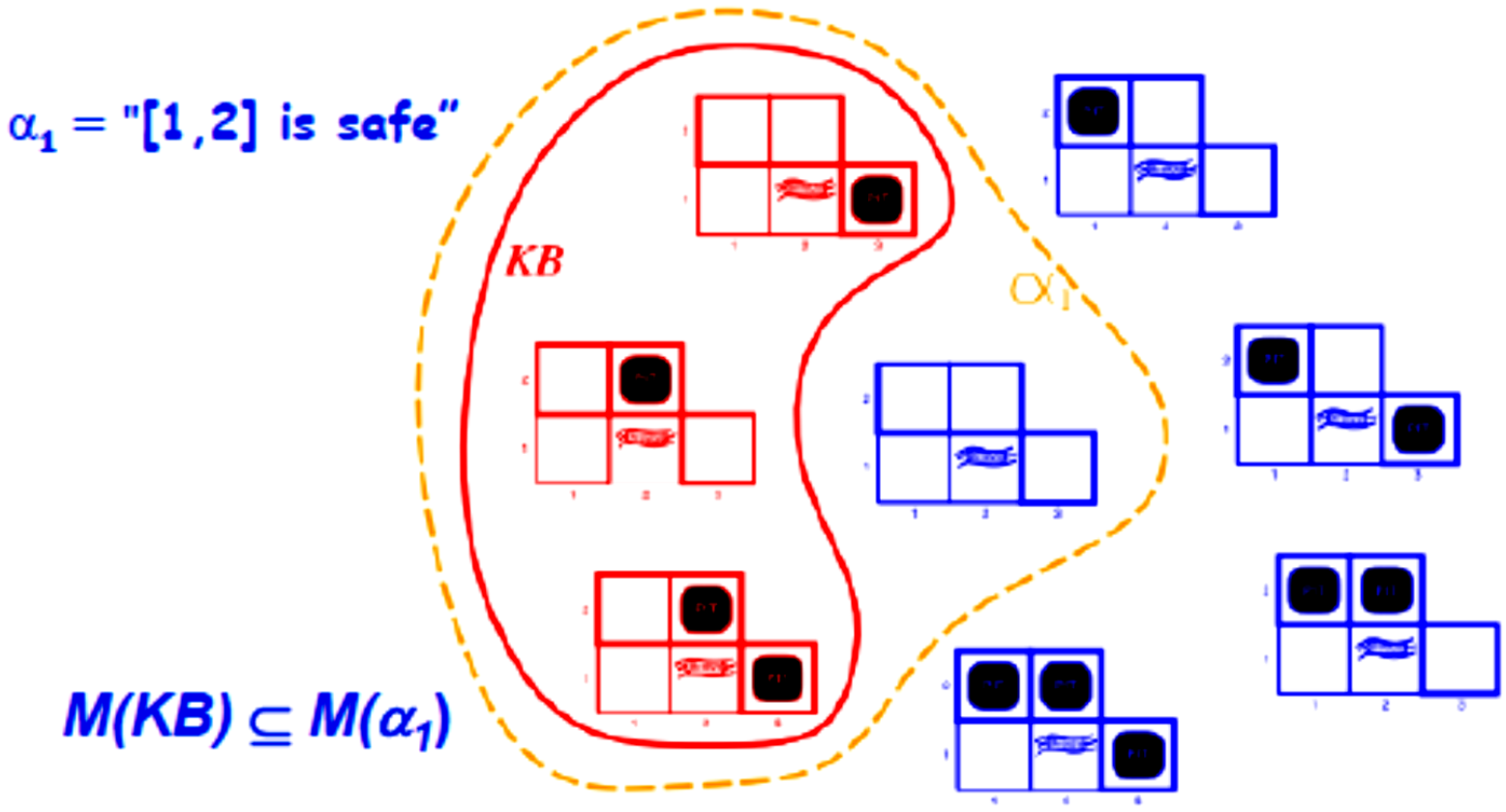
Interpretation consistent with rules + observations

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



- *KB* = wumpus-world rules + observations

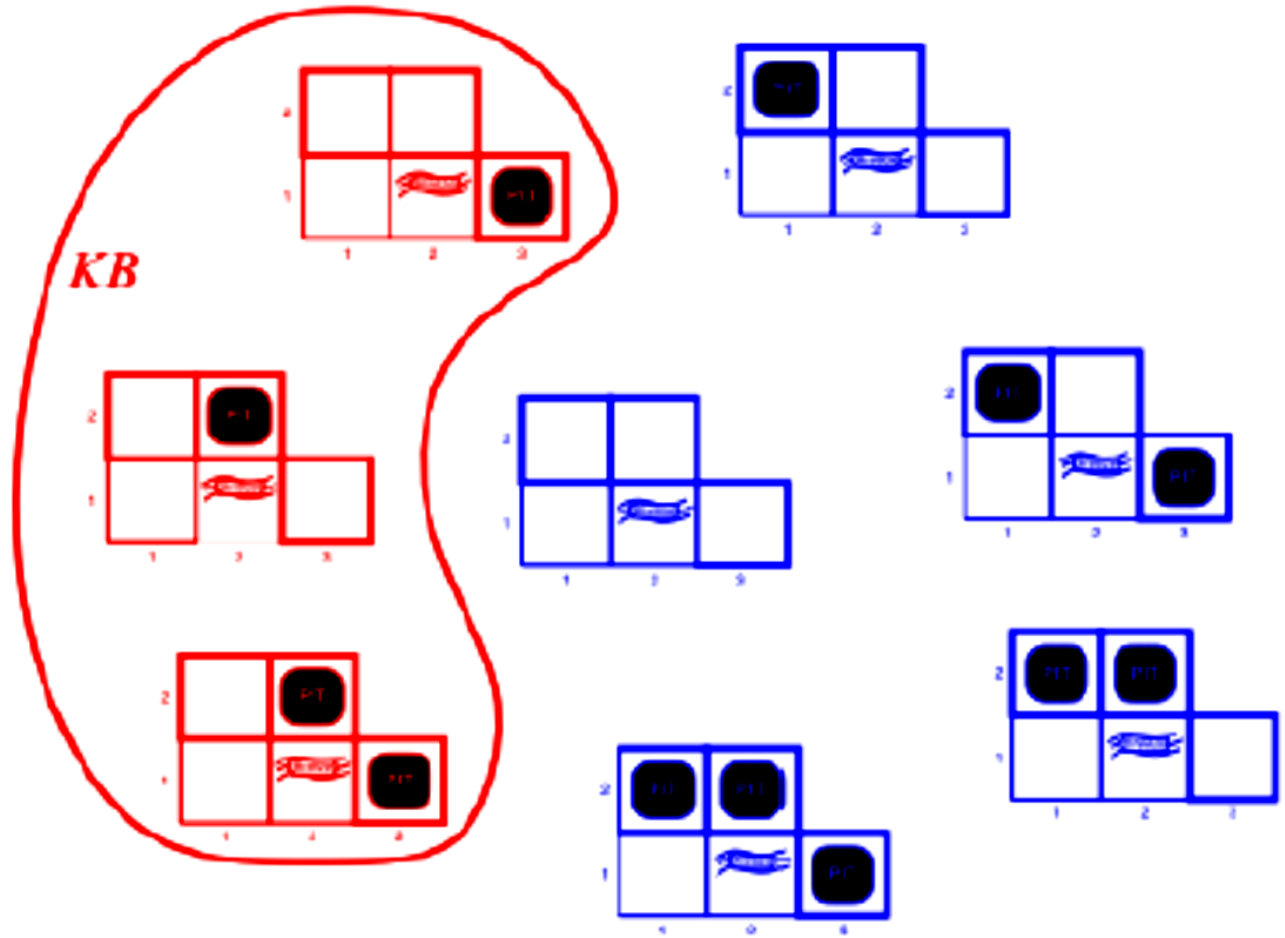
Entailment by Model Checking



- *KB* = wumpus-world rules + observations
- α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking

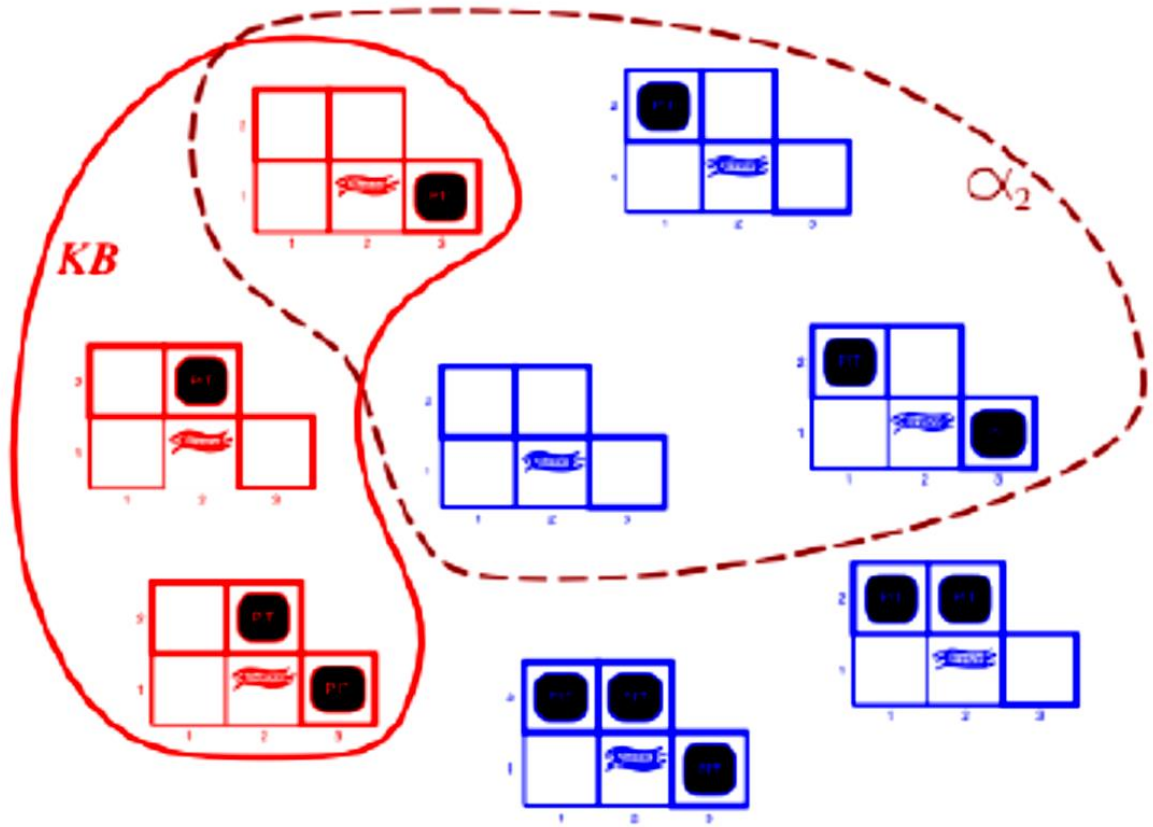
Another Example

Is [2,2] safe?



- *KB* = wumpus-world rules + observations

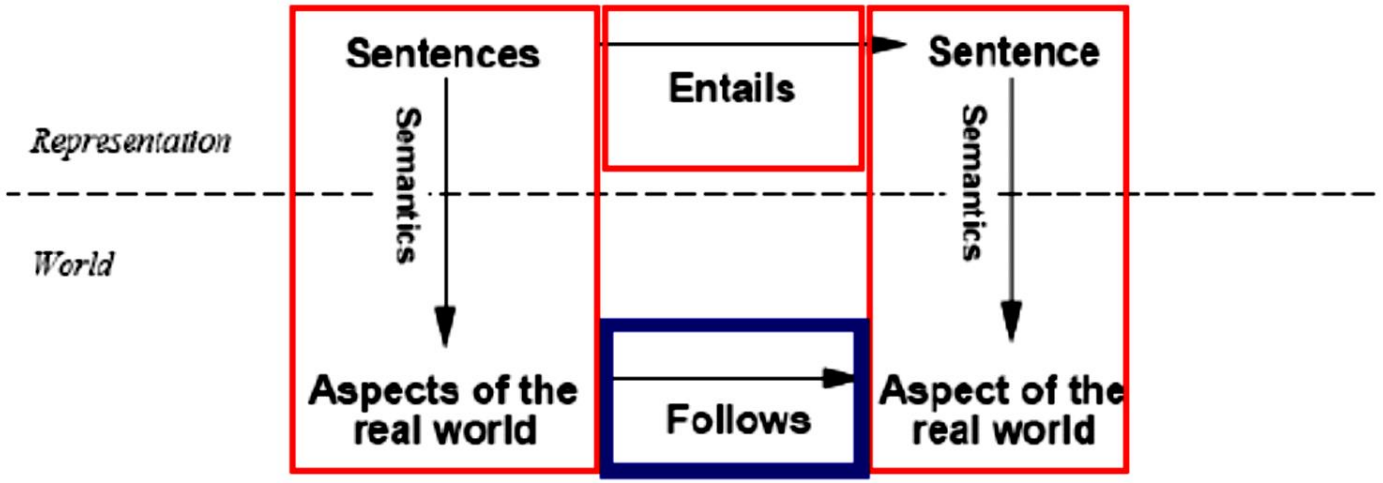
Another Example



$M(KB) \nsubseteq M(\alpha_2)$

- α_2 = "[2,2] is safe", $KB \nvdash \alpha_2$

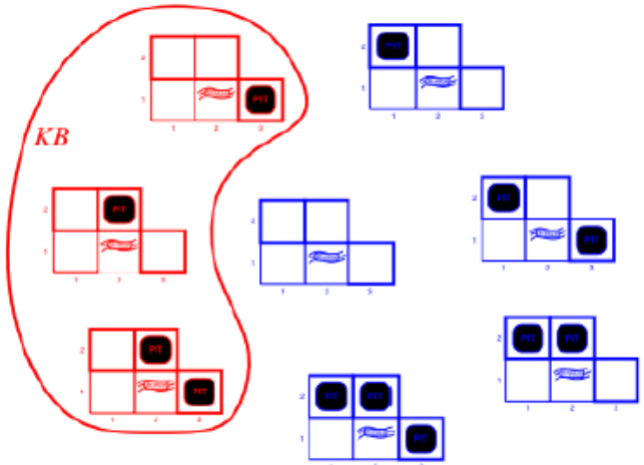
Relating to the Real World



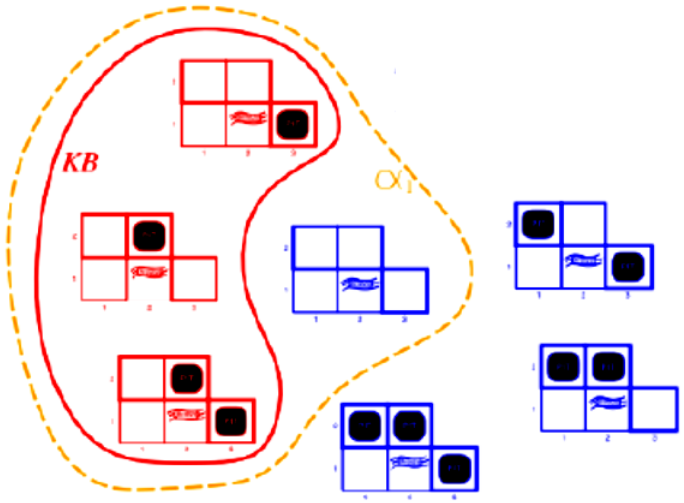
If a KB is true in the real world, then any sentence α derived from the KB by a sound inference procedure is also true in the real world

Illustration of real world follows "from: ¬ $P_{1,1}$ ^ $B_{2,1}$ follows ¬ $P_{1,2}$ "

Real world for ¬ $P_{1,1}$ ^ $B_{2,1}$:



Real world for ¬ $P_{1,2}$:



Propositional Logic: Syntax

Propositional logic is the simplest logic – illustrates basic ideas

Atomic sentences = proposition symbols = A, B, $P_{1,2}$, $P_{2,2}$ etc. used to denote properties of the world

- *Can be either True or False*

E.g. $P_{1,2}$ = "There's a pit in location [1,2]" is either true or false in the wumpus world

Propositional Logic: Syntax

Complex sentences constructed from simpler ones recursively using logical operators

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

An Interpretation I specifies true/false for each symbol

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

Rules for evaluating truth w.r.t. I :

$\neg S$ is true iff S is false

$S_1 \wedge S_2$ is true iff S_1 is true and S_2 is true

$S_1 \vee S_2$ is true iff S_1 is true or S_2 is true

$S_1 \Rightarrow S_2$ is true iff S_1 is false or S_2 is true

$S_1 \Leftrightarrow S_2$ is true iff both $S_1 \Rightarrow S_2$ and $S_2 \Rightarrow S_1$ are true

Truth Tables for Connectives

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

Propositional Logic: Semantics

Simple recursive process can be used to evaluate an arbitrary sentence

E.g., I :

$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
false	true	false

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

$= \mathit{true} \wedge (\mathit{true} \vee \mathit{false})$

$= \mathit{true} \wedge \mathit{true}$

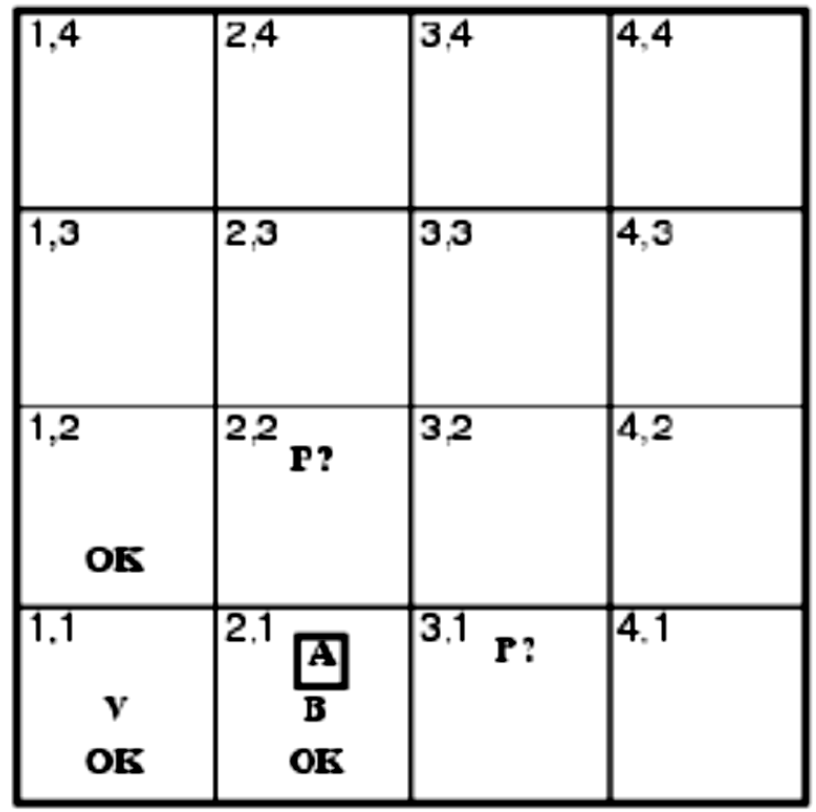
$= \mathit{true}$

Example: Wumpus World

Proposition Symbols and Semantics:

Let $P_{i,j}$ be true if there is a pit in [i, j].

Let $B_{i,j}$ be true if there is a breeze in [i, j].



Full Encoding of Wumpus World

In propositional logic:

$\neg P_{1,1}$
$\neg W_{1,1}$
$B_{x,y} \Leftrightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$
$S_{x,y} \Leftrightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$
$W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,4}$
$\neg W_{1,1} \vee \neg W_{1,2}$
$\neg W_{1,1} \vee \neg W_{1,3}$
…

$\Rightarrow$ 64 distinct proposition symbols, 155 sentences

Wumpus KB

Knowledge Base (KB) includes the following sentences:

Statements currently known to be true:

$\neg P_{1,1}$

$\neg B_{1,1}$

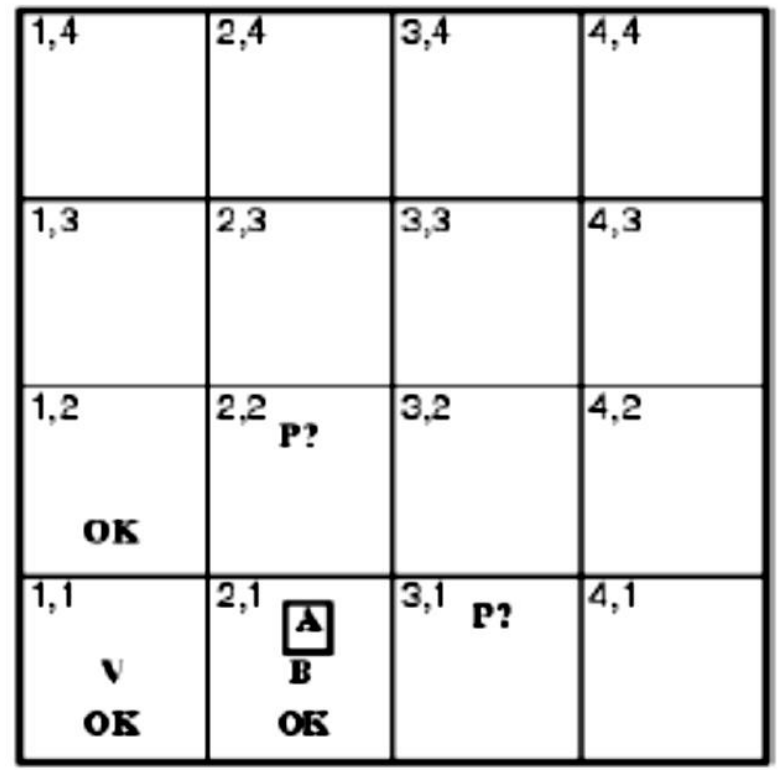
$B_{2,1}$

Properties of the world: E.g., "Pits cause breezes in adjacent squares"

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

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

(and so on for all squares)



Can a Wumpus-Agent use this logical representation and KB to avoid pits and the wumpus, and find the gold?

Is there no pit
in [1,2]?

$$KB \models \neg P_{1,2} \, ?$$

m is a model of a sentence α if α is true in *m*

$M(\alpha)$ is the set of <u>all models</u> of α

KB $\models \alpha$ (KB "entails" α) iff $M(KB) \subseteq M(\alpha)$

Inference by Truth Table Enumeration

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

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

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

KB

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	$\neg P_{1,2}$
false	false	false	false	false	false	false	false	true
false	false	false	false	false	false	true	false	true
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	true	false	false	false	false	false	false	true
false	true	false	false	false	false	true	true	true
false	true	false	false	false	true	false	true	true
false	true	false	false	false	true	true	true	true
false	true	false	false	true	false	false	false	true
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	true	true	true	true	true	true	false	false

In all Interpretations in which KB is true, $\neg P_{1,2}$ is also true

Therefore, KB $\models \neg P_{1,2}$

Inference by Truth Table Enumeration

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

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

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

KB

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*

$P_{2,2}$ is false in a Interpretation in which KB is true

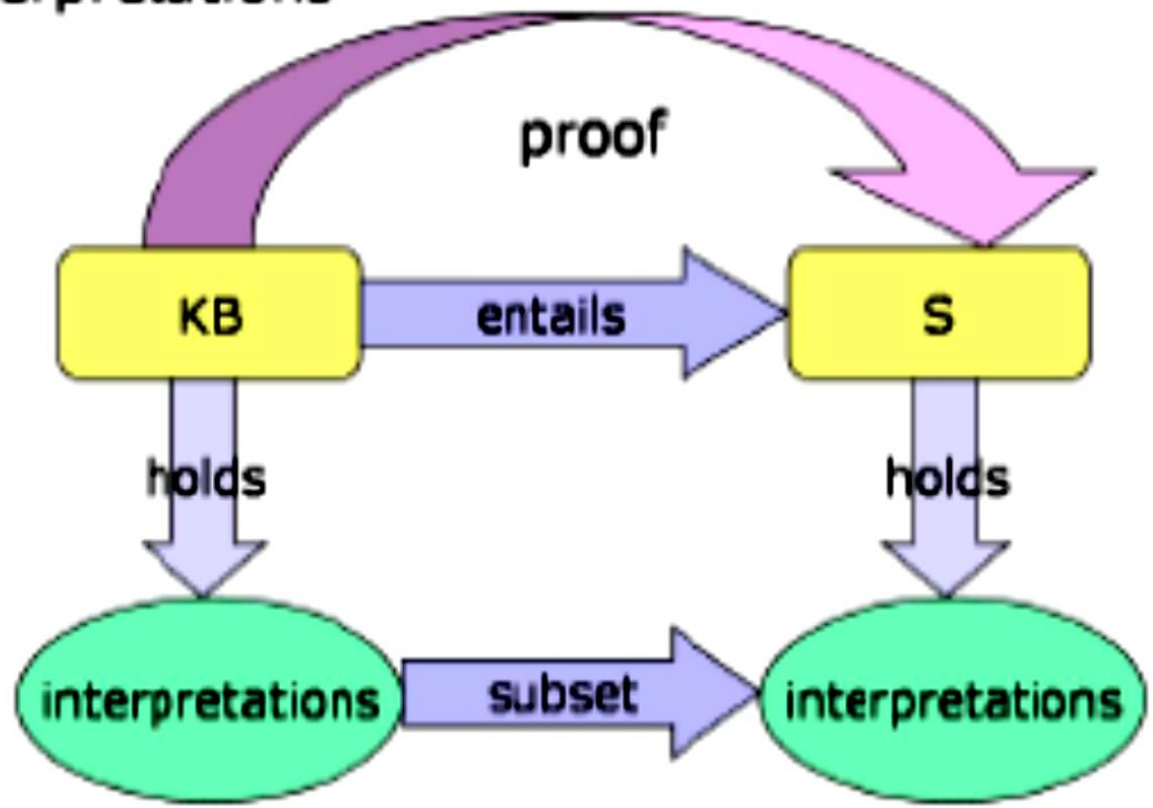
Therefore, KB $\not\vdash P_{2,2}$

Entailment using Proofs

- So far:
  - Enumerate all interpretations
  - Select those in which KB is true
  - Check to see if α is true in those interpretations
- Serious drawback - way too many interpretations, in general
- So we need methods that do entailment without exhaustive enumeration
- ***Proof*** methods test entailment without enumerating all interpretations

Entailment and Proof

A proof is a way to test whether a KB entails a sentence, without enumerating all possible interpretations



Inference using Proof Methods ⊢

- Proof is a sequence of sentences
- First ones are premises (KB)
- Then you can write down on the next line the result of applying an inference rule to previous lines
- When α is on a line you have proved α from KB, i.e. KB ⊢ α
- If inference rules are **sound**, then any α you can prove from KB is entailed by KB, i.e. **KB ⊢ α ⇒ KB ⊨ α**
- If inference rules are **complete**, then any α that is entailed by KB can be proved from KB, i.e. **KB ⊨ α ⇒ KB ⊢ α**

Some Inference Rules

$$\frac{\alpha \rightarrow \beta \quad \alpha}{\beta}$$ Modus ponens

$$\frac{\alpha \rightarrow \beta \quad \neg \beta}{\neg \alpha}$$ Modus tolens

$$\frac{\alpha \quad \beta}{\alpha \wedge \beta}$$ And-introduction

$$\frac{\alpha \wedge \beta}{\alpha}$$ And-elimination

How to say it - for modus ponens rule:

Given α and $\alpha \rightarrow \beta$ conclude β

Show example of soundness of modus ponens

Natural Deduction

Proof is a sequence of sentences

First ones are premises (KB)

Then, you can write down on line j the result of applying an inference rule to previous lines

When ϕ is on a line, you know $KB \vdash \phi$

If inference rules are sound, then $KB \models \phi$

Example:

Prove S

Step	Formula	Derivation
1	$P \wedge Q$	Given
2	$P \rightarrow R$	Given
3	$(Q \wedge R) \rightarrow S$	Given
4	P	1 And-Elim
5	R	4,2 Modus Ponens
6	Q	1 And-Elim
7	$Q \wedge R$	5,6 And-Intro
8	S	7,3 Modus Ponens

Another Example:

When it is raining, the ground is wet. When the ground is wet, it is slippery. It is raining. Prove that it is slippery.

1.	$raining \Rightarrow wet$	Premise
2.	$wet \Rightarrow slippery$	Premise
3.	$raining$	Premise
4.	wet	MP : 1, 3
5.	$slippery$	MP : 2, 4

Logical Equivalence

Two sentences are logically equivalent iff true in same models:

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

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

- Example:
  - Knowledge base is
    - Wumpus World

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

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

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

    - Percepts

$$R_4 : \neg \mathrm{B}_{1,1}$$

$$R_5 : \mathrm{B}_{2,1}$$

Natural Deduction:

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

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

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

From R_4 , R_8 (modes ponens):

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

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

Automated Proofs

- The proof process is mechanical
- Cast it as a search problem
- Successor function generates all 1-step consequences of all applicable inference rules
- But big branching factor
- Even more branching for proof by cases

Validity and Satisfiability

A sentence is valid if it is true in **all** Interpretations

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** Interpretation

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

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

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

Resolution

Terminology:

Literal = proposition symbol or its negation

E.g., A, $\neg A$, B, $\neg B$, etc.

Clause = disjunction of literals

E.g., $(B \vee \neg C \vee \neg D)$

Resolution assumes sentences are in *Conjunctive Normal Form (CNF):*

sentence = conjunction of clauses

E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Propositional Resolution

- One single rule of inference

Resolution Rule :

$$\frac{\alpha \vee \beta \qquad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

Resolution rule
in Wumpus problem

$$\frac{\text{There is a pit in [1,3] or There is a pit in [2,2]} \qquad \text{There is no pit in [2,2]}}{\text{There is a pit in [1,3]}}$$

General form of Resolution Rule

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

where ℓ_i and m_j are complementary literals

E.g., $\dfrac{P_{1,3} \vee P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$

Resolution is a sound inference rule

Conversion to CNF

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

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 ($\wedge$ over $\vee$) and flatten:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

This is in CNF - Done!

- Resolution rule:

$$\frac{\alpha \vee \beta \qquad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

- Resolution refutation:
  - Convert all sentences to CNF
  - Negate the desired conclusion (converted to CNF)
  - Apply resolution rule until either
    - Derive false (a contradiction)
    - Can't apply any more

Propositional Resolution Example

Prove R

1	P v Q
2	P → R
3	Q → R

Step	Formula	Derivation
1	P v Q	Given
2	¬ P v R	Given
3	¬ Q v R	Given
4	¬ R	Negated conclusion

Propositional Resolution Example

Prove R

1	$P \vee Q$
2	$P \rightarrow R$
3	$Q \rightarrow R$

Step	Formula	Derivation
1	$P \vee Q$	Given
2	$\neg P \vee R$	Given
3	$\neg Q \vee R$	Given
4	$\neg R$	Negated conclusion
5	$Q \vee R$	1,2
6	$\neg P$	2,4
7	$\neg Q$	3,4

Resolution Example

Prove R

1	$(P \rightarrow Q) \rightarrow Q$
2	$(P \rightarrow P) \rightarrow R$
3	$(R \rightarrow S) \rightarrow \neg(S \rightarrow Q)$

Convert to CNF

- $\neg(\neg P \vee Q) \vee Q$
- $(P \wedge \neg Q) \vee Q$
- $(P \vee Q) \wedge (\neg Q \vee Q)$
- $(P \vee Q)$

- $\neg(\neg P \vee P) \vee R$
- $(P \wedge \neg P) \vee R$
- $(P \vee R) \wedge (\neg P \vee R)$

- $\neg(\neg R \vee S) \vee \neg(\neg S \vee Q)$
- $(R \wedge \neg S) \vee (S \wedge \neg Q)$
- $(R \vee S) \wedge (\neg S \vee S) \wedge (R \vee \neg Q) \wedge (\neg S \vee \neg Q)$
- $(R \vee S) \wedge (R \vee \neg Q) \wedge (\neg S \vee \neg Q)$

1	$P \vee Q$	
2	$P \vee R$	
3	$\neg P \vee R$	
4	$R \vee S$	
5	$R \vee \neg Q$	
6	$\neg S \vee \neg Q$	
7	$\neg R$	Neg

1	$P \vee Q$	
2	$P \vee R$	
3	$\neg P \vee R$	
4	$R \vee S$	
5	$R \vee \neg Q$	
6	$\neg S \vee \neg Q$	
7	$\neg R$	Neg
8	S	4,7
9	$\neg Q$	6,8
10	P	1,9
11	R	3,10
12	•	7,11

Resolution algorithm

- To show KB $\models \alpha$, use proof by contradiction, i.e., show $KB \wedge \neg\alpha$ unsatisfiable

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

$clauses \leftarrow$ the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
$new \leftarrow \{\}$
loop do
  for each C_i, C_j **in** *clauses* **do**
    $resolvents \leftarrow$ PL-RESOLVE(C_i, C_j)
    if *resolvents* contains the empty clause **then return** *true*
    $new \leftarrow new \cup resolvents$
  if $new \subseteq clauses$ **then return** *false*
  $clauses \leftarrow clauses \cup new$

Resolution example

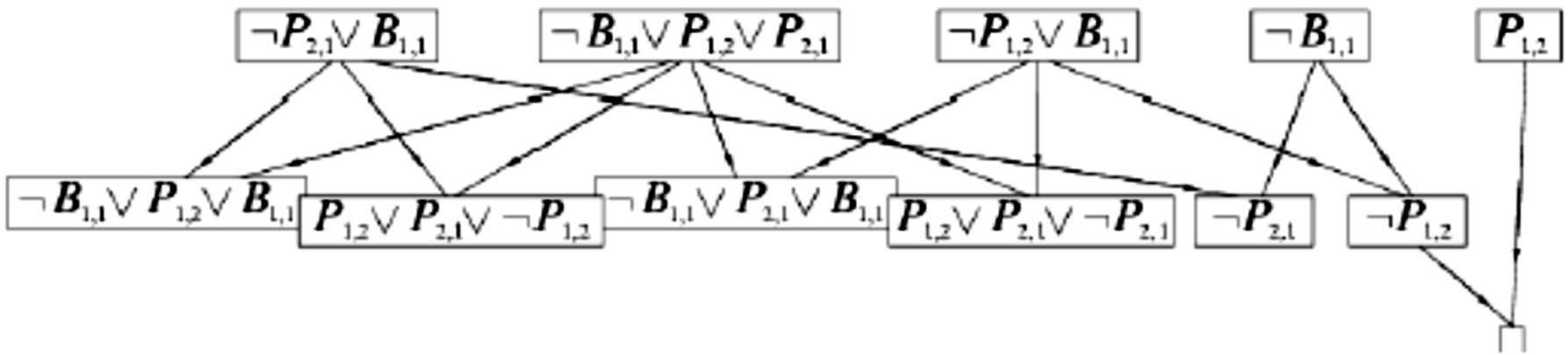
Given no breeze in [1,1], prove there's no pit in [1,2]

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

Resolution: Convert to CNF and show KB $\wedge \neg \alpha$ is unsatisfiable

Resolution example

Resolution example



Empty clause
(i.e., KB $\wedge \neg$ a unsatisfiable)

Completeness Proof of Resolution in Propositional Logic

Specialized Inference Algorithms

Forward/Backward Chaining

Horn Clauses

A *Horn clause* is a clause containing at most one positive literal.

Example: $\{r, \neg p, \neg q\}$
Example: $\{\neg p, \neg q, \neg r\}$
Example: $\langle p \rangle$
Non-Example: $\{q, r, \neg p\}$

NB: Every Horn clause can be written as a "rule".

$$\{\neg p, \neg q, r\} \quad \rightarrow \quad p \wedge q \Rightarrow r$$

- Require sentences to be in **Horn Form**:

  KB = conjunction of Horn clauses

  Horn clause =

  - proposition symbol or
  - "(conjunction of symbols) $\Rightarrow$ symbol"
    (i.e. clause with at most 1 positive literal)

  E.g., KB = $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$

- F/B chaining based on "Modus Ponens" rule:

$$\frac{\alpha_1, \ldots, \alpha_n, \qquad \alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

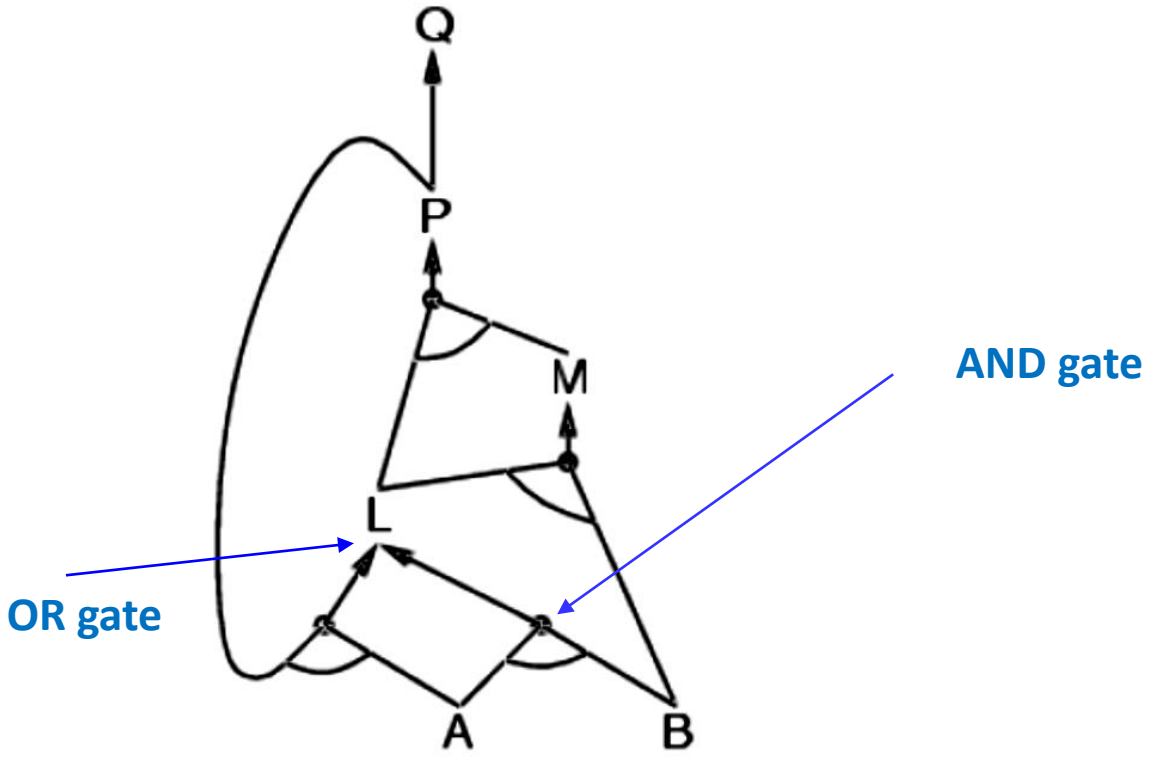
  Complete for Horn clauses

- Very natural and linear time complexity in size of KB

Forward chaining

- Idea: fire any rule whose premises are satisfied in *KB*,
  add its conclusion to *KB*, until query q is found

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



AND-OR Graph

Query = "Is Q true?"

Forward chaining algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true or false
  local variables: count, a table, indexed by clause, initially the number of premises
                   inferred, a table, indexed by symbol, each entry initially false
                   agenda, a list of symbols, initially the symbols known to be true

  while agenda is not empty do
      p ← POP(agenda)
      unless inferred[p] do
          inferred[p] ← true
          for each Horn clause c in whose premise p appears do
              decrement count[c]
              if count[c] = 0 then do
                  if HEAD[c] = q then return true
                  PUSH(HEAD[c], agenda)
  return false
```

Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

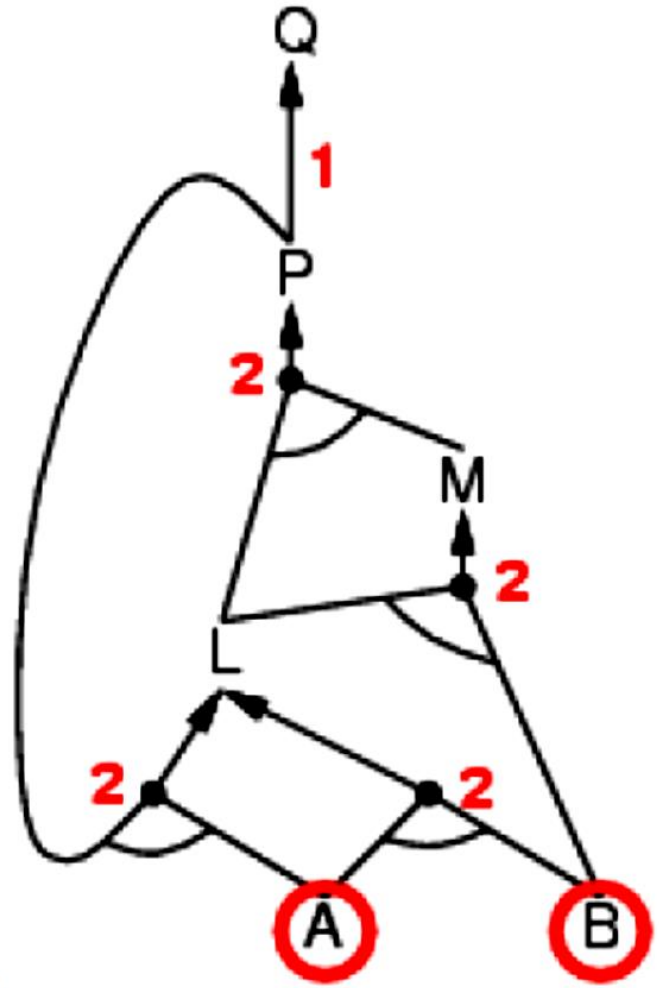
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Query = Q
(i.e. "Is Q true?")

Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

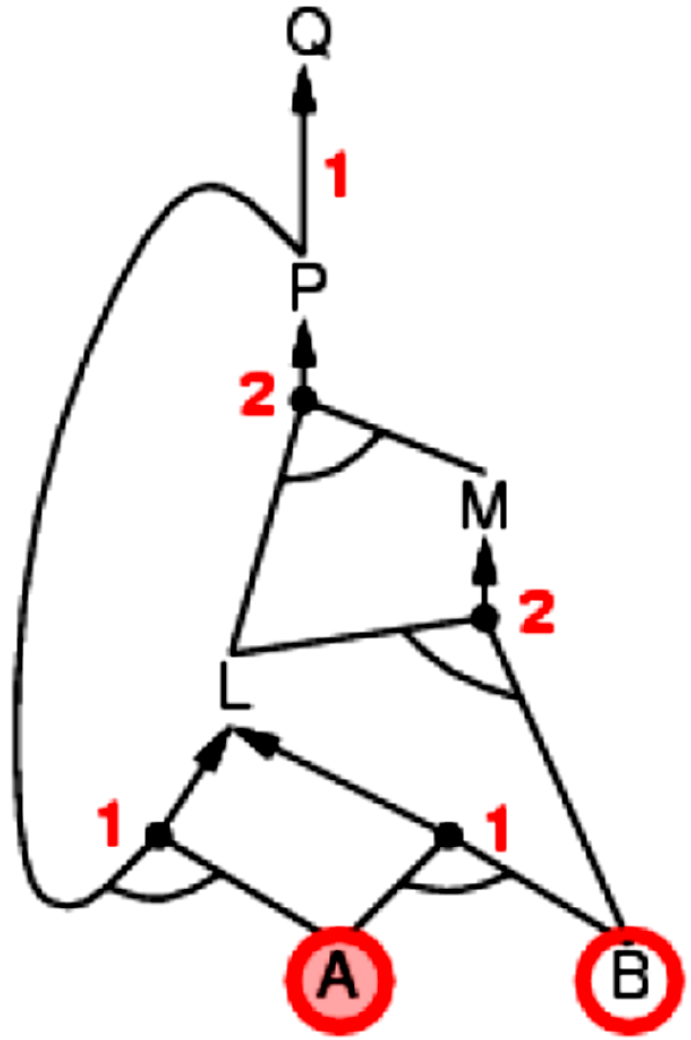
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

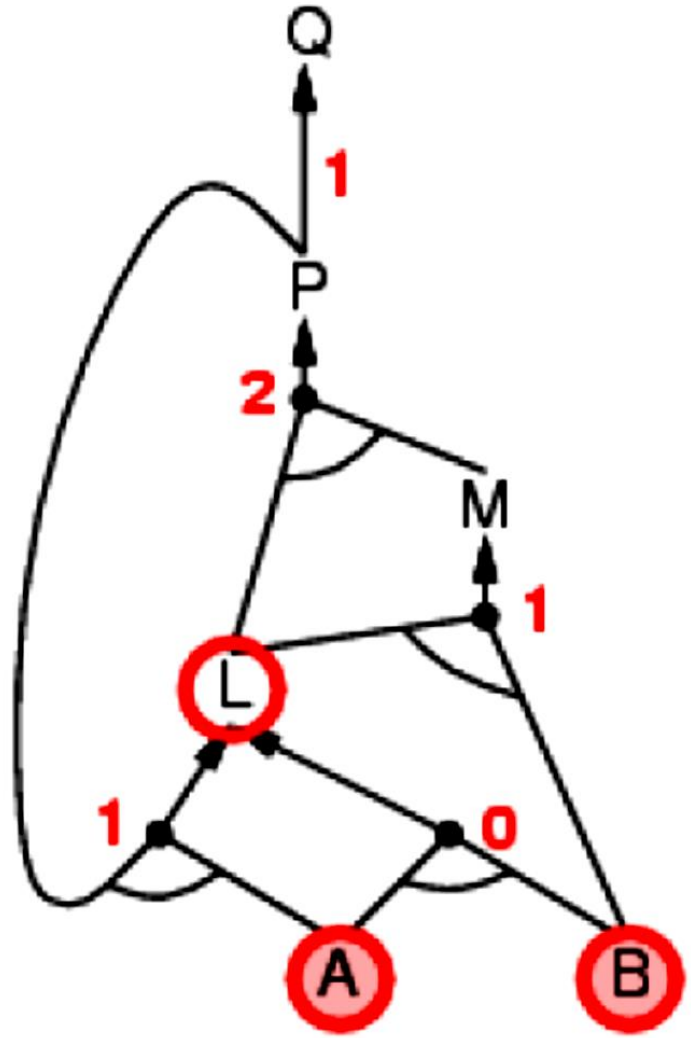
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

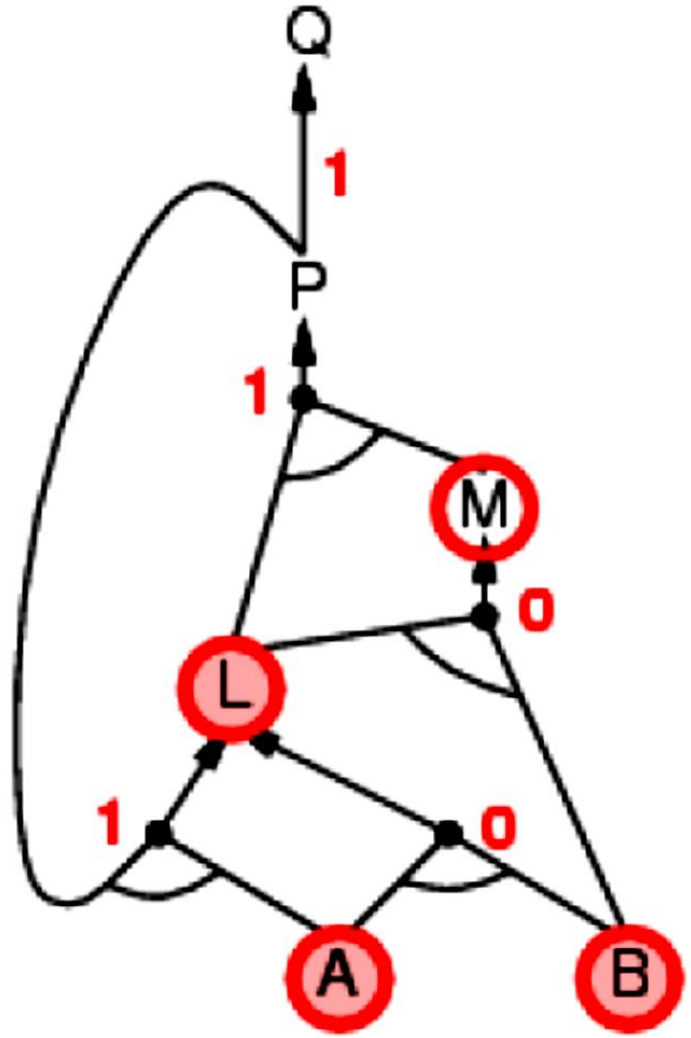
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

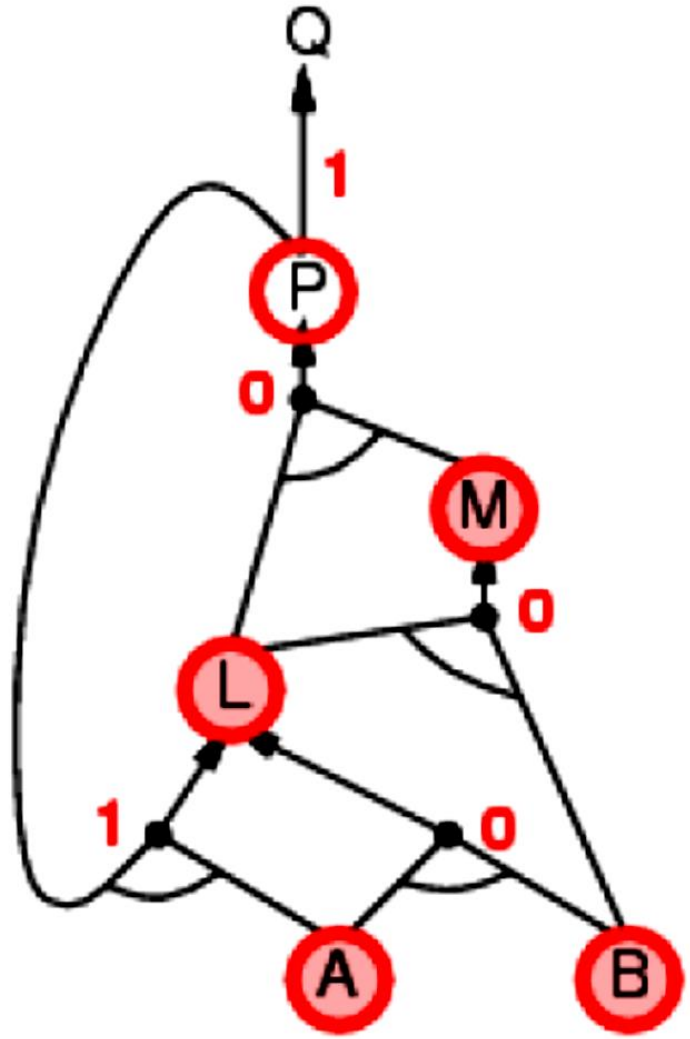
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

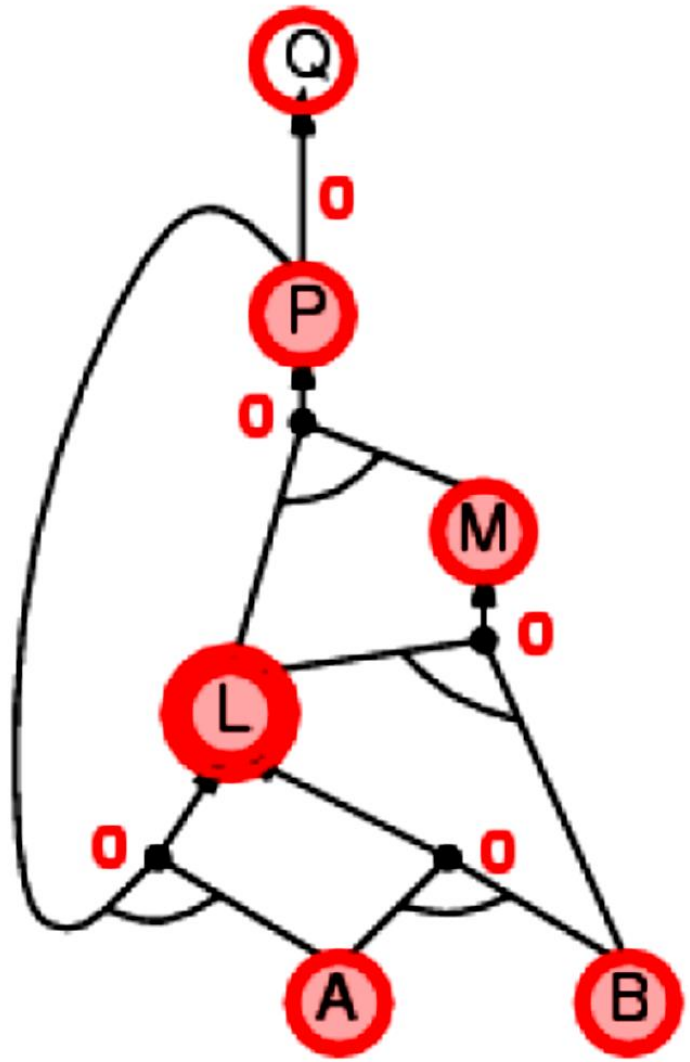
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward chaining is sound & complete for Horn KB

Completeness of Forward Chaining for Propositional Horn Logic

- FC derives every atomic sentence that is entailed by *KB*
  1. FC reaches a fixed point where no new atomic sentences are derived
  2. Consider the final state as a model *m*, assigning true/false to symbols
  3. Every clause in the original *KB* is true in *m*

     $a_1 \wedge \ldots \wedge a_k \Rightarrow b$
  4. Hence *m* is a model of *KB*
  5. If $KB \models q$, *q* is true in every model of *KB*, including *m*

Backward chaining

Idea: work backwards from the query *q*:

to prove *q* by BC,

check if *q* is known already, or

prove by BC all premises of some rule concluding *q*

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

Avoid repeated work: check if new subgoal

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

Backward chaining example

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

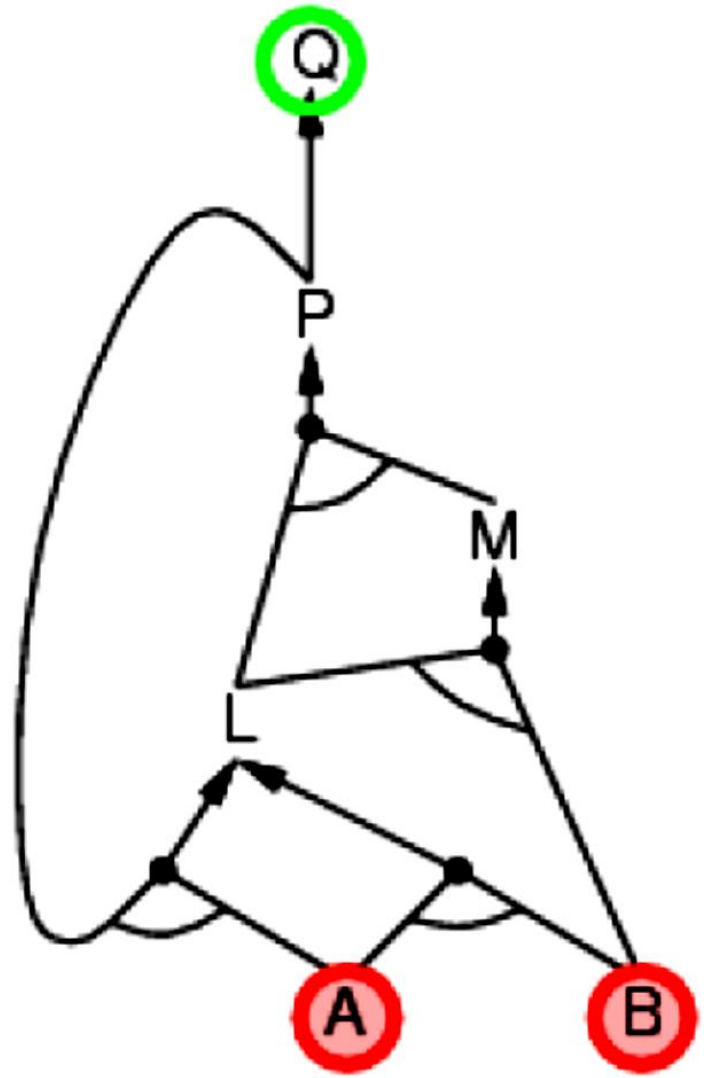
$B \wedge L \Rightarrow M$

$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B



Forward vs. backward chaining

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

- FC may do lots of work that is irrelevant to the goal

- BC is goal-driven, appropriate for problem-solving,
  e.g., How do I get an A in this class?
  e.g., What is my best exit strategy out of the classroom?

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

Why Satisfiability?

- Recall: $KB \models \alpha$ iff $KB \wedge \neg\alpha$ is unsatisfiable
  - Equivalent to proving sentence α by contradiction

- Thus, algorithms for satisfiability can be used for inference (entailment)

- However, determining if a sentence is satisfiable or not (the SAT problem) is NP-complete

  Finding a fast algorithm for SAT automatically yields fast algorithms for hundreds of difficult (NP-complete) problems

Efficient SAT Solvers

A *complete* SAT algorithm that runs in bounded time can be used to prove entailment, since entailment can be proven by showing unsatisfiability of $\alpha \wedge \neg \beta$.

Two commonly-used SAT algorithms, both based on model checking.

- **DPLL**. A complete backtracking model-enumeration approach. Because this algorithm is complete, it could be used to prove entailment.
- **WALKSAT.** A random-walk model-checking approach. Because the random walk could take an infinite time to explore all models, WALKSAT is not a good choice to prove unsatisfiability. However, for many problems, WALKSAT proves satisfiability quickly.

DPLL Algorithm –Ideas

Determine if an input propositional logic sentence (in CNF) is satisfiable.

Improvements over truth table enumeration:

1. **Early termination**
   - A clause is true if any literal is true.
   - A sentence is false if any clause is false.

2. **Pure symbol heuristic**
   - Pure symbol: always appears with the same "sign" in all clauses.
   - e.g., In the three clauses $(A \vee \neg B)$, $(\neg B \vee \neg C)$, $(C \vee A)$, A and B are pure, C is impure.
   - Make a pure symbol literal true.

3. **Unit clause heuristic**
   - Unit clause: only one literal in the clause
   - The only literal in a unit clause must be true.

DPLL Algorithm

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

inputs: s, a sentence in propositional logic

clauses ← the set of clauses in the CNF representation of s

symbols ← 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* ← FIND-PURE-SYMBOL(*symbols*, *clauses*, *model*)

if *P* is non-null **then return** DPLL(*clauses*, *symbols*–*P*, [*P* = *value*|*model*])

P, *value* ← FIND-UNIT-CLAUSE(*clauses*, *model*)

if *P* is non-null **then return** DPLL(*clauses*, *symbols*–*P*, [*P* = *value*|*model*])

P ← FIRST(*symbols*); *rest* ← REST(*symbols*)

return DPLL(*clauses*, *rest*, [*P* = *true*|*model*]) **or**
DPLL(*clauses*, *rest*, [*P* = *false*|*model*])

The WALKSAT Algorithm

- Local search algorithm

  Incomplete: may not always find a satisfying assignment even if one exists

- Evaluation function?

  = Number of satisfied clauses

  WalkSAT tries to maximize this function

- Balance between greediness and randomness

  Each iteration:

  Randomly select a symbol for flipping

  Or select symbol that maximizes # satisfied clauses

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

max-flips, number of flips allowed before giving up

(*m* is an Interpretation)

m ← a random assignment of *true*/*false* to the symbols in *clauses*

for $i = 1$ **to** *max-flips* **do**

if m satisfies *clauses* **then return** m

clause ← a randomly selected clause from *clauses* that is false in m

with probability p flip the value in m of a randomly selected symbol from *clause*

else flip whichever symbol in *clause* maximizes the number of satisfied clauses

return *failure*

Greed

Randomness

Summary of Inference in Propositional Logic

Two main approaches to inference:

1. Inference by *Model Checking* - Truth Table, DPLL, (Incomplete) WALKSAT, ..
2. Inference by *Theorem Proving*: Use rules of inference to construct a proof of a sentence

- *Search for proof* based on modus ponens, and-elimination, logical equivalences

  One important equivalence: $A \Rightarrow B \equiv \neg A \vee B$
- *Forward and backward chaining* for KBs of Horn clauses (disjunctions of literals, at most 1 positive literal)

  If A and B are true and $A \wedge B \Rightarrow C$, then C true
- *Resolution*: A single complete and sound rule