Logical Agents: Knowledge Representation

CS3243: Introduction to Artificial Intelligence – Lecture 8

13 March 2023

Contents

Reference: AIMA 4th Edition, Section 7.1-7.4

Administrative Matters

Upcoming…

- Deadlines
  - TA6 (released last week)
    - *Due in your Week 9 tutorial session*
    - *Submit the a physical copy (more instructions on the Tutorial Worksheet)*
  - Prepare for the tutorial!
    - Participation marks = 5%
  - Project 2 (released Week 6)
    - *Due this Sunday (19 March), 2359 hrs*
  - Project 3 (released this week)
    - *Due Week 12 Sunday (9 April), 2359 hrs*

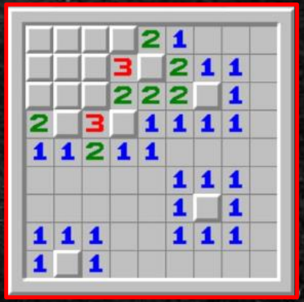
Midterm Appeals

- Appeals → to your tutor
- Deadline = Day of your Week 9 Tutorial

A Problem with Problem-Solving Agents

Problem-Solving Agents

- Problem-solving agents try to find a solution via Search
- No real model of what the agent knows
  - Each state contains knowledge on state of entire environment
    - Knows actions and transition model
    - Implicit general facts about the environment
      - Route Finding Agent – implicit knowledge that road lengths cannot be negative
      - 8-puzzle – implicit knowledge that two number-tiles cannot occupy the same grid
  - Atomic representations limiting
    - Imagine a game of minesweeper where the environment is only partially observable; the agent would not know where all the mines actually were
    - A problem-solving agent would typically use a representation that includes all possible mine positions (with accompanying adjacent mine numbers) in an attempt to search for a viable solution from the current board



On to agents with generalised knowledge representations: Knowledge-Based Agents

Knowledge-Based Agents: Logical Agents

Knowledge-Based Agents

- Represent agent domain knowledge using logical formulas
- General idea
  - Make inferences on existing information
    - Use existing knowledge to infer new information
  - States similar to CSPs
    - Represented as assignments of values to variables
- Agent structure

Inference Engine ← Domain-independent algorithms

Knowledge Base ← Domain-specific content

Knowledge Base (KB)

- What is a knowledge base (KB)?
  - Set of sentences in a formal language
    - Sentences are expressive and parsable
  - Pre-populate with domain knowledge
    - Example: game rules, general rules/knowledge
- Declarative approach to problem-solving
  - TELL it what it needs to know
    - Update with percept/state/action information
  - ASK itself what to do
    - Make inferences that help determine what actions to take
      - Answers should follow from the KB

KB Agent Function

What happened?

What did I do?

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

  TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))
  action ← ASK(*KB*, MAKE-ACTION-QUERY(*t*))
  TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))
  $t \leftarrow t + 1$
  return *action*

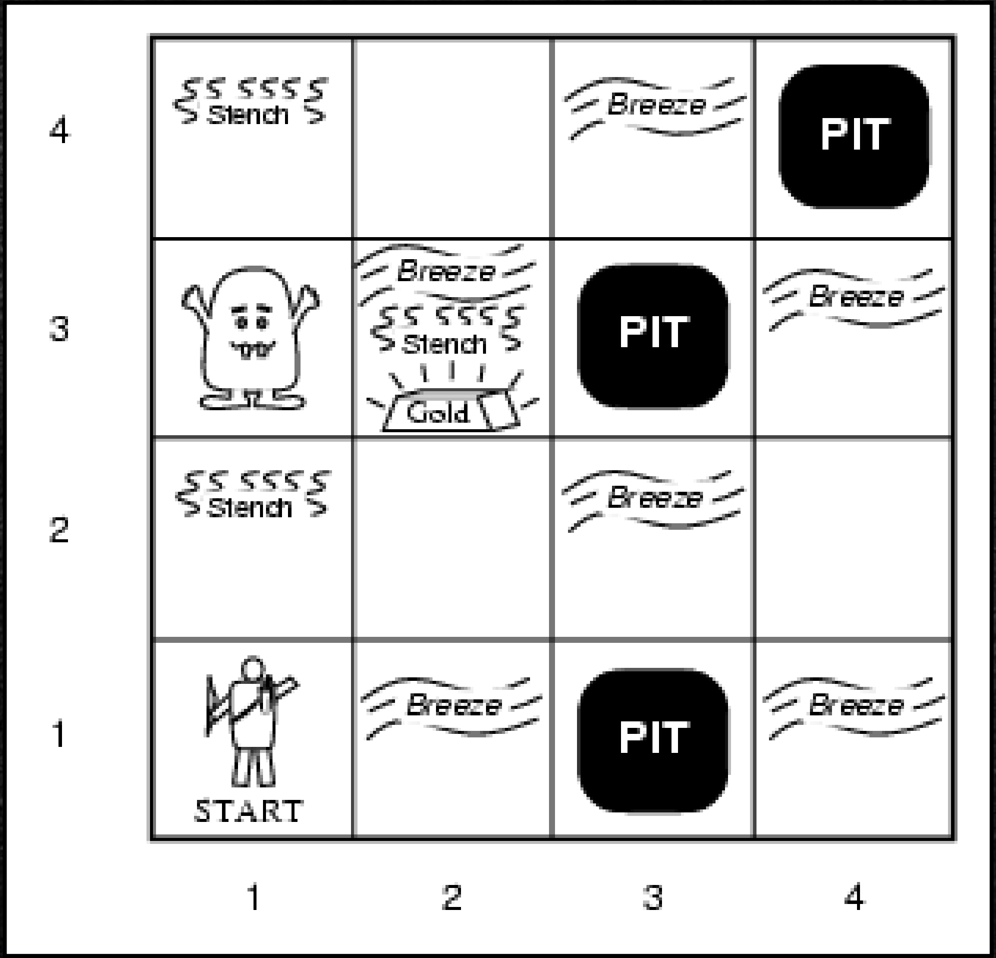
What did I perceive at time t?

What is the best action at time t?

- Agent must be able to
  - Represent states, actions, etc.
  - Incorporate new percepts
  - Update internal representation of environment
  - Deduce hidden environment properties, and deduce actions

An Example:
The Wumpus Dungeon

About Wumpus World



Performance Measure

- Optimise score
  - Obtain Gold: +1000
  - Death: -1000
  - Each Action: -1
  - Fire Arrow: -10

Environment

- 4×4 grid of rooms
  - Agent
  - Wumpus
  - Gold
  - Pits

Actuators

- Turn left/right
- Move forward
- Fire arrow (kills Wumpus if facing it; uses up arrow)
- Grab gold
- Exit Wumpus dungeon (by climbing out at (1,1))

Sensors

- Rooms adjacent to Wumpus are SMELLY
- Rooms adjacent to Pit are BREEZY
- Gold glitters (can detect it if in same room)
- Bump into walls
- Hear scream if Wumpus killed

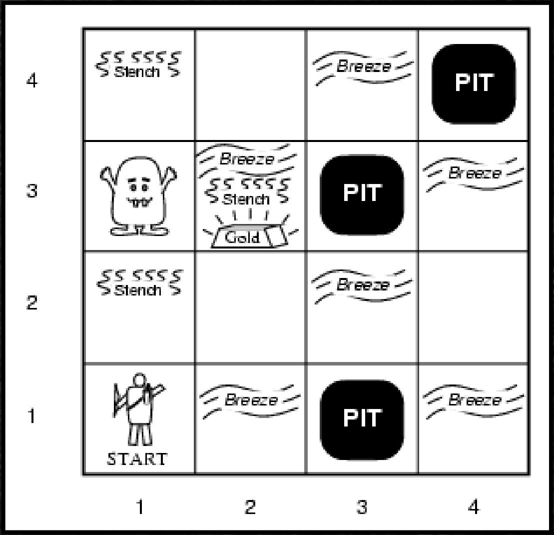
Properties of Wumpus World

- Not fully observable
  - Only local perception
  - Don’t know what is in unexplored rooms
- Deterministic
- Sequential
- Static
- Discrete
- Single Agent

Exploring the Wumpus Dungeon

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 **A** **OK V**	2,1 **A** **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



Start at (1,1); Infer that (1,2) and (2,1) are OK (i.e., safe)

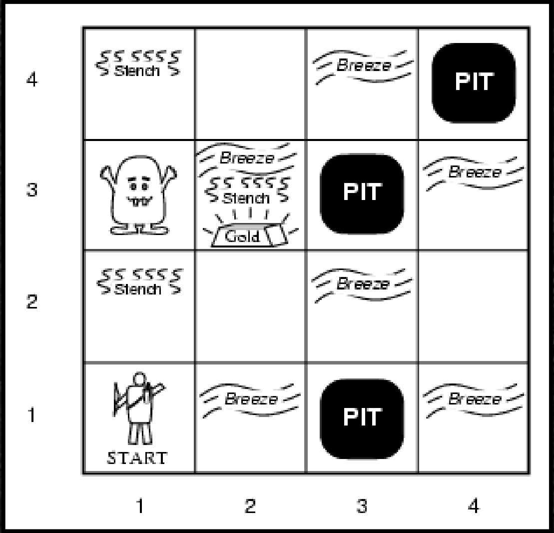
Iteration 1: Move to (2,1)

Iteration 2: Move back to (1,1)

Exploring the Wumpus Dungeon

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

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



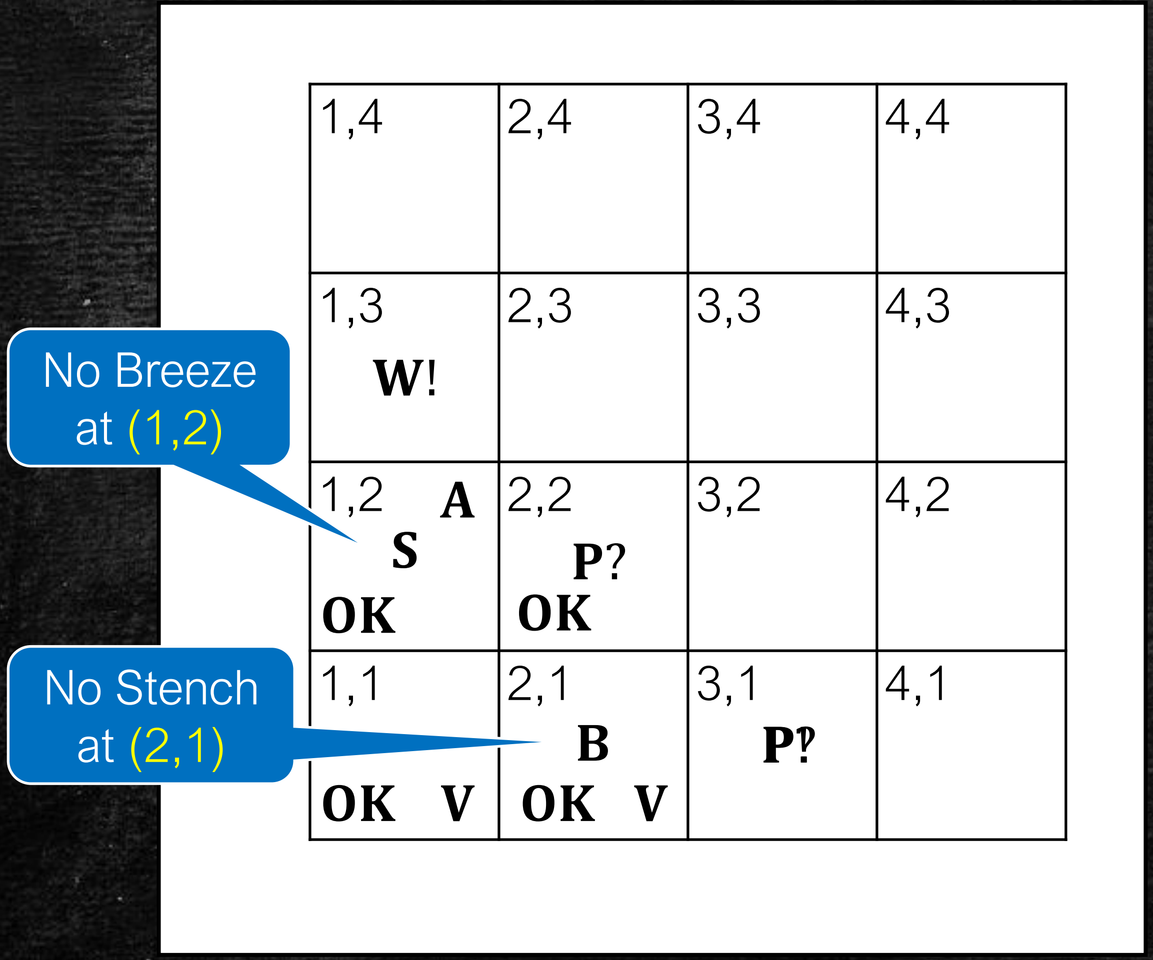
Start at (1,1); Infer that (1,2) and (2,1) are OK (i.e., safe)

Iteration 1: Move to (2,1)

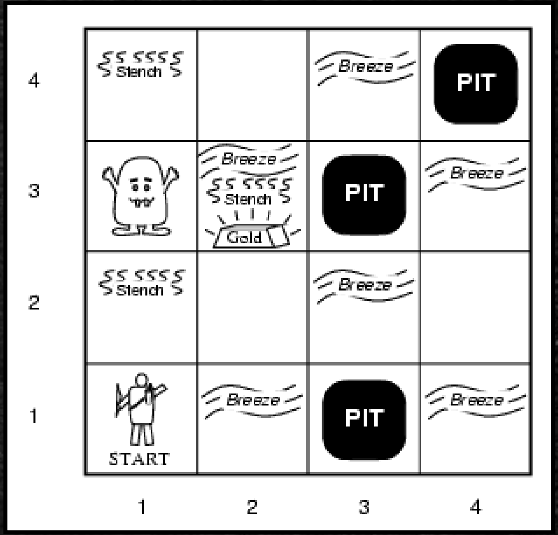
Iteration 2: Move back to (1,1)

Iteration 3: Move to (1,2)

Exploring the Wumpus Dungeon



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



Start at (1,1); Infer that (1,2) and (2,1) are OK (i.e., safe)

Iteration 1: Move to (2,1)

Iteration 2: Move back to (1,1)

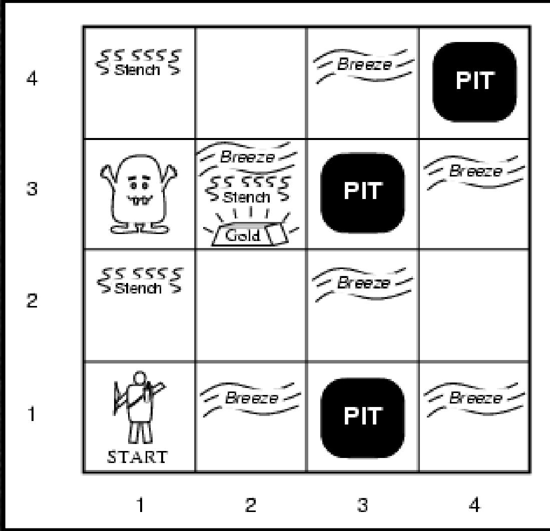
Iteration 3: Move to (1,2)

Iteration 4: Move to (2,2)

Exploring the Wumpus Dungeon

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

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



Start at (1,1); Infer that (1,2) and (2,1) are OK (i.e., safe)

Iteration 1: Move to (2,1)

Iteration 2: Move back to (1,1)

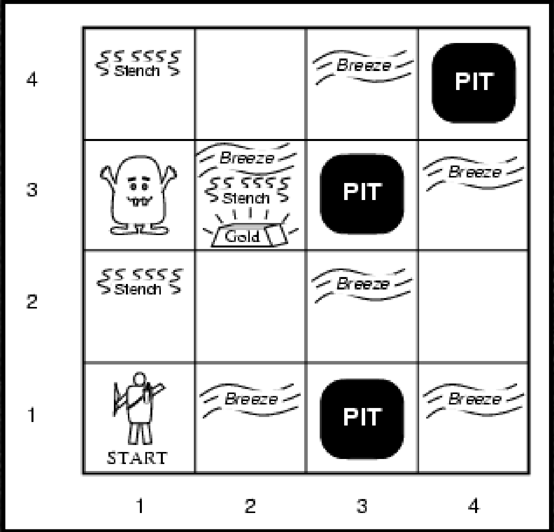
Iteration 3: Move to (1,2)

Iteration 4: Move to (2,2)

Exploring the Wumpus Dungeon

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

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



Start at (1,1); Infer that (1,2) and (2,1) are OK (i.e., safe)

Iteration 1: Move to (2,1)

Iteration 2: Move back to (1,1)

Iteration 3: Move to (1,2)

Iteration 4: Move to (2,2)

Iteration 5: Move to (2,3)

Logic

Review of Logic

- Logic
  - Formal language for knowledge representation (KR)
  - Allows the inference of conclusions about environment
- Syntax
  - Defines sentences in the language
- Semantics
  - Defines meaning of sentences
- Truth value
  - Statement result given observed values
    - Defines validity of a sentence within the environment
      - i.e., given value assignments that hold in the environment

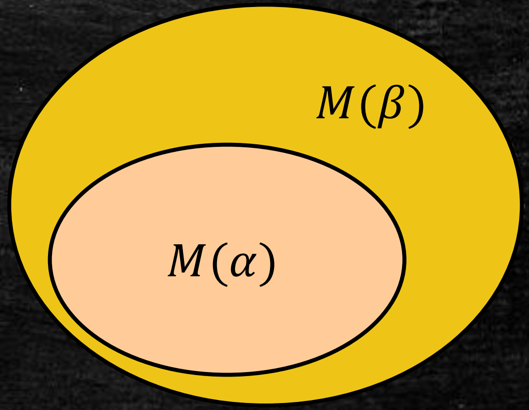
Review of Logic

- Example of KR: language of arithmetic
  - Syntax
    - $x + 2 \geq y$ is a sentence
    - $x2y +>$ is not a sentence
  - Truth values
    - $x + 2 \geq y$ is true in a world where $x = 7, y = 1$
    - $x + 2 \geq y$ is false in a world where $x = 0, y = 6$

Entailment

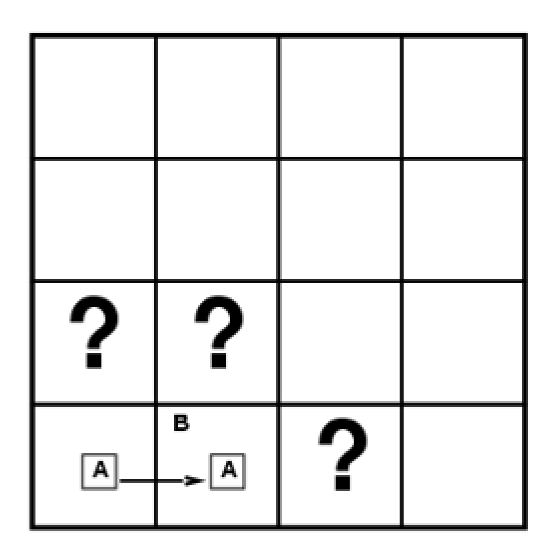
Entailment

- Modelling
  - v models α if α is true under v
    - v corresponds to one set of value assignments (applied to sentences α)
    - v corresponds to one instance of the environment (known part of a state)
  - For example
    - $\alpha = (q \in \mathbb{Z}_+) \wedge (\forall n, m \in \mathbb{Z}_+ : q = nm \Rightarrow n \vee m = 1)$
    - For which values of q will α be true?
- Let $M(\alpha)$ be the set of all models for α
- Entailment ($\vDash$) means that one thing follows from the another
  - $\alpha \vDash \beta$ or equivalently $M(\alpha) \subseteq M(\beta)$
  - Example:
    - $[\alpha = (q \text{ is prime})] \vDash [\beta = (q \text{ is odd}) \vee (q = 2)]$



Entailment in the Wumpus World

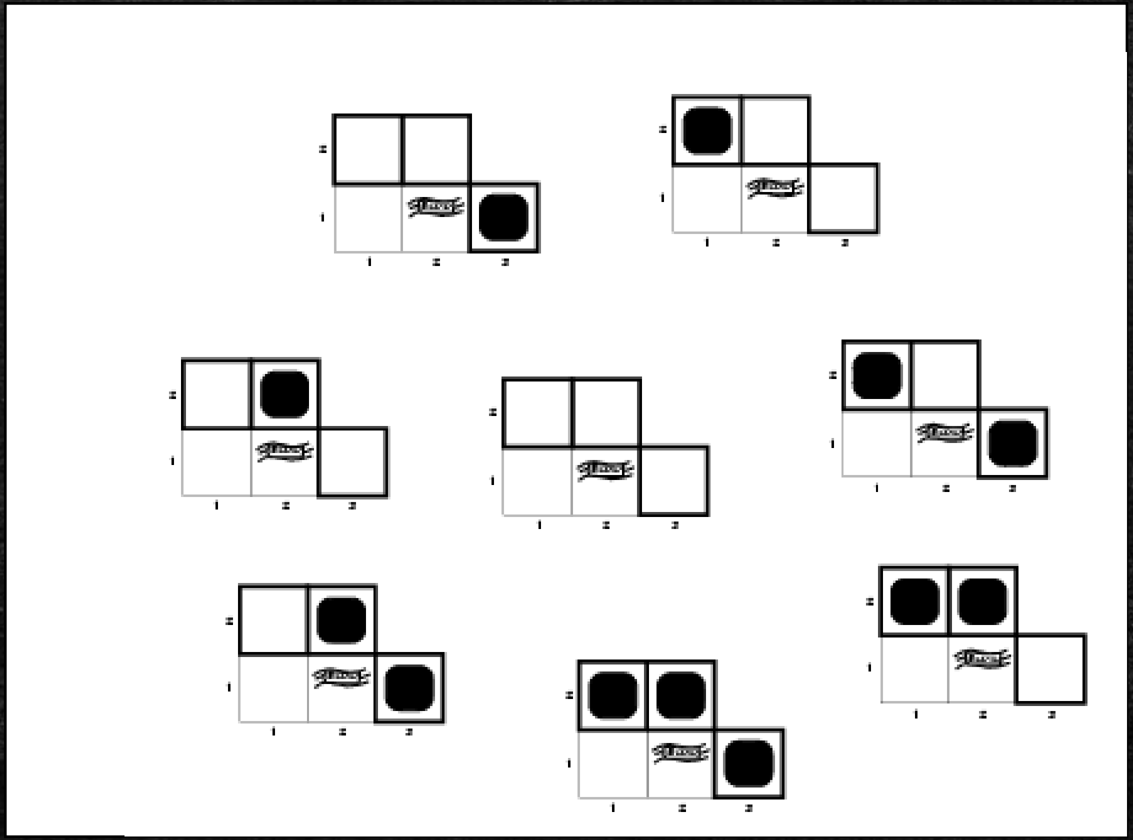
- Situation:
  - Detected Nothing at (1,1)
  - Moved Right to (2,1)
  - Detected Breeze at (2,1)
- Consider possible models for KB with pits
  - 3 Boolean choices ⇒ 8 possible models
    - Pit or No Pit at: (1,2), (2,2), (3,1)
    - All ($2^3 = 8$) permutations for the above (each a possible model)



Within this example, we will only deal with pits only; we ignore the Wumpus.

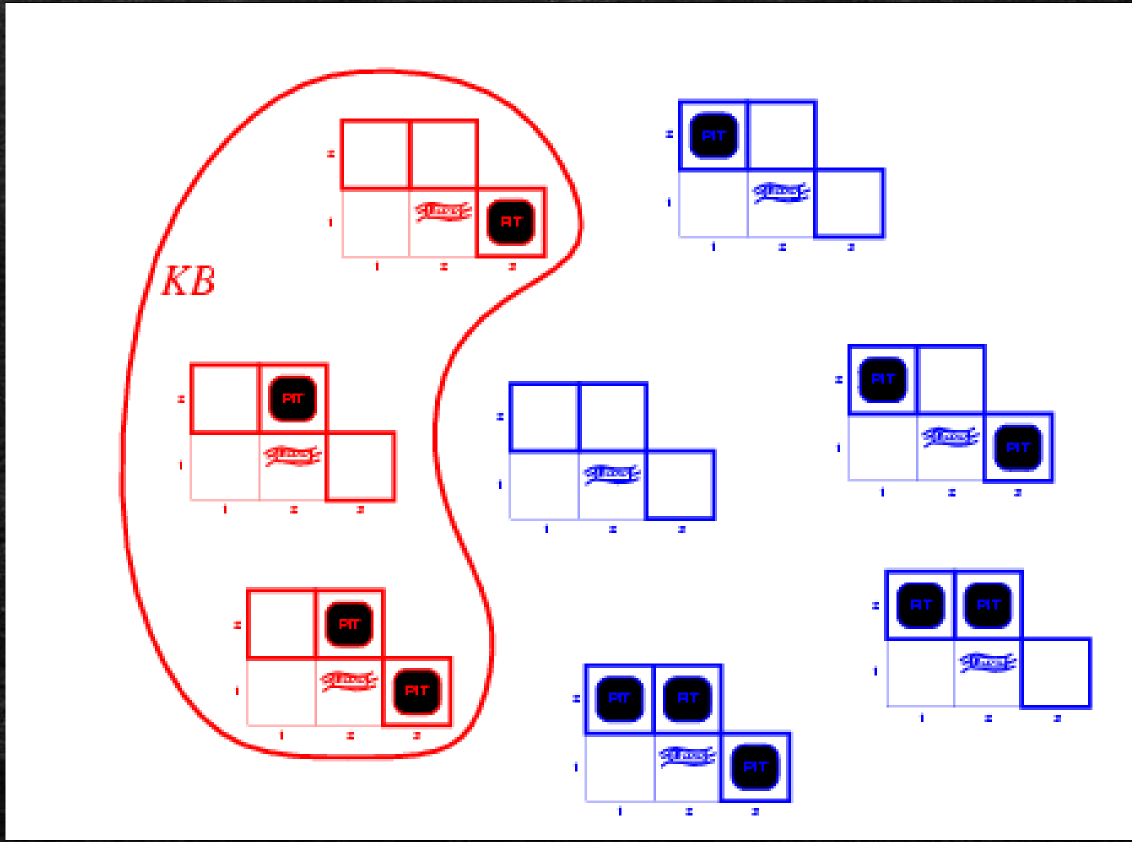
Entailment in the Wumpus World

- Possible 8 models



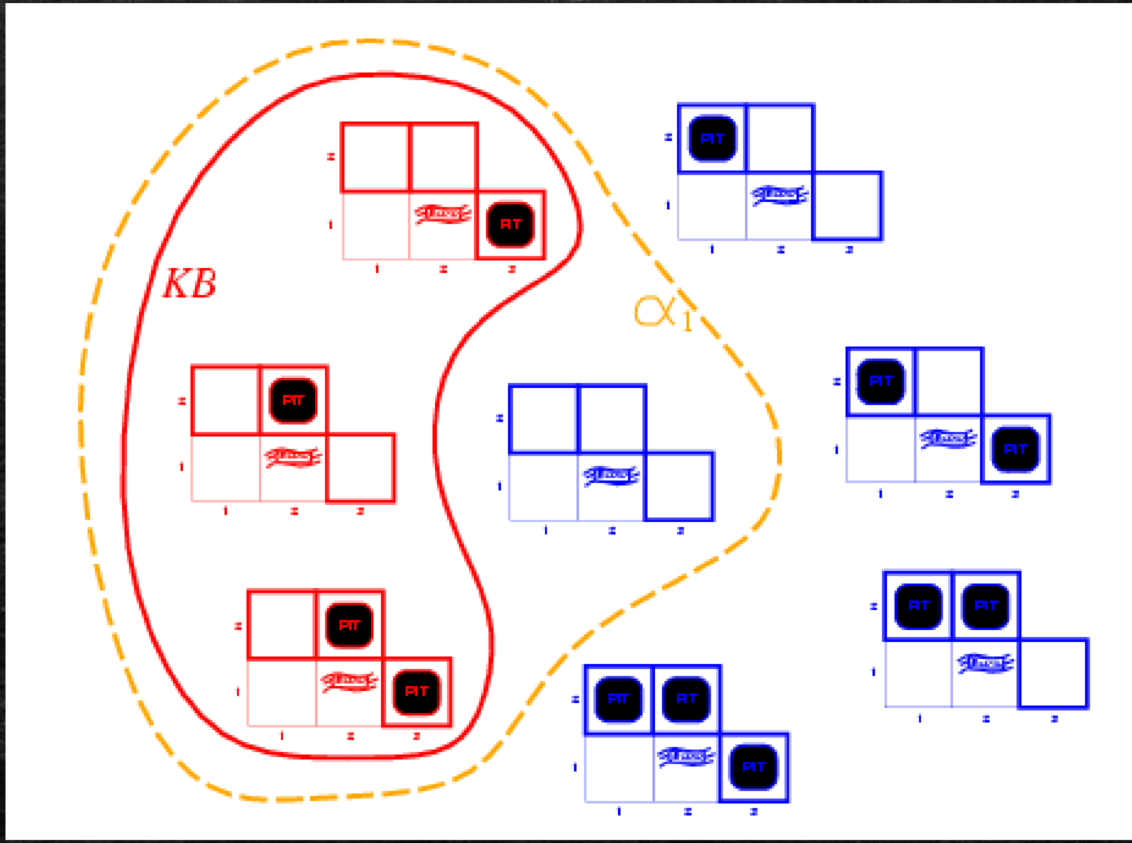
Entailment in the Wumpus World

- KB = rules + percepts
- Percepts
  - No Breeze at (1,1)
  - Breeze at (2,1)
- Relevant rules
  - No Pit at (1,1)
  - No Pit at (1,2)
  - No Pit at (2,1)
  - Breeze ⇒
    Pit in Adjacent Room



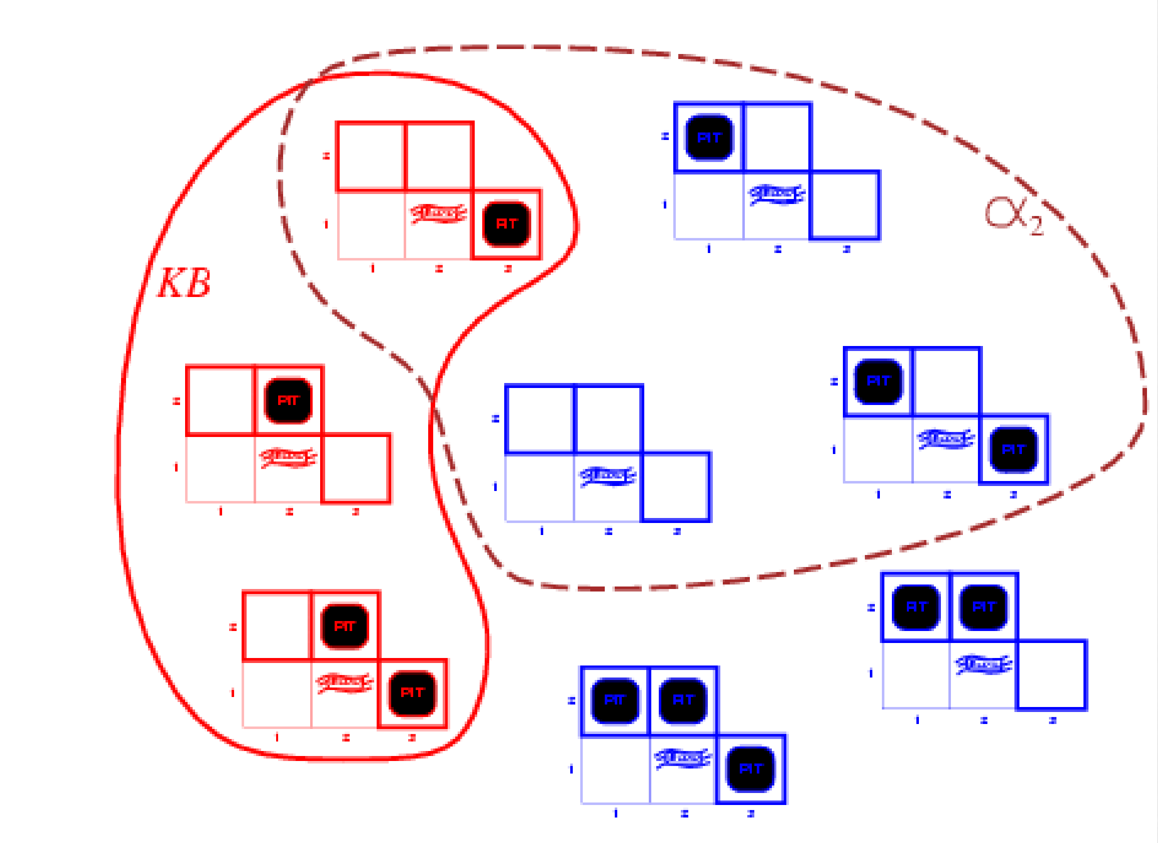
Entailment in the Wumpus World

- KB = rules + percepts
- Let α_1 = (1,2) is Safe
- We observe that:
  $M(\text{KB}) \subseteq M(\alpha_1)$
- Or rather:
  $\text{KB} \vDash \alpha_1$
- We may thus infer that it is safe for the agent to move to (1,2)



Entailment in the Wumpus World

- KB = rules + percepts
- Let α_2 = (2,2) is Safe
- We observe that: KB $\not\models \alpha_2$
- Since: $M(\text{KB}) \not\subseteq M(\alpha_1)$
- May NOT infer that it is safe for the to move to (2,2)
  - Exist some models where KB is True but α_2 is False
  - For entailment, we want all α_2 True when KB True
- Also, cannot infer unsafe!



Questions about the Lecture?

- Was anything unclear?
- Do you need to clarify anything?
- Ask on Archipelago
  - Specify a question
  - Upvote someone else's question

Invitation Link (Use NUS Email --- starts with E)
https://archipelago.rocks/app/resend-invite/12166893023

Propositional Logic

Review of Propositional Logic: Syntax

- A simple language for logic – illustrates basic ideas
- Defines allowable sentences
- Sentences are represented by symbols – e.g., s_1, s_2
  - Formed over basic variables
- Logical connectives for constructing complex sentences from simpler ones
  - 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)
    - $s_1 \vee s_2$ is a sentence (disjunction)
    - $s_1 \Rightarrow s_2$ is a sentence (implication)
    - $s_1 \Leftrightarrow s_2$ is a sentence (biconditional – *iff.*)

Review of Propositional Logic: Semantics

- A model
  - Truth assignment to the given basic variables
  - Given n variables, 2^n truth assignments
- All other sentences' truth value are derived according to logical rules
  - Example
    - Given x_1 = True; x_2 = False; x_3 = True
    - What is the truth value for $(x_1 \wedge \neg x_2) \Rightarrow \neg(x_3 \vee (\neg x_1 \wedge x_2))$?
      - Recall that X ⇒ Y is true if X false, or X true and Y true

Wumpus World KB

- Notation
  - P_{ij} = True ⇔ Pit at (i, j)
  - B_{ij} = True ⇔ Breeze at (i, j)
- Given
  - $R_1: \neg P_{1,1}$
  - $R_2: \neg B_{1,1}$
  - $R_3: B_{2,1}$
- Rules: "Pits cause a breeze in adjacent squares"
  - $R_4: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ – i.e., $\neg B_{1,1} \Leftrightarrow \neg(P_{1,2} \vee P_{2,1})$
  - $R_5: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

Within this example, again, we will only deal with pits only; we ignore the Wumpus.

KB is true iff $\bigwedge_{k=1,\ldots,5} R_k$ is true

Inference: Objectives & Application

- Given a KB, infer something non-obvious about the environment
- Mimic logical human reasoning

- After exploring 3 squares, we have some understanding of the Wumpus World

- Inference ⇒ Deriving knowledge out of percepts

Given KB and α, we want to know if KB $\vDash \alpha$	
What α?	Based on domain: e.g., is (1,2) safe?

Properties of Inference Algorithms

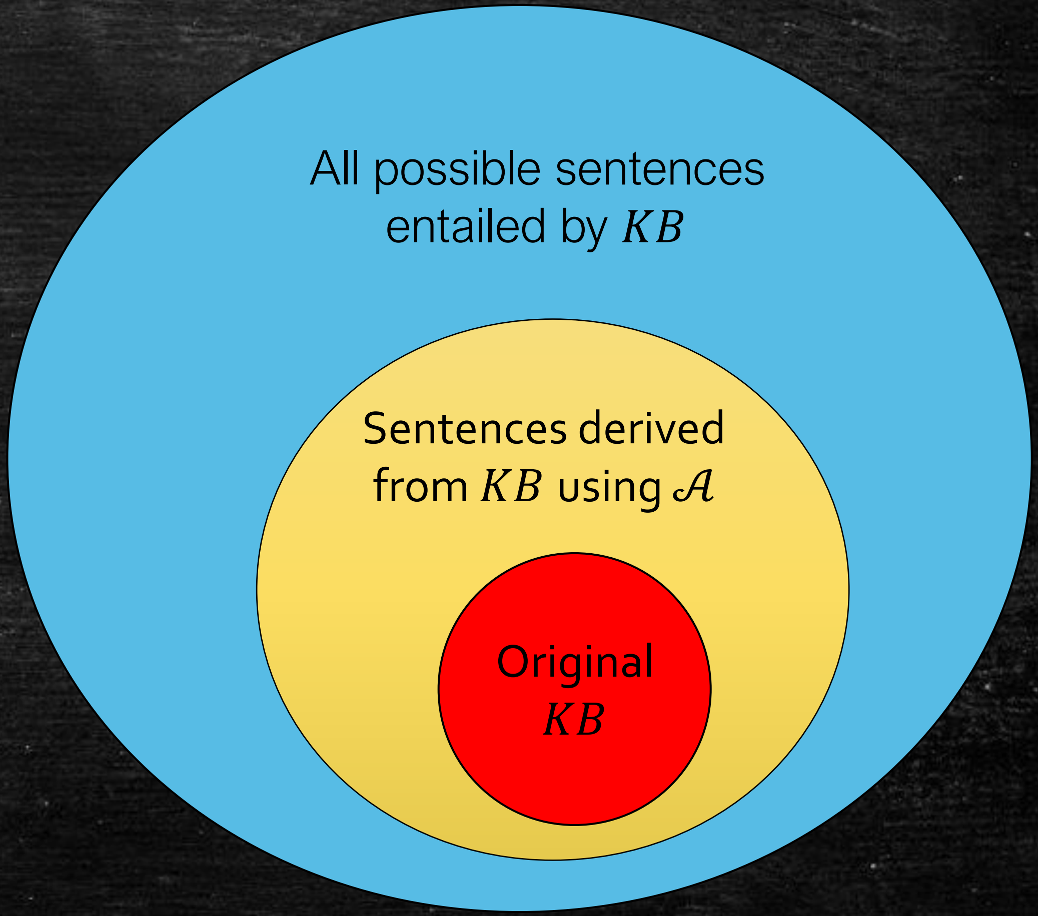
Soundness & Completeness

- KB $\vdash_{\mathcal{A}} \alpha$
  - Means: "sentence α is derived (i.e., inferred) from KB by inference algorithm $\mathcal{A}$"
- Soundness
  - $\mathcal{A}$ is sound if KB $\vdash_{\mathcal{A}} \alpha$ implies KB $\vDash \alpha$
  - This means that $\mathcal{A}$ will not infer nonsense
    - For all sentences inferred from the KB by $\mathcal{A}$, S
    - The KB will entail each α in S
- Completeness
  - $\mathcal{A}$ is complete if KB $\vDash \alpha$ implies KB $\vdash_{\mathcal{A}} \alpha$
- This means that $\mathcal{A}$ can infer any sentence that the KB entails
  - If KB entails a sentence (any sentence describing a superset of the KB)
  - $\mathcal{A}$ can infer that sentence

Determine if an inference algorithm is complete and sound

Soundness & Completeness

- More on completeness
  - If $\mathcal{A}$ is incomplete
  - $\mathcal{A}$ cannot reach all possible conclusions
- Given
  - Y = all possible sentences entailed by KB
  - X = all sentences derived from KB using $\mathcal{A}$
- Then
  - X = Y: sound, complete
  - X ⊂ Y: sound, not complete
  - Y ⊂ X: not sound, complete
  - X ⊄ Y, Y ⊄ X, X ≠ Y: not sound, not complete



Truth Table Enumeration

Truth Table Enumeration Example: Wumpus World

Can we infer that (1,2) is safe from pits?

$\alpha_1 = \neg P_{1,2}$

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*false*

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

$R_2: \neg B_{1,1}$

$R_3: B_{2,1}$

$R_4: \neg B_{1,1} \Leftrightarrow \neg(P_{1,2} \vee P_{2,1})$

$R_5: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

KB true

KB is true
iff $\wedge_{k=1,\ldots,5} R_k$ is true

Recall that a truth table contains every possible truth assignment (2^7 models in this example)

Does KB entail α_1? (Whenever KB true, α_1 true?)

Truth Table Enumeration

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

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

function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
   if EMPTY?(symbols) then
      if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
      else return true        // when KB is false, always return true
   else
      P ← FIRST(symbols)
      rest ← REST(symbols)
      return (TT-CHECK-ALL(KB, α, rest, model ∪ {P = true})
              and
              TT-CHECK-ALL(KB, α, rest, model ∪ {P = false }))
```

- Checks all 2^n truth assignments to verify KB entails α
- Depth-first enumeration
- Recursive step generates the 2^n possible assignments to the n symbols
- $O(2^n)$ time complexity
  $O(n)$ space complexity
- Implements definition of entailment directly
  (guarantees soundness)
- Finite models to check
  (guarantees completeness)

Questions about the Lecture?

- Was anything unclear?
- Do you need to clarify anything?

- Ask on Archipelago
  - Specify a question
  - Upvote someone else's question

Invitation Link (Use NUS Email --- starts with E)
https://archipelago.rocks/app/resend-invite/12166893023