Logical Agents: Knowledge Representation I

CS3243: Introduction to Artificial Intelligence – Lecture 8

NUS | Computing
National University of Singapore

1 Administrative Matters

Upcoming...

- Deadlines
  - Tutorial Assignment 6 (released last week)
    - Post-tutorial submission: Due this Friday!
  - Tutorial Assignment 7 (released today)
    - Pre-tutorial Submission: Due this Sunday!
    - Post-tutorial Submission: Due next Friday!
  - Project 2.2 (released 16 September)
    - Due THIS Sunday (i.e., Week 9)!

      20 October 2024
  - Project 3 (released 14 October)
    - Due in Week 12 (10 November)

Late Penalties
- < deadline +24 hours = 80% of score
- < deadline +48 hours = 50% of score
- ≥ deadline +48 hours = 0% of score

Project Consultations:
Thursdays (1900 hrs) via Zoom
Canvas > Announcements

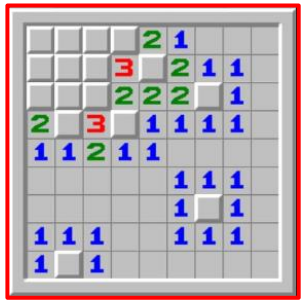
Contents

- Knowledge-Based Agents
- Wumpus World Example
- Knowledge Inference and Entailment
- Soundness and Completeness of Inference Algorithms
- Inference via Truth Table Enumeration

2 Knowledge-Based Agents: Logical 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
      - e.g., route-finding agent – implicit knowledge that road lengths cannot be negative
      - e.g., n-puzzle agent – implicit knowledge that two number-tiles cannot occupy the same grid
  - Atomicity of representations limiting
    - Imagine a game of minesweeper where the environment is only partially observable; the agent would not know where all the mines actually are
    - 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 (KR): Knowledge-Based (KB) Agents

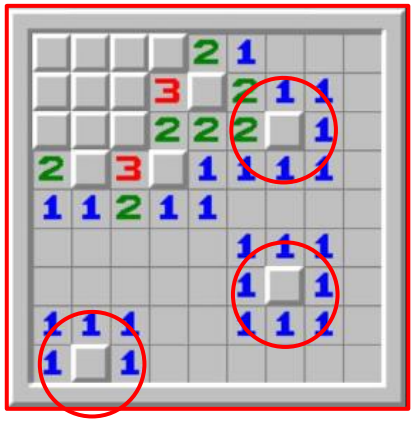
Knowledge-Based Agents

- Represent agent domain knowledge using logical formulas
  - Main components of a logical agent

Inference Engine ← Domain-independent algorithms

Knowledge Base ← Domain-specific content

  - General idea
    - Initialise knowledge base (KB)
    - Make inferences on existing information (i.e., KB)
    - Use existing knowledge to infer new information
    - Use inferred knowledge to determine actions
      - Implies that possible actions are contingent on knowledge
      - KB will also be updated with inferred knowledge



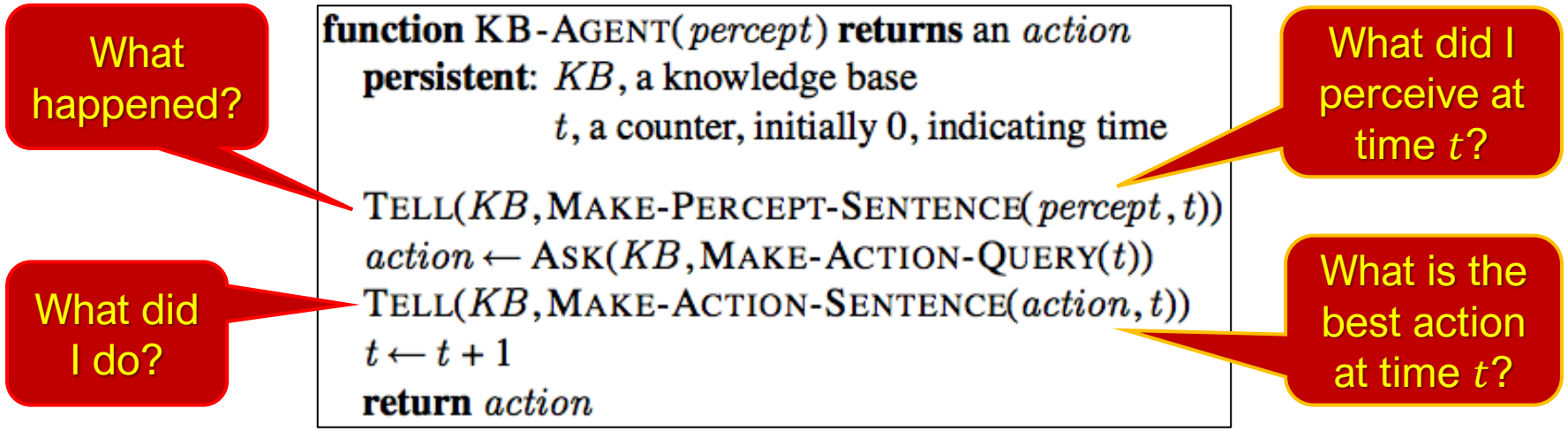
e.g., to determine actions in Minesweeper:

- Use rules about the game as KB (e.g., if number of covered cells adjacent to cell C == number in cell C, then cell C must contain a mine)
- Infer new information from KB (e.g., those isolated cells with 1s surrounding them contain mines; any others?)
- Take actions using inferred knowledge (e.g., mark the cells identified above as mines)

The Knowledge Base (KB)

- What is a knowledge base (KB)?
  - Set of sentences in a formal language
    - Sentences are expressive and can be parsed
  - Pre-populate with domain knowledge
    - Example: game rules, general rules/knowledge
- Declarative approach to problem-solving
  - TELL agent 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



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

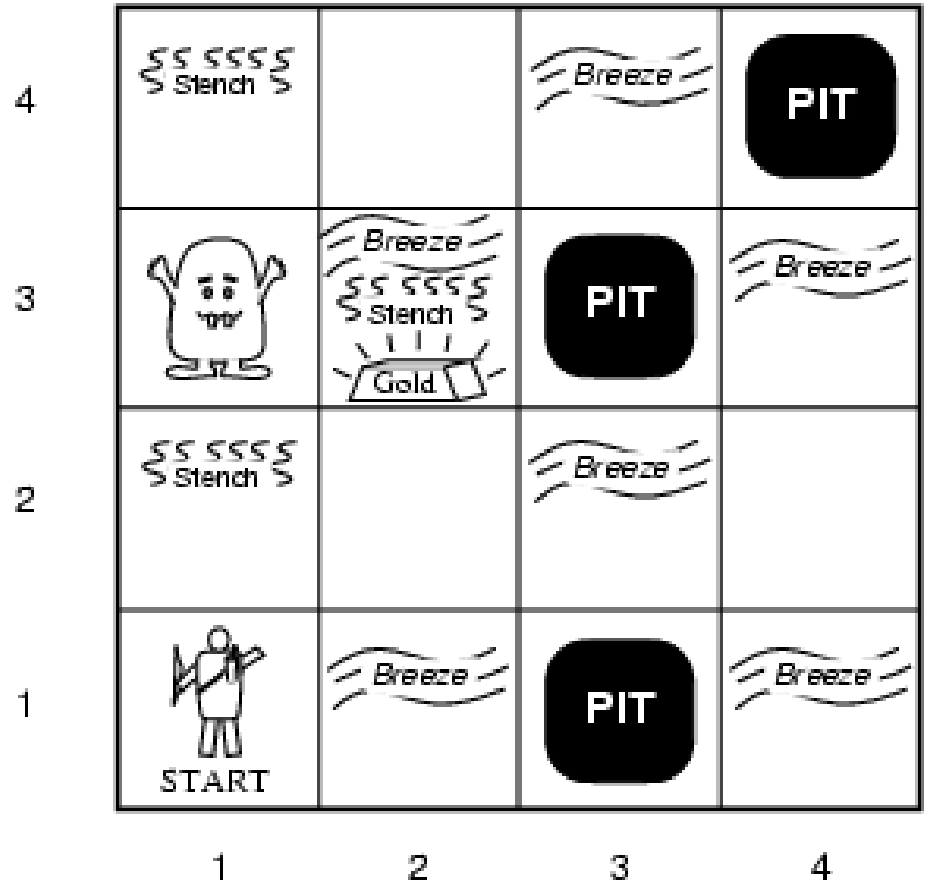
No longer atomic states!

Requires mapping from desired environment properties (i.e., variables) to actions!

3 An Example: The Wumpus Dungeon

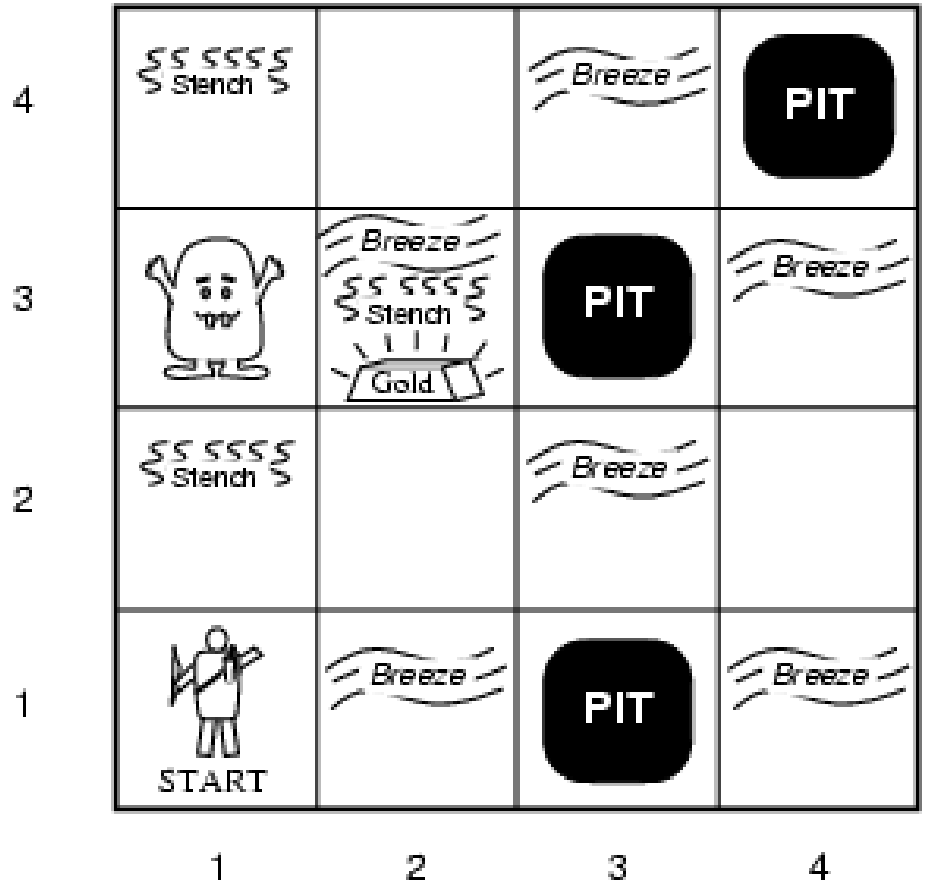
About Wumpus World

- Wumpus World Environment
  - Wumpus world is very dark - agent is unable to see/sense beyond the room it is currently
    - i.e., partially observable environment
  - Agent can change orientation
    - i.e., can face: {`N`, `S`, `E`, `W`}
  - Agent can move forward
  - In each room, the agent can sense: `STENCH`, `BREEZE`, `GLITTER`
    - Rooms adjacent to a `PIT` will contain a `BREEZE`
    - Rooms adjacent to `WUMPUS` contain a `STENCH`
    - Agent detects `GLITTER` if in room with `GOLD`
  - Agent is killed when walking into a room with a `PIT` or `WUMPUS`
  - Agent has `ONE` arrow it can fire in the direction faced
    - The arrow kills the `WUMPUS` if it is present in the direction fired
  - Goal: Find the gold and leave via the exit without getting killed



Properties of Wumpus World

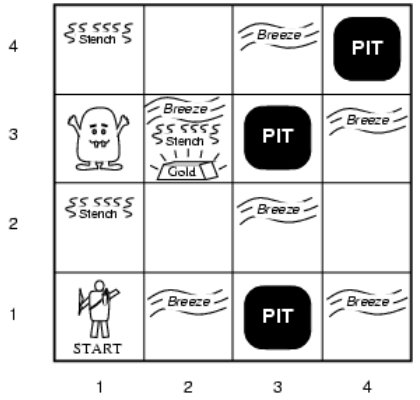
- Not fully observable
  - Only local perception
  - Don't know what is in unexplored rooms
- Deterministic
- Sequential
- Static
- Discrete
- Single Agent



Exploring the Wumpus Dungeon

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2	2,2	3,2	4,2
1,1	2,1	3,1	4,1

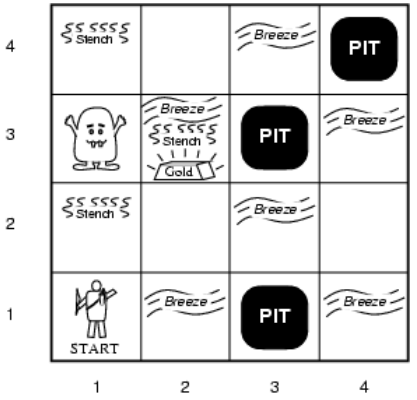
A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe Square
P = Pit
S = Stench
V = Visited
W = Wumpus



Exploring the Wumpus Dungeon

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

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



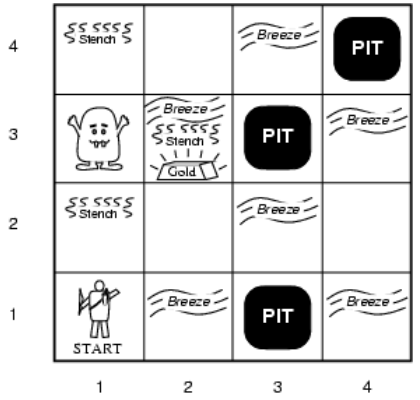
STATE 0 – Read Percepts

- Current Position: `(1,1)` // Initial position is safe
- `¬BREEZE(1,1) ∧ ¬STENCH(1,1)`

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

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



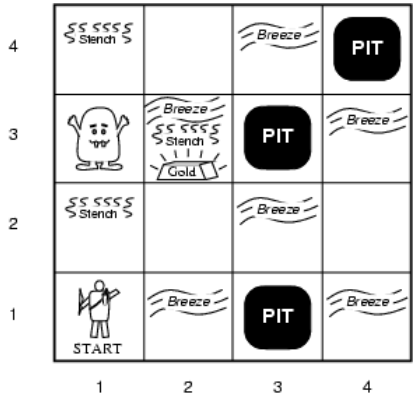
STATE 0 – Make Inferences

- Current Position: `(1,1)` // Initial position is safe
- `¬BREEZE(1,1) ∧ ¬STENCH(1,1)`
  `⇒ ¬PIT(1,2) ∧ ¬WUMPUS(1,2)`
  `⇒ ¬PIT(2,1) ∧ ¬WUMPUS(2,1)`

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

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



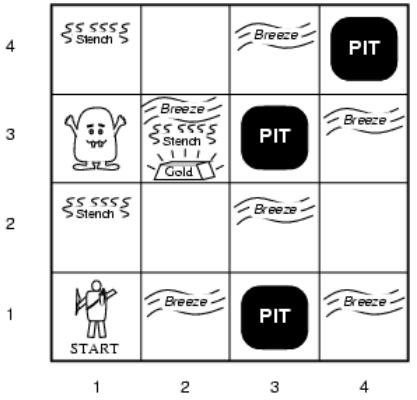
STATE 0 – Decide and Execute Next Action(s)

- Current Position: `(1,1)` // Initial position is safe
- `¬BREEZE(1,1) ∧ ¬STENCH(1,1)`
  `⇒ ¬PIT(1,2) ∧ ¬WUMPUS(1,2)`
  `⇒ ¬PIT(2,1) ∧ ¬WUMPUS(2,1)`
- `MOVE` to `(2,1)`

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	3,2	4,2
1,1 **OK** **V**	2,1 **A** **B** **OK**	3,1	4,1

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



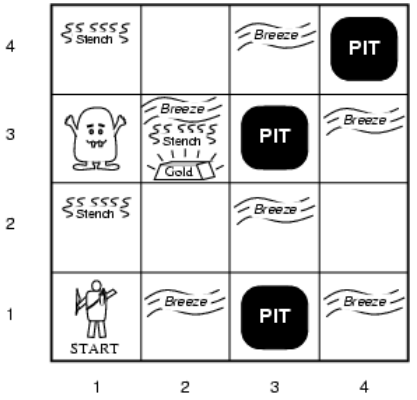
STATE 1 – Read Percepts

- Current Position: `(2,1)`
- `BREEZE(2,1) ∧ ¬STENCH(2,1)`

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 **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



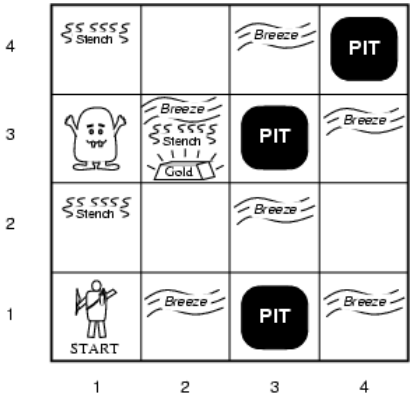
STATE 1 – Make Inferences

- Current Position: `(2,1)`
- `BREEZE(2,1) ∧ ¬STENCH(2,1)`
  `⇒ ¬WUMPUS(2,2) ∧ ¬WUMPUS(3,1)`
  `⇒ PIT(2,2) ∨ PIT(3,1)`

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 **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



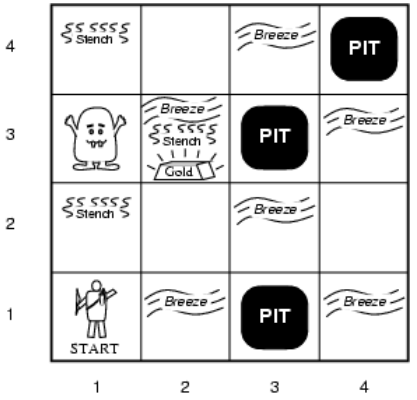
STATE 1 – Decide and Execute Next Action(s)

- Current Position: `(2,1)`
- `BREEZE(2,1) ∧ ¬STENCH(2,1)`
  `⇒ ¬WUMPUS(2,2) ∧ ¬WUMPUS(3,1)`
  `⇒ PIT(2,2) ∨ PIT(3,1)`
- `MOVE` back to `(1,1)` and then to `(1,2)`

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 **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



STATE 2 – Read Percepts

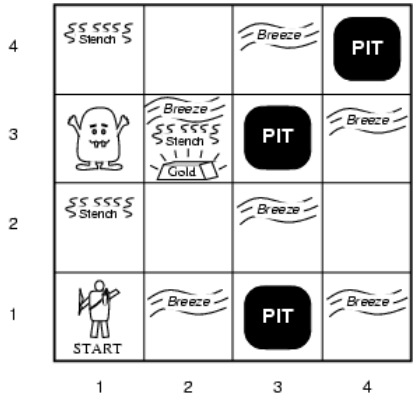
- Current Position: `(1,2)`
- `¬BREEZE(1,2) ∧ STENCH(1,2)`

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

¬BREEZE(1,2)

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



STATE 2 – Make Inferences

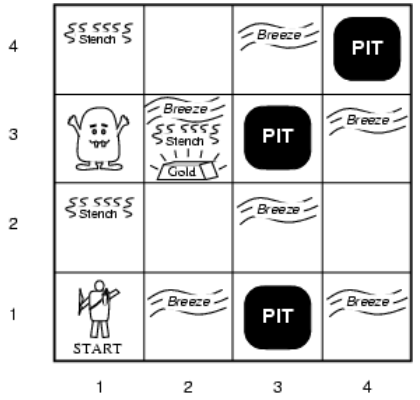
- Current Position: (1,2)
- ¬BREEZE(1,2) ∧ STENCH(1,2)

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

¬BREEZE(1,2)

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



STATE 2 – Make Inferences

- Current Position: (1,2)
- ¬BREEZE(1,2) ∧ STENCH(1,2) ⇒ ¬PIT(2,2)

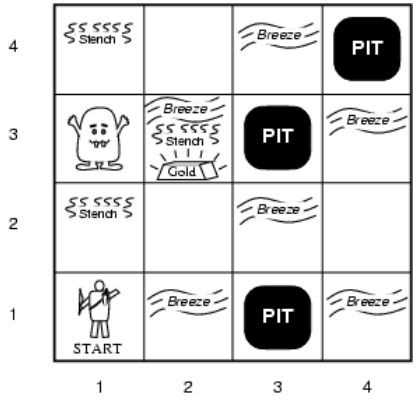
Exploring the Wumpus Dungeon

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

¬PIT(2,2)

⇒ PIT(2,2) V PIT(3,1)

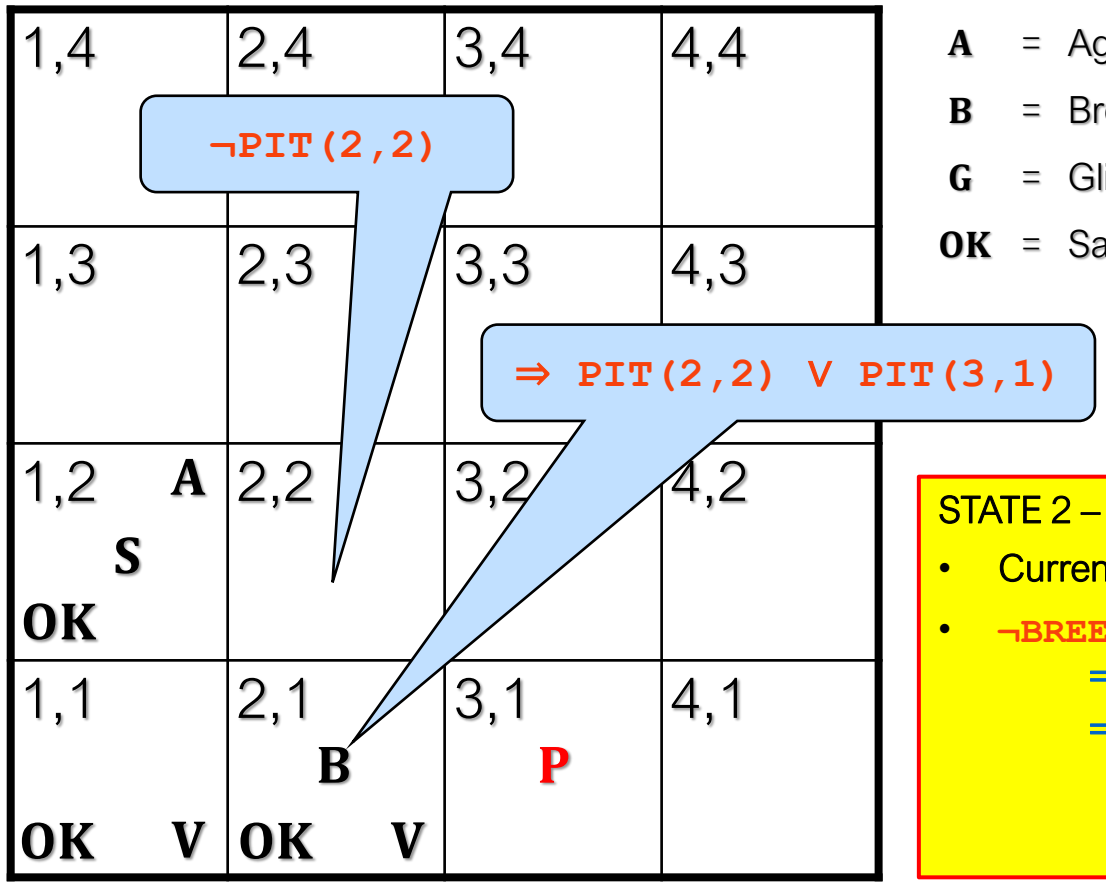
A	=	Agent	**P**	=	Pit
B	=	Breeze	**S**	=	Stench
G	=	Glitter, Gold	**V**	=	Visited
OK	=	Safe Square	**W**	=	Wumpus



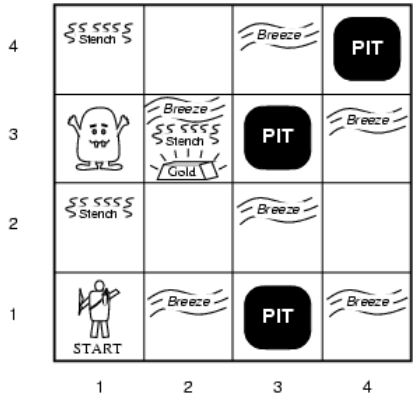
STATE 2 – Make Inferences

- Current Position: (1,2)
- ¬BREEZE(1,2) ∧ STENCH(1,2)
  ⇒ ¬PIT(2,2)

Exploring the Wumpus Dungeon



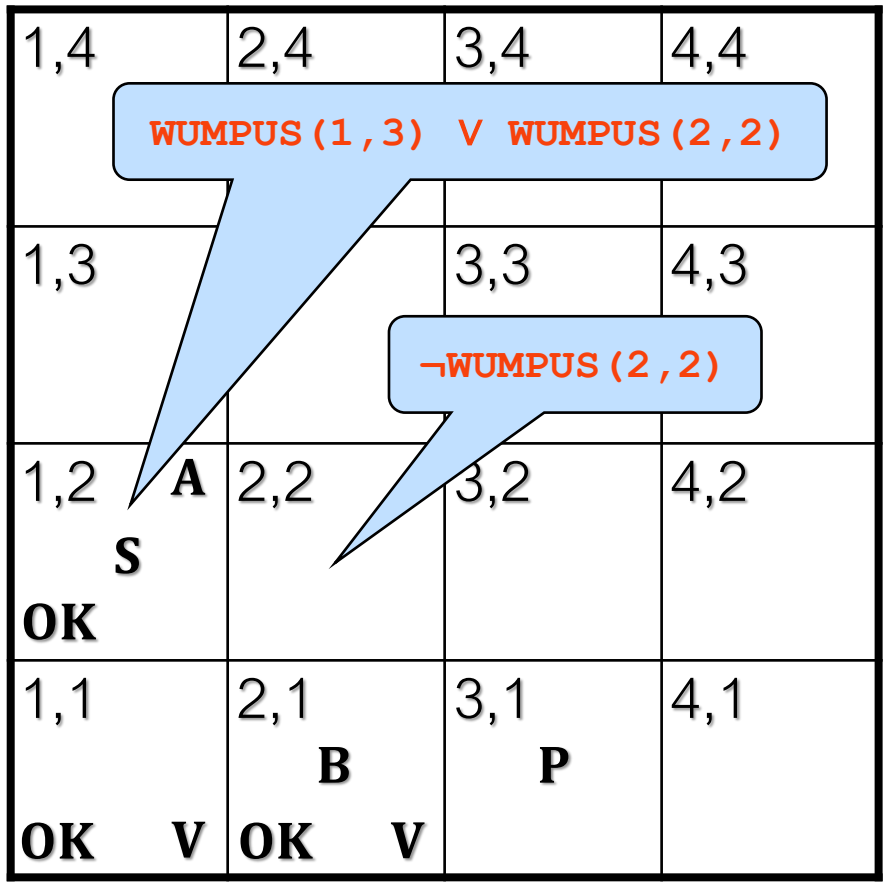
A	=	Agent	**P**	=	Pit
B	=	Breeze	**S**	=	Stench
G	=	Glitter, Gold	**V**	=	Visited
OK	=	Safe Square	**W**	=	Wumpus



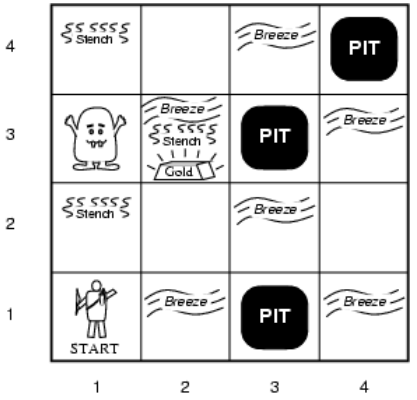
STATE 2 – Make Inferences

- Current Position: `(1,2)`
- `¬BREEZE(1,2) ∧ STENCH(1,2)`
  `⇒ ¬PIT(2,2)`
  `⇒ PIT(3,1)`

Exploring the Wumpus Dungeon



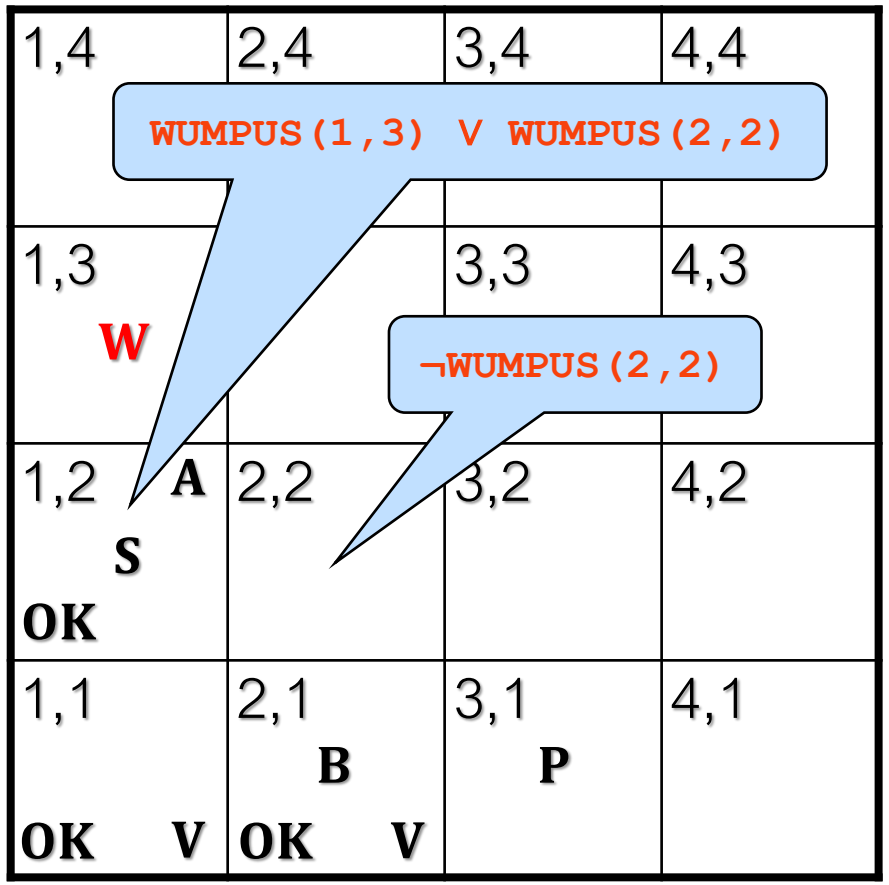
A	=	Agent	**P**	=	Pit
B	=	Breeze	**S**	=	Stench
G	=	Glitter, Gold	**V**	=	Visited
OK	=	Safe Square	**W**	=	Wumpus



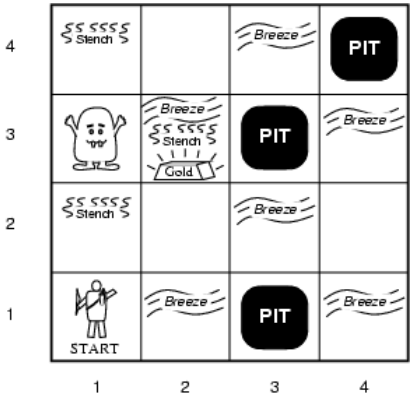
STATE 2 – Make Inferences

- Current Position: `(1,2)`
- `¬BREEZE(1,2) ∧ STENCH(1,2)`
  `⇒ ¬PIT(2,2)`
  `⇒ PIT(3,1)`

Exploring the Wumpus Dungeon



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



STATE 2 – Make Inferences

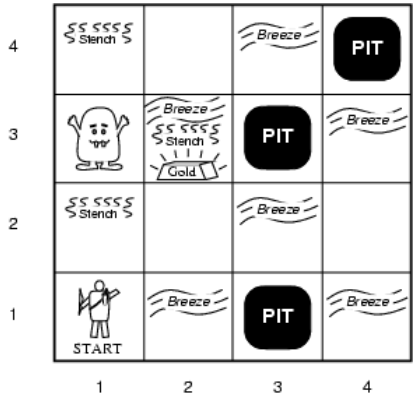
- Current Position: `(1,2)`
- `¬BREEZE(1,2) ∧ STENCH(1,2)`
  `⇒ ¬PIT(2,2)`
  `⇒ PIT(3,1)`
  `⇒ WUMPUS(1,3)`

Exploring the Wumpus Dungeon

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 **OK** **V**	2,1 **B** **OK** **V**	3,1 **P**	4,1

¬PIT(2,2) V ¬WUMPUS(2,2)

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



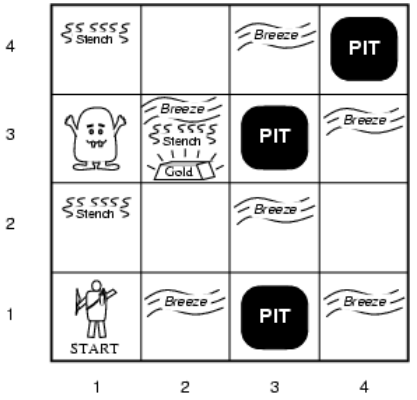
STATE 2 – Make Inferences

- Current Position: (1,2)
- ¬BREEZE(1,2) ∧ STENCH(1,2)
  - ⇒ ¬PIT(2,2)
  - ⇒ PIT(3,1)
  - ⇒ WUMPUS(1,3)

Exploring the Wumpus Dungeon

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 **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



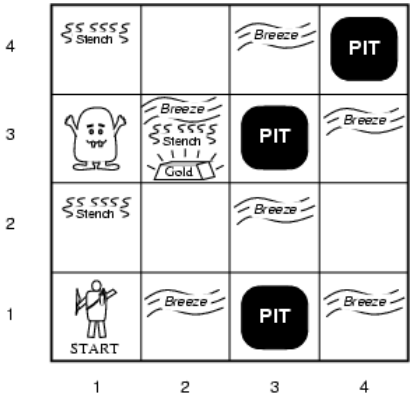
STATE 2 – Decide and Execute Next Action(s)

- Current Position: `(1,2)`
- `¬BREEZE(1,2) ∧ STENCH(1,2)`
  `⇒ ¬PIT(2,2)`
  `⇒ PIT(3,1)`
  `⇒ WUMPUS(1,3)`
- `MOVE` to `(2,2)`

Exploring the Wumpus Dungeon

1,4	2,4	3,4	4,4
1,3 **W**	2,3	3,3	4,3
1,2 **S** **OK** **V**	2,2 **A** **OK**	3,2	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



STATE 3 – Read Percepts

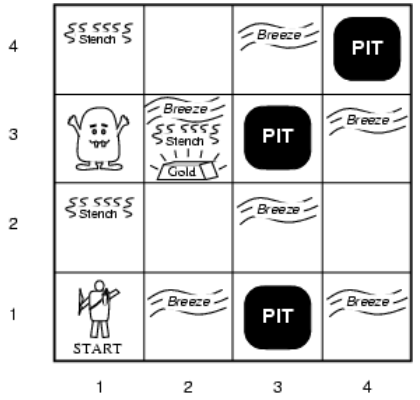
- Current Position: `(2,2)`
- `¬BREEZE(2,2) ∧ ¬STENCH(2,2)`

Exploring the Wumpus Dungeon

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

¬BREEZE(2,2) ∧ ¬STENCH(2,2)

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



STATE 3 – Make Inferences

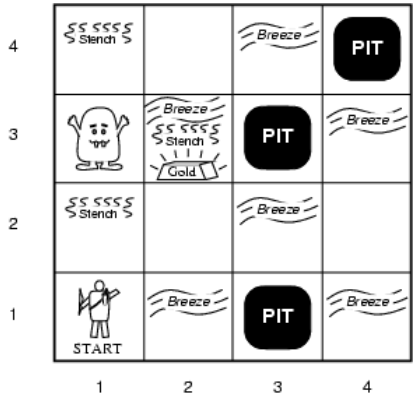
- Current Position: (2,2)
- ¬BREEZE(2,2) ∧ ¬STENCH(2,2)

Exploring the Wumpus Dungeon

1,4	2,4	3,4	4,4
1,3 **W**	2,3 **OK**	,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

¬BREEZE(2,2) ∧ ¬STENCH(2,2)

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



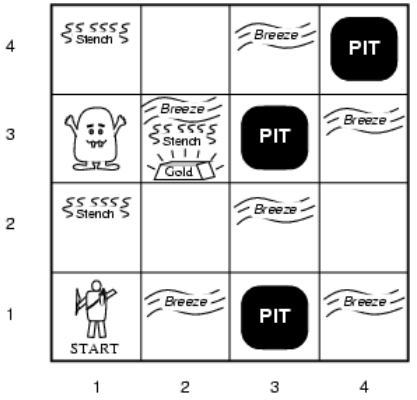
STATE 3 – Make Inferences

- Current Position: (2,2)
- ¬BREEZE(2,2) ∧ ¬STENCH(2,2)
  ⇒ ¬PIT(2,3) ∧ ¬WUMPUS(2,3)
  ⇒ ¬PIT(3,2) ∧ ¬WUMPUS(3,2)

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



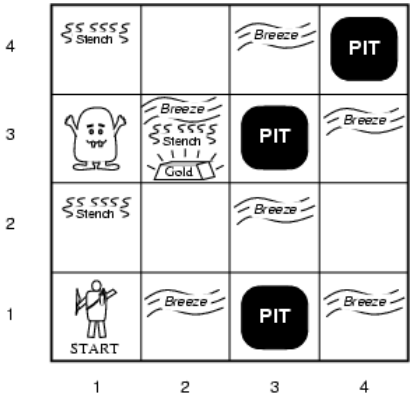
STATE 3 – Decide and Execute Next Action(s)

- Current Position: `(2,2)`
- `¬BREEZE(2,2) ∧ ¬STENCH(2,2)`
  `⇒ ¬PIT(2,3) ∧ ¬WUMPUS(2,3)`
  `⇒ ¬PIT(3,2) ∧ ¬WUMPUS(3,2)`
- `MOVE` to `(2,3)`

Exploring the Wumpus Dungeon

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



STATE 4 – Read Percepts; Make Inferences; Decide and Execute Next Action(s)

- Current Position: `(2,3)`
- `BREEZE(2,2) ∧ STENCH(2,3) ∧ GLITTER(2,3)`
  `⇒ GOLD(2,3)`
- `TAKE GOLD` then exit cave via `(2,2)`, `(2,1)`, `(1,1)`

4 | Defining the Knowledge Base: Knowledge Representation via Logic

Knowledge Representation & Logic

- Logic
  - Formal language for knowledge representation (KR)
  - Allows us to define the rules or laws that govern an environment
  - Allows the inference of conclusions about environment
- Syntax
  - Defines sentences in the language
- Semantics
  - Defines meaning of sentences
- Truth value
  - Sentence result (true/false outcome) given observed values
    - Defines validity of a sentence within the defined environment based on value assignments
    - i.e., determines if given value assignments will be consistent with defined environment

We need semantics (environment variables) to make sense of rules in the KB and thus determine variable-action mappings

Knowledge Representation & Logic

- Example of KR: language of arithmetic
  - Syntax
    - $x + 2 \geq y$ is a valid sentence
    - $x2y + >$ is not a valid sentence
  - Truth values
    - $x+2 \geq y$ is `true` in a world (i.e., set of value assignments) where $x=7$ and $y=1$
    - $x+2 \geq y$ is `false` in a world (i.e., set of value assignments) where $x=0$ and $y=6$

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 possible model
  - Truth assignment to the given Boolean variables
  - Given n variables, 2^n possible truth assignments
- All other sentences' truth value are derived according to logical rules
  - For example:
    - Given $x_1 = \text{True}\ x_2 = \text{False}\ x_3 = \text{True}$
    - 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 $X \Rightarrow Y$ is true if X false, or X true and Y true

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/11468403214

5 Inference via Entailment

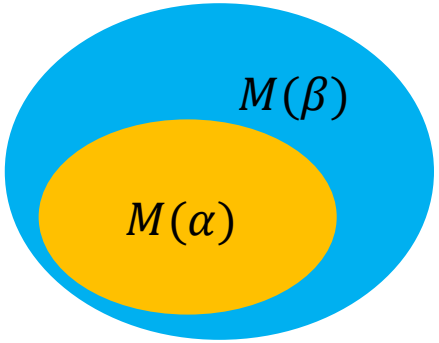
Inference

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

- Inference ⇒ deriving new knowledge from the KB
  - KB: environment rules/laws and percepts
- Prove that new statements are true (given the state of the environment)

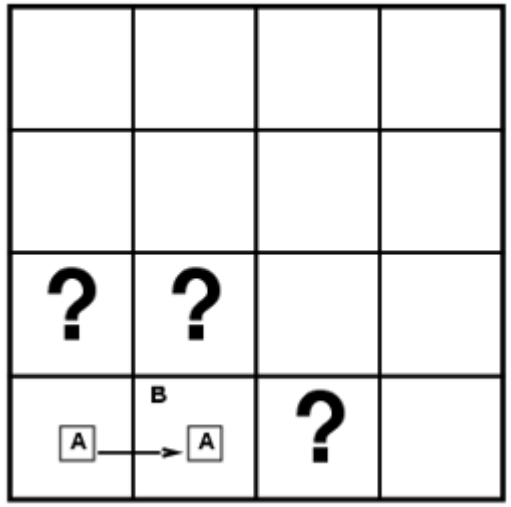
Entailment

- Modelling
  - v models α if α is true under v
    - v corresponds to one set of value assignments (applied to sentence(s) 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 ($\models$) means that one thing (right) follows from the other (left)
  - $\alpha \models \beta$ or equivalently $M(\alpha) \subseteq M(\beta)$
  - Continuing example above:
    - $[\alpha = (q \text{ is prime})] \models [\beta = (q \text{ is odd}) \vee (q = 2)]$



Entailment in the Wumpus World

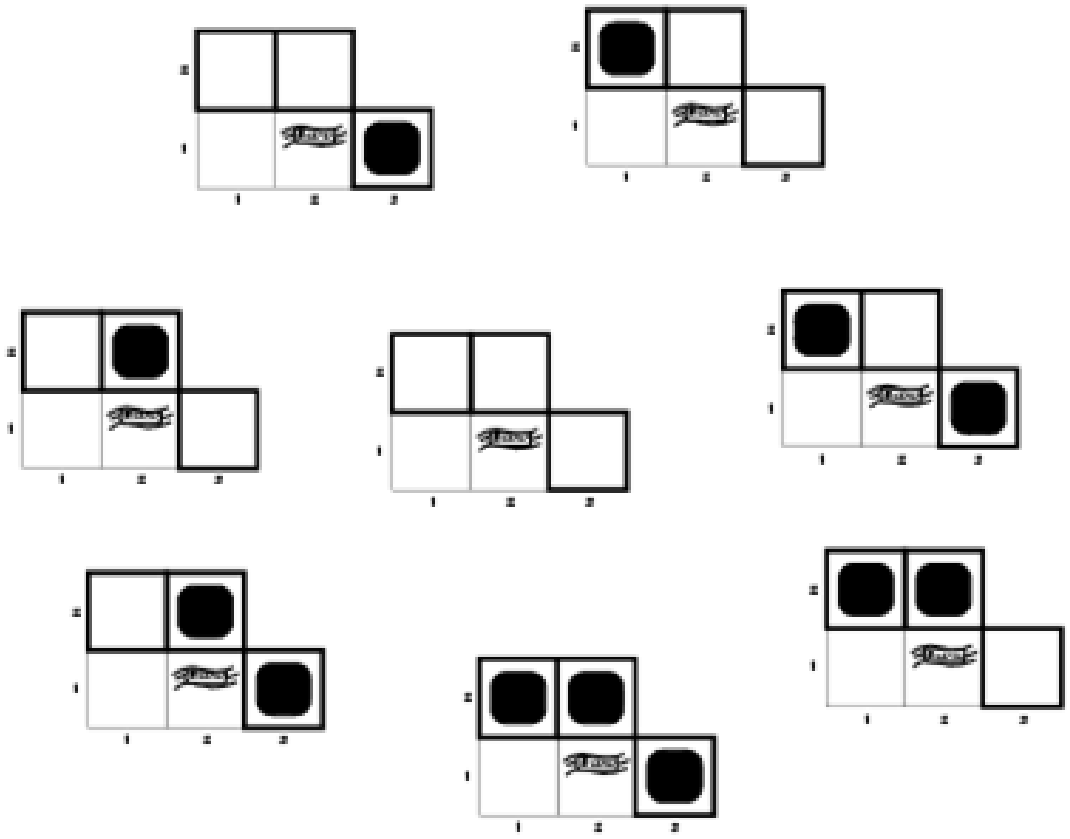
- Situation:
  - Detected Nothing at `(1,1)`
  - Moved Right to `(2,1)`
  - Detected Breeze at `(2,1)`
- Consider possible models for KB with pits
  - 3 Boolean choices ⇒ 8 possible models
    - `PIT` or `¬PIT` at: `(1,2)`, `(2,2)`, `(3,1)`
    - All ($2^3 = 8$) permutations for the above (each a possible model)



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

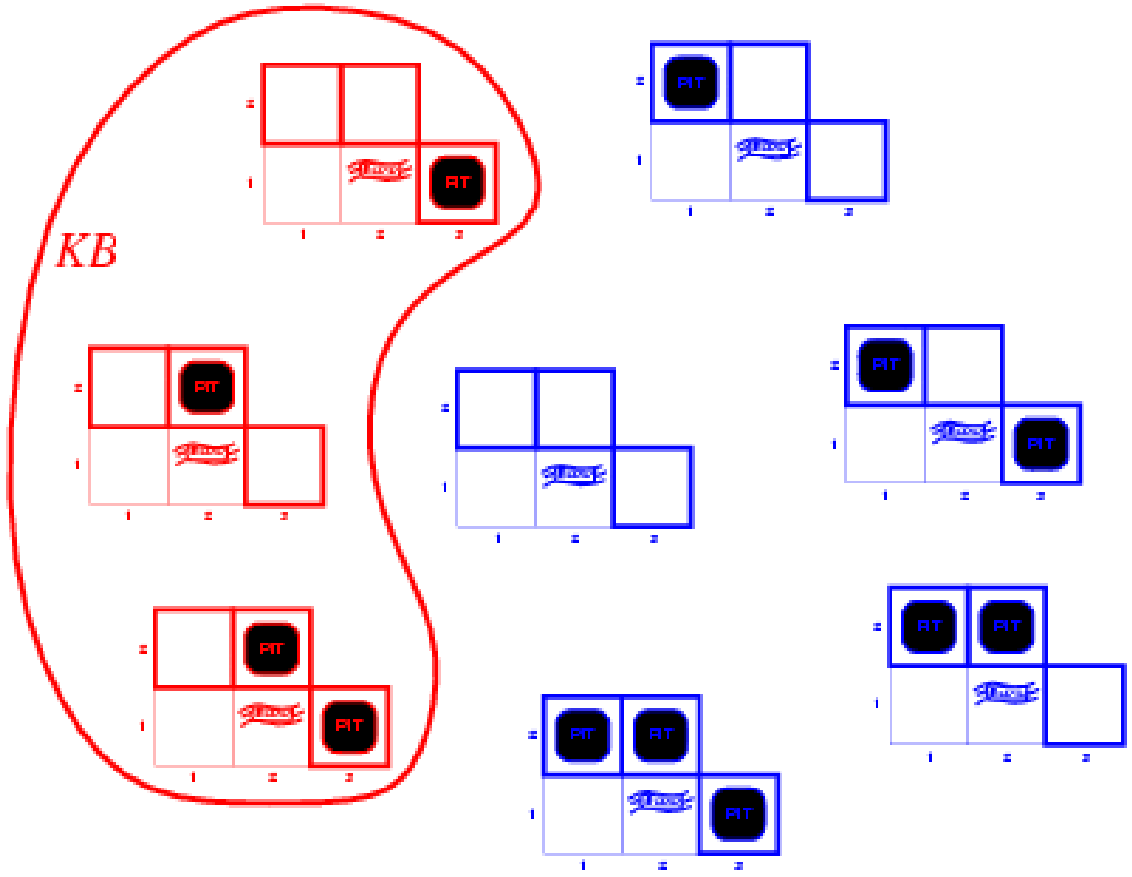
Entailment in the Wumpus World

- Possible 8 models



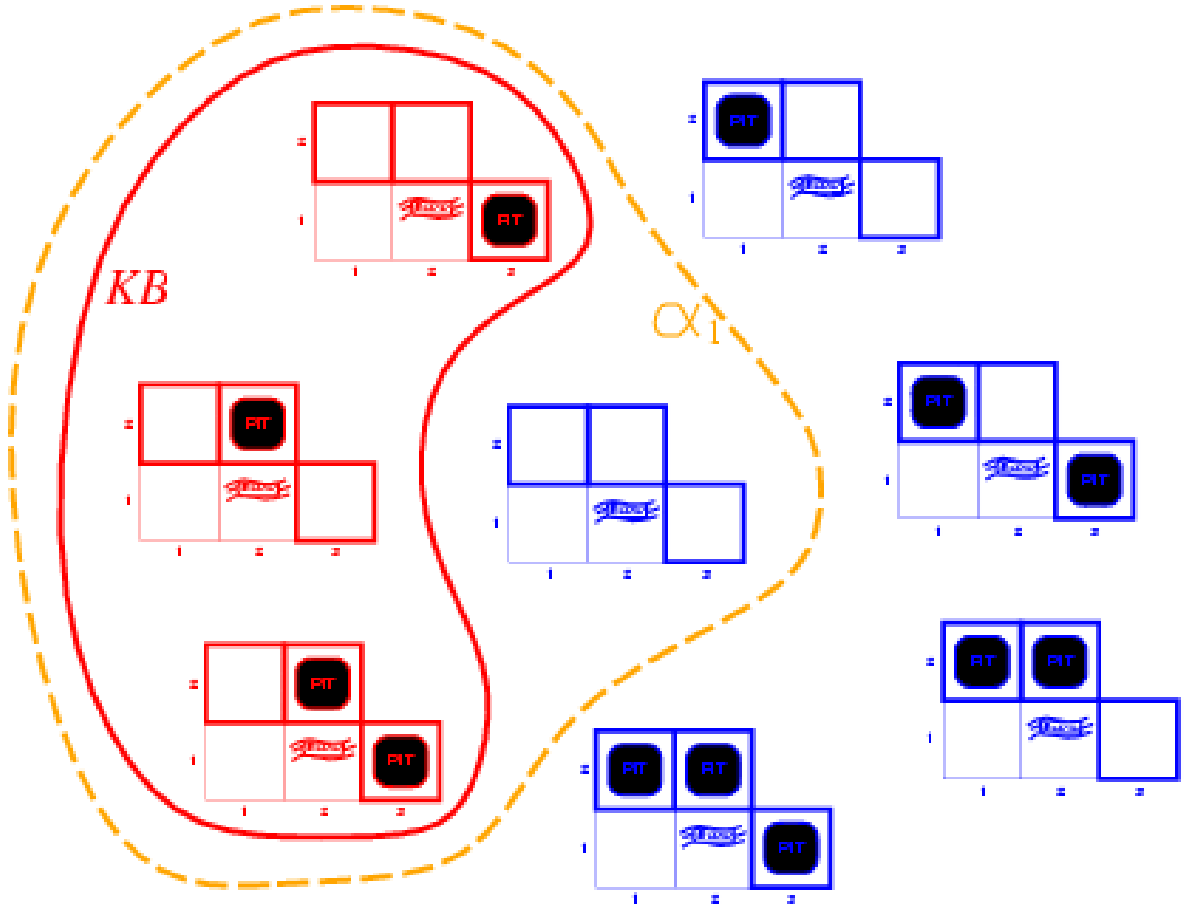
Entailment in the Wumpus World

- KB = rules + percepts
- Percepts
  - ¬PIT at (1,1)
  - ¬BREEZE at: (1,1)
  - ¬PIT at (2,1)
  - BREEZE at (2,1)
- Relevant rules
  - BREEZE ⇒ PIT in adjacent room
    - ¬PIT at (1,2)
    - ¬PIT at (2,1)
    - PIT at (1,1) ∨
      PIT at (2,2) ∨
      PIT at (3,2)



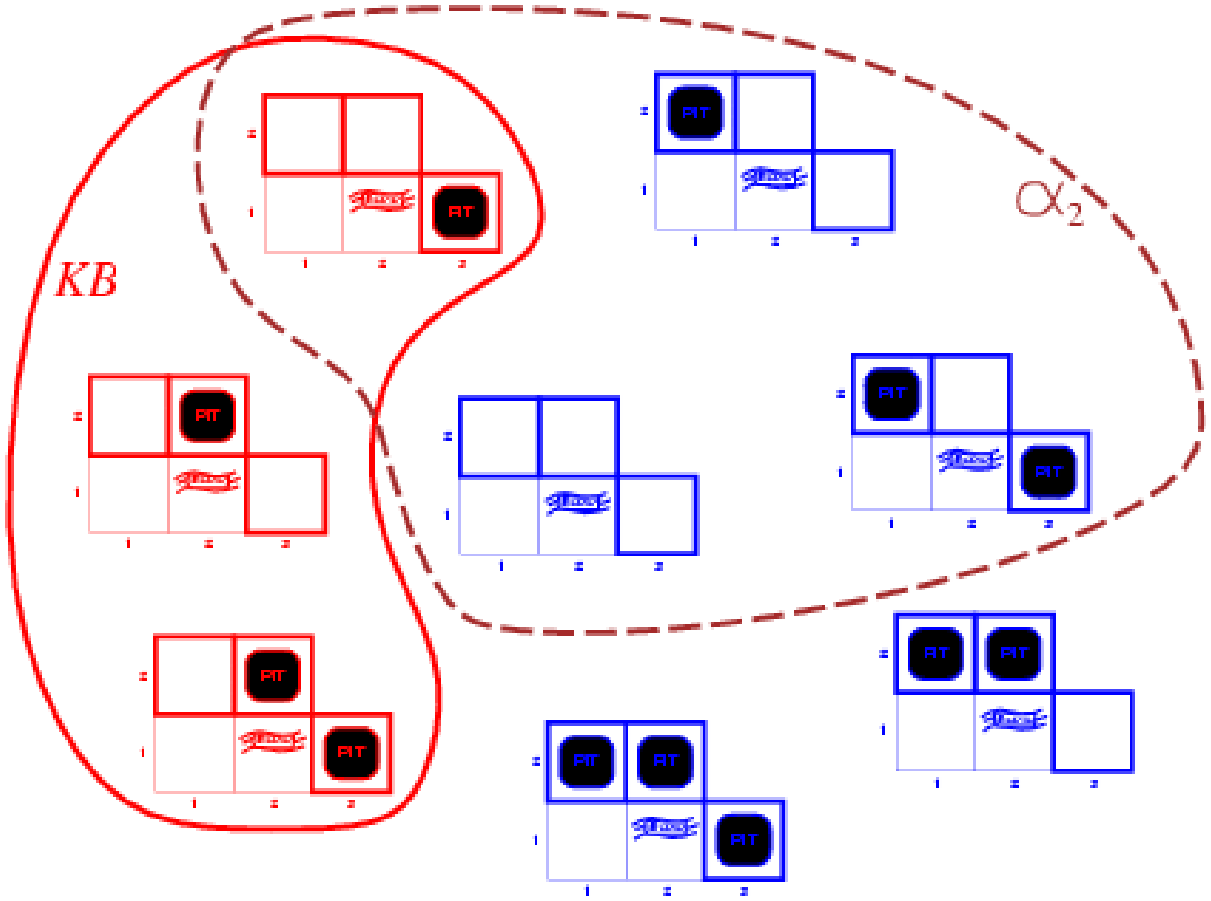
Entailment in the Wumpus World

- Let α_1 = `(1,2)` is Safe
  = ¬`PIT` at `(1,2)`
- We observe that:
  $M(KB) \subseteq M(\alpha_1)$
- Or rather:
  $KB \models \alpha_1$
- We may thus infer that it is safe for the agent to move to `(1,2)`



Entailment in the Wumpus World

- Let α_2 = `(2,2)` is Safe
  = ¬`PIT` at `(2,2)`
- We observe that: $KB \models \alpha_2$
- Since: $M(KB) \subseteq M(\alpha_2)$
- We may **NOT** infer that it is safe for the to move to `(2,2)`
  - i.e., there exist some models where KB is True but α_2 is False
  - For entailment, we want all α_2 True when KB True
- Note: we also cannot infer that `(2,2)` is unsafe!



Inference

- Inference ⇒ deriving new knowledge from the KB
  - `KB`: environment rules/laws and percepts

- From the previous (Wumpus) example
  - After exploring 2 squares, we have some understanding of the Wumpus World (current `KB`)
  - Prove that some query α is true (given the state of the environment; i.e., given the `KB`)

Given `KB` and α, we want to know if `KB` $\models \alpha$

What α?

Based on domain: e.g., is `(1,2)` safe?

6 Properties of Inference Algorithms

Soundness & Completeness

- KB $\vdash_{\mathcal{A}} \alpha$
  - Means: "sentence α is derived (i.e., inferred) from KB by inference algorithm $\mathcal{A}$"

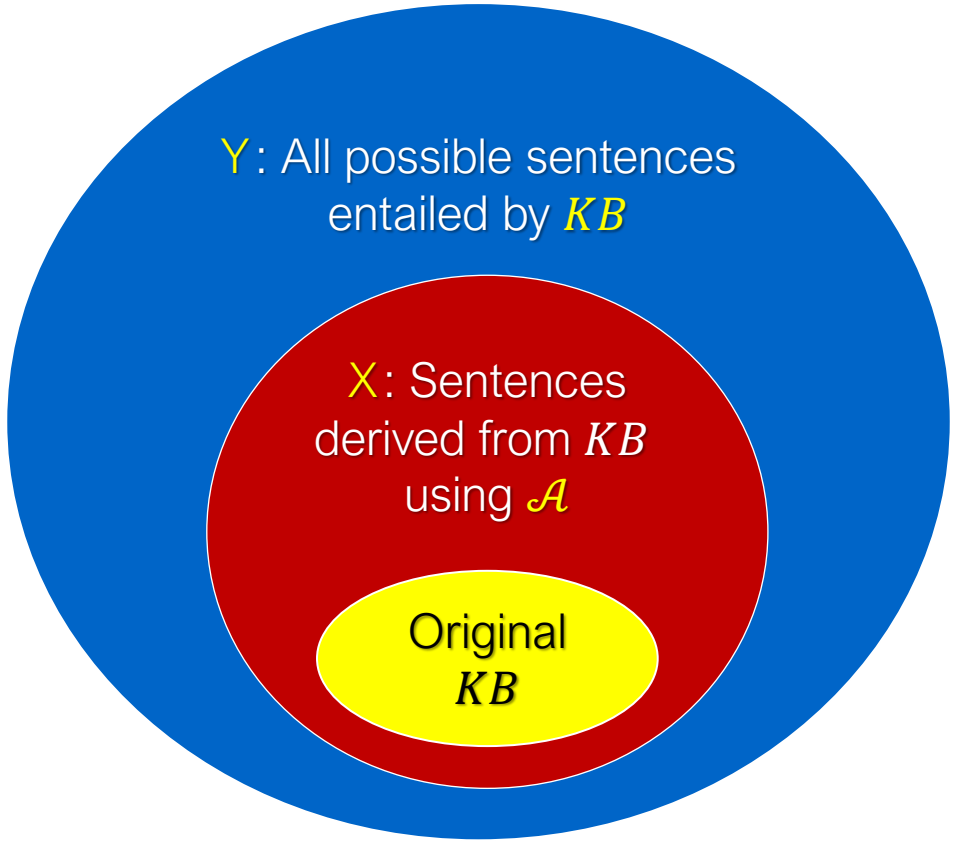
- Soundness
  - $\mathcal{A}$ is sound if KB $\vdash_{\mathcal{A}} \alpha$ implies KB $\models \alpha$
  - This means that $\mathcal{A}$ will not infer nonsense
    - For all sentences inferred from the KB by $\mathcal{A}$, S
    - The KB will entail each α in S

- Completeness
  - $\mathcal{A}$ is complete if KB $\models \alpha$ 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

Must determine if an inference algorithm is both complete and sound

Soundness & Completeness

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



7 Truth Table Enumeration

Entailment in the Wumpus World

- Notation
  - P_{ij} = True ⇔ Pit at (i, j)
  - B_{ij} = True ⇔ Breeze at (i, j)
- Given
  - $R_1: \neg P_{1,1}$
  - $R_2: \neg B_{1,1}$
  - $R_3: \neg P_{2,1}$
  - $R_4: B_{2,1}$
- Rules: "Pits cause a breeze in adjacent squares"
  - $R_5: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ – i.e., $\neg B_{1,1} \Leftrightarrow \neg(P_{1,2} \vee P_{2,1})$
  - $R_6: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

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

KB is true iff $\wedge_{k=1,\ldots,6} R_k$ is true

Truth Table Enumeration Example: Wumpus World

Can we infer that (1,2) is safe from pits? | $\alpha_1 = \neg P_{1,2}$ | Does KB entail α_1? (Whenever KB true, α_1 true?)

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

KB

KB is true
iff $\wedge_{k=1,\ldots,6} R_k$ is true

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

Truth Table Enumeration Example: Wumpus World

Can we infer that (1,2) is safe from pits? | $\alpha_1 = \neg P_{1,2}$ | Does KB entail α_1? (Whenever KB true, α_1 true?)

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

KB

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

KB is true
iff $\wedge_{k=1,\ldots,6} R_k$ is true

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

Truth Table Enumeration Example: Wumpus World

Can we infer that (1,2) is safe from pits? | $\alpha_1 = \neg P_{1,2}$ | Does KB entail α_1? (Whenever KB true, α_1 true?)

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

KB

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

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

KB is true iff $\wedge_{k=1,\ldots,6} R_k$ is true

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

Truth Table Enumeration Example: Wumpus World

Can we infer that (1,2) is safe from pits? | $\alpha_1 = \neg P_{1,2}$ | Does KB entail α_1? (Whenever KB true, α_1 true?)

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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

KB

$R_1: \neg P_{1,1}$ $R_3: \neg P_{2,1}$

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

KB is true iff $\wedge_{k=1,\ldots,6} R_k$ is true

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

Truth Table Enumeration Example: Wumpus World

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

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

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

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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

KB

$R_1: \neg P_{1,1}$ $R_3: \neg P_{2,1}$

$R_2: \neg B_{1,1}$ $R_4: B_{2,1}$

KB is true iff $\wedge_{k=1,\ldots,6} R_k$ is true

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

Truth Table Enumeration Example: Wumpus World

Can we infer that (1,2) is safe from pits? | $\alpha_1 = \neg P_{1,2}$ | Does KB entail α_1? (Whenever KB true, α_1 true?)

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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

KB

$R_1: \neg P_{1,1}$ $R_3: \neg P_{2,1}$

$R_2: \neg B_{1,1}$ $R_4: B_{2,1}$

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

KB is true iff $\wedge_{k=1,\ldots,6} R_k$ is true

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

Truth Table Enumeration Example: Wumpus World

Can we infer that (1,2) is safe from pits? | $\alpha_1 = \neg P_{1,2}$ | Does KB entail α_1? (Whenever KB true, α_1 true?)

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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

KB

$R_1: \neg P_{1,1}$ $R_3: \neg P_{2,1}$

$R_2: \neg B_{1,1}$ $R_4: B_{2,1}$

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

$R_6: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

KB is true iff $\wedge_{k=1,\ldots,6} R_k$ is true

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

Truth Table Enumeration Example: Wumpus World

Can we infer that (1,2) is safe from pits? | $\alpha_1 = \neg P_{1,2}$ | Does KB entail α_1? (Whenever KB true, α_1 true?)

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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

KB

$R_1: \neg P_{1,1}$ $R_3: \neg P_{2,1}$

$R_2: \neg B_{1,1}$ $R_4: B_{2,1}$

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

$R_6: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

As $M(\text{KB}) \subseteq M(\alpha_1)$, KB $\vDash \alpha_1$

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)

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 }))
```

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

Given KB and α, run DFS

- State:
  - *Unassigned Symbols*
  - *Current Model* (Value Assignments)
- Actions:
  - Assign T/F to a symbol in *Unassigned Symbols*
- Transition Model:
  - Remove assigned symbol from *Unassigned Symbols*
  - Update *Current Model*

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

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

function TT-CHECK-ALL($KB, \alpha, symbols, model$) **returns** *true* or *false*
  if EMPTY?($symbols$) **then**
    if PL-TRUE?($KB, model$) **then return** PL-TRUE?($\alpha, model$)
    else return *true* // when KB is false, always return true
  else
    $P \leftarrow$ FIRST($symbols$)
    $rest \leftarrow$ REST($symbols$)
    return (TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = true\}$)
      and
      TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = false\}$)

Depth-first enumeration

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($KB, \alpha, symbols, \{\}$)

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

Depth-first enumeration

Base case:

- *Unassigned Symbols* is empty
- Check that KB implies α
  - If KB False under given model, then return True
  - Else if KB and α both True under given model, then return True
  - Else return False

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

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

function TT-CHECK-ALL($KB, \alpha, symbols, model$) **returns** *true* or *false*
  if EMPTY?($symbols$) **then**
    if PL-TRUE?($KB, model$) **then return** PL-TRUE?($\alpha, model$)
    else return *true* // when KB is false, always return true
  else
    $P \leftarrow$ FIRST($symbols$)
    $rest \leftarrow$ REST($symbols$)
    return (TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = true\}$)
      and
      TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = false\}$))

Depth-first enumeration

Recursive case:

- With each recursive call, take a symbol, *P*, from *Unassigned Symbols*, and then branch the existing model with *P* = True, and *P* = False
  - i.e., generates the 2^n possible assignments to the n symbols
- Notice that the recursive call performs a conjunction over all model applications to KB and α

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 }))
```

Algorithm properties

- $O(2^n)$ time complexity
  - Given n symbols
  - Abstract away cost of FIRST, REST, UNION, EMPTY?, and PL-TRUE?
- 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/11468403214