Department of
Mathematics &
Computer Science

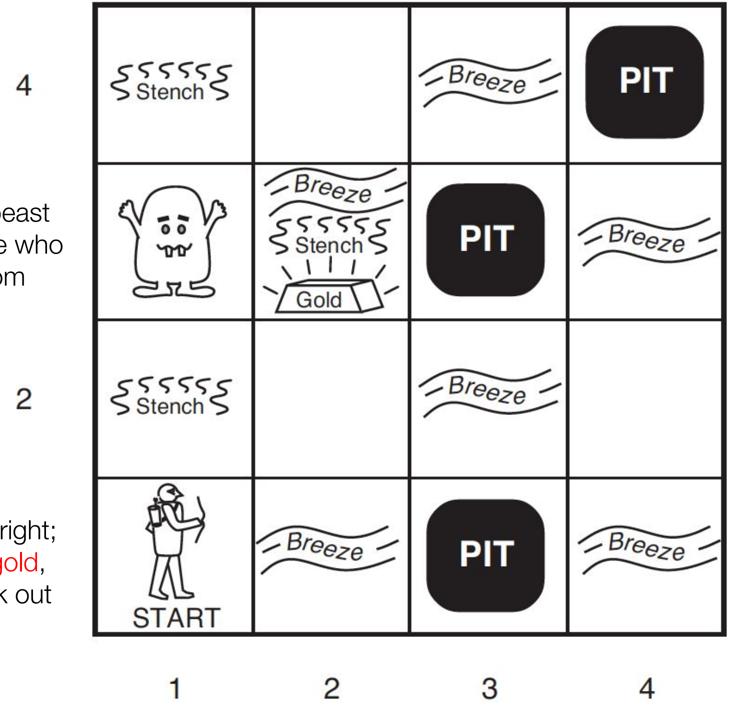
7. Logical Agents

7.2 The wumpus world

"The wumpus world is a cave consisting of rooms connected by passageways"

Wumpus is a beast that eats anyone who enters its room

Agent is facing right; wants to find gold, and come back out



Some rooms have pits that trap anyone who wander into

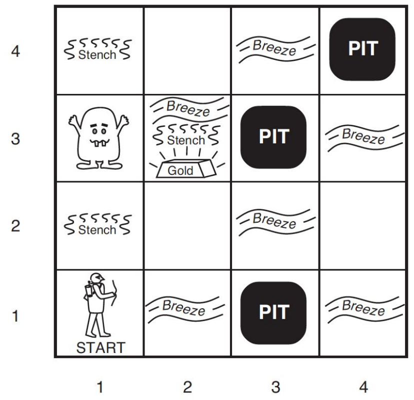
Rules:

- The agent has only one arrow to shoot the wumpus
- The game ends either when the agent dies or when the agent climbs out of the cave
- The agent dies if it enters a square containing a pit or a live wumpus
- It is safe to enter a square with a dead wumpus

7.2 PEAS for wumpus world

1. Performance measure

- “+1000” for climbing out of the cave with the gold
- “–1000” for falling into a pit or being eaten by the wumpus
- “–1” for each action taken, and “–10” for using up the arrow



2. Environment

- A 4×4 grid of rooms
- The agent always starts in the square labeled [1,1], facing to the right
- The locations of the gold and the wumpus are chosen randomly, with a uniform distribution, from the squares other than the start square
- Each square other than the start can be a pit, with probability 0.2
- ~21% of the environments are unfair, because the gold is in a pit or surrounded by pits

7.2 PEAS for wumpus world

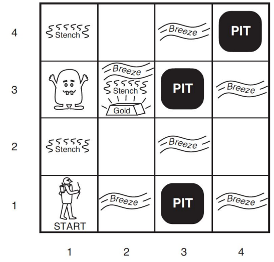
3. Actuators

- The agent can move Forward, TurnLeft by 90^0, or TurnRight by 90^0
  - If an agent tries to move forward and bumps into a wall, then the agent does not move
- The action Grab can be used to pick up the gold if it is in the same square as the agent
- The action Shoot can be used to fire an arrow in a straight line in the direction the agent is facing
  - The arrow continues until it either hits (and hence kills) the wumpus or hits a wall
  - The agent has only one arrow, so only the first Shoot action has any effect
- The action Climb can be used to climb out of the cave, but only from square [1,1]

4. Sensors

The agent has five sensors, each of which gives a single bit of information:

- In the square containing the wumpus and in the directly (not diagonally) adjacent
  squares, the agent will perceive a Stench
- In the squares directly adjacent to a pit, the agent will perceive a Breeze
- In the square with gold, agent will perceive Glitter
- When an agent walks into a wall, it will perceive a Bump
- When the wumpus is killed, it emits a woeful Scream that can be perceived anywhere in the cave



7.2 Percept sequence

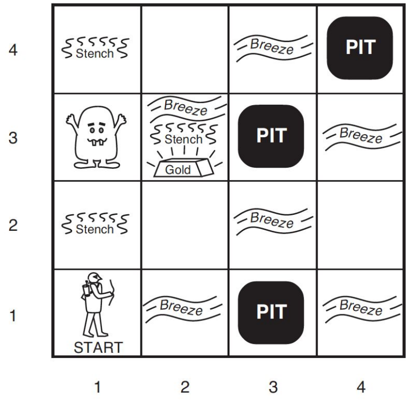
- The percepts will be given to the agent program in the form of a list of five symbols
- Example:
  - If there is a stench and a breeze, but no glitter, bump, or scream, the agent will get

```
[Stench, Breeze, None, None, None]
```

In which squares and when will the agent perceive the following?

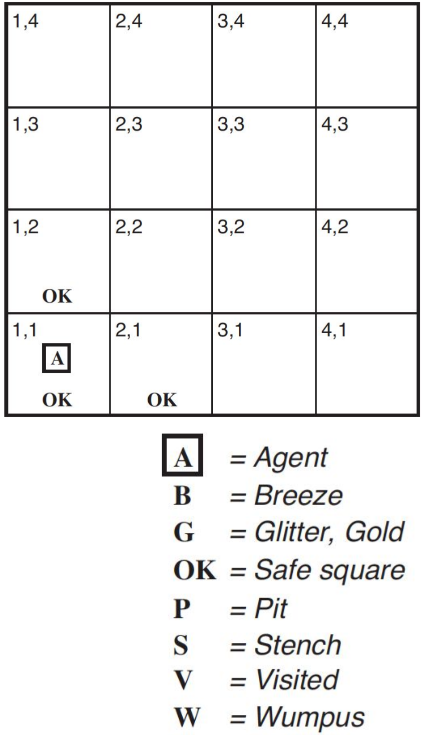
- `[None, None, None, None, None]`
- `[None, None, None, None, Scream]`
- `[None, Breeze, None, None, None]`
- `[None, None, Glitter, Breeze, Scream]`

THINK PAIR SHARE



7.2 A knowledge-based agent exploring an environment

- Consider that the agent can use an informal knowledge representation language
    - i.e. it can write down symbols in a grid
- The agent's initial knowledge base contains the rules of the environment
    - As described in PEAS
        - It knows that it is in [1,1], denoted with 'A'
        - [1,1] is a safe square, denoted with 'OK'
- The first percept is `[None, None, None, None, None]`
    - From this the agent can conclude that its neighboring squares, [1,2] and [2,1], are free of dangers - they are OK



7.2 A knowledge-based agent exploring an environment

1) Assuming that you (as a human player agent) can maintain a knowledge-base in a grid shown beside, update your knowledge state so you return safely to the start square.

2) Show how the percept sequence updates after each move.

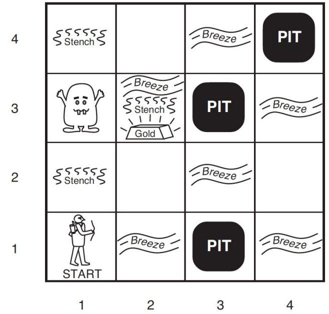
The format for percept sequence is [stench/none, breeze/none, glitter/none, bump/none, scream/none]

Solution:

- Agent is at [1,1] and percept sequence is [None, None, None, None, None]
- The agent decides to move forward to [2,1] and at [2,1] the agent perceives [0, 1, 0, 0, 0]
  - We mark 'B' at [2,1] and 'A' at [2,1]
  - There must be a pit in a neighboring square
  - The pit cannot be in [1,1], by the rules of the game, so there must be a pit in [2,2] or [3,1] or both
  - We mark 'P?' to indicate a possible pit in [2,2] and [3,1]
  - There is only one unvisited square with 'OK', that is [1,2]
- The agent moves to [1,2] and perceives [1, 0, 0, 0, 0]
  - The agent perceives a stench in [1,2]
  - There must be a wumpus nearby
  - But the wumpus cannot be in [1,1]
  - By the rules of the game, and it cannot be in [2,2]
    - or the agent would have detected a stench when it was in [2,1]
  - Therefore, we can infer that the wumpus is in [1,3], we mark with 'W!'
  - The lack of a breeze in [1,2] implies that there is no pit in [2,2]
    - Yet the agent has already inferred that there must be a pit in either [2,2] or [3,1], so this means it must be in [3,1] - a fairly difficult inference
  - The agent has now proved to itself that there is neither a pit nor a wumpus in [2,2]
- The agent decides to move forward to [ ] and the percept sequence will be [, , , ,]

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

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



How can we build logical agents that can represent information and draw conclusions?

For example, the way we used knowledge state to solve the agent's task

Topics we will cover next:

1. What is the central component of a knowledge-based agent?
2. Fundamental concepts of logical representation and reasoning
3. Propositional Logic - a simple but powerful logic

7.1 Central component of a knowledge-based agent

- Central component of a knowledge-based agent is its **knowledge base** (KB)
  - A KB is a set of **sentences**
  - Each sentence is expressed in a language called a **knowledge representation language** and represents some assertion about the world
  - KB may initially contain some **background knowledge**
- There must be ways to (a) add new sentences to the KB and (b) query what is known
  - The standard names for these operations are TELL and ASK
  - Both operations (TELL & ASK) may involve **inference**
    - i.e. deriving new sentences from old
- Inference
  - When one ASKs a question of the KB, the answer should follow from what has been told to the knowledge base previously

When you design a KB for an agent we need to decide how our sentences are structured in the knowledge representation language.
For example, we may come up with different KRL for navigating the wumpus world.

What do the operations TELL and ASK do? How can we implement them?

THINK PAIR SHARE

7.1 An example of knowledge-based agent program

Each time the agent program is called, it does three things: 1

1. First, it TELLs the knowledge base what it perceives
2. Second, it ASKs the knowledge base what action it should perform

   In the process of answering this query, extensive reasoning may be done about the current state of the world, about the outcomes of possible action sequences, and so on

3. Third, the agent program TELLs the knowledge base which action was chosen, and the agent executes the action

A generic knowledge-based agent. Given a percept, the agent adds the percept to its knowledge base, asks the knowledge base for the best action, and tells the knowledge base that it has in fact taken that action.

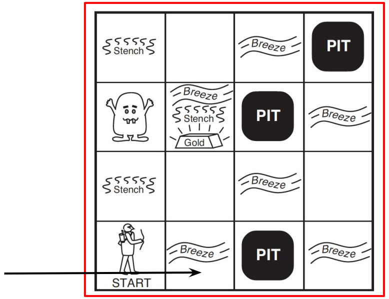
```
function KB-AGENT(percept) returns an action
  persistent: KB, a knowledge base
              t, a counter, initially 0, indicating time

  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

2

Example: 4
Agent moves to [2,1]
[none, breeze, none, none, none]

1. ‘Perceive Breeze’ is TELLed to KB
2. ASK returns: ‘Move to [2,2] or [1,1]’
3. ‘Moved to [2,2]’ is TELLed to KB



- MAKE-PERCEPT-SENTENCE
  - constructs a sentence asserting that the agent perceived the given percept at the given time
- MAKE-ACTION-QUERY
  - constructs a sentence that asks what action should be done at the current time
- MAKE-ACTION-SENTENCE 3
  - constructs a sentence asserting that the chosen action was executed
- The details of the inference mechanisms are hidden inside TELL and ASK

7.3 Logic - syntax and semantics

- Knowledge bases consist of sentences
    - Sentences are expressed according to the **syntax** of the representation language
    - The representation language specifies all the sentences that are well formed
    - For example, in ordinary arithmetic:
      "x + y = 4" is a well-formed sentence, whereas "x4y+ =" is not

- The **semantics** defines the truth of each sentence with respect to each "possible world" or "model"

  For example, each possible wumpus world environment

    - For example, the semantics for arithmetic specifies that the sentence "x + y =4" is true in a world where x is 2 and y is 2, but false in a world where x is 1 and y is 1
    - In standard logics, every sentence must be either true or false in each possible world
        - there is no "in between"

7.3 Model = Possible world

Example:

- ‘x’ men and ‘y’ women are sitting at a table playing bridge
- The sentence ‘x + y = 4’ is true when there are four people in total
- The possible models are all possible assignments of real numbers to the variables x and y
    - i.e. [x=1, y=3] is a model, [x=2, y=2] is another model

M(α) = the set of all models of sentence ‘α’

M(“x+y=4”) = { [x=0, y=4], [x=1, y=3], .., [x=4, y=0] }

If a sentence ‘α’ is true in a model ‘m’ we say that ‘**m satisfies α**’ or ‘**m is a model of α**’

For a sentence x+y=1, what are the possible models? What is M(α)? Which of the models satisfy x+y=1?

THINK PAIR SHARE

7.3 The concept of 'logical entailment'

- $\alpha \models \beta$ means that sentence α entails sentence β (mathematical notation)
    - Mary broke the window $\models$ The window broke
    - Sue and Fred went to the party $\models$ Sue went to the party
    - Walk $\models$ Movement
- $\alpha \models \beta$ if and only if, in every model in which α is true, β is also true (formal definition)

Thus,

- $\alpha \models \beta$ if and only if $M(\alpha) \subseteq M(\beta)$ ← Counter intuitive

Examples:

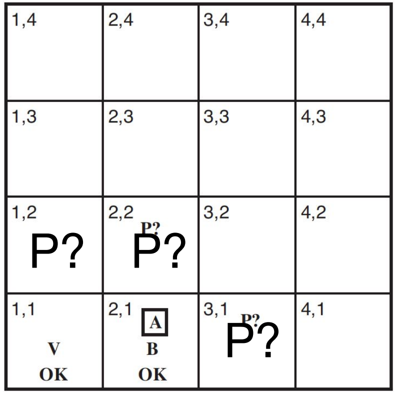
Walk $\models$ Movement if and only if "a model where walking is possible" $\subseteq$ "a model where movement is possible"

M. broke window $\models$ The window broke iff "a model where M. breaks window" $\subseteq$ "a model where window is broken"

If the sentence $x = 0$ entails the sentence $xy = 0$, what is the relationship between a model which satisfies $x = 0$ and a model that satisfies $xy = 0$?

THINK PAIR SHARE

7.3 Logical entailment for the wumpus world



Consider that the agent is in [2,1]

- The agent detected nothing in [1,1] and Breeze in [2,1]
- The knowledge base for the agent =

  Current percepts + rules of the wumpus world

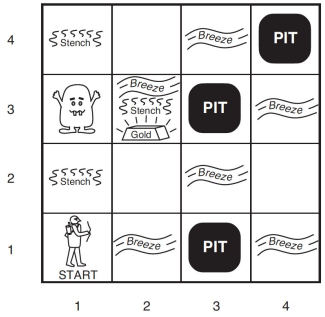
- Do the adjacent squares [1, 2], [2, 2], & [3, 1] contain pits?
  - Each of the three squares might or might not contain a pit
  - So, there are $2^3 = 8$ possible models

Agent can maintain a list of all the “visitable” nodes (including those skipped along the path of making choices, such as [1,2])

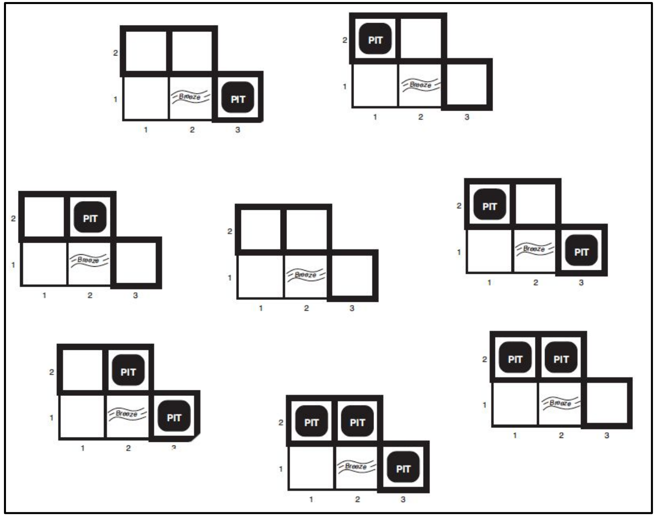
What are the 8 possible models?

THINK PAIR SHARE

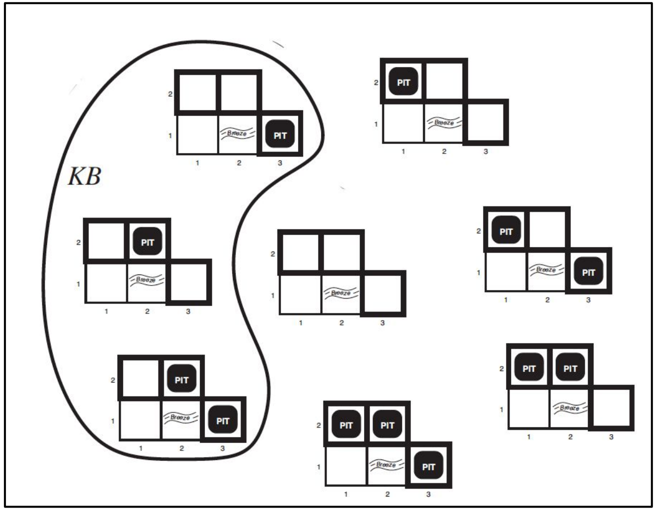
- KB can be used to decrease this model space
  - The KB is false in models that contradict what the agent knows
    - For example, the KB is false in any model in which [1,2] contains a pit, because there is no breeze in [1,1]



7.3 Logical entailment for wumpus world



All possible models



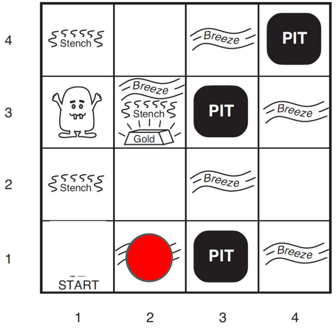
Models for which the KB is true

- According to current percept of Breeze there must be a PIT at either [2,2] or [3,1]

- Agent did not perceive anything in [1,1] so [1,2] must be safe

KB can be used to reduce possible models!

This gives us M(KB)



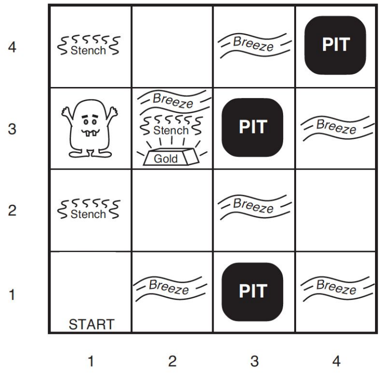
7.3 Logical entailment for wumpus world

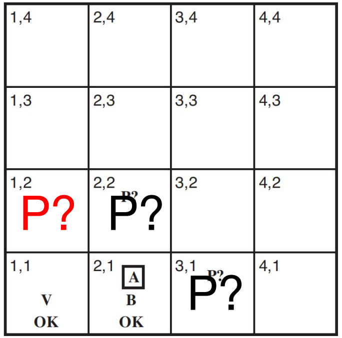
Do the adjacent squares [1, 2], [2, 2], & [3, 1] contain pits?

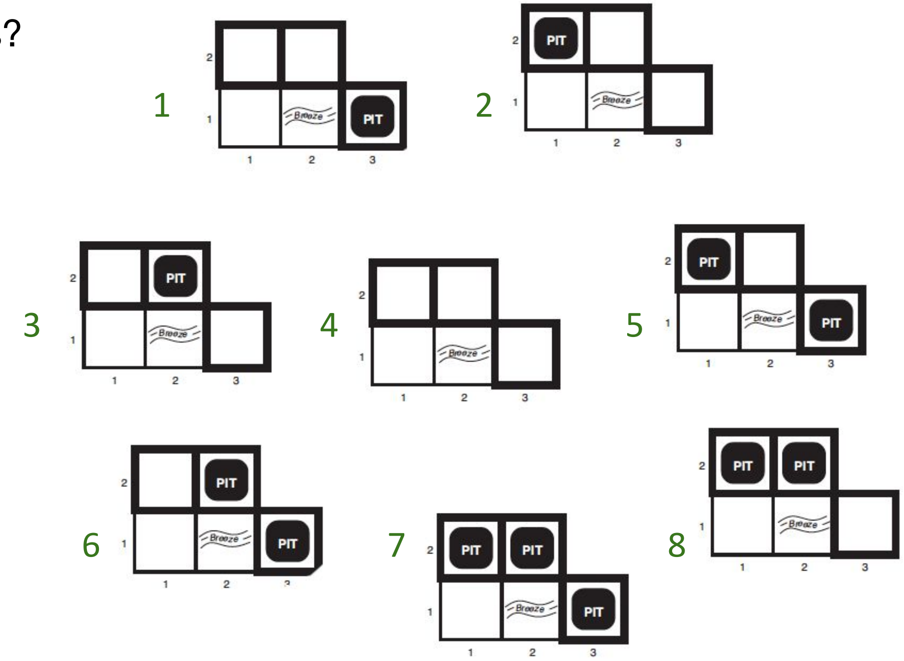
The agent can possibly draw three conclusions:

1. α1 = "There is no pit in [1,2]"
2. α2 = "There is no pit in [2,2]"
3. α3 = "There is no pit in [3,1]"

This gives us M(α)!







Q1. Select the models that satisfy α1

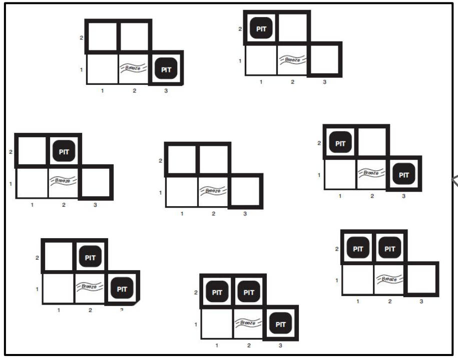
Q2. Select the models that satisfy α2

Q3. Select the models that satisfy α3

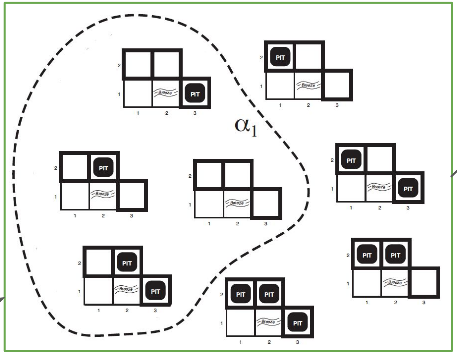
THINK PAIR SHARE

7.3 Logical entailment for wumpus world

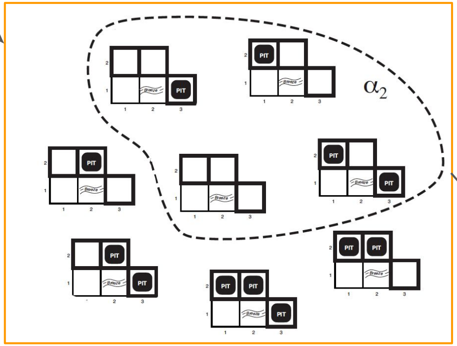
Say, we focus only on α1 and α2

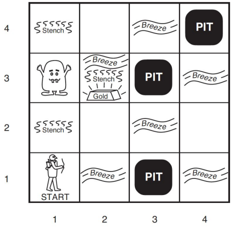


α1 = "There is no pit in [1,2]"



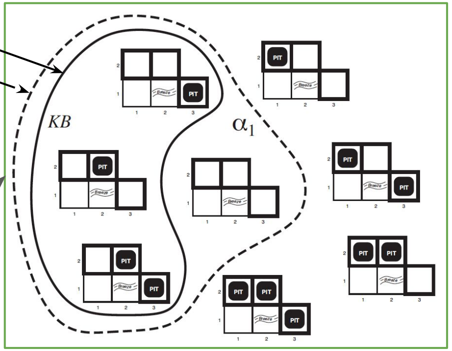
α2 = "There is no pit in [2,2]"



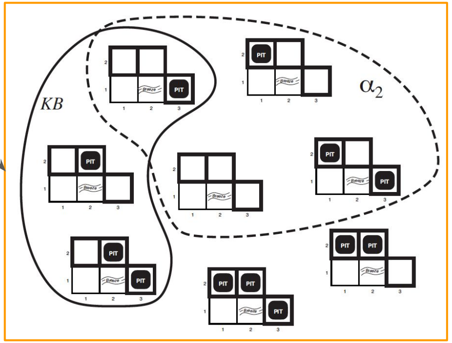


This algorithm for inference is a **model-checking approach**. It is a direct implementation of the definition of entailment: enumerate the models, and check that α is true in every model in which KB is true.

M(KB)
M(α)



- **The KB is false in models that contradict what the agent knows**
- It can be used to decrease the model space



"By inspection":

In every model in which KB is true, α1 is also true

Is M(KB) ⊆ M(α1)?

Hence, KB ⊨ α1: **there is no pit in [1,2]**

KB entails α1

Formal Definition of Entailment:
α ⊨ β if and only if, in every model in which α is true, β is also true. i.e. α ⊨ β if and only if M(α) ⊆ M(β)

By inspection:

In some models in which KB is true, α2 is false

Hence, KB ⊭ α2: **the agent cannot conclude that there is no pit in [2,2]**

7.3 Entailment vs Inference

- Entailment can be applied to carry out logical inference
- The inference algorithm (in the previous slide) is called **model checking**
  - because it enumerates all possible models to check that α is true in all models in which KB is true, i.e. $M(KB) \subseteq M(\alpha)$
- If an inference algorithm 'i' can derive α from KB
  - We say "α is derived from KB by i" or "i derives α from KB"

$$KB \vDash_i \alpha$$

- An inference algorithm that derives only entailed sentences is called **sound** or **truth preserving**
  - An unsound inference procedure makes things up as it goes along—it announces the discovery of nonexistent needles
  - Model checking (for wumpus world example), which is a method of inference, is a sound procedure

Entailment

Inference

Entailment = the needle being in the haystack

Inference = finding it

7.3 Inference and completeness

- The property of completeness is desirable in an inference algorithm
  - an inference algorithm is complete if it can derive any sentence that is entailed

- For real haystacks (which are finite in extent)
  - It seems obvious that a systematic examination can always decide whether the needle is in the haystack
  - However, for many knowledge bases, the haystack of consequences is infinite, and completeness becomes an important issue
  - For example, a huge wumpus world

7.4 Propositional logic

- A simple yet powerful logic (a logic language)

7.4.1 Syntax of propositional logic (symbols)

- The syntax of propositional logic defines the allowable sentences
- The **atomic sentences** consist of a **single proposition symbol**
  - Each such symbol stands for a proposition that can be true or false
    - e.g. S_s = “The phrase St. Louis starts with S”
  - We use symbols that start with an uppercase letter and may contain other letters or subscripts, for example: P, Q, R, $W_{1,3}$ and North
  - The names are arbitrary but are often chosen to have some mnemonic value
    - e.g. we use $W_{1,3}$ to stand for the proposition that the “wumpus is in [1,3]”
  - Symbols such as $W_{1,3}$ are atomic, i.e., W, 1, and 3 are not meaningful parts of the symbol
- There are two proposition symbols with fixed meanings:
  - True is the always-true proposition
  - False is the always-false proposition

7.4.1 Syntax of propositional logic (connectives)

- Complex sentences are constructed from simpler sentences
  - using parentheses and logical connectives
- ¬ (not) - A sentence such as $\neg W_{1,3}$ is called the negation of $W_{1,3}$
  - A literal is either an atomic sentence (positive literal) or a negated atomic sentence (negative literal)
  - For example, $W_{1,3}$ is a literal and $\neg W_{1,3}$ is also a literal
- ∧ (and) - $W_{1,3} \land P_{3,1}$, is called a conjunction; its parts are the conjuncts
- ∨ (or) - $(W_{1,3} \land P_{3,1}) \lor W_{2,2}$, is a disjunction of the disjuncts $(W_{1,3} \land P_{3,1})$ and $W_{2,2}$
- ⇒ (implies) - $(W_{1,3} \land P_{3,1}) \Rightarrow \neg W_{2,2}$ is called an implication (or conditional)
  - Its premise (or antecedent) is $(W_{1,3} \land P_{3,1})$, and its conclusion (or consequent) is $\neg W_{2,2}$
  - Implications are also known as rules or if–then statements
- ⇔ (if and only if) - The sentence $W_{1,3} \Leftrightarrow \neg W_{2,2}$ is a biconditional

7.4.1 Syntax of propositional logic (connectives)

In arithmetic, how much is −2+4 ? Why?

THINK PAIR SHARE

$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} & \rightarrow & \mathit{True} \mid \mathit{False} \mid P \mid Q \mid R \mid \ldots \\
\mathit{ComplexSentence} & \rightarrow & (\ \mathit{Sentence}\ ) \mid [\ \mathit{Sentence}\ ] \\
 & \mid & \neg\ \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \land \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \lor \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \Rightarrow \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \Leftrightarrow \mathit{Sentence} \\
\\
\textsc{Operator Precedence} & : & \neg, \land, \lor, \Rightarrow, \Leftrightarrow
\end{array}
$$

Square brackets mean the same thing as parentheses; the choice of square brackets or parentheses is solely to make it easier for a human to read a sentence

Formal grammar of sentences in propositional logic, along with operator precedences, from highest to lowest

7.4.2 Semantics of propositional logic

- Semantics defines the rules for determining the truth of a sentence **with respect to a particular model**
  - $W_{1,3}$ = wumpus in [1,3] and $P_{1,3}$= pit in [1,3]
    - $W_{1,3} \land P_{1,3}$ = false, if either of them are false
- A model simply fixes the truth value—true or false—for every proposition symbol
  - For example, if the sentences in the knowledge base make use of the proposition symbols $P_{1,2}$, $P_{2,2}$, and $P_{3,1}$, then one possible model is $m_1 = \{P_{1,2} = \text{false}, P_{2,2} = \text{false}, P_{3,1} = \text{true}\}$
  - With three proposition symbols, there are $2^3 = 8$ possible models
- The models and symbols are purely mathematical objects with no necessary connection to wumpus worlds
  - $P_{1,2}$ is just a symbol; it might mean “there is a pit in [1,2]” or “I’m in Paris today and tomorrow”
- The semantics for propositional logic must specify how to compute the truth value of any sentence (short or long), given a model
  - This is done recursively

7.4.2 Semantics of propositional logic

- The 5 rules which hold for any sub-sentences ‘P’ and ‘Q’ in any model ‘m’:
  a. ¬P is true iff P is false in m
  b. P ∧ Q is true iff both P and Q are true in m
  c. P ∨ Q is true iff either P or Q is true in m
  P implies Q d. P ⇒ Q is true unless P is true and Q is false in m
  P & Q are biconditional e. P ⇔ Q is true iff P and Q are both true or both false in m
- These rules can also be expressed as truth tables:

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*

Figure 7.8 Truth tables for the five logical connectives. To use the table to compute, for example, the value of $P \vee Q$ when P is true and Q is false, first look on the left for the row where P is *true* and Q is *false* (the third row). Then look in that row under the $P \vee Q$ column to see the result: *true*.

“The truth value of any sentence ‘s’ can be computed w.r.t. any model ‘m’ by a simple recursive evaluation”

Propositional logic does not require any relation of causation or relevance between P and Q.

Example1: The sentence “5 is odd implies Tokyo is the capital of Japan” is a true sentence of propositional logic.

Example2: “5 is even implies Sam is smart” is true, regardless of whether Sam is smart. This seems bizarre, but it makes sense if you think of “P ⇒ Q” as saying, “If P is true, then I am claiming that Q is true. Otherwise I am making no claim.”

The only way for this sentence to be false is if P is true but Q is false.

7.4.2 Semantics of propositional logic (examples)

Symbols in Model 1 and 2 are

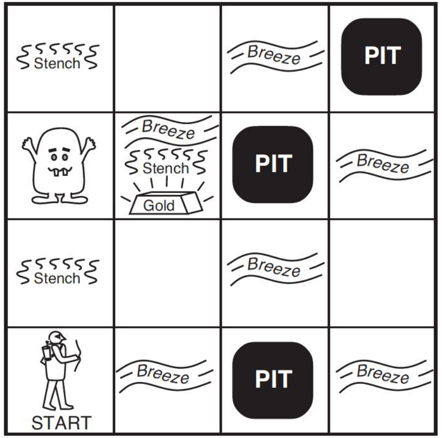
$W_{1,2}$, $W_{1,1}$, $W_{2,1}$, ..., $P_{1,2}$, $P_{1,1}$, $P_{2,1}$, ..., $A_{1,2}$, $A_{1,1}$, $A_{2,1}$, ..., $B_{1,2}$, $B_{1,1}$, $B_{2,1}$, ...,$OK_{1,2}$, $OK_{1,1}$, $OK_{2,1}$, ..., etc.

$P_{x,y}$ is true if there is a pit in [x, y]
$W_{x,y}$ is true if there is a wumpus in [x, y], dead or alive
$B_{x,y}$ is true if the agent perceives a breeze in [x, y]
$S_{x,y}$ is true if the agent perceives a stench in [x, y]

Consider a wumpus world (figures on the right) where the agent is at [2,3]. Assuming that the agent has fully explored the wumpus world, and the symbols have usual meaning (see table above), **evaluate the following expressions in this model.**

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

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

This table won't be provided in Test!

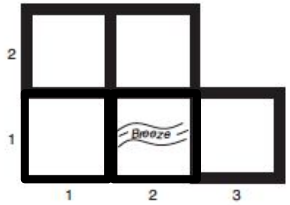
PL Semantics - Reference Table

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*

Figure 7.8 Truth tables for the five logical connectives. To use the table to compute, for example, the value of $P \vee Q$ when P is true and Q is false, first look on the left for the row where P is *true* and Q is *false* (the third row). Then look in that row under the $P \vee Q$ column to see the result: *true*.

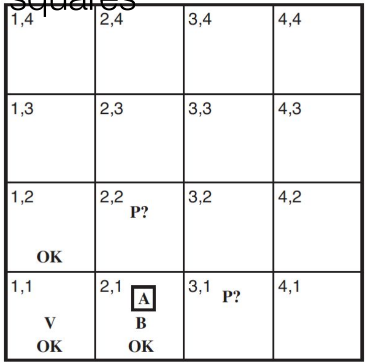
7.4.3 A simple knowledge base for propositional logic

- Immutable aspects of the wumpus world, are good candidates for KB
    - For example, “$P_{x,y}$ is true if there is a pit in [x, y]” and “$W_{x,y}$ is true if there is a wumpus in [x, y], dead or alive”



Consider that the agent is at [2,1]:

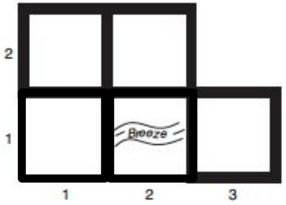
- KB consists of sentences (true in all wumpus worlds models)
    - There is no pit in [1,1]
        - $R_1 : \neg P_{1,1}$
    - A square is breezy if and only if there is a pit in a neighboring square
        - This has to be stated for each square; for now, let’s include just the relevant squares
        - $R_2 : B_{1,1} \Leftrightarrow P_{1,2} \vee P_{2,1}$
        - $R_3 : B_{2,1} \Leftrightarrow P_{1,1} \vee P_{2,2} \vee P_{3,1}$
- KB also consists of sentences for the agent (as the agent moves)
    - Breeze percepts for the first two squares visited
        - $R_4 : \neg B_{1,1}$
        - $R_5 : B_{2,1}$

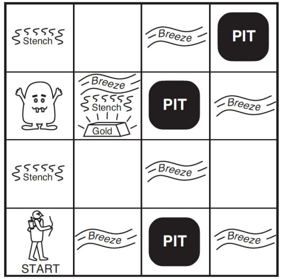


7.4.3 A simple knowledge base for propositional logic

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

Figure 7.9 A truth table constructed for the knowledge base given in the text. KB is true if R_1 through R_5 are true, which occurs in just 3 of the 128 rows (the ones underlined in the right-hand column). In all 3 rows, $P_{1,2}$ is false, so there is no pit in [1,2]. On the other hand, there might (or might not) be a pit in [2,2].

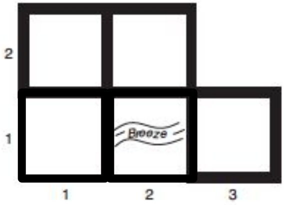


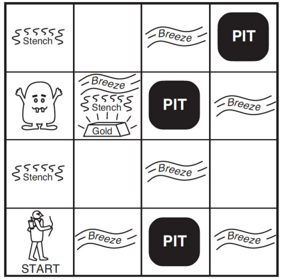


7.4.4 A simple inference proc. in propositional logic

The goal of inference is to decide whether KB $\vDash \alpha$ for some sentence α

- In the wumpus-world example, say that the agent is at [2,2]
- **Is $\neg P_{1,2}$ entailed by our KB? i.e. Does $KB_{2,2} \vDash \neg P_{1,2}$?**
  - Step 1: Find all possible models
    - The relevant proposition symbols are $B_{1,1}$, $B_{2,1}$, $P_{1,1}$, $P_{1,2}$, $P_{2,1}$, $P_{2,2}$, and $P_{3,1}$
    - With seven symbols, there are 2^7 =128 possible models
  - Step 2: Select the models for which KB is true (by looking at the **Truth Table**)
    - Of all the 128 models, in three of these models, KB is true
    - [$B_{1,1}$= false, $B_{2,1}$ = true, $P_{1,1}$ = false, $P_{1,2}$= false, $P_{2,1}$= false, $P_{2,2}$= true, $P_{3,1}$= false]
    - [$B_{1,1}$= false, $B_{2,1}$ = true, $P_{1,1}$ = false, $P_{1,2}$= false, $P_{2,1}$= false, $P_{2,2}$= false, $P_{3,1}$= true]
    - [$B_{1,1}$= false, $B_{2,1}$ = true, $P_{1,1}$ = false, $P_{1,2}$= false, $P_{2,1}$= false, $P_{2,2}$= true, $P_{3,1}$= true]
  - Step 3: In all the three models in KB, $\neg P_{1,2}$ is true
    - Hence there is no pit in [1,2]
- Another example: Is $P_{2,1}$ entailed by our KB?
  - $P_{2,1}$ is also true in all of the three models
  - So, there is no pit in [2,2]





Summary

- Knowledge is contained in agents in the form of sentences in a knowledge representation language that are stored in a knowledge base
- A knowledge-based agent is composed of a knowledge base and an inference mechanism
  - It operates by storing sentences about the world in its knowledge base, using the inference mechanism to infer new sentences, and using these sentences to decide what action to take
- A representation language is defined by its syntax, which specifies the structure of sentences, and its semantics, which defines the truth of each sentence in each possible world or model
- The relationship of entailment between sentences is crucial to our understanding of reasoning
  - A sentence α entails another sentence β if β is true in all worlds where α is true
- Inference is the process of deriving new sentences from old ones
  - Sound inference algorithms derive only sentences that are entailed
  - Complete algorithms derive all sentences that are entailed
- Prop. Logic is a simple language consisting of proposition symbols & logical connectives
  - It can handle propositions that are known true, known false, or completely unknown

7.4.4 A simple inference proc. in propositional logic

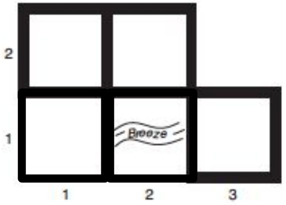
function TT-ENTAILS?(KB, α) **returns** *true* or *false*
  inputs: KB, the knowledge base, a sentence in propositional logic
      α, the query, a sentence in propositional logic

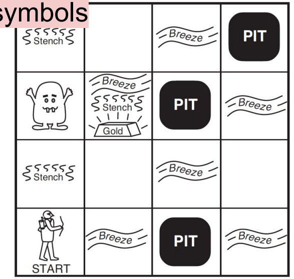
  $symbols \leftarrow$ a list of the proposition symbols in KB and α
  return TT-CHECK-ALL($KB, \alpha, symbols, \{\,\}$) $KB, \neg P_{1,2}, \{B_{1,1}, B_{2,1}, P_{1,1}, P_{1,2}, P_{2,1}, P_{2,2}, P_{3,1}\}, \{\}$

---

function TT-CHECK-ALL($KB, \alpha, symbols, model$) **returns** *true* or *false*
  if EMPTY?($symbols$) **then**   *'symbols'* will be empty when model has all symbols
    if PL-TRUE?($KB, model$) **then return** PL-TRUE?($\alpha, model$)
    else return *true* // *when KB is false, always return true*
  else do
    $P \leftarrow$ FIRST($symbols$)   Corresponds to Step 2   Corresponds to Step 3
    $rest \leftarrow$ REST($symbols$)
    return (TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = true\}$) $\{B_{1,1} = T\}$
        and $\{B_{1,1} = T, B_{2,1} = T\}$
        TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = false\}$))   ...

Figure 7.10 A truth-table enumeration algorithm for deciding propositional entailment. (TT stands for truth table.) PL-TRUE? returns *true* if a sentence holds within a model. The variable *model* represents a partial model—an assignment to some of the symbols. The keyword "**and**" is used here as a logical operation on its two arguments, returning *true* or *false*.





When we don't have a pre-computed Truth Table