COE206 – Principles of Artificial Intelligence

Mustafa MISIR

Istinye University, Department of Computer Engineering

mustafa.misir@istinye.edu.tr

http://mustafamisir.github.io
http://memoryrlab.github.io

L7: Logical Agents

Outline

- Knowledge-Based Agents
- The Wumpus World
- Logic
- Propositional Logic
- Propositional Theorem Proving

Knowledge-based Agents

The central component of a **knowledge-based agent** is its **knowledge base** (**KB**)

- A **knowledge base** is a set of **sentences**[1]
- Each **sentence** is expressed in a language called a **knowledge representation language** and represents some assertion[2] about the **world**
- Sometimes dignify a **sentence** with the name **axiom**, when the **sentence** is taken as given without being derived from other **sentences**.

[1] here **sentence** is used as a technical term. It is related but not identical to the sentences of English and other natural languages

[2] a confident and forceful statement of fact or belief

Knowledge-based Agents

There must be a way to add new **sentences** to the **knowledge base** and a way to query what is known.

- The standard names for these operations are `TELL` and `ASK`, respectively
- Both operations may involve **inference** – that is, deriving new **sentences** from old

Knowledge-based Agents

Given a **percept**,

1. the agent adds the **percept** to its **knowledge base**, which is TELLing the **knowledge base** what it **perceives**
2. ASKS the **knowledge base** for the **best action**
3. 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
```

Knowledge-based Agents

The **knowledge-based agent** is not an arbitrary program for calculating **actions**

- It is amenable to a description at the **knowledge level**, where we need specify only what the **agent** knows and what its **goals** are, in order to fix its behavior
- e.g., an automated taxi might have the goal of taking a passenger from San Francisco to Marin County and might know that the Golden Gate Bridge is the only link
- Then expect to cross the Golden Gate Bridge because it knows that that will achieve its **goal**.

This analysis is independent of how the taxi works at the **implementation level**

Knowledge-based Agents

A **knowledge-based agent** can be built simply by `TELL`ing it what it needs to know

- Starting with an empty **knowledge base**, the **agent** designer can `TELL` sentences one by one until the **agent** knows how to operate in its **environment**

This is called the **declarative** approach to system building.

In contrast, the **procedural** approach encodes desired behaviors directly as program code

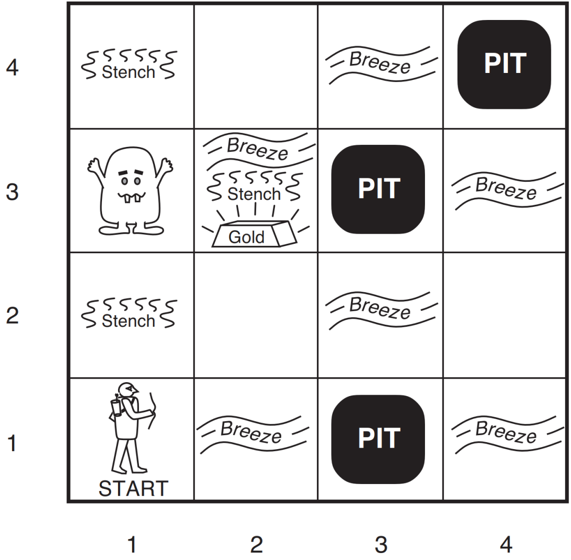
Wumpus World

An **environment** in which **knowledge-based agents** can show their worth.

A cave consisting of rooms connected by passageways

- Hidden somewhere in the cave is the terrible **wumpus**, a beast that eats anyone who enters its room (not moving)
- The **wumpus** can be shot by an **agent**, but the **agent** has only one arrow
- Soom rooms contain bottomless pits that will trap anyone who wanders into these rooms (except for the **wumpus**, which is too big to fall in)
- The only good feature of this environment is the possibility of finding gold

Wumpus World[3]



[3] a typical **wumpus** world. The **agent** is in the bottom left corner, facing right

Wumpus World — PEAS – 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.

The game ends either when the **agent** dies or when the **agent** climbs out of the cave

Wumpus World — PEAS – 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.

- In addition, each square other than the start can be a pit, with probability 0.2

Wumpus World — PEAS – Actuators

The **agent** can move **Forward**, **TurnLeft** by 90 degrees, or **TurnRight** by 90 degrees.

- 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 **action Climb** can be used to climb out of the cave, but only from square [1,1]

The **agent** dies if it enters a square containing a pit or a live **wumpus** – It is safe, albeit smelly, to enter a square with a dead **wumpus** – If an agent tries to move forward and bumps into a wall, then the **agent** does not move

Wumpus World — PEAS – Sensors

The **agent** has five **sensors**:

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

Wumpus World Environment Characterization

- (**Fully**) **Observable**[4]: No – only local perception, so **Partially Observable**
- **Deterministic**[5]: Yes – outcomes exactly specified
- **Episodic**[6]: No – sequential at the level of actions
- **Static**[7]: Yes – **Wumpus** and **Pits** do not move
- **Discrete**[8]: Yes
- **Single-agent**: Yes – **Wumpus** is essentially a natural feature

---

[4] An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data

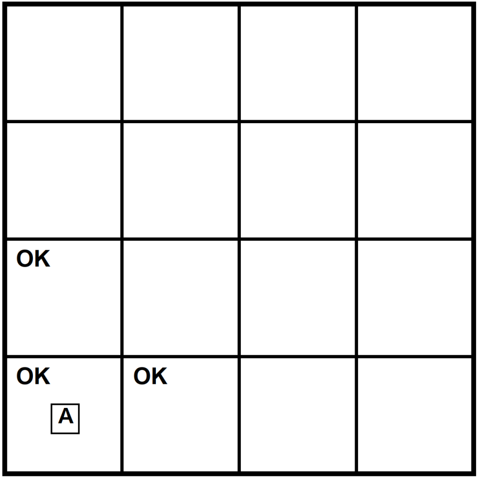
[5] If the next state of the environment is completely determined by the current state and the action executed by the agent, then we say the environment is deterministic

[6] In an episodic task environment, the agent's experience is divided into atomic episodes. In each episode the agent receives a percept and then performs a single action. Crucially, the next episode does not depend on the actions taken in previous episodes. In sequential environments, on the other hand, the current decision could affect all future decisions

[7] If the environment can change while an agent is deliberating, then we say the environment is dynamic for that agent; otherwise, it is static

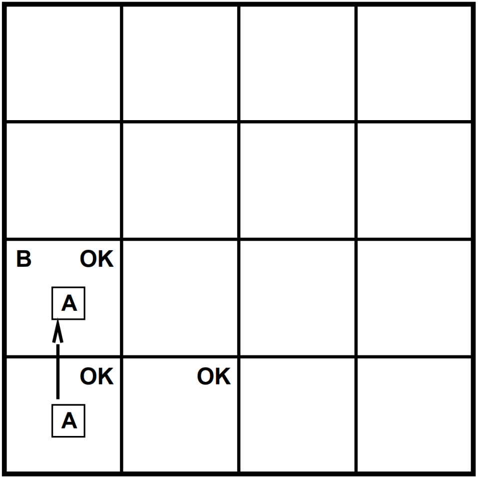
[8] The discrete/continuous distinction applies to the state of the environment, to the way time is handled, and to the percepts and actions of the agent

Exploring a Wumpus World



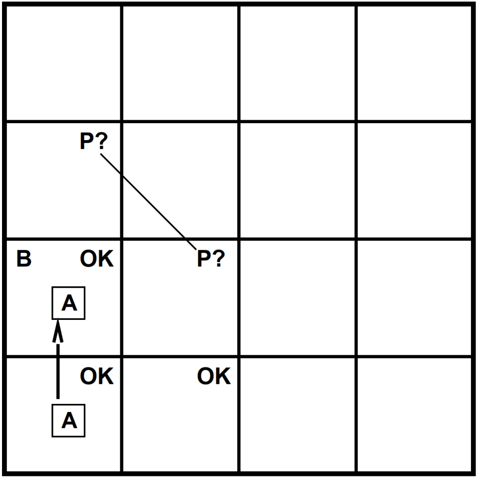
A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

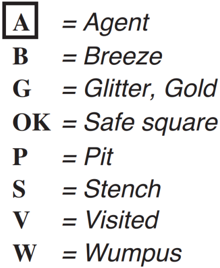
Exploring a Wumpus World



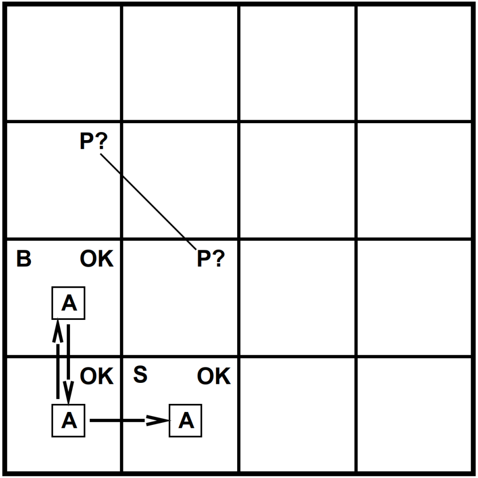
A = Agent
B = Breeze
G = Glitter, Gold
OK = Safe square
P = Pit
S = Stench
V = Visited
W = Wumpus

Exploring a Wumpus World



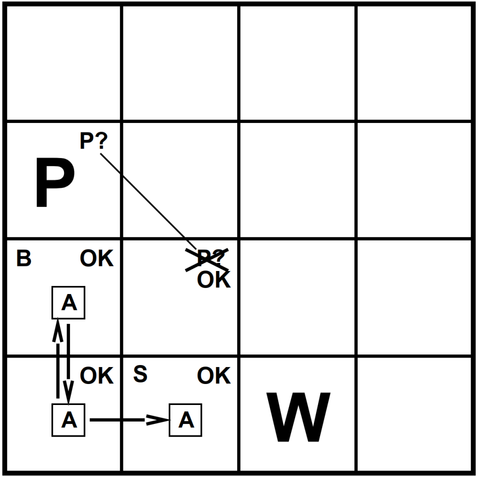


Exploring a Wumpus World



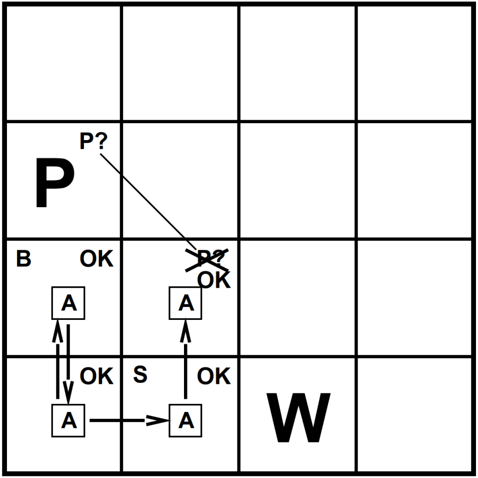
A	= *Agent*
B	= *Breeze*
G	= *Glitter, Gold*
OK	= *Safe square*
P	= *Pit*
S	= *Stench*
V	= *Visited*
W	= *Wumpus*

Exploring a Wumpus World



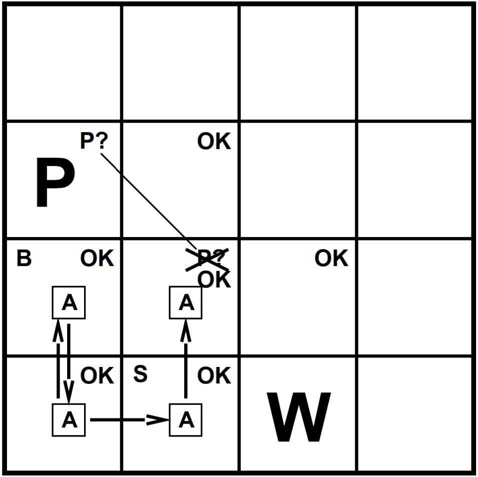
A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

Exploring a Wumpus World



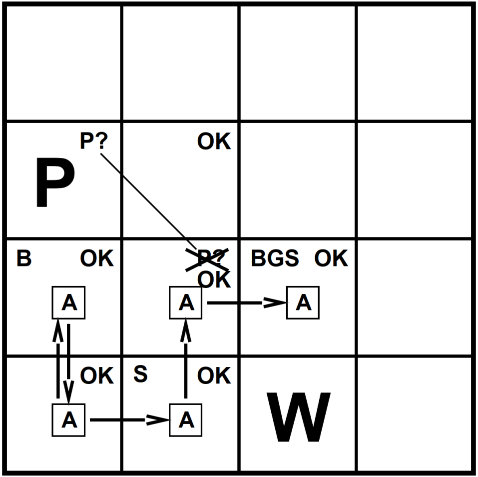
A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

Exploring a Wumpus World



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

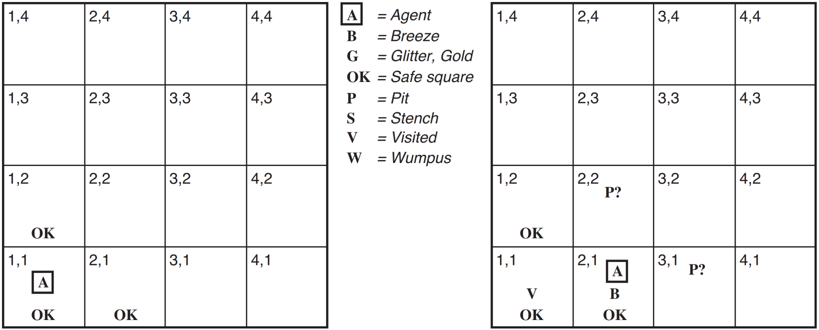
Exploring a Wumpus World



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

Exploring a Wumpus World

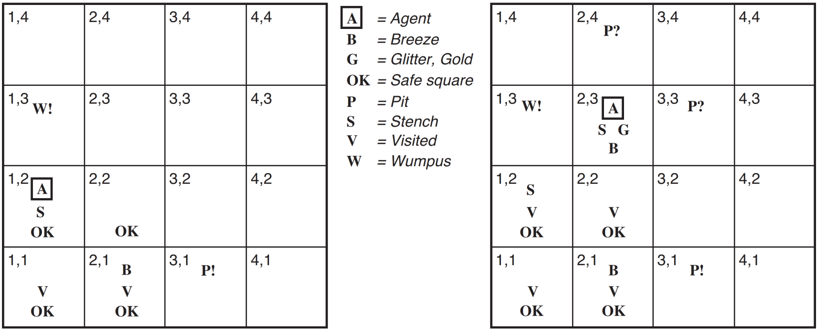
The first step taken by the agent in the wumpus world



(a) The initial situation, after **percept** (each referring to a sensor) [*None, None, None, None, None*] – from which the **agent** can conclude that its neighboring squares, [1,2] and [2,1], are free of dangers—they are OK

(b) After one move, with **percept** [*None, Breeze, None, None, None*]

Exploring a Wumpus World



(a) After the third move, with **percept** [*Stench, None, None, None, None*] – the `Stench` in [1,2] means that there must be a **wumpus** nearby, but **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, the **agent** can infer that the **wumpus** is in [1,3], **W!**

(b) After the fifth move, with **percept** [*Stench, Breeze, Glitter, None, None*]

Logic

- **Logics** are formal languages for representing information such that conclusions can be drawn
- **Syntax** defines the **sentences** in the language – a set of words or expressions that are obtained using an alphabet and rules
- **Semantics** define the meaning of sentences – truth of a **sentence** in a **possible world**

In standard **logics**, every **sentence** must be either true or false in each **possible world** – there is no in-between

e.g., the language of arithmetic

$x + 2 \geq y$ is a **sentence**; $x2 + y >$ is not a **sentence**

$x + 2 \geq y$ is true iff $x + 2$ is no less than y

$x + 2 \geq y$ is true in a **world** where $x = 7$, $y = 1$

$x + 2 \geq y$ is false in a **world** where $x = 0$, $y = 6$

Logic

The term **model** can be used in place of **possible world**

- Whereas **possible worlds** might be thought of as (potentially) real environments that the **agent** might or might not be in, **models** are mathematical abstractions, each of which simply fixes the truth or falsehood of every relevant **sentence**

e.g. for **possible world**, having x men and y women sitting at a table playing bridge, and the **sentence** $x + y = 4$ is true when there are four people in total

- the **possible models** are just all possible assignments of real numbers to the variables x and y
- If a **sentence** α is true in **model** m, we say that m satisfies α or sometimes m is a **model** of α – use the notation $M(\alpha)$ to mean the set of all **models** of α

Logic — Entailment[10]

Logical **entailment** means that a **sentence** follows logically from another **sentence**:

- A set of **sentences** (called **premises**) **logically entails** a **sentence** (called a **conclusion**) if and only if every truth assignment that satisfies the **premises** also satisfies the **conclusion**

$\alpha \models \beta$ means that **sentence** α **entails** the **sentence** β

if and only if, in every **model** in which α is true, β is also true[9]

$$\alpha \models \beta \text{ if and only if } M(\alpha) \subseteq M(\beta)$$

The relation of **entailment** is familiar from arithmetic; e.g. the **sentence** $x = 0$ **entails** the **sentence** $xy = 0$ regardless of the value of y

---

[9] the direction of the $\subseteq$ here: if *alpha* $\models \beta$, then α is a stronger assertion than β: it rules out more **possible worlds**

[10] an **entailment** is a deduction or implication, that is, something that follows logically from or is implied by something else

Logic — Entailment[11]

$$\{p\} \models (p \vee q)$$
$$\{p\} \not\models (p \wedge q)$$
$$\{p, q\} \models (p \wedge q)$$

Logical entailment does not guarantee **logical equivalence**

$$\{p\} \models (p \vee q)$$
$$(p \vee q) \not\models \{p\}$$

[11] http://logic.stanford.edu/classes/cs157/2011/lectures/lecture03.pdf

Logic — Entailment – Truth Table Method

Check for **logical entailment** by comparing tables of all possible interpretations

- In the first table, eliminate all rows that do not satisfy **premises**
- In the second table, eliminate all rows that do not satisfy the **conclusion**

If the remaining rows in the first table are a subset of the remaining rows in the second table, then the **premises logically entail** the **conclusion**

$$\alpha \models \beta \text{ if and only if } M(\alpha) \subseteq M(\beta)$$

Logic — Entailment – Truth Table Method

Does p **logically entail** $(p \vee q)$?

p	q
T	T
T	F
F	T
F	F

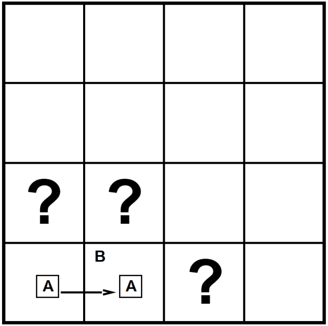
p	q
T	T
T	F
F	T
F	F

Logic — Entailment – Wumpus World

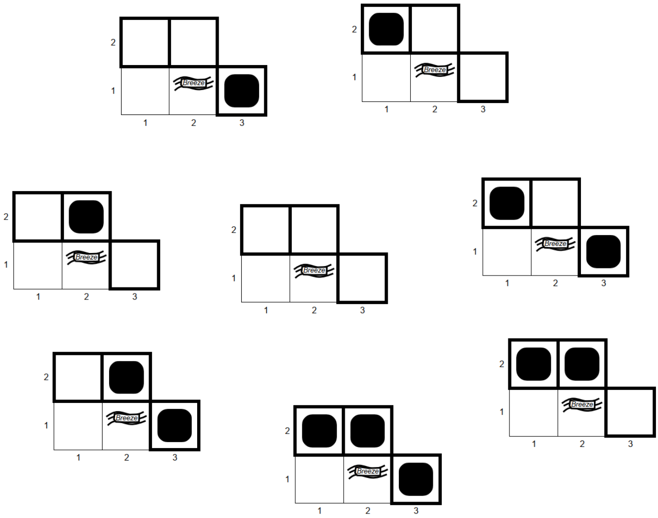
Situation after detecting nothing in $[1, 1]$, moving right, breeze in $[2, 1]$

Consider **possible models** for **KB** assuming only pits

- 3 Boolean choices $\Rightarrow$ 8 possible models

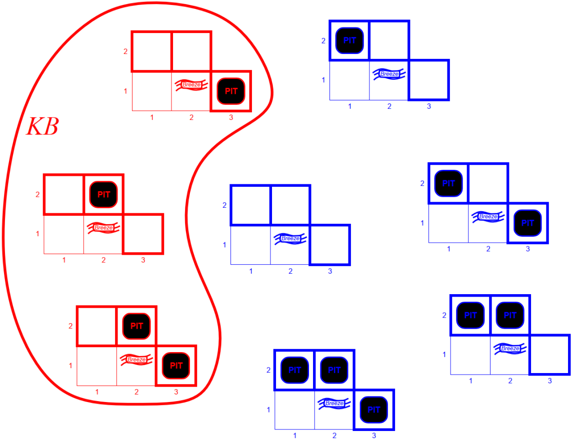


Logic — Entailment – Wumpus World[12]



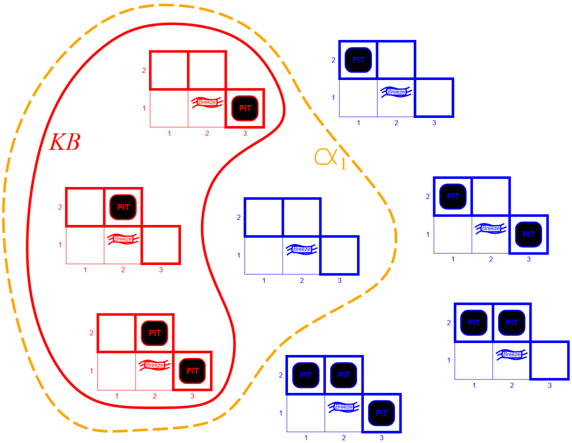
[12] 3 Boolean choices of having pits $\Longrightarrow$ 8 possible models

Logic — Entailment – Wumpus World



KB = Wumpus World Rules + Observations

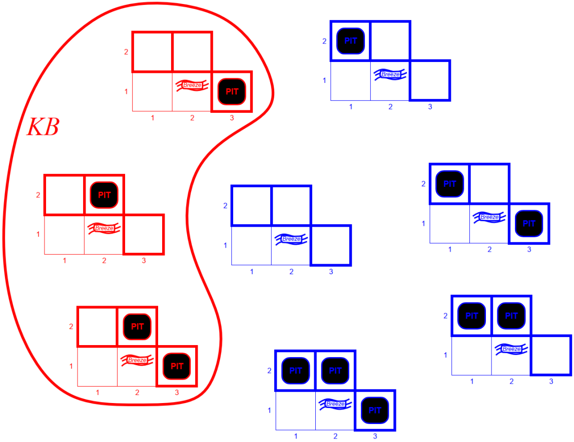
Logic — Entailment – Wumpus World



KB = Wumpus World Rules + Observations

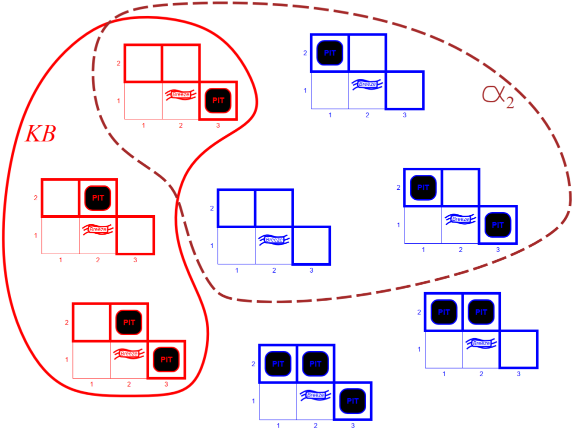
- $\alpha_1 = [1, 2]$ is safe, $KB \models \alpha_1$ proved by **model checking**

Logic — Entailment – Wumpus World



KB = Wumpus World Rules + Observations

Logic — Entailment – Wumpus World

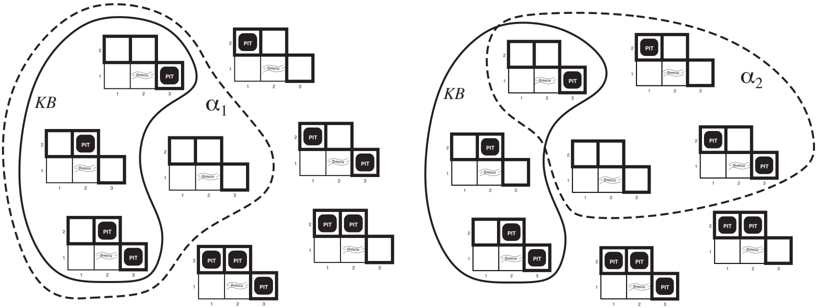


KB = Wumpus World Rules + Observations

- $\alpha_2 = [2, 2]$ is safe, $KB \not\models \alpha_2$

Logic — Entailment – Wumpus World

Possible models for the presence of pits in squares [1,2], [2,2], and [3,1]. The **KB** corresponding to the observations of nothing in [1,1] and a breeze in [2,1] is shown by the solid line.



(a) Dotted line shows models of α_1 (no pit in [1,2])

(b) Dotted line shows models of α_2 (no pit in [2,2])

Logic — Entailment

Logical **entailment** means that a **sentence** follows logically from another **sentence**:

$$KB \models \alpha$$

Knowledge base KB entails sentence α if and only if α is true in all **worlds** where **KB** is true

- e.g., the **KB** containing the Giants won and the Reds won **entails** either the Giants won or the Reds won
- e.g., $x + y = 4$ **entails** $4 = x + y$

Entailment is a relationship between **sentences** (i.e., **syntax**) that is based on **semantics**

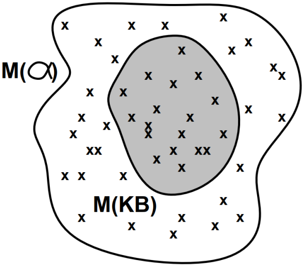
Logic — Models

Logicians typically think in terms of **models**, which are formally structured **worlds** with respect to which truth can be evaluated

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

- $M(\alpha)$ is the set of all **models** of α
- Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$

e.g. **KB** = Giants won and Reds won, α = Giants won



Propositional Logic

Propositional logic[13] consists of a **formal language** and **semantics** that give meaning to the well-formed strings, which are called **propositions**.

The **syntax** of **propositional logic** defines the allowable **sentences**.

The **atomic sentences** consist of a single **proposition symbol**, e.g. P, Q etc.; each such symbol stands for a **proposition** that can be true or false

[13] also known as **sentential logic** and **statement logic**, is the branch of logic that studies ways of joining and/or modifying entire **propositions**, **statements** or **sentences** to form more complicated **propositions**, **statements** or **sentences**, as well as the logical relationships and properties that are derived from these methods of combining or altering **statements** – https://www.iep.utm.edu/prop-log/

Propositional Logic — Propositions, e.g.[14]

- $5 + 2 = 8 \rightsquigarrow$ false
- How are you? $\rightsquigarrow$ a question is not a **proposition**
- 2 is a prime number $\rightsquigarrow$ true
- She is very talented $\rightsquigarrow$ since she is not specified, neither true nor false
- There are other life forms on other planets in the universe $\rightsquigarrow$ either true or false

[14] https://people.cs.pitt.edu/~milos/courses/cs2740/Lectures/class4.pdf

Propositional Logic

Propositional Logic — Syntax

Complex sentences are constructed from simpler **sentences**, using parentheses and **logical connectives**. There are 5 **connectives** in common use[15]:

- $\neg$ (not). A **sentence** such as $\neg W_{1,3}$ is the **negation** of $W_{1,3}$. A **literal** is either an **atomic sentence** (a positive literal) or a **negated atomic sentence** (a negative literal)
- $\wedge$ (and)[16]. A sentence whose main connective is $\wedge$, such as $W_{1,3} \wedge P_{3,1}$, is a **conjunction**; its parts are the **conjuncts**
- $\vee$ (or)[17]. A **sentence** using $\vee$, such as $(W_{1,3} \wedge P_{3,1}) \vee W_{2,2}$, is a **disjunction** of the **disjuncts** $(W_{1,3} \wedge P_{3,1})$ and $W_{2,2}$
- $\Rightarrow$ (implies)[18]. A **sentence** such as $(W_{1,3} \vee P_{3,1}) \Rightarrow \neg W_{2,2}$ is called an implication (or conditional). Its **premise** or **antecedent** is $(W_{1,3} \wedge P_{3,1})$, and its **conclusion** or **consequent** is $\neg W_{2,2}$. Implications are also known as rules or **if–then statements**
- $\Leftrightarrow$ (if and only if)[19]. The **sentence** $W_{1,3} \Leftrightarrow \neg W_{2,2}$ is a **biconditional**

15 the first symbol is **unary connective**, while the remaining ones are **binary connectives** – you might additionally refer (..) as a grouping symbol

16 the $\wedge$ looks like an A for And

17 the $\vee$ comes from the Latin vel, which means or. It might be easier to remember $\vee$ as an upside-down $\wedge$

18 sometimes written as $\supset$ or $\rightarrow$

19 sometimes denoted as $\equiv$

Propositional Logic

The **composite** / **compound sentences** are constructed from **valid sentences** via **connectives**, **e.g.** if P and Q are **sentences**, then

- $\neg P$ and $(P \wedge Q)$ are **composite sentences**

Propositional Logic[20]

Suppose the **propositions** P and Q stand for these **statements** about the **world**:

P: It is raining outside

Q: The ground is wet

Then the following **compound propositions** stand for these **statements** about the **world**:

$\neg P$: It is not raining outside

$P \wedge Q$: It is raining outside and the pavement is wet

$P \vee Q$: It is raining outside or the pavement is wet

$P \Rightarrow Q$: If it is raining outside, then the ground is wet

$P \Leftrightarrow Q$: It is raining outside if and only if the ground is wet

[20] Artificial Intelligence - With an Introduction to Machine Learning by Richard E. Neapolitan, Xia Jiang, 2018 - CRCPress

Propositional Logic — Syntax – BNF (Backus–Naur Form)

A formal **grammar** of **propositional logic** in the form of a **context-free grammar** as each expression has the same form in any context

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

$$\textsc{Operator Precedence} \quad : \quad \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$$

Propositional Logic — Syntax – BNF

There are 4 components to a **BNF grammar**:

- A set of **terminal symbols** that are the symbols or words that make up the strings of the language[21]
- A set of **nonterminal symbols** that categorize subphrases of the language[22]
- A **start symbol**, which is the **nonterminal symbol** that denotes the complete set of strings of the language[23]
- A set of **rewrite rules**, of the form $LHS \rightarrow RHS$, where LHS is a **nonterminal symbol** and RHS is a sequence of zero or more symbols[24]

---

21 they could be letters (**A**, **B**, **C**, . . .) or words (**a**, **aardvark**, **abacus**, . . .), or whatever symbols are appropriate for the domain.

22 e.g. the **nonterminal symbol** *NounPhrase* in English denotes an infinite set of strings including *you* and *the big slobbery dog*

23 in English, this is *Sentence*; for arithmetic, it might be *Expr*, and for programming languages it is *Program*

24 these can be either **terminal** or **nonterminal symbols**, or the symbol ϵ, which is used to denote the empty string

Propositional Logic — Syntax – BNF

The **BNF grammar** by itself is ambiguous; a **sentence** with several operators can be parsed by the **grammar** in multiple ways.

To eliminate the ambiguity we define a **precedence** for each operator:

- **not** operator $\neg$ has the highest **precedence**, e.g. $\neg A \land B$ is the equivalent of $(\neg A) \land B$

Propositional Logic — Semantics

The **semantic** gives the meaning to **sentences**

The **semantics** in the **propositional logic** is defined by:

- **Interpretation** of propositional symbols and constants – **semantics** of **atomic sentences**
- Through the meaning of **connectives** – meaning (semantics) of composite sentences

Propositional Logic — Semantics

The **semantics** defines the rules for determining the truth of a **sentence** with respect to a particular **model**.

In **propositional logic**, a **model** simply fixes the truth value—true or false — for every proposition symbol.

- e.g. 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:

$$m_1 = \{P_{1,2} = \mathit{false}, P_{2,2} = \mathit{false}, P_{3,1} = \mathit{true}\}$$

With 3 proposition symbols, there are $2^3 = 8$ **possible models**

- The **models** 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*

Propositional Logic — Semantics

The **semantics** for **propositional logic** must specify how to compute the truth value of any **sentence**, given a **model** – done recursively.

All **sentences** are constructed from **atomic sentences** and the **five connectives**[25]

Atomic sentences are easy:

- True is true in every **model** and False is false in every **model**
- The truth value of every other proposition symbol must be specified directly in the model; e.g. in the model m_1 given earlier, $P_{1,2}$ is false.

[25] therefore, need to specify how to compute the truth of **atomic sentences** and how to compute the truth of **sentences** formed with each of the **5 connectives**

Propositional Logic — Semantics

For complex **sentences**, we have five rules, which hold for any subsentences P and Q in any **model** m (*iff* means *if and only if*):

- is true iff P is false in m
- $P \wedge Q$ is true iff both P and Q are true in m
- $P \vee Q$ is true iff either P or Q is true in m
- $P \Rightarrow Q$ is true unless P is true and Q is false in m
- $P \Leftrightarrow Q$ is true iff P and Q are both true or both false in m

The truth value of any **sentence** s can be computed with respect to any **model** m by a simple recursive evaluation from truth tables

- e.g. the **sentence** $\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1})$, evaluated in m_1, gives $\mathit{true} \wedge (\mathit{false} \vee \mathit{true}) = \mathit{true} \wedge \mathit{true} = \mathit{true}$

Propositional Logic — Semantics

The **rules** can also be expressed with truth tables that specify the truth value of a complex sentence for each possible assignment of truth values to its components.

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

Propositional Logic — Knowledge Base (KB)

Construct a KB for the **Wumpus World**, $[x, y]$ refers to the location

- $P_{x,y}$ 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 $[x, y]$

These sentences will suffice to derive $\neg P_{1,2}$ (no pit in $[1, 2]$), as

was done informally earlier.

Propositional Logic — Knowledge Base (KB)

Label each sentence as R_i:

- There is no pit in $[1,1] \rightsquigarrow R_1 : \neg P_{1,1}$
- A square is breezy if and only if there is a pit in a neighboring square. This has to be stated for each square; for now, we include just the relevant squares:
  - $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  - $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
- The preceding **sentences** are true in all **wumpus worlds**. Now we include the breeze **percepts** for the first two squares visited in the specific world the agent is in
  - $R_4 : \neg B_{1,1}$
  - $R_5 : B_{2,1}$

Propositional Logic — Inference

Goal now is to decide whether $KB \models \alpha$ for some **sentence** α

- e.g. is $\neg P_{1,2}$ entailed by the **KB**?

First algorithm to try for **inference** is a **model checking** approach that is a direct implementation of the definition of **entailment**:

- enumerate the **models**
- check that α is true in every **model** in which **KB** is true

Models are assignments of true or false to every **proposition symbol**

Propositional Logic — Inference

KB is true if R_1 through R_5 are true, which occurs in just 3 of the 128 rows[26] (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 $[2, 2]$.

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	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*	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	<u>*true*</u>
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*

[26] there are 7 of B and P **proposition symbols**, each with two value options $\{$*true*, *false*$\}$, so there are $2^7 = 128$ variations / models

Propositional Logic — Inference

A **truth-table** (TT) enumeration algorithm[27] for deciding **propositional entailment**

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 do
    $P \leftarrow$ FIRST(*symbols*)
    $rest \leftarrow$ REST(*symbols*)
    return (TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = true\}$)
      and
      TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = false\}$))

---

27 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 as a logical operation on its two arguments, returning true or false

Propositional Theorem Proving

Until now, we have seen how to determine **entailment** by **model checking**: enumerating **models** and showing that the **sentence** must hold in all **models**

Now, investigate how **entailment** can be done by **theorem proving** – applying rules of **inference** directly to the **sentences** in a **knowledge base** to construct a proof of the desired **sentence** without consulting **models**.

If the number of **models** is large but the length of the proof is short, then **theorem proving** can be more efficient than **model checking**.

Propositional Theorem Proving — Logical Equivalence

The first concept is **logical equivalence**: two **sentences** α and β are **logically equivalent** if they are true in the same set of **models**; $\alpha \equiv \beta$

- e.g. using truth tables, $P \wedge Q$ and $Q \wedge P$ are **logically equivalent**

An alternative definition of **equivalence** is as follows: any two sentences α and β are equivalent only if each of them **entails** the other:

$$\alpha \equiv \beta \text{ if and only if } \alpha \models \beta \text{ and } \beta \models \alpha$$

Propositional Theorem Proving — Logical Equivalence

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

Propositional Theorem Proving — Validity

A **sentence** is **valid** if it is true in all **models**; e.g. the sentence $P \vee \neg P$ is **valid**

Valid sentences are also known as **tautologies**[28]

- Because the **sentence** True is true in all **models**, every valid sentence is **logically equivalent** to True

What good are **valid sentences**? From the definition of **entailment**, can derive the **deduction theorem**:

- For any **sentences** α and β, $\alpha \models \beta$ if and only if the **sentence** $(\alpha \Rightarrow B)$ is **valid**

28 the saying of the same thing twice over in different words, generally considered to be a fault of style, e.g. they arrived one after the other in succession)

Propositional Theorem Proving — Satisfiability

A **sentence** is **satisfiable** if it is true in, or satisfied by, some **model**.

- e.g. the **knowledge base** given earlier, $(R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5)$, is satisfiable because there are 3 **models** in which it is true

Satisfiability can be checked by enumerating the **possible models** until one is found that satisfies the **sentence**

- The problem of determining the **satisfiability** of sentences SAT in propositional logic – the SAT problem – can be computationally challenging

Many problems in computer science are **satisfiability** problems

- e.g. all the **constraint satisfaction problems** ask whether the constraints are satisfiable by some assignment

Propositional Theorem Proving — Inference & Proofs

Inference rules that can be applied to derive a proof — a chain of conclusions that leads to the desired goal.

- The best-known rule is called **Modus Ponens** (Latin for mode that affirms) and is written

$$\frac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$$

Means that, whenever any **sentences** of the form $\alpha \Rightarrow \beta$ and α are given, then the sentence β can be inferred

- e.g. if $(\mathit{WumpusAhead} \wedge \mathit{WumpusAlive}) \Rightarrow \mathit{Shoot}$ and $(\mathit{WumpusAhead} \wedge \mathit{WumpusAlive})$ are given, then Shoot can be inferred.

Propositional Theorem Proving — Inference & Proofs

Another useful **inference rule** is **And-Elimination**, which says that, from a **conjunction**, any of the **conjuncts** can be inferred:

$$\frac{\alpha \wedge \beta}{\alpha}$$

e.g., from (*WumpusAhead* $\wedge$ *WumpusAlive*), *WumpusAlive* can be inferred