KNOWLEDGE AND REASONING

KNOWLEDGE-BASED AGENTS

"An agent can represent knowledge ofits world, its goals and the current situation by sentences in logic and decide what to do by inferring that a certain action or course of action is appropriate to

LOGICAL SYSTEM

- **Intelligent agents** need knowledge about the world to choose good actions/decisions.

- A **logical system** is a system that knows about its **partially observable environment** and can reason about possible actions by **inferring from the hidden information**. **Reasoning** is also known as **inferencing**. An agent that acts upon logical system is known as **Knowledge based Agent.**

- Issues in construction of a logical system:

  1. **Knowledge representation:** how do we represent information? Knowledge representation should
     be somewhat natural, expressive and efficient.

  2. **Knowledge reasoning:** how do we use information to reach decisions and/or derive new facts?

- **Knowledge** in the form of a set of **facts** about our **environment** are **stored in a knowledge base (KB).**

  - **Facts** are claims about the environment which are either **true** or **false**. **Facts** are represented by sentences

  - A **sentence** is an assertion about the world. **Sentences** are expressed in a representation language.

KNOWLEDGE BASED AGENT

- A knowledge-based agent(Logic Agent)comprises of 2 features:

  1.Knowledge base: domain-specific content i.e. a list of facts that are known to the agent.
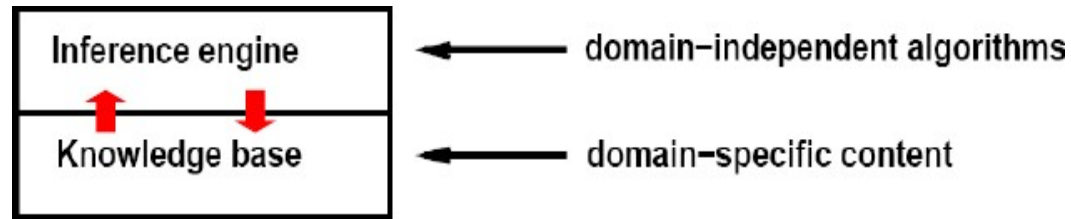
  2. **Inference engine:** domain-independent algorithms for inferencing new knowledge. Current percepts to infer hidden aspects of the current state using Rules of inference .

- **Knowledge base:** A set of sentences in in a formal knowledge representation language that encodes assertions about the world.

A KNOWLEDGE BASED AGENT

- The agent must be able to:
  - Represent states, actions, etc.
  - Incorporate new percepts
  - Update internal representations of the world
  - Deduce hidden properties of the world
  - Deduce appropriate actions

A KNOWLEDGE BASED AGENT

- **Declarative approach** to build a knowledge based agent

- The agent operates as follows:
  - **Add new sentences:** It TELLs the knowledge base what it perceives based on what it wants to know.
  - **Query what is known:** It ASKs the knowledge base what action it should perform. The answers should follow from the KB.
  - **Execute Action:** It performs the chosen action.

- **Procedural approach** to build a knowledge based agent
  - Encode desired behaviors directly as program code
  - Minimizing the role of explicit representation and reasoning can result in a much more efficient system. In this approach, knowledge is stored into an empty system in the form of program code. It designs the behavior of the system via coding

Mechanism of an Agent Program

LEVELS OF A KNOWLEDGE-BASED AGENT

● **Knowledge Level:** In this level, the behavior of an agent is decided by specifying the following :

- The agent's current knowledge it has percieved.
- The goal of an agent.

● **Implementation Level:** This level is the physical representation of the knowledge level. Here, it is understood that "how the knowledge-based agent actually implements its stored knowledge."

KNOWLEDGE BASED AGENT

```
function KB-AGENT(percept) returns an action
    static: 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
```

TECHNIQUES USED FOR KNOWLEDGE REPRESENTATION

- **Logic:** It is the basic method used to represent the knowledge of a machine. The term logic means to apply intelligence over the stored knowledge.

Logic can be further divided as:

1. **Propositional Logic:** This technique is also known as **propositional calculus, statement logic, or sentential logic**. It is used for representing the knowledge about **what is true and what is false**.

2. **First-order Logic:** It is also known as **Predicate logic or First-order predicate calculus (FOPL)**. This technique is used to represent the objects in the form of **predicates or quantifiers**. It is different from Propositional logic as it removes the complexity of the sentence represented by it. In short, FOPL is an advance version of propositional logic.

Techniques used for Knowledge Representation

- Propositional logic
- First oder logic
- Rule-based System
- Semantic Networks
- Frames
- Script

TECHNIQUES USED FOR KNOWLEDGE REPRESENTATION

3.Rule-based System: . In the rule-based system, we impose rules over the propositional logic and first-order logic techniques. If-then clause is used for this technique. **For example,** if there are two variables A and B. Value of both A and B is True. Consequently, the result of both should also be True and vice-versa. **It is represented as:** If the value of A and B is True, then the result will be True. So, such a technique makes the propositional as well as FOPL logics bounded in the rules.

4.Semantic Networks: The technique is based on storing the knowledge into the system in the form of a graph. Nodes of a graph represent the objects which exist in the real world, and the arrow represents the relationship between these objects. Such techniques show the connectivity of one object with another object. **For example,** Consider the given knowledge stored in a machine:

- **Ram has a cycle.**
- **Ram is a boy.**
- **Cycle has a bell.**
- **Ram is 12 years old.**
- **Cycle has two paddles.**
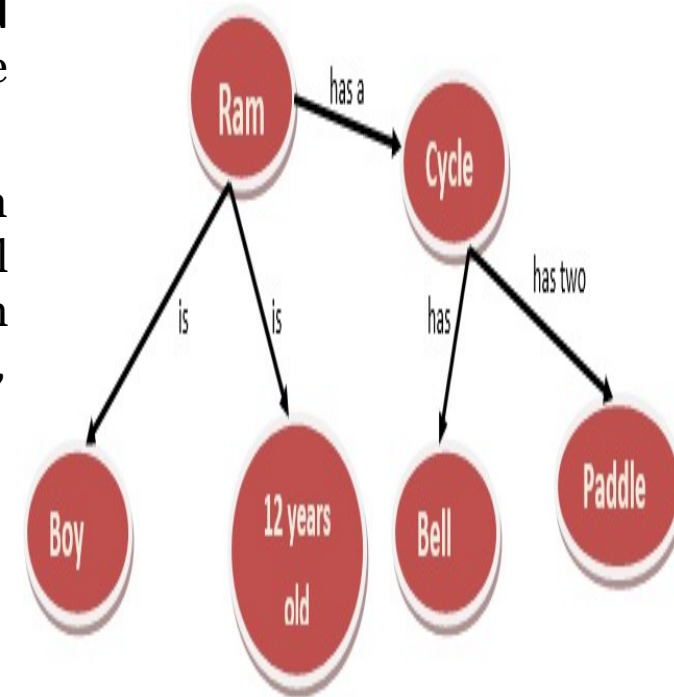
TECHNIQUES USED FOR KNOWLEDGE REPRESENTATION

5.Frames: In this technique, the knowledge is stored via **slots and fillers**. Slots are the entities and Fillers are its attributes similar to database. They are together stored in a frame. So, whenever

there is a requirement, the machine infers the necessary information to take the decision. **For example,** Tomy is a dog having one tail. It can be framed as:
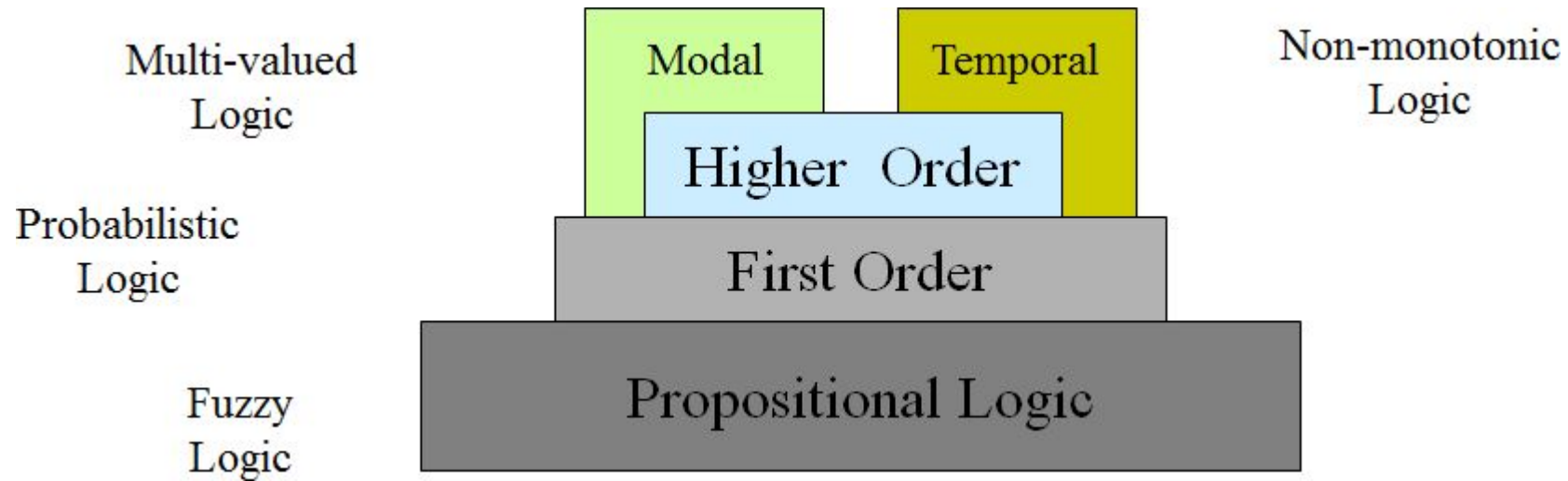
Tomy((Species (Value = Dog))

(Feature (Value = Tail)))

6.Script: It is an advanced technique over the Frames. Here, the information is stored in the form of a script. The script is stored in the system containing all the required information. The system infers the information from that script and solves the problem

LOGIC AS A KR LANGUAGE

WUMPUS WORLD

WUMPUS WORLD GAME DESCRIPTION

The Wumpus World is a cave consisting of rooms connected by passageways. Lurking somewhere in the cave is the Wumpus, a beast that eats any agent that enters its room. The Wumpus can be shot by an agent, but agent has only one arrow. Some rooms contain bottomless pits that trap any agent that wanders into the room. Occasionally, there is a heap of gold in a room. The goal is to collect the gold and exit the world <u>without being eaten.</u>
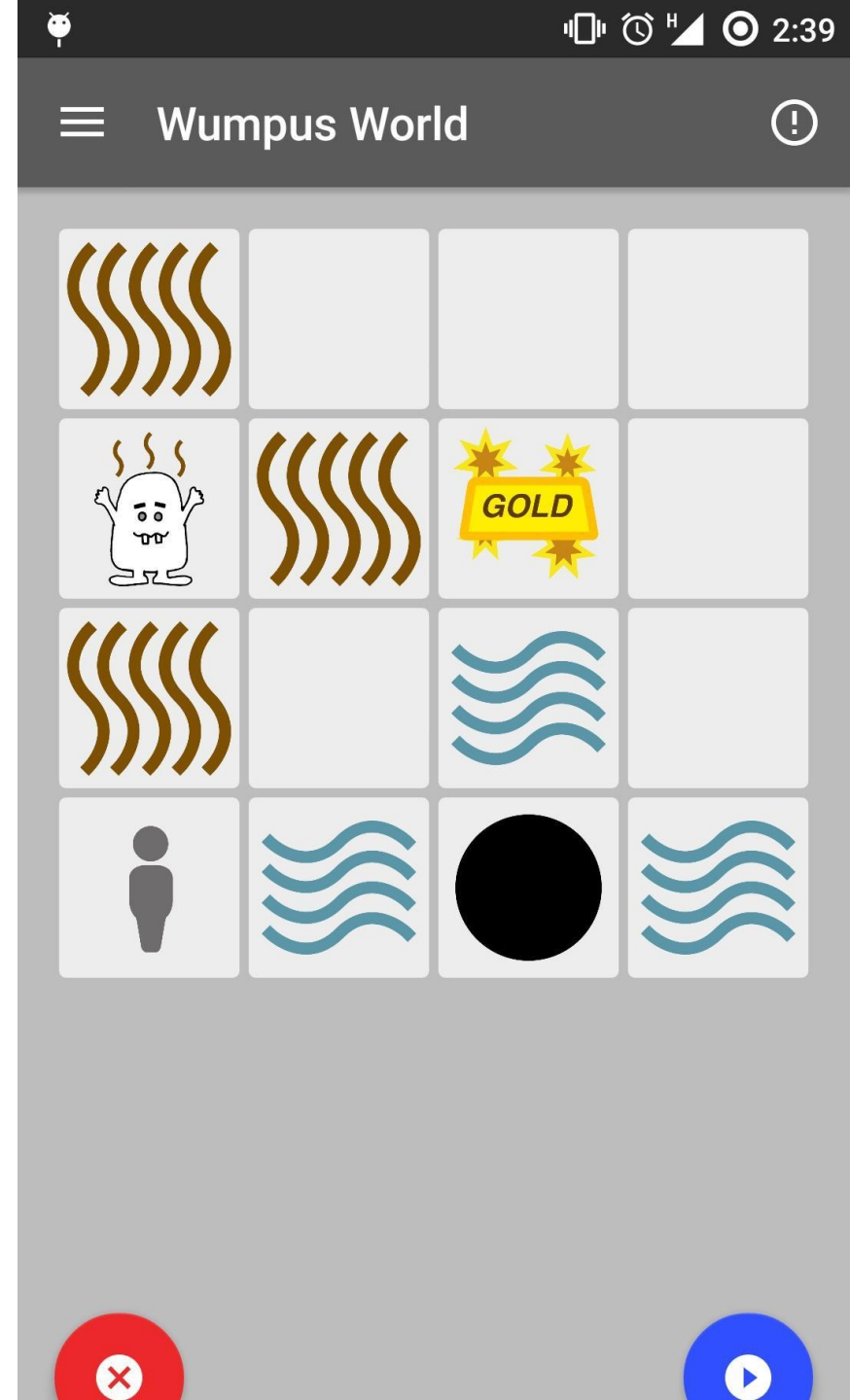
WUMPUS WORLD ENVIRONMENT

- The agent always **starts** in the **field [1,1]**.

- The task of the agent is to **find the gold**, **return to the field [1,1]** and **climb out of the cave**.

- Squares adjacent to Wumpus are smelly and squares adjacent to pit are breezy (not diagonal)

- Glitter iff gold is in the same square

- Shooting kills Wumpus if you are facing it

- Wumpus emits a horrible scream when it is killed that can be heard anywhere

- Shooting uses up the only arrow

- Grabbing picks up gold if in same square

- Releasing drops the gold in same square
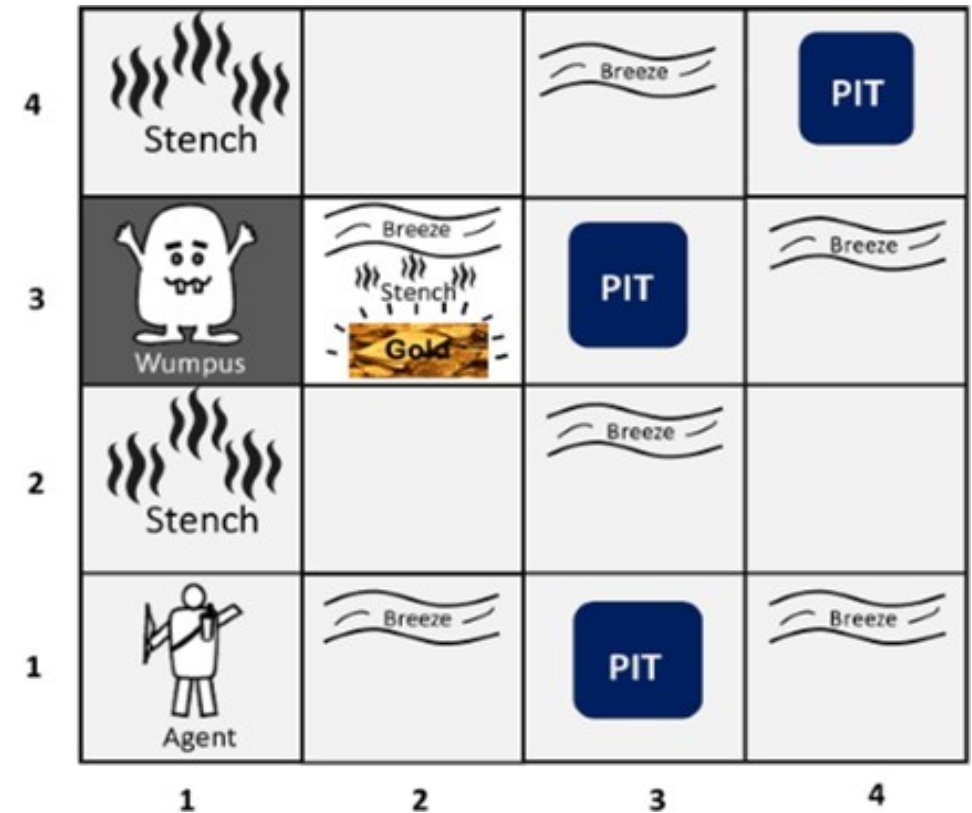
PEAS DESCRIPTION

- **Performance measure**
  - gold: +1000, death: -1000
  - -1 per step , -10 for using the arrow

- **Environment**
  - Squares adjacent to Wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Gold is picked up by reflex, can't be dropped
  - Shooting kills Wumpus if you are facing it. It screams
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square
  - You bump if you walk into a wall

- **Actuators:** Face , Move, Grab, Release, Shoot

- **Sensors:** Stench, Breeze, Glitter, Bump, , Scream

Percept->there is a stench and a breeze, but no glitter, bump, or scream, the agent program will get [Stench, Breeze, None, None, None]

WUMPUS WORLD CHARACTERIZATION

1. **Deterministic**   **Yes** – outcomes exactly specified

2. **Static**   **Yes** – Wumpus and Pits do not move

3. **Discrete**   **Yes**

4. **Single-agent**   **Yes** – Wumpus is essentially a natural feature

5. **Fully Observable**   **No** – only local perception

6. **Episodic**   **No**—What was observed before (breezes, pits, etc) is very useful.

EXPLORING THE WUMPUS WORLD
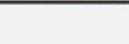
(a)

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

(b)

1. The KB initially contains the rules of the environment.
2. **Location: [1,1]**
   Percept: [¬Stench, ¬Breeze, ¬Glitter, ¬Bump]=**[None, None, None, None, None]**
   Action: Move to safe cell e.g. 2,1
3. **Location: [2,1]**
   Percept: [¬Stench, Breeze, ¬Glitter,¬ Bump]
   INFER: Breeze indicates that there is a pit in [2,2] or [3,1]
   Action: Return to [1,1] to try next safe cell

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

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

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

(a)

(b)

4. **Location: [1,2]** (after going through [1,1])
   Percept: [Stench, ¬Breeze, ¬Glitter, ¬Bump]
   INFER: Wumpusis in [1,1] or [2,2] or [1,3]
   INFER... stench not detected in [2,1], thus not in [2,2]
   REMEMBER....Wumpus not in [1,1]
   THUS... Wumpus is in [1,3]
   THEREFORE[2,2] is safe because of lack of breeze in [2,1]
   Action: Move to [2,2]
   REMEMBER: Pit in [2,2] or [3,1] (as breeze was found in [2,1]
   ~~**THEREFORE:** Pit in [3,1]!~~
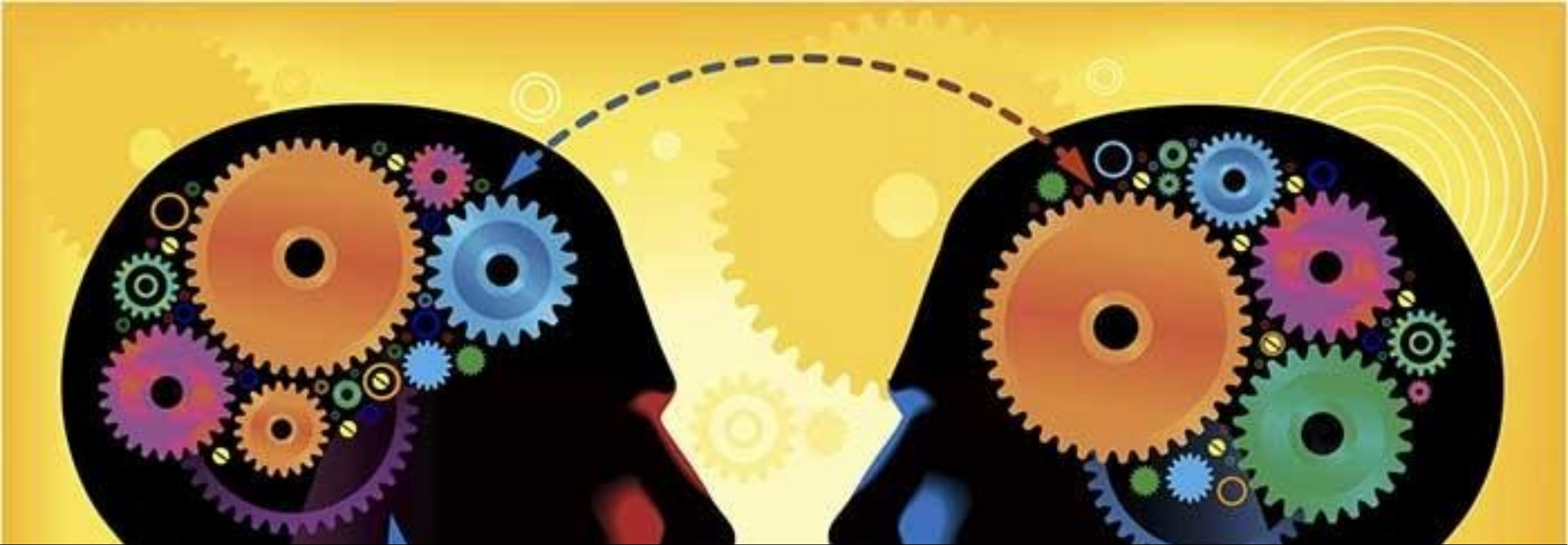
LOGIC

LOGIC

- The objective of **knowledge representation** is to express knowledge in a **computer-tractable form,** so that agents can perform well.

- *Logics are formal languages for representing information such that conclusions can be drawn.*

- A **formal knowledge representation language** is defined by:
    - its **syntax**, which **defines all possible sequences of symbols that can be put together to constitute sentences of the language.**
    - its **semantics**, which **determines the facts in the world to which the sentences refer.** It define the **"meaning"** of sentences.

- Each **sentence** **makes a claim about the world**. An agent is said to believe a sentence about the world.

- E.g., the language of arithmetic
    - x+2 ≥ y is a sentence; x2y +> {} is not a sentence
    - x+2 ≥ y is true iff the number x+2 is no less than the number y
    - x+2 ≥ y is true in a world where x = 7, y = 1
    - x+2 ≥ y is false in a world where x = 0, y = 6

INFERENCING WITH KNOWLEDGE AND ENTAILMENT

- **Inferencing** is how we derive:
  - **Conclusions** from **existing knowledge;**
  - **New information** from **existing information.** Inferencing might be used in both ASK and TELL operations.
- **Entailment is the generation or discovery that a new sentence is TRUE given existing sentences. Entailment means that one thing follows logically from another. Entailment** is a **relationship between sentences** (i.e., syntax) that is **based on semantics.** Knowledge base KB entails sentence α if and only if α is true in all worlds where KB is true ie, **KB $\models \alpha$**
- **E.g.**
  1. KB containing "**the Phillies won**" and "**the Reds won**" entails "**Either the Phillies won or the Reds won**".
  2. **KB containing "the Giants won and the Reds won " entails "The Giants won".**
  3. **x+y = 4** entails **4 = x+y**

MODELS

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

- We say 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 $KB \models \alpha$ iff $M(KB) \models M(\alpha)$

- E.g.

  1. *KB* = **Phillies won and Yankees won**

  2. α = **Phillies won**
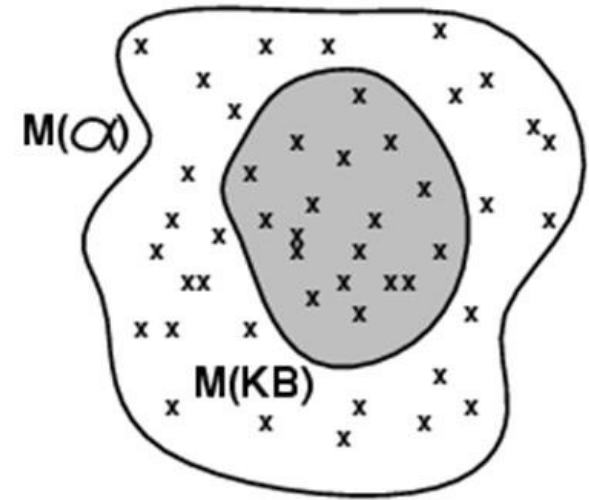
THE CONNECTION BETWEEN SENTENCES AND FACTS
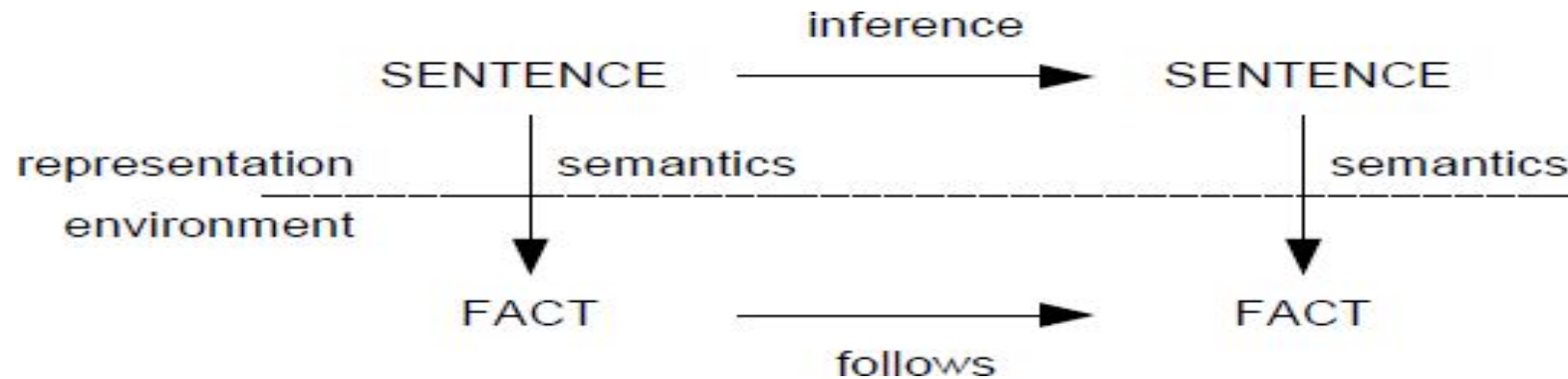
- **Semantics** maps **sentences in logic to facts** in the world.

- The property of one fact following from another is mirrored by the property of **one sentence being entailed**
  by another.

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

ENTAILMENT IN THE WUMPUS WORLD

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

- Consider possible models for *KB* assuming only pits and a reduced Wumpus world

- 3 Boolean choices$\Rightarrow$ 8 possible models/ways to fill in the ?' s.
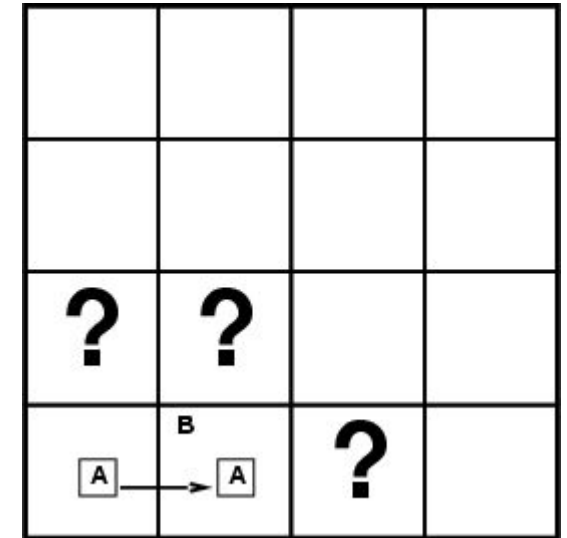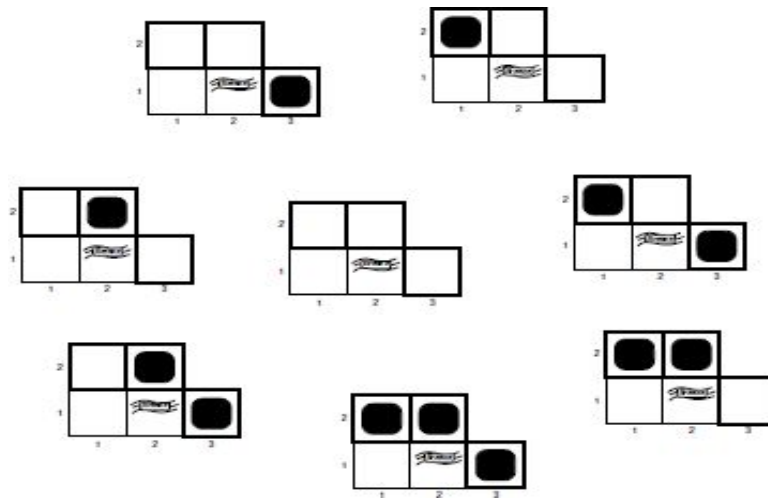
WUMPUS MODELS

KB = all possible Wumpus-worlds consistent with the
observations and the "physics" of the Wumpus world

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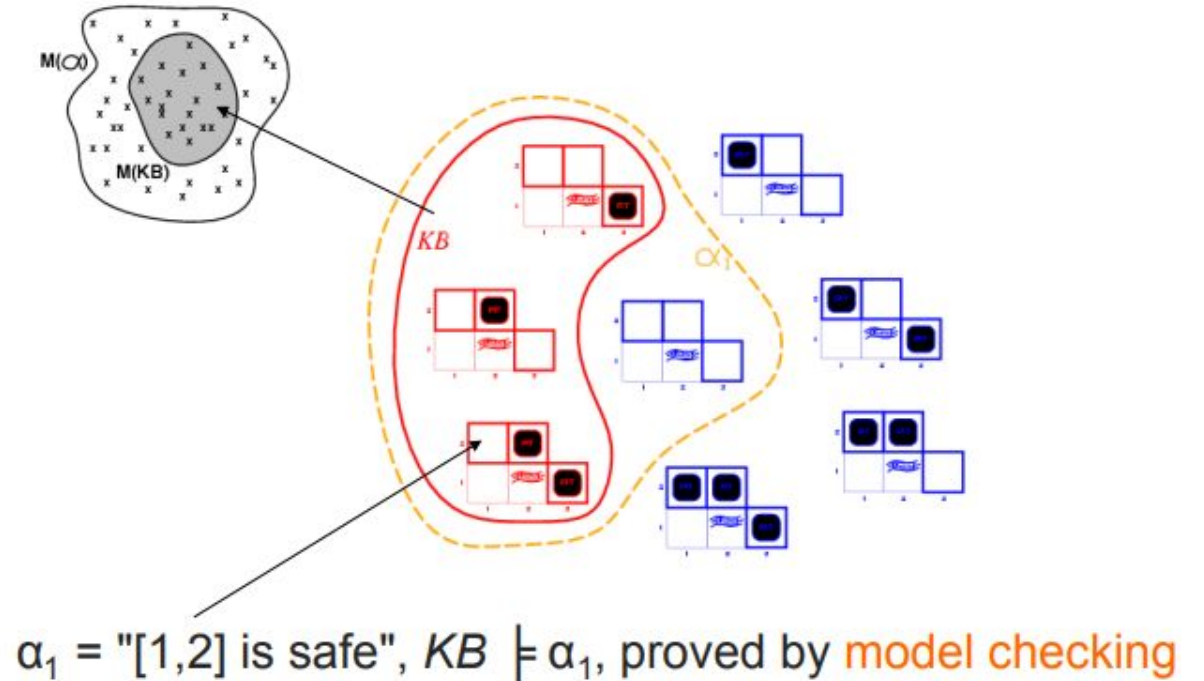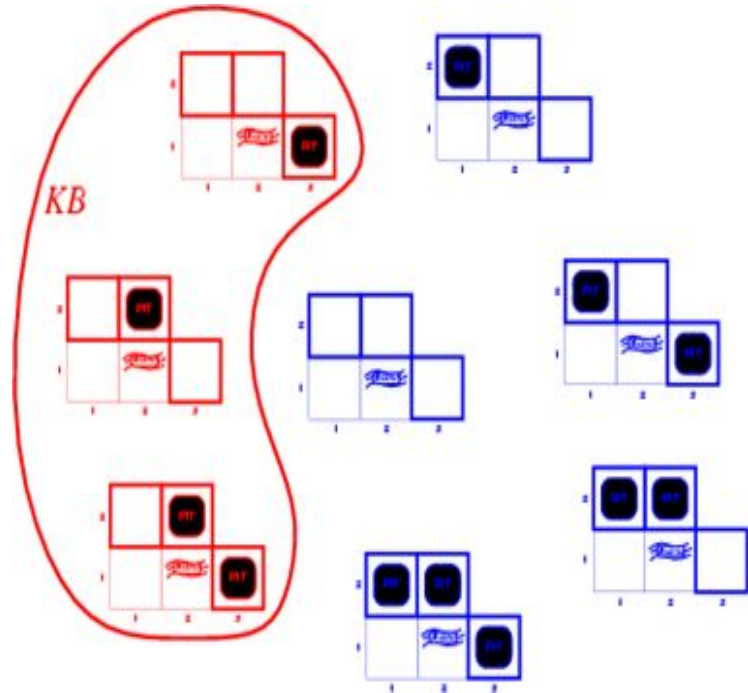
•*KB* = wumpus-world rules + observations

WUMPUS MODELS
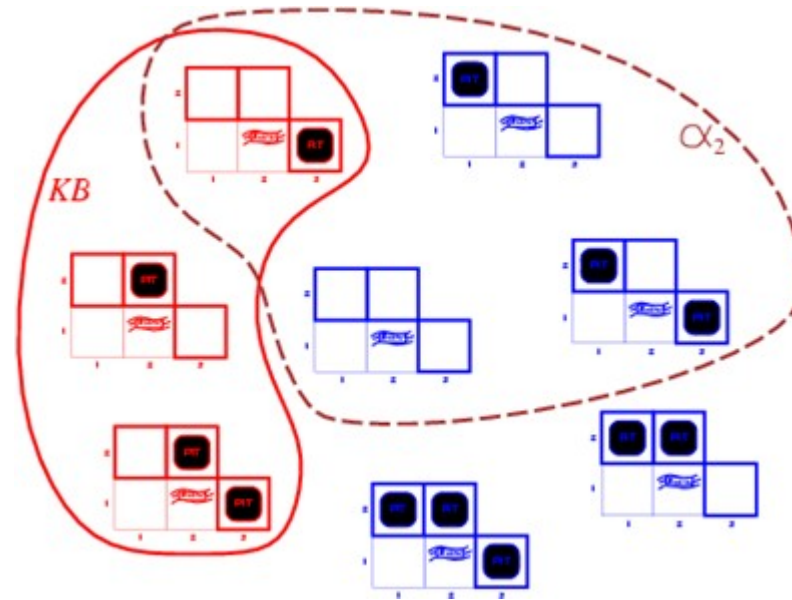
- α_2 = " [2,2] is safe with no pit ", $KB \models \alpha_2$

SOUNDNESS AND COMPLETENESS

- A **sound** inference method derives only entailed sentences. Ie, $KB \vdash_i \alpha$ = sentence α can be derived

  from KB by inference procedure I

- **Soundness:** i is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$ (i.e., no wrong inferences, but maybe not all inferences)

- Analogous to the property of **completeness** in search, a *complete* inference method can derive any sentence that is entailed.

- **Completeness:** i is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$ (i.e., all inferences can be made, but maybe some wrong extra ones as well)

- Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure. That is, the procedure will answer any question whose answer follows from what is known by the KB.

PROPOSITIONAL LOGIC

- **Propositional Logic** also known as simply "Boolean logic"is a method to achieve **knowledge representation** and **logical inferencing**.

- Propositional logic consists of **Syntax** and **Semantics**

SYNTAX

- The **symbols** and the **connectives** together define the **syntax of the language**. Again, syntax is like grammar.

- **TRUTH SYMBOLS:** T (true) and F (false) are provided by the language. Either T or F.

- **PROPOSITIONAL SYMBOLS:** P, Q, R, etc. mean something in the environment. Proposition symbols are **sentences**.

- E.g: **P** means "It is hot", **Q** means "It is humid", **R** means "It is raining", "If it is hot and humid, then it is raining" **P^Q=>R**

- **Syntax** can have:
    - **ATOMIC SENTENCE:** Truth and propositional symbols are considered ATOMIC SENTENCES. Atomic sentences must have truth assigned (i.e., be assigned T or F).
    - **COMPLEX SENTENCES:** More complex sentences are formed using connectives. Sentences formed in this way can be called **Well-Formed Formula (WFF**). The evaluation of complex sentences is done using truth tables for the connectives.

SEMANTICS

- Need to be able to evaluate sentences to true or false. The **truth tables** define the **semantics of the language.**

LOGICAL CONNECTIVES

- ¬ or **NOT** or **NEGATION**: If S1 is a sentence, then ¬S1 is a sentence

- **∧** or **AND** or **CONJUNCTION**: If S1, S2 are sentences, then S1 ∧ S2 is a sentence

- **∨** or **OR** or **DISJUNCTION**: If S1, S2 are sentences, then S1 ∨ S2 is a sentence

- ⇒ or **IFTHEN** or **IMPLICATION**: If S1, S2 are sentences, then S1 ⇒ S2 is a sentence

- ⇔ or **IFF** or **BICONDITIONAL**: If S1, S2 are sentences, then S1 ⇔ S2 is a sentence

- Parentheses can be used to indicate precedence.

- **KB is conjunction (AND) of all facts.**

PROPOSITIONAL LOGIC TRUTH TABLE

P	Q	¬P	P∧Q	P ∨ Q	P⟹ Q	P⟺ 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

PRECEDENCE OF OPERATORS

- Just like arithmetic operators, there is an operator precedence when evaluating logical operators as
  follows:

1. **Expressions in parentheses are processed (inside to outside)**

2. **Negation**

3. **AND**

4. **OR**

5. **Implication**

6. **Biconditional**

7. **Left to right**

- Use parentheses whenever you have any doubt!

PROPOSITIONAL LOGIC EXAMPLES

- **Example 1: If it is humid, then it is raining.**
  - P=It is humid. And Q=It is raining.
  - It is represented as **(P→Q).**

- **Example 2: It is noon and Ram is sleeping.**
  - **Solution:** A= It is noon. And B= Ram is sleeping.
  - It is represented as **(A Λ B).**

- **Example 3: If it is raining, then it is not sunny.**
  - **Solution:** P= It is raining. And Q= It is sunny.
  - It is represented as **P → (~Q)**

- **Example 4: Ram is a man or a boy.**
  - **Solution:** X= Ram is a man. And Y= Ram is a boy.
  - It is represented as **(X V Y).**

- **Example 5: I will go to Delhi if and only if it is not humid.**
  - **Solution:** A= I will go to Delhi. And B= It is humid.
  - It is repesentation as **(A ⇔ ~ B).**

HOW CAN WE REPRESENT THE WUMPUS WORLD?

- We can represent the **Wumpus world** (things we know and things we discover) in terms of logic
  as follows:

- Consider the propositional symbols (partial formulation):
  - **P(i,j)** is T if there is a **pit in (I,J)**, otherwise F.
  - **B(i,j)** is T if there is a **breeze in (I,J)**, otherwise F.

- We can update as we explore:
  - **¬B(1,1)** – no breeze in square (1,1).
  - **B(2,1)** – breeze in square (2,1).
  - **¬P(1,1)**- no pit in starting square.

- "Pits cause breezes in adjacent squares"

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

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

LOGICAL EQUIVALENCE

- Two sentences are **logically equivalent**, denoted by $\alpha \equiv \beta$ iff they are true in the same models,
  i.e., iff: $\alpha \models \beta$ and $\beta \models \alpha$.

- If the **value of P** and **Q** is true in the same set of models, then they are said to be **logically equivalence.**

- It can be used as **inference rule**s in **both directions.**

Example

- $(A \Rightarrow B) \equiv (\neg B \Rightarrow \neg A)$ (contraposition)

INFERENCE RULES WITH LOGICAL EQUIVALENCES

Rule Name	Rule
Idempotency Law	$(A \wedge A) \equiv A$ $(A \vee A) \equiv A$
Commutative Law	$(A \wedge B) \equiv (B \wedge A)$ $(A \vee B) \equiv (B \vee A)$
De morgan's Law	$\sim(A \wedge B) \equiv \sim A \vee \sim B)$ $\sim(A \vee B) \equiv (\sim A \wedge \sim B)$
Associative Law	$A \vee (B \vee C) \equiv (A \vee B) \vee C$ $A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$
Distributive Law	$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$ $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$
Contrapositive Law	$A \rightarrow B \equiv \sim B \rightarrow \sim A$
Implication Removal	$A \rightarrow B = \sim A \vee B$
Biconditional Removal	$A \leftrightarrow B = (A \rightarrow B) \wedge (B \rightarrow A)$
Absorption Law	$A \wedge (A \vee B) \equiv A$ $A \vee (A \wedge B) \equiv A$
Double-negation elimination	$\sim(\sim A) = A$

INFERENCE RULES IN PROPOSITION LOGIC

- **Inference rules** are those rules which are used to **describe certain conclusions.** The inferred conclusions lead to the desired goal state.

- In propositional logic, there are various inference rules which can be applied to prove the given statements and conclude them.

$$\frac{p \qquad p \rightarrow q}{q}$$

$$\frac{\neg q \qquad p \rightarrow q}{\neg p}$$

if P and P → Q are both true then Q must be true

Modus Ponens

Modus Tollens

COMMON RULES

1. **Addition:**

$$\frac{p}{p \lor q}$$

2. **Simplification:**

$$\frac{p \land q}{q}$$

3. **Disjunctive-syllogism:**

$$\frac{\begin{array}{c} p \lor q \\ \neg p \end{array}}{q}$$

4. **Hypothetical-syllogism:**

$$\frac{\begin{array}{c} p \to q \\ q \to r \end{array}}{p \to r}$$

Hypothetical Syllogism can be represented as: If **(P→Q) ∧ (Q→R)= (P→R)**

5. **Introduction:**

$$\frac{\begin{array}{c} A \\ B \end{array}}{A \land B}$$

6. **And Elimination:**

$$\frac{A \land B}{A}$$

VALIDITY AND SATISFIABILITY

- **Validity:** If a sentence is **valid in all set of models**, then it is a valid sentence. Validity is also known as **tautology,** where it is necessary to have true value for each set of model.

Eg: **A V ¬A, A ⇒ A,**

- **Satisfiability:** If a sentence is true **atleast for some set of value**s, it is a **satisfiable sentence.**

- It can be done by **truthtable enumeration.**

- **(P V Q)   →   (P Λ Q)**

P	Q	P V Q	P Λ Q	(P V Q) → (P Λ Q)
False	False	False	False	True
False	True	True	False	False
True	False	True	False	False
True	True	True	True	True

- from the above truth table, it is clear that the given expression is satisfiable but not valid.

EXAMPLE 2:

- **((A → B) Λ A) → B**

A	B	A → B	(A → B) Λ A	((A → B) Λ A) → B
False	False	True	False	True
False	True	True	False	True
True	False	False	False	True
True	True	True	True	True

- the given expression is valid as well as satisfiable.

LOGICAL INFERENCE PROBLEM

- Given a **knowledge base KB** (a set of sentences) and a **sentence** α(called a **theorem**). Does a **KB semantically entail** α? In other words in all interpretations in which sentences in the **KB** are true, is also α true? Ie, **KB |= α** ?

- Three approaches:

  - **Truth-table approach**

  - **Deduction using Inference rules**

  - **Proof by Contradiction or Resolution-refutation**

DEDUCTION THEOREM & PROOF BY CONTRADICTION

Deduction Theorem (connects inference and validity)

- KB $\models \alpha$ if and only if KB $\Rightarrow \alpha$ is valid

Proof By Contradiction or Refutation or reductio ad absurdum

- KB $\models$ a is valid if and only if the sentence KB$\land\neg\alpha$ is a contradiction.

- **Monotonic**
  - If we have a proof, adding information to the DB will not invalidate the proof ie set of entailed
    sentences can only increase information to KB.

DEDUCTION EXAMPLE

- **P:** "It is hot", **Q :** "It is humid" and **R :** "It is raining". (SYMBOLS).

- Given **KB** as:
  1. "If it is hot and humid, then it is raining": **P^Q=>R**
  2. "If it is humid, then it is hot": **Q=>P**
  3. "It is humid"**: Q**

- **Question:** Is it raining? (i.e., is R entailed by KB?)

SOLUTION

Step		Reason
1	Q	(premise)
2	$Q \Rightarrow P$	(premise)
3	P	(modus ponens) (1,2)
4	$(P \wedge Q) \Rightarrow R$	(premise)
5	$P \wedge Q$	(and-intro) (1,3)
6	R	(and-elim) (4,5)

CHALLENGE

- Given KB.
    - P∧ Q
    - P→ R
    - Q∧ R →S

- Can you conclude
  S

SOLUTION

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

PROOF BY CONTRADICTION(Resolution Refutation)

- Assume our conclusion is false, and look for a contradiction. If found, the opposite of our

  assumption must be true.

1. A   ion R

1	P ∨ Q
2	P → R
3	Q → R

SOLUTION

Step	Formula	Derivation
1	P v Q	Given
2	P → R	Given
3	Q → R	Given
4	¬ R	Negated Conclusion
5	Q v R	1,2
6	¬ P	2,4
7	¬ Q	3,4
8	R	5,7
9	F	4,8

FORMALIZING THE WW IN PL

- The Wumpus World knowledge base:

- There is no pit in [1, 1] (agent percept): $R_1 : \neg P_{11}$

- A square is breezy if and only if there is a pit in a neighboring square. (Rule of the WW). We state this for the square $B11$ only: $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$

- There is no breeze in square [1; 1]. (agent percept) $R_3 : \neg B11$

- The agent can now use the PL inference rules and logical equivalences to prove the following: **There is no pit in squares [1,2] or [2, 1]**

- **Theorem:** $\neg P12 \land \neg P21$

FORMALIZING THE WW IN PL

- Apply biconditional elimination to R_2:

- **R4 : (B11 $\Rightarrow$ (P12 $\vee$ P21)) $\wedge$ ((P12 $\vee$ P21) $\Rightarrow$ B11)**

- Apply And-elimination to R_4:

- **R5 : (P12 $\vee$ P21) $\Rightarrow$ B11**

- Apply logical equivalence for contrapositives to R_5:

- **R6 : $\neg$ B11 $\Rightarrow$ $\neg$(P12 $\vee$ P21)**

- Apply modus ponens to *R*6 and *R*3:

- **R7 : $\neg$ (P12 $\vee$ P21)**

- Apply de Morgan's rule to *R*7:

- **R8 : $\neg$ P12 $\wedge$ $\neg$ P21**

KB IN RESTRICTED FORMS

- If the sentences in the KB are restricted to some special forms some of the sound inference rules may become complete

- **Example:**

- **Horn form (Horn normal form)**

- **CNF (Conjunctive Normal Forms)**

PROPOSITIONAL THEOREM PROVING

- **Searchfor proofs** is a more efficient way than enumeratingmodels (Wecan ignore irrelevant information). Truth tables have an exponential number of models.

- The idea of inference is to repeat applying inference rules to the KB.

- Inference can be applied whenever suitable premises are found in the KB

- **Theorem proving** **means to apply rules of inference directly to the sentences.**

- Two ways to ensure completeness:

  1. **Proof by resolution**: use sequence of powerful inference rules (resolution rule) and construction
     of / search for a proof. **Resolution** works best when the formula is of the special form **CNF**.

     Properties

     - Typically requires translation of sentences into a normal form.

  2. **Forward or Backward chaining**: use of modus ponens on a restricted form of propositions (**Horn clauses**)

NORMAL FORMS

- **Literal:** A literal is an atomic sentence (propositional symbol), or the negation of an atomic sentence. Eg:- p (positive literal), $\neg p$ (negative literal)

- **Clause:** A disjunction of literals. Eg:- $\neg p \lor q$

- **Conjunctive Normal Form (CNF):** A conjunction of disjunctions of literals, i.e., a conjunction of clauses Eg:- $(A \lor \neg B) \land (B \lor \neg C \lor \neg D)$

- **DNF(Disjunctive Normal Form):** This is a reverse approach of CNF which is disjunction of conjunction of literals. Eg:- $(A1 \land B1) \lor (A2 \land B2) \lor ... \lor (An \land Bn)$

- In **DNF**, it is OR of AND's, a sum of products, or a cluster concept, whereas, in **CNF**, it is ANDsof OR's a product of sums.

CNF TRANSFORMATION

- In propositional logic, the resolution method is applied only to those clauses which are disjunction of literals. **There are following steps used to convert into CNF:**

1) Eliminate **bi-conditional implication** by replacing **A ⇔ B** with **(A → B) ⋀ (B →A)**

2) Eliminate **implication** by replacing **A → B** with **¬A V B.**

3) In CNF, negation(¬) appears only in literals, therefore we **move negation inwards** as:

      ¬ (¬A) ≡ A (double-negation elimination

      ¬ (A ⋀ B) ≡ (¬A V ¬B) (De Morgan)

      ¬(A V B) ≡ (¬A ⋀ ¬B) (De Morgan)

4) Finally, **using distributive law** on the sentences, and form the CNF as:

      $(A_1 \lor B_1) \land (A_2 \lor B_2) \land \ldots \land (A_n \lor B_n)$.

- **Note: CNF can also be described as AND of ORS**

- Transform to CNF: **$B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$**

CNF TRANSFORMATION EXAMPLE

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

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$.

   $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \lor \beta$.

   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1})$

3. Move $\neg$ inwards using de Morgan's rules and double-negation:

   $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1})$

4. Apply distributivity law ($\land$ over $\lor$) and flatten:

$(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$

METHOD 1: RESOLUTION METHOD IN FOL

- In propositional logic, **resolution method is by application of inference rule gives a new clause when two or more clauses and are coupled together to prove theorem**.

- Using propositional resolution, it becomes easy to make a theorem prover sound and complete for all. **The process followed to convert the propositional logic into resolution method is known as** Resolution refutation **contains the below steps:**

1. **Convert the given axiom(all sentences) into clausal form, CNF.**

2. **Negate the desired conclusion (converted to CNF)**

3. **Apply resolution rule until either – Derive false (a contradiction) – Can't apply any more**

4. **If we derive a contradiction, then the conclusion follows from the axioms**

5. **If we can't apply any more, then the conclusion cannot be proved from the axioms.**

- This is known as **resolution Algorithm**.

- **Resolution refutation** is **sound** and **complete.**

EXAMPLE

- Prove R from:

1	(P → Q) → Q
2	(P → P)→ R
3	(R → S) → ¬ (S → Q)

SOLUTION

- Convert to CNF

1. $(P \rightarrow Q) \rightarrow Q \equiv \neg(\neg P \vee Q) \vee Q$

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

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

    $\equiv (P \vee Q) \wedge T$

2. $(P \rightarrow P) \rightarrow R \equiv \neg(\neg P \vee P) \vee R$

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

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

3. $(R \rightarrow S) \rightarrow \neg(S \rightarrow Q) \equiv \neg(\neg R \vee S) \vee \neg(\neg S \vee Q)$

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

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

    $\equiv (R \vee S) \wedge T \wedge (R \vee \neg Q) \wedge (\neg S \vee \neg Q)$

| 1	P v Q	
2	P v R	
3	¬ P v R	
4	R v S	
5	R v ¬ Q	
6	¬ S v ¬ Q	
7	¬ R	Neg
8	S	4,7
9	¬ Q	6,8
10	P	1,9
11	R	3,10
12	F	7,11

PROPOSITIONAL RESOLUTION EXAMPLE

- Consider the following Knowledge Base:

    1. The humidity is high or the sky is cloudy.

    2. If the sky is cloudy, then it will rain.

    3. If the humidity is high, then it is hot.

    4. It is not hot.

- **Goal:** It will rain.

- Use propositional logic and apply resolution method to prove that the goal is derivable from the
  given knowledge base.

SOLUTION

- **Solution:** Let's construct propositions of the given sentences one by one:

Let, **P:** Humidity is high. **Q:** Sky is cloudy. **R:** It will rain   **S:** It is hot.

1.   It will be represented as **P V Q.**

2.   It will be represented as **Q →   R.**

3.   It will be represented as **P   →   S.**

4.   It will be represented as **¬S.**

P V Q   ¬Q V R

P V R   ¬P V S

R V S   ¬S

R   ¬R

Ø(Null)
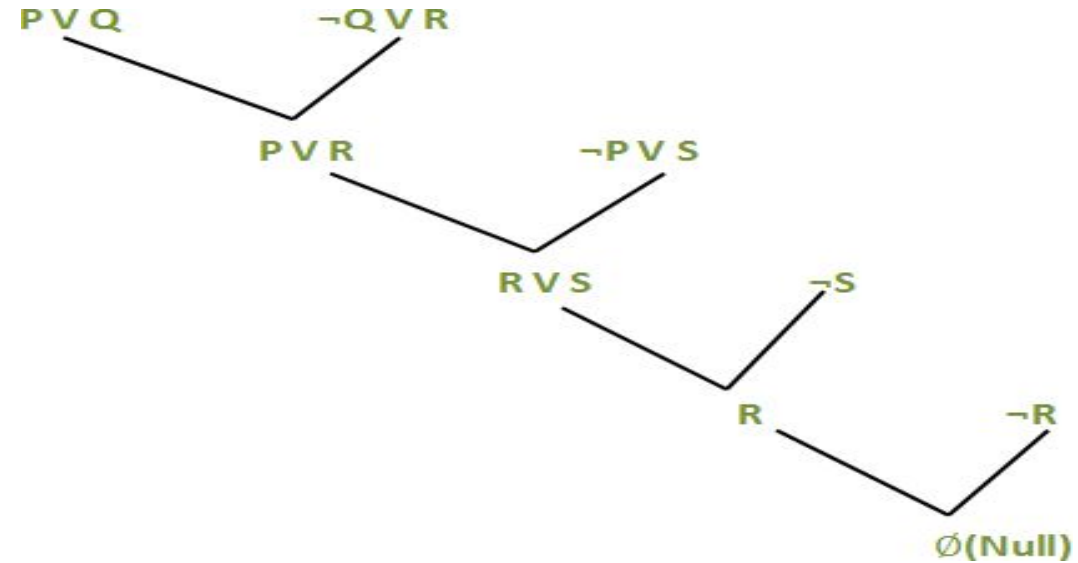
CHALLENGES

1. Given KB = $(B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$ Prove $\alpha: \neg P_{1,2}$

SOLUTION

Given KB

- **R1:** $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$

- **R2:** $\neg B_{1,1}$

- **R3:** Negation of theorem $= \neg(\neg P_{1,2}) = P_{1,2}$

- Given R1 can be split up as   **R4:** : $(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1})$ **R5:** $(\neg P_{1,2} \lor B_{1,1})$ **R6:** $(\neg P_{2,1} \lor B_{1,1})$

- Consider **R5** and **R2** apply Modus Ponens **R6:** $\neg P_{1,2}$

- Consider R6 and R3 which leads to a negation

HORN CLAUSES AND DEFINITE CLAUSES

- **DEFINITE CLAUSE:** A disjunction of literals of which **exactly one** is positive.

    - $(\neg L_{1,1} \lor \neg breeze \lor B_{1,1})$ **Yes**
    - $(\neg B_{1,1}$ **No** $\lor P_{1,2} \lor P_{2,1})$

- **HORN CLAUSE:** A disjunction of literals of which **atmost one** is positive, ie is a CNF clause with exactly one positive literal. The positive literal is called the **head**. The negative literals are called the **body**. All **definite clauses are Horn Clauses**.

    - $(\neg L_{1,1} \lor \neg breeze \lor B_{1,1})$ **Yes**
    - $(\neg B_{1,1} \lor \neg P_{1,2} \lor \neg P_{2,1})$ **Yes**
    - $(\neg B_{1,1}$ **No** $\lor P_{1,2} \lor P_{2,1})$

- Horn clauses are **closed under resolution,** ie if 2 Horn closes are resolved we get back a horn clause.

- Not all sentences in propositional logic can be converted into the Horn form

- **GOAL CLAUSE:** A clause with **no** positive literal.

    - $(\neg L_{1,1} \lor breeze \lor B_{1,1})$ **No**
    - $(\neg B_{1,1} \lor \neg P_{1,2} \lor \neg P_{2,1})$ **Yes**

HORN CLAUSES

- **Horn clauses** can be re-written as **implications ie,** logic proposition of the form: p1 ^

  ..... ^ pn $\rightarrow$ q .

- Eg: $\neg C \vee \neg B \vee A$ can be written as $C \wedge B \rightarrow A$

- **KB = conjunction of Horn clauses.**

- Modus Pone

$$\frac{B \Rightarrow A, \quad B}{A}$$

  – More general version of the rule:

$$\frac{(B_1 \wedge B_2 \wedge \ldots B_k \Rightarrow A), B_1, B_2, \ldots B_k}{A}$$

- **Inference with Horn Clauses** can be done using **forward and backward chaining algorithms**.

- The **Prolog language** is based on **Horn Clauses**.

- Deciding entailment with Horn Clauses is *linear in the size of the knowledge base.*

FORWARD AND BACKWARD CHAINING

- These algorithms are very natural and run in linear time

FORWARD CHAINING:

- Based on rule of **modus ponens**. If know P1, …,Pn& know (P1 Λ... ΛPn)→Q. Then can conclude Q. Whenever the premises of a rule are satisfied, infer the conclusion. Continue with rules that became satisfied.

- Forward chaining is also known as a forward deduction or forward reasoning method when using an inference engine. Forward chaining is a form of reasoning which start with atomic sentences in the knowledge base and applies inference rules (Modus Ponens) in the forward direction to extract more data until a goal is reached.

BACKWARD CHAINING:

- In Backward chaining, we will start with our goal predicate and then infer further rules.

- Search start from the query and go backwards.

FORWARD CHAINING

- **IDEA: It begins from facts(positive literals) in knowledge base and determines if the query can be entailed by knowledge base of definite clauses. If all premises of an implication are known its conclusion is added to set of known facts.** **Eg: Given L$_{1,1}$ and Breeze and** $(L_{1,1} \wedge \text{Breeze}) \rightarrow B_{1,1}$ is in knowledge base then **B$_{1,1}$** can be added.

- Every *inference* is an application of **modus ponens** ie $\dfrac{p1,......, pn \qquad p1 \wedge ..... \wedge pn \rightarrow q}{q}$ Can be used with forward chaining.

FORWARD CHAINING STEPS

1. Start with given proposition symbols (atomic sentence).

2. Iteratively try to infer truth of additional proposition symbols

3. Continue until
   – no more inference can be carried out, or
   – goal is reached

FORWARD CHAINING

- Fire any rule whose premises are satisfied in the KB, add its conclusion to the KB, until query is found

- **AND-OR graph: Multiple links joined by an arc indicates a conjunction where every link has to be proved, while multiple links without an arc indicates disjunction, where any link has to be proved.**

$P \Rightarrow Q$

$L \wedge M \Rightarrow P$

$B \wedge L \Rightarrow M$

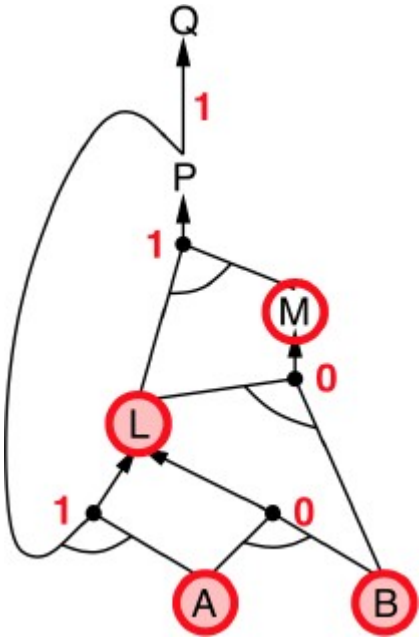$A \wedge P \Rightarrow L$

$A \wedge B \Rightarrow L$

A

B
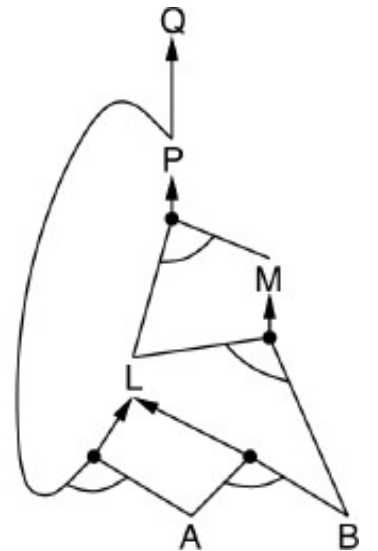
AND-OR GRAPH

FORWARD CHAINING

$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$
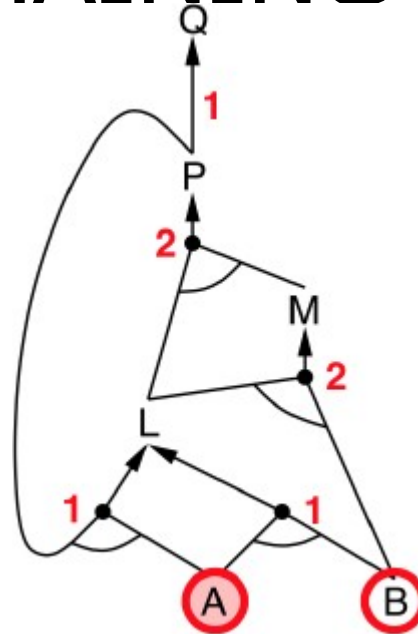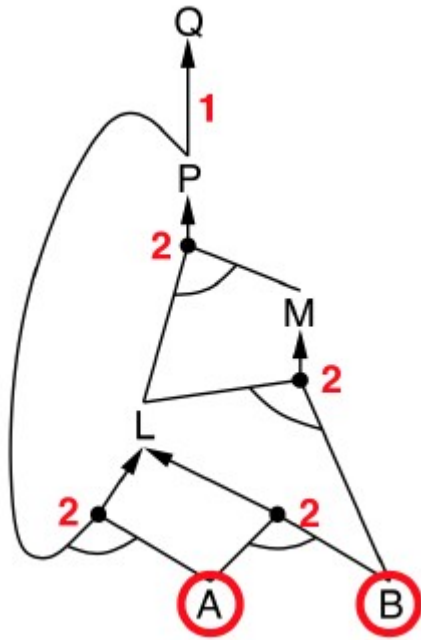
- Process agenda item A
- Decrease count for horn clauses in which A is premise
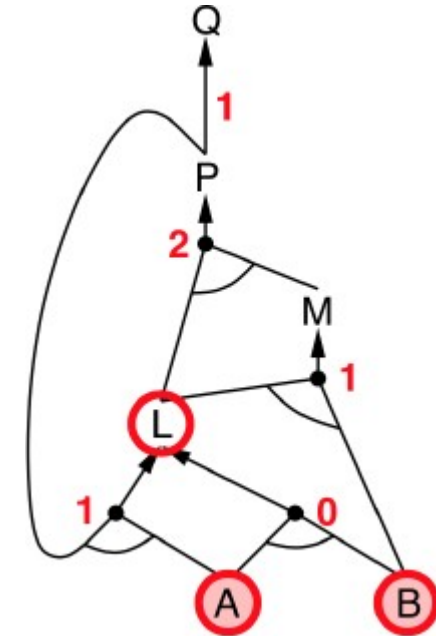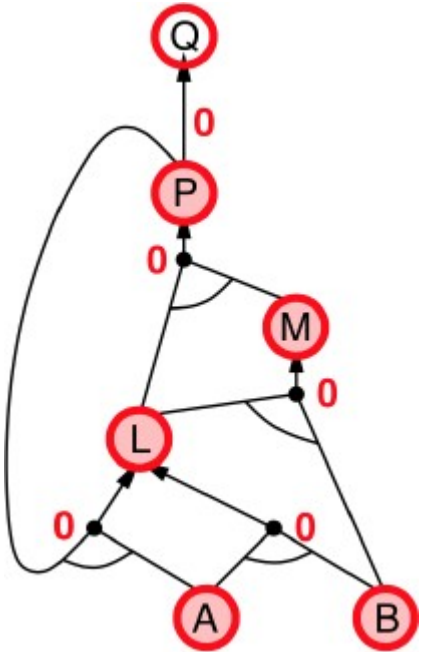
- Process agenda item B
- Decrease count for horn clauses in which B is premise
- A ^ B → L has now fulfilled premise
- Add L to agenda

FORWARD CHAINING


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

- Process agenda item L
- Decrease count for horn clauses in which L is premise
- B ^ L → M has now fulfilled premise
- Add M to agenda
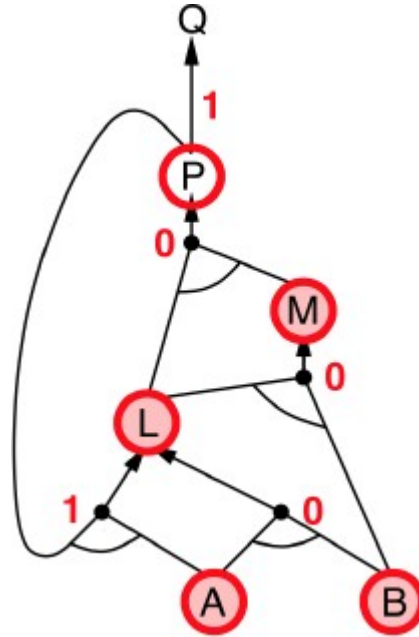
- Process agenda item M
- Decrease count for horn clauses in which M is premise
- L ^ M→P has now fulfilled premise
- Add P to agenda
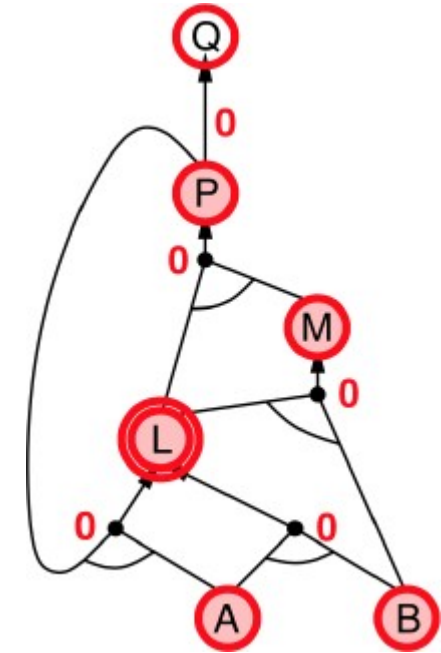
- Process agenda item P
- Decrease count for horn clauses in which P is premise
- P → Q has now fulfilled premise
- Add Q to agenda
- A ^ P → L has now fulfilled premise

FORWARD CHAINING

- Process agenda item P
- Decrease count for horn clauses in which P is premise
- P → Q has now fulfilled premise
- Add Q to agenda
- A ^ P → L has now fulfilled premise
- But L is already inferred
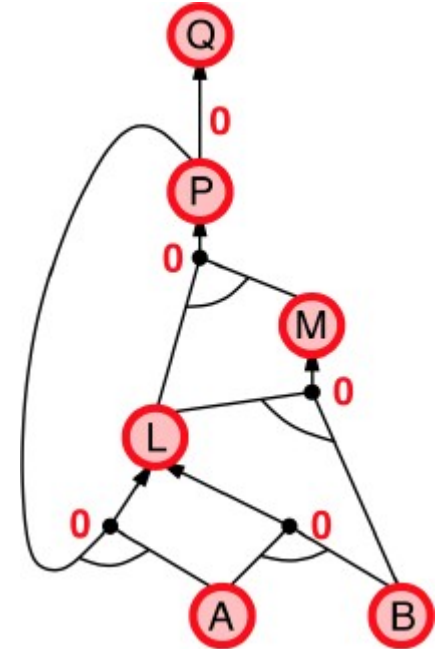
- Process agenda item Q
- Q is inferred
- Done

FORWARD CHAINING CHALLENGE

Assume the KB with the following rules and facts:

KB: R1: $A \wedge B \Rightarrow C$

      R2: $C \wedge D \Rightarrow E$

      R3: $C \wedge F \Rightarrow G$

      F1: A
      F2: B
      F3: D

Theorem: E ?

SOLUTION

Theorem: E

KB: R1: $A \wedge B \Rightarrow C$

R2: $C \wedge D \Rightarrow E$

R3: $C \wedge F \Rightarrow G$

F1: A
F2: B
F3: D
Rule R1 is satisfied.
F4: C
Rule R2 is satisfied.
F5: E ✓

BACKWARD CHAINING

- Idea: Works backwards from the query q

- to prove q by Backward Chaining:

- Check if q is known already, or

- Prove by Backward Chaining all premises of some rule concluding q

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

- Avoid repeated work: check if new subgoal

- has already been proved true, or has already failed

BACKWARD CHAINING
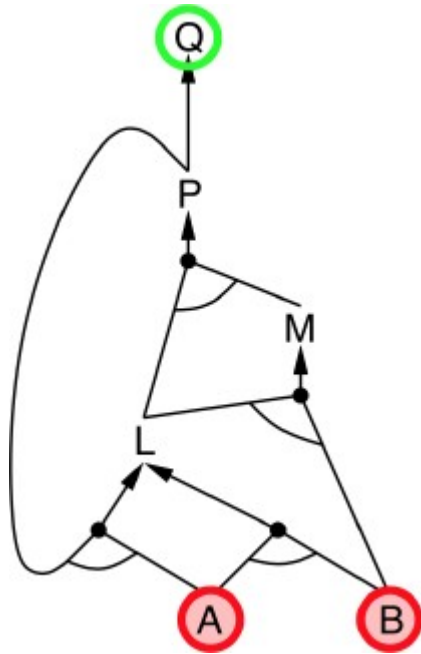
- A and B are known to be true
- Q needs to be proven
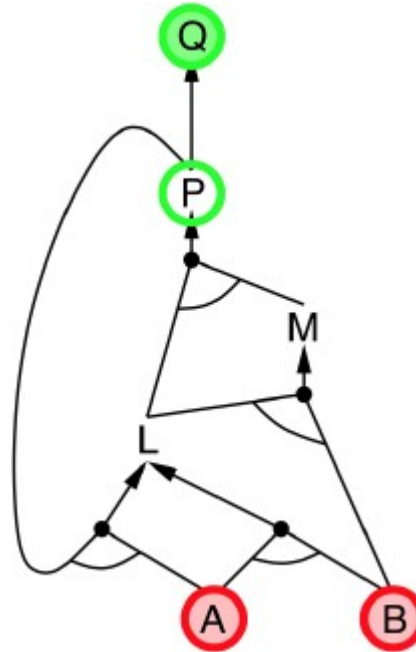
- Current goal: Q
- Q can be inferred by P → Q
- P needs to be proven
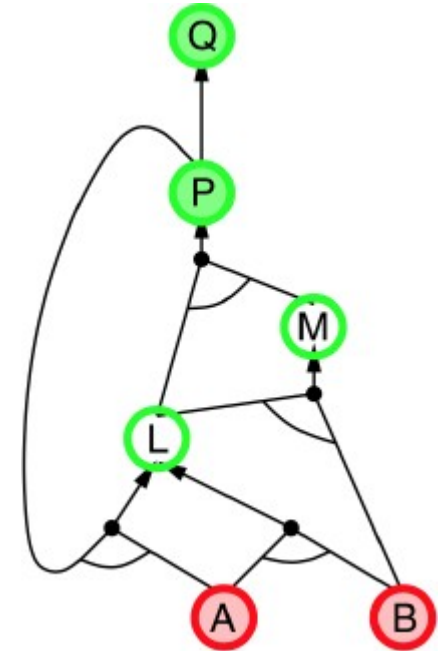
- Current goal: P
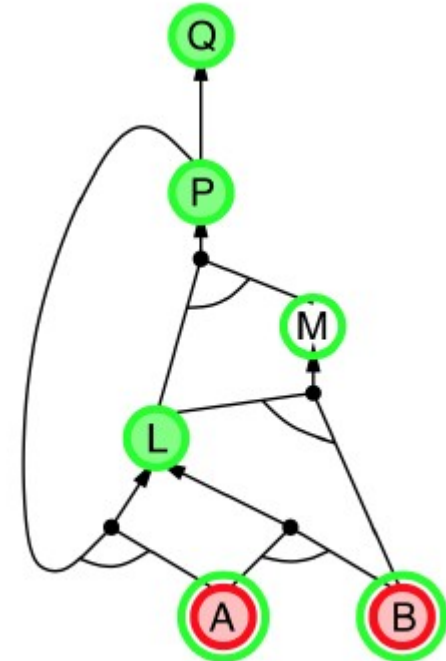- P can be inferred by L ^ M → P
- L and M need to be proven

BACKWARD CHAINING
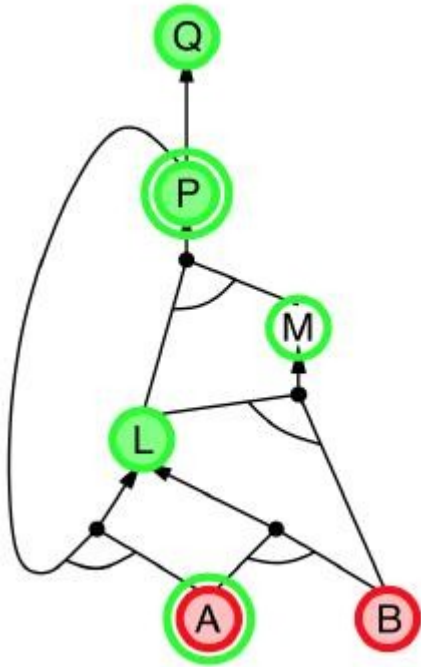
- Current goal: L
- L can be inferred by A ^ P → L
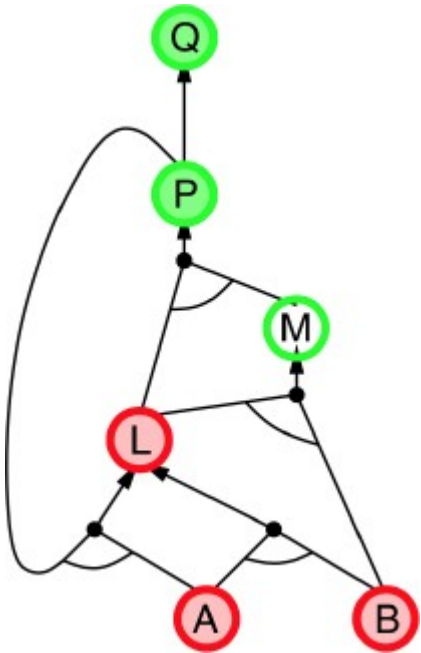- A is already true
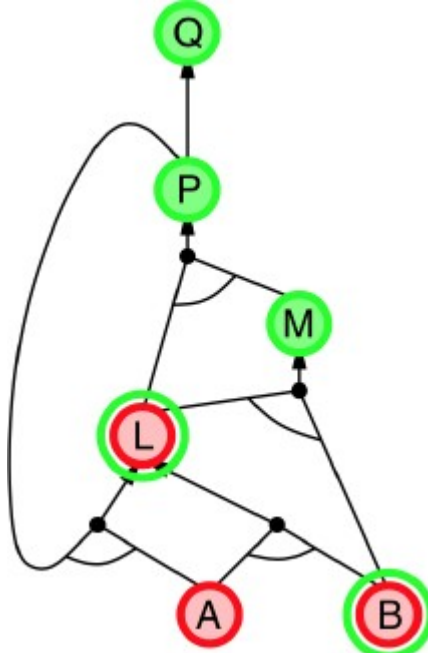- P is already a goal
- repeated sub-goal
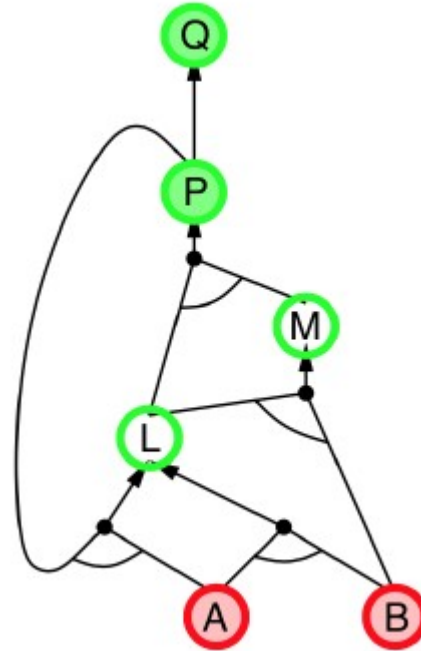
- Current goal: L

- Current goal: L
- L can be inferred by A^ B → L
- Both are true

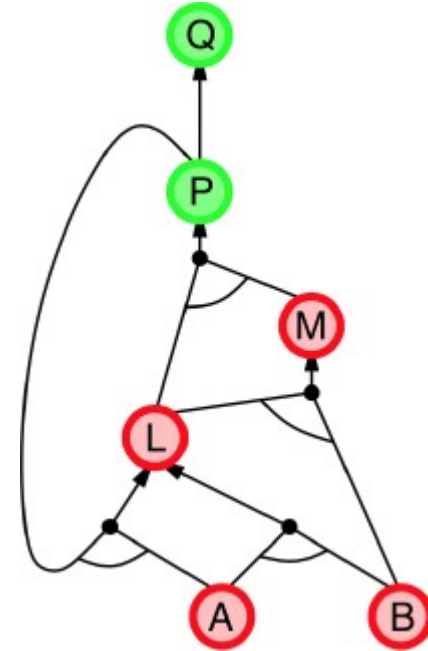# BACKWARD CHAINING

- Current goal: L
- L can be inferred by A ^ B → L
- Both are true
- L is true
- Current goal: M

- Current goal: M
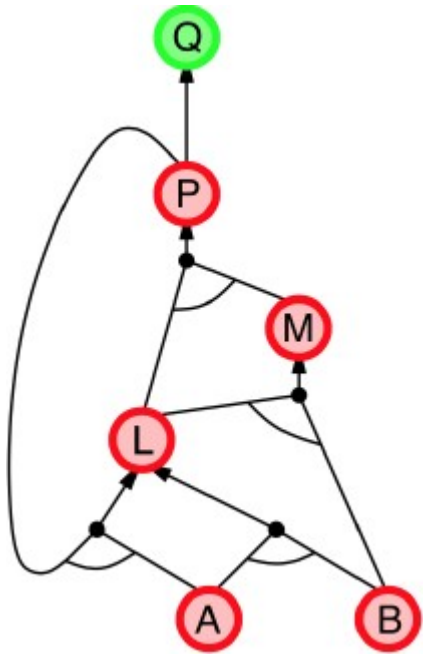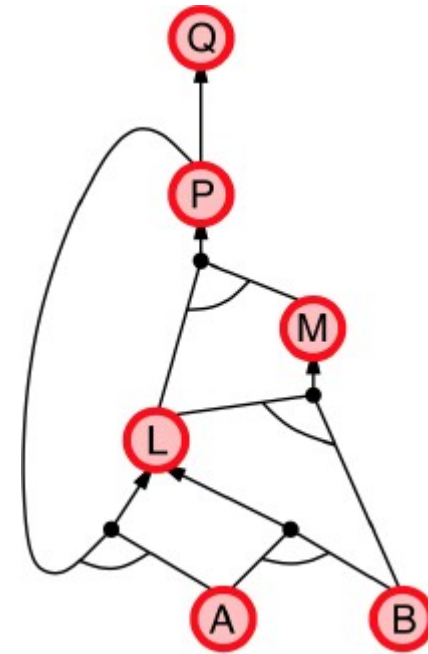- M can be inferred by B^L→ M

- Current goal: M
- M can be inferred by B ^ L→ M
- Both are true
- M is true

BACKWARD CHAINING

- Current goal: P
- P can be inferred by L ^ M → P
- Both are true
- P is true

- Current goal: Q
- Q can be inferred by P → Q
- P is true
- Q is true

FORWARD VS BACKWARD

Forward chaining:

- Data-driven, automatic, unconscious processing.

- May do lots of work that is irrelevant to the goal

Backward chaining:

- Goal-driven, appropriate for problem-solving.

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

DISADVANTAGES OF PL

- Consider now the following WW rule: If a square has no smell, then neither the square nor any of its adjacent squares can
  house a Wumpus. How can we formalize this rule in PL?

- We have to write one rule for every relevant square! For example: $\neg S11 \Rightarrow \neg W11 \wedge \neg W12 \wedge W21$

- For an example having large environment say we have a vacuum cleaner (Roomba) to clean a 1010 squares in the classroom. Use PL to express information about the squares.

- This is a very disappointing feature of PL. There is no way in PL to make a statement referring to all objects of some kind (e.g., to all squares).

LIMITATION

1. PL **is not expressive enough to describe all the world around us**. It can't express information about different object and the relation between objects.

2. **PL is not compact**. It can't express a fact for a set of objects without enumerating all of them which is sometimes impossible.

3. Propositional logic is **declarative**: pieces of syntax correspond to facts

- Not to worry: this can be done in First order logic!

THANK YOU