CHAPTER 3

Unit III

Logic and Reasoning

Syllabus Topics

Knowledge Based Reasoning : Agents, Facets of Knowledge.

Logic and Inferences : Formal Logic, Propositional and First Order Logic, Resolution in Propositional and First Order Logic, Deductive Retrieval, Backward Chaining, Second order Logic

Knowledge Representation : Conceptual Dependency, Frames, Semantic nets.

Syllabus Topic : Knowledge Based Reasoning

3.1 Knowledge Based Reasoning

- As shown Fig. 3.1.1, a knowledge based agents can be described at different levels : **Knowledge Base (KB)** and an **Inference Engine.**

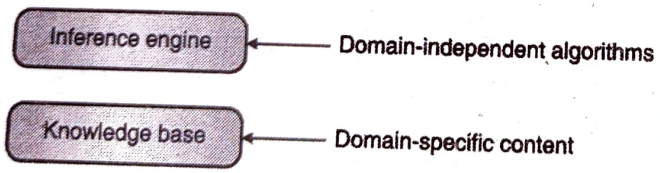


Fig. 3.1.1 : Levels of Knowledge Base

1. Knowledge level

- Knowledge level is a base level of an agent, which consists of domain-specific content.
- In this level agent has facts/information about the surrounding environment in which they are working, it does not consider the actual implementation.

2. Implementation level

- Implementation level consists of domain independent algorithms. At this level, agents can recognize the data structures used in knowledge base and algorithms which use them. For example, propositional logic and resolution. (We will be learning about logic and resolution in this chapter)
- Knowledge based agents are crucial to use in partially observable environments. Before choosing any action, knowledge based agents make use of the existing knowledge along with the current inputs from the environment in order to infer hidden aspects of the current state.
- As we have learnt that knowledge base is a set of representations of facts/information about the surrounding environment (real world).
- Every single representation in the set is called as a **sentence** and sentences are expresses with the help of formal representation language. We can say that sentence is a statement which is a set of words that express some truth about the real world with the help of knowledge representation language.
- Declarative approach of building an agent makes use of TELL and ASK mechanism.
  - TELL the agent, about surrounding environment (what it needs to know in order to perform some action). TELL mechanism is similar to taking input for a system.
  - Then the agent can ASK itself what action should be carried out to get desired output. ASK mechanism is

similar to producing output for a system. However ASK mechanism makes use of the knowledge base to decide what it should do.

- TELL and ASK mechanism involve inference. When you run ASK function, the answer is generated with the help of knowledge base, based on the knowledge which was added with TELL function previously.
- **TELL(K) :** Is a function that adds knowledge K to the knowledge base.
- **ASK(K) :** Is a function that queries the agent about the truth of K.

- An agent carries out following operations : First, it TELLs the knowledge base about facts/information it perceives with the help of sensors. Then, it ASKs the knowledge base what action should be carried out based on the input it has received. Lastly, it performs the selected action with the help of effectors.

Syllabus Topic : Agents

3.1.1 Agents

Knowledge based agents can be implemented at three levels namely, knowledge level, logical level and implementation level.

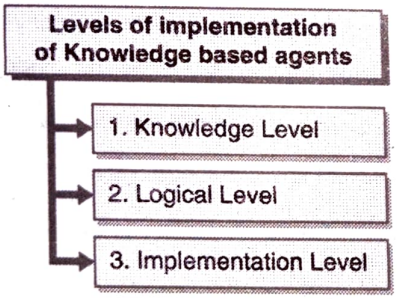


Fig. 3.1.2 : Level to implement knowledge based agents

➜ **1. Knowledge Level**

- It is the most abstract level of agent implementation. The knowledge level describes agent by saying what it knows. That is what knowledge the agent has as the initial knowledge.
- Basic data structures and procedures to access that knowledge are defined in his level. Initial knowledge of knowledge base is called as background knowledge.
- Agents at the knowledge level can be viewed as an agent for which one only need to specify what the agent knows and what its goals are in order to specify its behavior, regardless of how it is to be implemented.
- **For example :** A taxi driving agent might know that the Golden Gate Bridge connects San Francisco with the Marin county.

➜ **2. Logical Level**

- At the logical level, the knowledge is encoded into sentences. This level uses some formal language to represent the knowledge the agent has. The two types of representations we have are propositional logic and first order or predicate logic.
- Both these representation techniques are discussed in detail in the further sections.
- **For example :** Links (Golden Gate Bridge, San Francisco, Marin County)

➜ **3. Implementation Level**

- In implementation level, the physical representation of logical level sentences is done. This level also describes data structures used in knowledge base and algorithms that used for data manipulation.
- **For example :** Links(GoldenGate Bridge, SanFrancisco, MarinCounty)

```
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
    returns action
```

Fig. 3.1.3 : General function of knowledge based agent

- Fig. 3.1.3 is the general implementation of knowledge based agent. TELL and ASK are the sub procedures implemented to perform the respective actions.
- The knowledge base agent must be able to perform following tasks :
  - Represent states, actions, etc.
  - Incorporate new precepts.
  - Update internal representations of the world.
  - Deduce hidden properties of the world.
  - Deduce appropriate actions.

3.2 The WUMPUS World Environment

- You have learnt about vacuum world problem, block world problem so far. Similarly we have WUMPUS world problem. Fig. 3.2.1 shows the WUMPUS world.
- WUMPUS is an early computer game also known as "Hunt the Wumpus". WUMPUS was developed by Gregory Yob in 1972/1973. It was originally written in **BASIC (Beginner's All-purpose Symbolic Instruction Code).**
- WUMPUS is a map-based game. Let's understand the game :
  - WUMPUS world is like a cave which represents number of rooms, rooms, which are connected by passage ways. We will take a 4 × 4 grid to understand the game.
  - WUMPUS is a monster who lives in one of the rooms of the cave. WUMPUS eats the player (agent) if player (agent) comes in the same room. Fig. 3.2.1 shows that room (3, 1) where WUMPUS is staying.
  - Player (agent) starts from any random position in cave and has to explore the cave. We are starting from (1, 1) position.
- There are various sprites in the game like pit, stench, breeze, gold, and arrow. Every sprite has some feature. Let's understand this one-by-one :
  - Few rooms have bottomless pits which trap the player (agent) if he comes to that room. You can see in the Fig. 3.2.1 that room (1,3), (3,3) and (4,4) have bottomless pit. Note that even WUMPUS can fall into a pit.
  - Stench experienced in a room which has a WUMPUS in its neighborhood room. See the Fig. 3.2.1, here room (2,1), (3,2) and (4,1) have Stench.
  - Breeze is experienced in a room which has a pit in its neighborhood room. Fig. 3.2.1 shows that room (1,2), (1,4), (2,3), (3,2), (3,4) and (4,3) consists of Breeze.
  - Player (Agent) has arrows and he can shoot these arrows in straight line to kill WUMPUS.
  - One of the rooms consists of gold, this room glitters. Fig. 3.2.1 shows that room (3,2) has Gold.
- Apart from above features player (agent) can accept two types of percepts which are : Bump and scream. A bump is generated if player (agent) walks into a wall. While a sad scream created everywhere in the cave when the WUMPUS is killed.

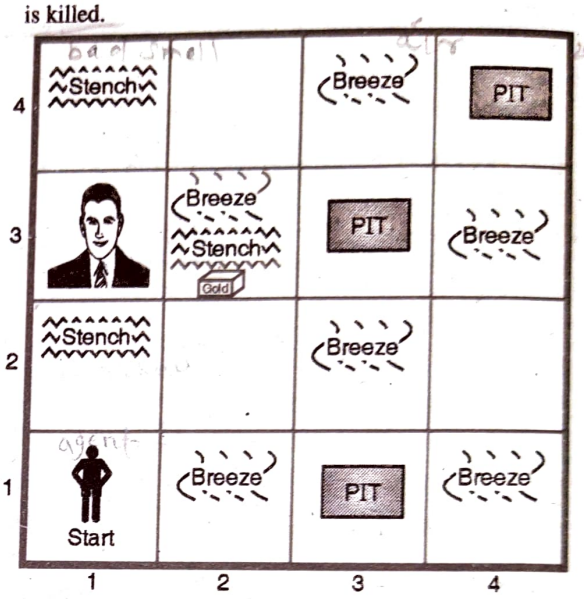


Fig. 3.2.1 : The WUMPUS World

3.2.1 Description of the WUMPUS World

Q. 3.2.1 Describe WUMPUS WORLD Environment. (Refer section 3.2.1) (8 Marks)

- An agent receives percepts while exploring the rooms of cave. Every percepts can be represented with the help of five element list, which is [stench, breeze, glitter, bump, scream]. Note that player (agent) cannot perceive its own location.
- If the player (agent) gets percept as [Stench, Breeze, None, None, None]. Then it means that there is a stench and a breeze, but no glitter, no bump, and no scream in the WUMPUS world at that position in the game.
- Let's take a look at the actions which can be performed by the player(agent) in WUMPUS World :
  - **Move :** To move in forward direction,
  - **Turn :** To turn right by 90 degrees or left by 90 degrees,
  - **Grab :** To pick up gold if it is in the same room as the player(agent),

- **Shoot** : To Shoot an arrow in a straight line in the direction faced by the player (agent).

- These actions are repeated till the player (agent) kills the WUMPUS or if the player (agent) is killed. If the WUMPUS is killed then it is a winning condition, else if the player (agent) is killed then it is a losing condition and the game is over.
- Game developer can keep a restriction on the number of arrows which can be used by the player(agent). So if we allow agent to have only one arrow, then only the first shoot action will have some effect. If this shoot action kills the WUMPUS then you win the game, otherwise it reduces the probability of winning the game.
- Lastly there is a direction : It takes places automatically if the agent enters in a room with a bottomless pit or in a room with WUMPUS. Die action is irreversible.

☞ **Goal of the game**

- Main aim of the game is that player (agent) should grab the gold and return to starting room (here its (1,1)) without being killed by the monster (WUMPUS).
- Award and punishment *points* are assigned to a player (Agent) based on the actions it performs. Points can be given as follows :
  - 100 points are awarded if player (agent) comes out of the cave with the gold.
  - 1 point is taken away for every action taken.
  - 10 points are taken away if the arrow is used.
  - 200 points are taken away if the player (agent) gets killed.

3.2.2 PEAS Properties of WUMPUS World

Q. 3.2.2 Specify PEAS properties and type of environment for the same. (Refer section 3.2.2) **(8 Marks)**

Q. 3.2.3 Give PEAS descriptors for WUMPUS world. (Refer section 3.2.2) **(6 Marks)**

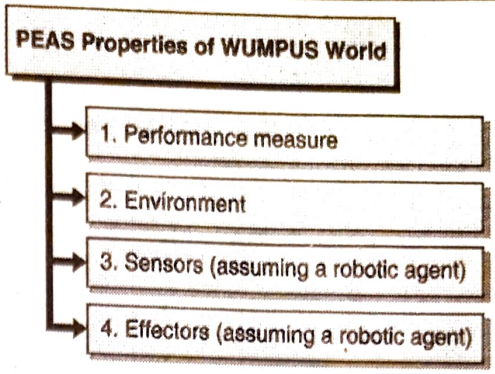


Fig. 3.2.2 : PEAS Properties of WUMPUS World

➜ **1. Performance measure**

- +100 for grabbing the gold and coming back to the starting position,
- – 200 if the player(agent) is killed.
- – 1 per action,
- – 10 for using the arrow.

➜ **2. Environment**

- Empty Rooms.
- Room with WUMPUS.
- Rooms neighbouring to WUMPUS which are smelly.
- Rooms with bottomless pits.
- Rooms neighbouring to bottomless pits which are breezy.
- Room with gold which is glittery.
- Arrow to shoot the WUMPUS.

➜ **3. Sensors (assuming a robotic agent)**

- Camera to get the view.
- Odour sensor to smell the stench.
- Audio sensor to listen to the scream and bump.

➜ **4. Effectors (assuming a robotic agent)**

- Motor to move left, right.
- Robot arm to grab the gold.
- Robot mechanism to shoot the arrow.

☞ **The WUMPUS world agent has following characteristics**

1.	Fully observable	2.	Deterministic
3.	Episodic	4.	Static
5.	Discrete	6.	Single agent

3.2.3 Exploring a WUMPUS World

Let's try to understand the WUMPUS world problem in step by step manner. Keep Fig. 3.2.3 as a reference figure.

4.1	4.2	4.3	4.4
3.1	3.2	3.3	3.4
2.1	2.2	2.3	2.4
1.1 [A] OK	1.2 OK	1.3	1.4

[A] - Agent
B – Breeze
G - Glitter, Gold
OK – Safe square
P - Pit
S – Stench
V – Visited
W –Wumpus

Fig. 3.2.3(a) : WUMPUS world with player in room (1, 1)

- The knowledge base initially contains only the rules (facts) of the WUMPUS world environment.

Step 1 : Initially the player(agent) is in the room (1, 1).

See Fig. 3.2.3(a).

The first percept received by the player is [none, none, none, none, none]. (remember percept consists of [stench, breeze, glitter, bump, scream])

Player can move to room(1,2) or (2,1) as they are safe cells.

Step 2 : Let us move to room (1,2). See Fig. 3.2.3(b).

4.1	4.2	4.3	4.4
3.1	3.2	3.3	3.4
2.1	2.2	2.3	2.4
1.1 V OK	1.2 [A] B OK	1.3 P?	1.4

[A] - Agent
B – Breeze
G - Glitter, Gold
OK – Safe square
P - Pit
S – Stench
V – Visited
W –Wumpus

Fig. 3.2.3(b) : WUMPUS world with player in room (1, 2)

- As room (1, 1) is visited you can see "V" mark in that room. The player receives following percept : [none, breeze, none, none, none].
- As breeze percept is received room (1, 2) is marked with "B" and it can be predicted that there is a bottomless pit in the neighboring room.
- You can see that room (1, 3) and room (2, 2) is marked with "P?". So room (1, 3) and (2, 2) is not safe to move in. Thus player should return to room (1, 1)and try to find other, safe room to move to.

Step 3 :

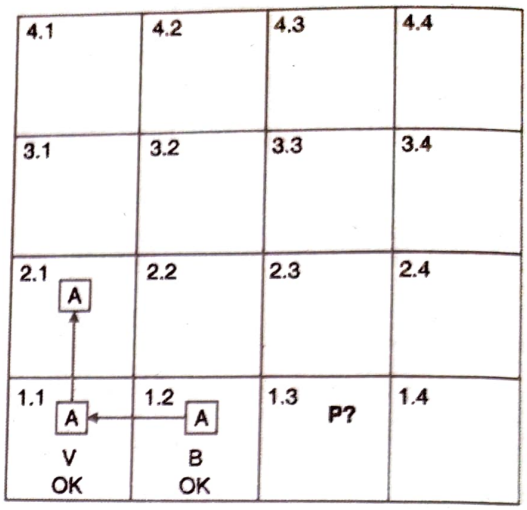


Fig. 3.2.3(c) : WUMPUS world with player moving back to room (1, 1) and then moves to other safe room (2, 1)

- As seen in Fig. 3.2.3(c). Player in now in room (2, 1), where it receives a percept as follows : [stench, none, none, none, none] which means that there is a WUMPUS in neighboring room (i.e. either room (2, 2) or (3, 1) has WUMPUS).
- As we did not get breeze percept in this room, we can understand that room (2, 2) cannot have any pit and from step 2 we can understand that room (2, 2) cannot have WUMPUS because room (1, 2) did not show stench percept.
- Thus room(2, 2) is safe to move in.

Step 4 : Player receives [none, none, none, none, none] percept when it comes to room (2, 2). From Fig. 3.2.3(d) you can understand that room (2, 3) and room (3, 2) are safe to move in.

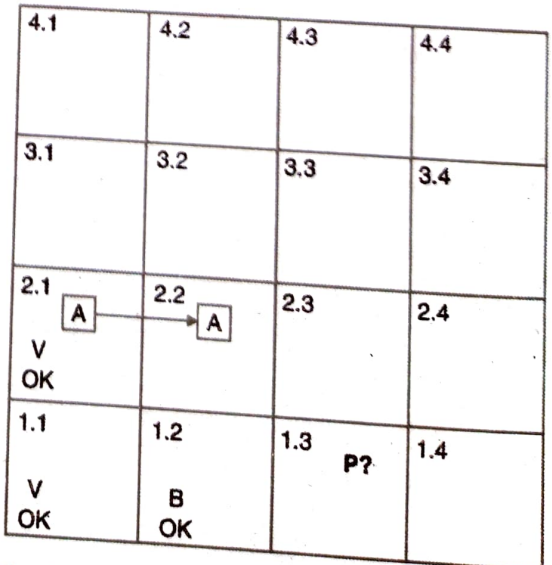


Fig. 3.2.3(d) : WUMPUS world with player moving to room (2, 2)

Step 5 : Let's move to room (3, 2). Here, player receives [stench, breeze, glitter, none, none] percept. See Fig. 3.2.3(e). Field 1 of the percept shows that room (3, 1), (3, 3) and (4, 2) can have WUMPUS. Field 2 of the percept shows that room (3, 1), (3, 3), (2, 2) and (4, 2) can have bottomless pit. Field 3 of the percept shows that room (3, 2) has gold. So, the player grabs the gold first. As the aim of this game is to grab the gold and go back to the starting position, without being killed by the WUMPUS.

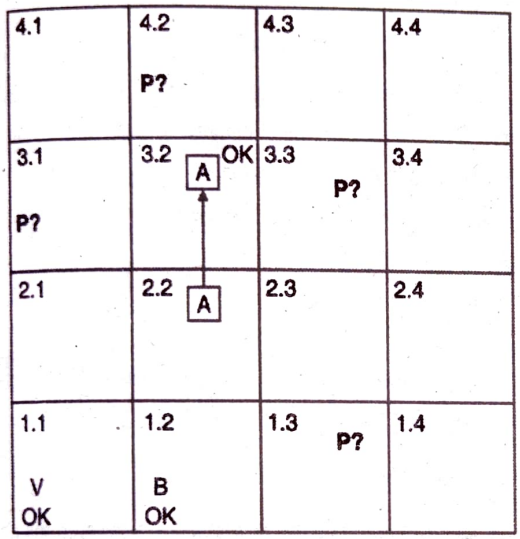


Fig. 3.2.3(e) : WUMPUS world with player moving to room (3,2)

- Now, we have to go back to the starting position i.e. room (1, 1) without getting killed by WUMPUS. From steps 1, 2, 3 and 4 We know that room (1, 1), (1, 2), (2, 1) and (2, 2) are safe rooms. so, we can go back to room (1, 1) by following any of the two paths : i.e. (2, 2), (2, 1), (1, 1) or (2, 2), (1, 2), (1, 1).

Step 6 : As can be seen in Fig. 3.2.3(f). We will go from room (2, 2) to room (2, 1) and from room (2, 1) to room (1, 1). Thus we won the WUMPUS World game!!!

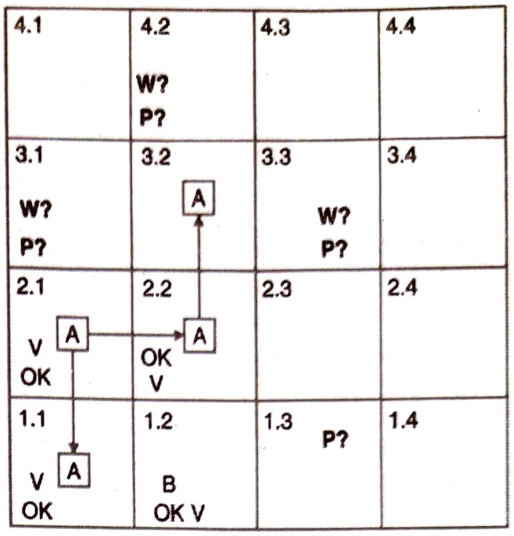


Fig. 3.2.3(f) : WUMPUS world with player moving back to room (1,1) with gold

Syllabus Topic : Facets of Knowledge

3.3 Facets of Knowledge

- The term "facet" was introduced into the field of library classification systems by Ranganathan in the 1930's. A facet is a viewpoint or aspect. Facets can describe different aspects on the same level of abstraction or the same aspect on different levels of abstraction. The elements of faceted knowledge representation are units, relations and facets.
  1. Units are atomic elements or tuples of atomic elements.
  2. Relations are sequences or matrices of 0's and 1's. They only obtain a meaning if they are applied to a domain (i.e., sets of units).
  3. Facets are conceptual relations or roles that are not binary or unary are modelled as higher level facets but are not "relations" .
- A **facet** is a viewpoint or aspect of given uniformities and their relations. These can be constrained by rules, which are constructed from uniformities and relational operators.
- Relations are meaningless unless they are applied to sets of units as domain and co-domain. This is formalized in basic facets.
- A **basic facet** consists of a relation and a set of units as domain and, in the case of binary relations, a set of units as co-domain. The notation is $f = (N; r)$ or $f = (N_1, N_2; r)$,

respectively. For the units that correspond to a 1 in the sequence or matrix, the relation is written as $n \in r_f$ or nr_f and $(n_1, n_2) \in r_f$ or $n_1\ r_f\ n_2$, respectively. The index "f" can be omitted in context. The set of basic facets (denoted by F_B) is a subset of U.

☞ **There are six facets of knowledge as follows**

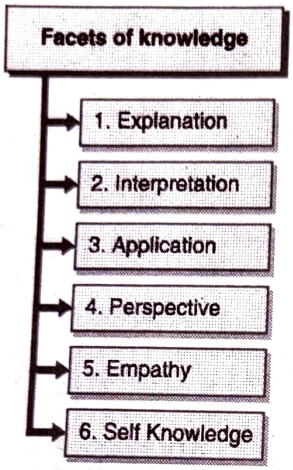


Fig. 3.3.1 : Six facets of knowledge

➜ **1. Explanation**

- To ensure that, one understand why an answer or approach is the right we make use of explanation type of knowledge. We explain or justify our responses or justify our course of action.

Example

- Students develop an illustrated brochure to explain the principles and practices of a particular type of technology i.e., transportation, construction, medical, information.

➜ **2. Interpretation**

- To ensure that one avoid the pitfall of looking for the "right answer" and demand answers that are principles. Using interpretation, one can encompass as many salient facts and points of view as possible.

Example

- Students develop a 'biography' of the development of a particular type of technology.

➜ **3. Application**

- To ensure key performances are conscious and explicit reflection, self-assessment, and self-adjustment, with reasoning made evident. Authentic assessment requires a real or simulated audience, purpose, setting, and options for personalizing the work, realistic constraints, and "background noise."

Example

- Students analyze a design of a product, taking it apart in order to determine how it works. Students design, develop, test, and revise a solution to a local issue, such as a new roadway system, a water treatment system, or long-term storage of various materials.

➜ **4. Perspective**

- This kind of facet is required to understand the importance or significance of an idea and to grasp its importance or unimportance. Encourage students to step back and ask, "What of it?" "Of what value is this knowledge?" "How important is this idea?" "What does this idea enable us to do ?"

Example

- Students investigate about a technological artifact from the perspective of different regions and countries.

➜ **5. Empathy**

- This kind of knowledge facet enable us to develop the ability to see the world from different viewpoints in order to understand the diversity of thought and feeling in the world.

Example

- Students imagine they are politicians debating the value of nuclear power. They write their thoughts and feelings explaining why they agree or disagree with the use of nuclear power.

➜ **6. Self Knowledge**

- This enables us to be aware of the boundaries of our own and others' understanding and make us able to recognize our own prejudices and projections.

Example

- Students reflect on their own progress of understanding. They evaluate the extent to which they have improved, what task or assignment was the most challenging and why, and which project or product of work they are most proud of and why.

Syllabus Topic : Logic and Inferences

3.4 Logic and Inferences

- Logic can be called as **reasoning** which is carried out or it is a review based on strict rule of validity to perform a specified task.
- In case of intelligent systems we say that any of logic's particular form cannot bind logical representation and reasoning, they are independent of any particular form of logic.
- Make a note that logic is beneficial only if the knowledge is represented in small extent and when knowledge is represented in large quantity the logic is not considered valuable.
- Fig. 3.4.1 depicts that sentences are physical configurations of an agent, also it is shows that sentences need sentence. This means that reasoning is a process of forming new physical configurations from old ones.

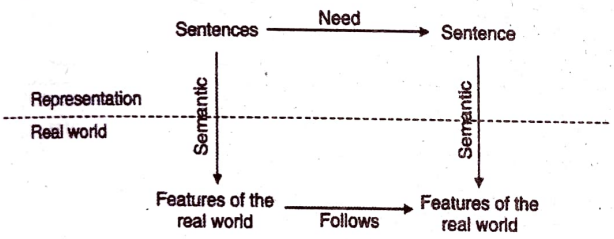


Fig. 3.4.1 : Correspondence between real world and its representation

- Logical reasoning should make sure that the new configurations represent features of the world that actually follow the features of the world that the old configurations represented

3.4.1 Role of Reasoning in AI

- Fig. 3.4.2 shows how logic can be seen as a knowledge representation language. There are various levels to the logic and most fundamental type of logic is propositional logic.

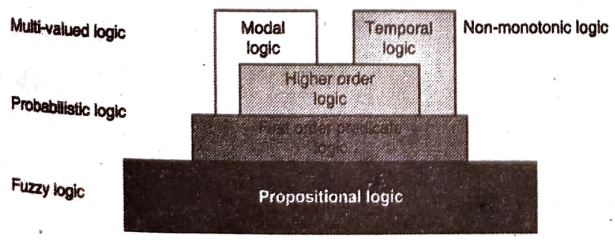


Fig. 3.4.2 : Logic as Knowledge Representation language

- Propositional logic can be considered at fuzzy logic level, where rules are values between range of 0 and 1. Next level is also called as **probabilistic logic level** using which first order predicate logic is implemented.
- In this Fig. 3.4.2 that there are two more levels above higher order logic which are multi-valued and non-monotonic logic levels and they consist of modal logic and temporal logic respectively. All these types of logic are basic building blocks of intelligent systems and they all use reasoning in order to represent sentences. Hence reasoning plays a very important role in AI.

Syllabus Topic : Formal Logic

3.4.2 Formal Logic

- **Formal Logic** is the study of inference with purely formal content. An inference possesses a *purely formal content* if it can be expressed as a particular application of a wholly abstract rule, that is, a rule that is not about any particular thing or property. Formal logic is used to layout formal system through which we can reason. It helps to make abstraction of the reasoning process, so that we can understand human reasoning process better.
- Formal logic representations enables computer to mimic the reasoning process of humans. Aristotle developed a symbolic system which makes it possible for a machine to mechanically generate conclusion, given initial premises.

Example

- **Minor premise :** Every mammal has spine.
- **Major premise :** Dog is a mammal.
- **Conclusion :** Dog has spine.

- Leibnitz introduced a first system of formal logic. He built a mechanical device intended to carry out mental operations.
- George Boole introduced his formal language for making logical inferences in 1864. His work was entitled as "An investigation on laws of thoughts", on which the mathematical theories of logic and probabilities are founded. His system was a precursor to fully developed propositional logic.
- Let's study propositional and predicate logic as the usable form of formal logic.

Syllabus Topic : Propositional and First Order Logic

3.5 Propositional and First Order Logic

- **Propositional Logic (PL)** is simple but powerful for some artificial intelligence problems. You have learnt simple mathematical logic in which uses atomic formulas are used. (Atomic formula is a formula that has no strict sub-formulas). Atomic logic formulas are called **propositions**.
- In case of artificial intelligence propositional logic is not categorized as the study of truth values, but it is based on relativity of truth values. (i.e. The relationship between the truth value of one statement to that of the truth value of other statement).

3.5.1 Syntax

Q. 3.5.1 Write syntax and semantics and example sentences for propositional logic. (Refer section 3.5.1) **(7 Marks)**

Basic syntax followed by the propositional logic can be given as follows :

- Propositional symbols are denoted with capital letters like : A, B, C, etc.
- Propositional logic constants have a truth value generally truth values have a crisp nature (i.e. 0 (false) and 1 (true)). But for fuzzy logic truth values can vary in the range of 0 and 1.
- Propositional logic makes use of wrapping parenthesis while writing atomic sentence. It is denoted as '(....)'.
- Literal is an atomic sentence or it can be negation of atomic sentence.

(A, ¬A) If A is a sentence, then ¬A is a sentence.

- Propositional logic makes use of relationships between propositions and it is denoted by connectives, if A and B are propositions. Connectives used in proposition logic can be seen in the Table 3.5.1.

Table 3.5.1 : Connectives used in Propositional logic

Connective symbol	Name of the Connective symbol	Relationship between Propositional symbols	Name of the Relationship between Propositional symbols
∧	And	A ⊥ B	Conjunction
∨	Or	A ∨ B	Disjunction
¬	Not	¬A	Negation
⇒	Implies	A ⇒ B	Implication / conditional
⇔	is equivalent/ if and only if	A ⇔ B	Biconditional

- To define logical connectives truth tables are used. Following truth table shows five logical connectives.

Table 3.5.2

A	B	A ∧ B	A ∨ B	¬A	A ⇒ B	A ⇔ B
false	false	false	False	true	true	true
false	true	false	True	true	true	false
true	false	false	True	false	false	false
true	true	true	True	false	true	true

- Let take one example, where **A ∧ B**, i.e. Find the value of **A ∧ B** where A is true and B is false.
- Third row of the Table 3.5.2 shows this condition, now see third row of the third column where, **A ∧ B** shows result as false. Similarly other logical connectives can be mapped in the truth table.

3.5.2 Semantics

- World is set of facts which we want to represent to form propositional logic.
- In order to represent these facts propositional symbols can be used where each propositional symbol's interpretation can be mapped to the real world feature.
- Semantics of a sentence is meaning of a sentence. Semantics determine the interpretation of a sentence.

Example

- You can define semantics of each propositional symbol in following manner :
  1. A means "It is hot"
  2. B means "It is humid", etc.
- Sentence is considered true when its interpretation in the real world is true. Every sentence results from a finite number of usages of the rules.

Example

If A and B are sentences then (A ∧ B), (A ∨ B), (B → A) and (A ↔ B) are sentences.

- The knowledge base is a set of sentences as we have seen in previous section.
- Thus we can say that real world is a model of the knowledge base when the knowledge base is true for that world. In other words a model can be thought of as a truth assignment to the symbols.
- If truth values of all symbols in a sentence are given then it can be evaluated for determining its truth value (i.e. we can say if it is true or false).

3.5.3 What is Propositional Logic ?

- A ∧ B and B ∧ A should have same meaning but in natural language words and sentences may have different meanings. Say for an example,
  1. Radha started feeling feverish and Radha went to the doctor and
  2. Radha went to the doctor and Radha started feeling feverish.
- Here, sentence 1 and sentence 2 have different meanings.
- In artificial intelligence propositional logic is a relationship between the truth value of one statement to that of the truth value of other statement.

3.5.4 PL Sentence - Example

Take example of a weather problem.

- Semantics of each propositional symbol can be defined as follows :
  - Symbol A is a sentence which means "It is hot".
  - Symbol B is a sentence which means "It is humid".
  - Symbol C is a sentence which means "It is raining".
- We can also choose symbols which are easy to understand, like :
  - HT for "It is hot".
  - HM for "It is humid".
  - RN for "It is raining".
- If you have B → A then that means "If it is humid, then it is hot".
- If you have (A ∧ B) → C then it means "If it is hot and humid, then it is raining" and so on.
- First we have to create the possible models for a knowledge base. To do this we need to consider all the possible assignments of true or false values for Sentence A, B and C. Then verify the truth table for the validity. There can be total 8 possibilities as shown below :

Sentence A	Sentence B	Sentence C	Validity
False	False	False	Valid
False	False	True	Valid
False	True	False	Not Valid
False	True	True	Not Valid
True	False	False	Valid
True	False	True	Valid
True	True	False	Not Valid
True	True	True	Valid

- Now, if the knowledge base is [B, B → A, (A ∧ B) → C](i.e. ["It is humid", "If it is humid, then it is hot" , "If it is hot and humid, then it is raining"]), then "True -True - True" is the only possible valid model.
  - **Tautology** means valid sentence. It is a sentence which is true for all the interpretations.

    Example : (A ∨ Ø A) ("A or not A") : "It is hot or It is not hot".
  - **Contradiction** means an inconsistent sentence. It is a sentence which is false for all the interpretations.

Example : A Ù ¬A ("*A* and not *A*") : "It is hot and it is not hot."

- **X entails Y**, is shown as X |= Y. It means that whenever sentence X is True, sentence Y will be True.

  Example : if, X = Priya is Pooja's Mother's Sister and Y = Priya is Pooja's Aunty. Then X |= Y (X entails Y).

3.5.5 Inference Rules

Q. 3.5.2 Explain the inference process in case of propositional logic with suitable examples. (Refer section 3.5.5) **(8 Marks)**

- New sentences are formed with the logical inference.
- **Example :** If A=B and B=C then A=C. You must have come across this example many times it implies that if knowledge base has "A=B" and "B=C" then we can infer that "A=C".
- In short inference rule says that new sentence can be create by logically following the set of sentences of the knowledge base.

Table 3.5.3 : Inference Rules

Inference Rules	Premise (KB)	Conclusion
Modus Ponens	X, X→Y	Y
Substitution	X →Z &Y → Z	A→B
Chain rule	X → Y, Y→Z	X →Z
AND introduction	X, Y	X ^ Y
Transposition	X →Y	~X →~Y

- Entailment is represented as : KB |= Q and Derivation is represented as : KB |- Q.

There are two types of inference rules

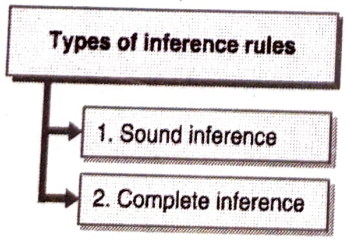


Fig. 3.5.1 : Types of inference rules

➜ **1. Sound inference**

- For **Modus Ponens (MP)** rule we assume that knowledge base has [A, A → B], from this we can conclude that knowledge base can have B. See following truth table :

A	B	A → B	Valid?
TRUE	TRUE	TRUE	Yes
TRUE	FALSE	FALSE	Yes
FALSE	TRUE	TRUE	Yes
FALSE	FALSE	TRUE	Yes

- Soundness property of inference says that, if "X is derived from the knowledge base" using given set of protocols of inference, then "X is entailed by knowledge base". Soundness property can be represented as : "If KB |- X then KB | = X".

➜ **2. Complete inference**

- Complete inference is converse of soundness. Completeness property of inference says that, if "X is entailed by knowledge base" then "X can be derived from the knowledge base" using the inference protocols.
- Completeness property can be represented as : " If KB |= Q then KB |- Q".

3.5.6 Horn Clause

Q. 3.5.3 Explain Horn Clause with example. (Refer section 3.5.6) **(6 Marks)**

- Clauses are generally written as sets of literals. Horn clause is also called as **horn sentence**. In a horn clause a conjunction of 0 or more symbols is to the left of "→" and 0 - 1 symbols to the right. See following formula :

  $A_1 \wedge A_2 \wedge A_3 \ldots \wedge A_n \rightarrow B_m$ where n >= 0 and m is in range{0,1}

- There can be following special cases in horn clause in the above mentioned formula :
  - For n = 0 and m = 1 : A (This condition shows that assert A is true)
  - For n > 0 and m = 0 : A Ù B® (This constraint shows that both A and B cannot be true)
  - For n = 0 and m = 0 : (This condition shows empty clause)

- Conjunctive normal form is a conjunction of clauses and by its set of clauses it is determined up to equivalence. For a horn clause conjunctive normal form can be used where, each sentence is a disjunction of literals with at most one non-negative literal as shown in the following formula : $\neg A_1 \vee \neg A_2 \vee \neg A_3 \ldots \vee \neg A_n \vee B$
- This can also be represented as : $(A \rightarrow B) = (\neg A \vee B)$

Significance of horn logic

- Horn sentences can be used in first order logic. Reasoning processes is simpler with horn clauses. Satisfiability of a propositional knowledge base is NP complete. (Satisfiability means the process of finding values for symbols which will make it true).
- For restricting knowledge base to horn sentences, satisfiability is in A. Due to this reason, first order logic horn sentences are the basis for prolog and datalog languages.
- Let's take one example which gives entailment for horn formulas.
- Find out if following horn formula is satisfiable?

  $(true \rightarrow X) \wedge (X \wedge Y \rightarrow Z) \wedge (Z \,!\, W) \wedge (Z \wedge W \rightarrow false) \wedge (true \rightarrow Y)$
- From the above equation, we entail if the query atom is false. Equation shows that there are clauses which state that $true \rightarrow X$ and $true \rightarrow Y$, so we can assign X and Y to true value (i.e. $true \rightarrow X \wedge Y$).
- Then we can say that all premises of $X \wedge Y \rightarrow Z$ are true, based on this information we can assign Z to true. After that we can see all premises of $Z \rightarrow W$ are true, so we can assign W to true.
- As now all premises of $Z \wedge W \rightarrow$ false are true, from this we can entail that the query atom is false. Therefore, the horn formula is not satisfiable.

3.5.7 Propositional Theorem Proving

- Sequence of sentences form a "Proof". A sentence can be premise or it can be a sentence derived from earlier sentences in the proof based on the inference rule. Whatever we want to prove is called as a query or a goal. Query/goal is the last sentence of the theorem in the proof.
- Take Example of the "weather problem" which we have seen above.
  - HT for "It is hot".
  - HM for "It is humid".
  - RN for "It is raining".

1.	HM	Premise (initial sentence)	"It's humid"
2.	$HM \rightarrow HT$	Premise(initial sentence)	"If it's humid, it's hot"
3.	HT	Modus ponens(1,2) (sentence derived from 1 and 2)	"It's hot"
4.	$(HT \wedge HM) \rightarrow RN$	Premise(initial sentence)	"If it's hot and humid, it's raining"
5.	$HT \wedge HM$	And introduction(1,3)	"It's hot and humid"
6.	RN	Modus ponens(4,5) (sentence derived from 4 and 5)	"It's raining"

3.5.8 Advantages of Propositional Logic

- Propositional logic is a simple knowledge representation language.
- It is sufficient and efficient technique for solving some artificial intelligence based problems.
- Propositional logic forms the foundation for higher logics like **First Order Logic (FOL)**, etc.
- Propositional logic is NP complete and reasoning is decidable.
- The process of inference can be illustrated by PL.

3.5.9 Disadvantages of Propositional Logic

Q. 3.5.4 Write a short note on : Drawbacks of propositional logic. (Refer section 3.5.9) **(5 Marks)**

- Propositional logic is cannot express complex artificial intelligence problems.
- Propositional logic can be impractical for even small worlds, think about WUMPUS hunter problem.

- Even if we try to make use of propositional logic to express complex artificial intelligence problems, it can be very wordy and lengthy.
- PL is a weak knowledge representation language because :
  - With PL it is hard to identify if the used entity is "individual".
    Example : If there are entities like : Priya, Mumbai, 123, etc.
  - PL cannot directly represent properties of individual entities or relations between individual entities. For example, Pooja is tall.
  - PL cannot express specialization, generalizations, or patterns, etc.
  - **Example :** All rectangles have 4 sides.

3.6 First Order Predicate Logic

Q. 3.6.1 What is first order logic? (Refer section 3.6) **(2 Marks)**

- Because of the inadequacy of PL discussed above there was a need for more expressive type of logic. Thus **First-Order Logic (FOL)** was developed. FOL is more expressive than PL, it can represent information using relations, variables and quantifiers, e.g., which was not possible with propositional logic.
  - "Every Gorilla is Black" can be represented as :
    $$\forall x \, (\text{Gorilla}(x) \rightarrow \text{Black}(x))$$
  - "There is a white Dog" can be represented as :
    $$\exists x \, (\text{Dog}(X) \wedge \text{white}(X))$$
- First Order Logic (FOL) is also called as **First Order Predicate Logic (FOPL)**. Since FOPL is much more expressive as a knowledge representation language than PL it is more commonly used in artificial intelligence.

3.6.1 Syntactic Elements, Semantic and Syntax

Q. 3.6.2 Write syntax and semantics of FOL with example. (Refer section 3.6.1) **(6 Marks)**

- FOPL symbol can be a constant term, a variable term or a function.
- Assuming that "X" is a domain of values, we can define a term with following rules :
  1. **Constant term :** It is a term with fixed value which belongs to the domain.
  2. **Variable term :** It is a term which can be assigned values in the domain.
  3. **Function :** Say "f" is a function of "n" arguments. If we assume that $t_1, t_2, .., t_n$ are terms then $f(t_1, t_2, .., t_n)$ is also called as **a term**.
- All the terms are generated by applying the above three protocols.
- First order predicate logic makes use of propositional logic as a base logic, so the connectives used in PL and FOPL are common. Apart from these connectives FOPL makes use of quantifiers like : Universal Quantifier $\forall$ (for all) and Existential Quantifier $\exists$ (there exists).
  - "$\forall x$ A" means A is true if every replacement for x makes A true.
  - "$\exists x$ A" means A is true if at least one replacement for x makes A true.
- If a term does not have any variables it is called as a **ground term**. A sentence in which all the variables are quantified is called as a "well-formed formula".
  - Interpretation has an *objects* in the world and a *mappings* of terms and predicates (condition).
  - Every ground term is mapped with an object.
  - Every condition (predicate) is mapped to a relation.
  - A ground atom is considered as true if the predicate's relation holds between the terms' objects.

3.7 Unification and Lifting

Q. 3.7.1 What is unification and lifting? (Refer section 3.7.1) **(4 Marks)**

- We have seen from the previous section that the propositional approach was not efficient enough for knowledge representation. From propositional logic we tend to get irrelevant inferences and sentences.

- To demonstrate this let us consider an example given below :

  "x King(x) ∧ Brave(x) ⇒ Noble(x)

  King(Ram)

  Brave(Ram)

  Brother(Laxman, Ram)

- If for the above expressions the query is Noble(x) and Universal instantiation is applied to it we get,

  King (Ram) ∧ Brave (Ram) ⇒ Noble (Ram)

  King (Laxman) ∧ Brave (Laxman) ⇒ Noble (Laxman)

- The fact that Noble (Ram) is true, but Noble (Laxman) is totally irrelevant. Now let us observe what we get when we add the offsprings (Luv, Kush) to our knowledge data base.

  "x King(x) ∧ Brave(x) ⇒ Noble(x)

  King(Ram)

  Brave(Ram)

  Brother(Laxman, Ram)

  Offspring(Luv,Kush)

- Now with two new values in our knowledge base we apply the universal instantiation to get :

  King (Ram) ∧ Brave (Ram) ⇒ Noble (Ram)

  King (Laxman) ∧ Brave (Laxman) ⇒ Noble (Laxman)

  King (Luv) ∧ Brave (Luv) ⇒ Noble (Luv)

  King (Kush) ∧ Brave (Kush) ⇒ Noble (Kush)

- All these do not make any sense and these sentences are not necessarily in our knowledge base. Therefore the computer must be taught to make better inferences using some algorithms that are robust.

- If we desire an 'x' where 'x' is a king and 'x' is brave (Then x is noble) then ideally what we want is θ = {substitution set}
  i.e. θ = {x/ Ram}

- Now suppose instead of knowing Brave (Ram), we know that everyone is brave :
  - "y Brave (y) = "for all values of y, y=brave" or equivalently "Everyone is brave".
  - Now our query θ = {x/Ram, y/Ram}, so we can now infer Noble (x).
  - The process of encapsulating inference rule is called as **Generalized Modus Ponens.**

3.7.1 Generalized Modus Ponens

- For atomic sentences p_i, p_i', and q, where there is a substitution θ such that

  $$\text{SUBST}(\theta, p_i) = \text{SUBST}(\theta, p_i') \quad \text{for all i,}$$

  $$\frac{p_1', p_2', p_3', \ldots p_n', (p_1 \wedge p_2 \wedge p_3 \wedge \ldots \ldots p_n \Rightarrow q)}{\text{SUBST}(\theta, q)}$$

- N + 1 premises = N atomic sentences + one implication. Applying SUBST (θ , q) produces the conclusion we seek.

  $p_1' = King(Ram)$ $\quad$ $p_2' = Brave(y)$

  $p_1 = King(x)$ $\quad$ $p_2 = Brave(x)$

  $\theta = \{x / Ram, y / Ram\}$ $\quad$ $q = Noble(x)$

  SUBST(θ, q) is *Noble(Ram)*

3.7.2 Unification

The processes of finding legal substitutions that make different logical expressions look identical. The unification algorithm is a recursive algorithm; the problem of **unification** is : given two atoms, to find if they unify, and, if they do, return an MGU (Most General Unifier) of them.

```
Procedure Unify(t1,t2)
 Inputs
  t1,t2: atoms Output
  most general unifier of t1 and t2 if it exists or ⊥ otherwise
 Local
  E: a set of equality statements
  S: substitution
 E ←{t1=t2}
 S={}
 while (E≠{})
  select and remove x=y from E
  if (y is not identical to x) then
   if (x is a variable) then
     replace x with y everywhere in E and S
       S←{x/y} ∪S
   else if (y is a variable) then
     replace y with x everywhere in E and S
       S←{y/x} ∪S
   else if (x is f(x1,...,xn) and y is f(y1,...,yn)) then
     E←E ∪{x1=y1,...,xn=yn}
   else
     return ⊥
 return S
```

Fig. 3.7.1 : Unification algorithm for Datalog

3.7.3 Lifting

- Generalized Modus Ponens is a **lifted** version of Modus Ponens it raises Modus Ponens from ground (variable-free) propositional logic to first-order logic.
- Here "lifted" indicates transformed from.
- The major advantage of lifted inference rules over propositional logic is that only those substitutions are made that are required so as particular inferences are allowed to proceed.

Syllabus Topic : Resolution in Propositional and First Order Logic

3.8 Resolution in Propositional and First Order Logic

- Resolution is a valid inference rule. Resolution produces a new clause which is implied by two clauses containing complementary literals. This resolution rule was discovered by Alan Robinson in the mid 1960's.
- We have seen that a literal is an atomic symbol or a negation of the atomic symbol (i.e. A, ¬A).
- Resolution is the only interference rule you need, in order to build a sound (soundness means that every sentence produced by a procedure will be "true") and complete (completeness means every "true" sentence can be produced by a procedure) theorem proof maker.
- Take an example where we are given that :
  - A clause X containing the literal : Z
  - A clause Y containing the literal : ¬ Z
- Based on resolution and the information given above we can conclude :

  $(X - \{Z\}) \cup (Y - \{\neg Z\})$
- Take a generalized version of the above problem :

  Given :
  - A clause X containing the literal : Z
  - A clause Y containing the literal : ¬Y
  - A most general unifier G of Z and ¬Y
- We can conclude that : $((X - \{Z\}) \cup (Y - \{\neg Y\})) \mid G$

3.8.1 The Resolution Procedure

- Let knowledge base be a set of true sentences which do not have any contradictions, and Z be a sentence that we want to prove.
- The Idea is based on the proof by negation. So, we should assume ¬Z and then try to find a contradiction (You must have followed such methods while solving geometry proofs). Then based on the Intuition that, if all the knowledge base sentences are true, and assuming ¬Z creates a contradiction then Z must be inferred from knowledge base. Then we need to convert knowledge base U {¬Z} to clause form.
- If there is a contradiction in knowledge base, that means Z is proved. Terminate the process after that.
- Otherwise select two clauses and add their resolvents to the current knowledge base. If we do not find any resolvable clauses then the procedure fails and then we terminate. Else, we have to start finding if there is a contradiction in knowledge base, and so on.

3.8.2 Conversion from FOL Clausal Normal Form (CNF)

Q. 3.8.1 Explain steps to convert logical statements to clausal form. (Refer section 3.8.2) **(8 Marks)**

Q. 3.8.2 Explain the steps involved in converting the propositional logic statement into CNF with a suitable example. (Refer section 3.8.2) **(8 Marks)**

1. Elimination of implication i.e. Eliminate all '→' : Replace $P \rightarrow Q$ with $\neg P \vee Q$
2. Distribute negations : Replace ¬ ¬P with P, ¬(PV Q) with $\neg P \wedge \neg Q$ and so on.
3. Eliminate existential quantifiers by replacing with Skolem constants or Skolem functions :

   e.g. $\forall X \exists Y(P_1(X,Y) \vee (P_2(X,Y)) \equiv \forall X (P_1(X, f(X)) \vee (P_2(X, f(X)))$
4. Rename variables to avoid duplicate quantifiers.
5. Drop all universal quantifiers
6. Place expression into conjunctive Normal Form.
7. Convert to clauses i.e. separates all conjunctions as separate clause.
8. Rename variables to avoid duplicate clauses.

Ex. 3.8.1 : Convert following propositional logic statement into CNF:

Aà(Bà C)

Soln. :

- FOL : $A \rightarrow (B \leftrightarrow C)$
- Normalizing the given statement.
  - (i) $A \rightarrow (B \rightarrow C \wedge C \rightarrow B)$
  - (ii) $(A \rightarrow (B \rightarrow C)) \wedge (A \rightarrow (C \rightarrow B))$
- Converting to CNF.
- Applying Rule, $\alpha \rightarrow \beta \equiv \sim \alpha \vee \beta$

  $\sim A \vee (\sim B \vee C) \wedge \sim A \vee (\sim C \vee B)$

  i.e. $\sim A \vee ((\sim B \vee C) \wedge (\sim C \vee B))$

3.8.3 Facts Representation

- To show how facts can be represented let's take a simple problem :
  - "Heads X wins, Tails Y loses."
  - Our goal is to show that X always wins with the help of resolution.
- Solution can be given as follows :
  1. H⇒ Win(X)
  2. T ⇒ Loose(Y)
  3. ¬H⇒ T
  4. Loose(Y) ⇒ Win(X)
- Thus we have : Win(X)
- We can write a proof for this problem as follws :
  1. {¬H, Win(X)}
  2. {¬T, Loose(Y)}
  3. {H, T}
  4. {¬Loose(Y), Win(X)}
  5. {¬ Win(X)}
  6. {¬T, Win(X)} ...(From 2 and 4)
  7. {T, Win(X)} ...(From 1 and 3)
  8. {Win(X)} ...(From 6 and 7)
  9. {} ...(From 5 and 8)

3.8.4 Example

Let's take the same example of forward and backward chaining to learn how to write proofs for resolution.

☞ **Step 1**

The given facts are :

1. It is a crime for an American to sell weapons to the enemy nations.
   - American(x) Λ Weapon(y) Λ sell (x,y,z) Λ enemy(z, America) => Criminal (x)
2. Country Nono is an enemy of America.
   - Enemy (Nono, America)
3. Nono has some missiles.
   - Owns (Nono, x)
   - Missile(x)
4. All the missiles were sold to Nono by Colonel West.
   - Missile(x)Λ owns(Nono, x)=> Sell(West, x, Nono)
5. Missile is a weapon.
   - Missile(x)=> weapon(x)
6. Colonel West is American.
   - American (West)

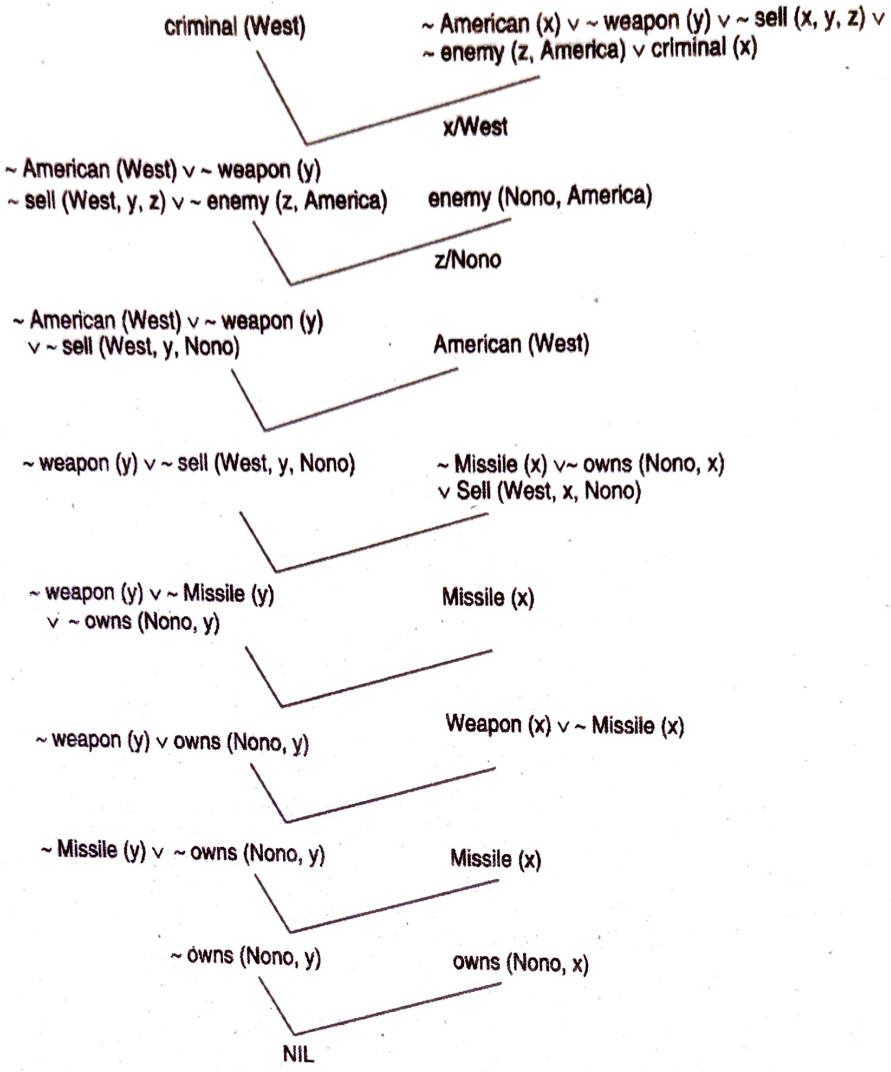
☞ **Step 2**

Let's convert them to CNF

1. ~ American(x) ∨ ~ Weapon(y) ∨~ sell (x, y, z) ∨ ~ enemy(z, America) ∨ Criminal (x)
2. Enemy (Nono, America)
3. Owns (Nono, x)
4. Missile(x)
5. ~ Missile(x) ∨ ~ owns(Nono, x) ∨ Sell(West, x, Nono)
6. ~Missile(x) ∨ weapon(x)
7. American (West)

☞ **Step 3**

To prove that West is criminal using resolution.



Hence our assumption was wrong. Hence proved that West is criminal.

Ex. 3.8.2 : Consider following statements :

(a) Ravi Likes all kind of food.

(b) Apple and Chicken are food

(c) Anything anyone eats and is not killed is food.

(d) Ajay eats peanuts and still alive.

(e) Rita eats everything that Ajay eats.

Prove that Ravi Likes Peanuts using resolution. What food does Rita eat?

Soln. :

(A) Proof by Resolution

Step 1 : Negate the statement to be proved.

– ~ likes (Ravi, Peanuts)

Step 2 : Convert given facts to FOL

(a) Ú x, food (x) → likes (Ravi, x)

(b) food (Apple)

(c) food (Chicken)

(d) Ú x Úy : eats (x, y) ∧ ~ killed (x) → food (y)

(e) eats (Ajay, Peanuts) ∧ alive (Ajay)

(f) Ú x : eats (Ajay, x) → eats (Rita, x)

In this case we have to add few common sense predicate which are always true.

(g) Ú x : ~ killed (x) → alive (x)

(h) Ú x : alive (x) → ~ killed (x)

Step 3 : Converting FOLs to CNF

(a) ~ food (x) v likes (Ravi, x)

(b) food (Apple)

(c) food (Chicken)

(d) ~ eats (x, y) ∨ killed (x) ∨ food (y)

(e) eats (Ajay, Peanuts)

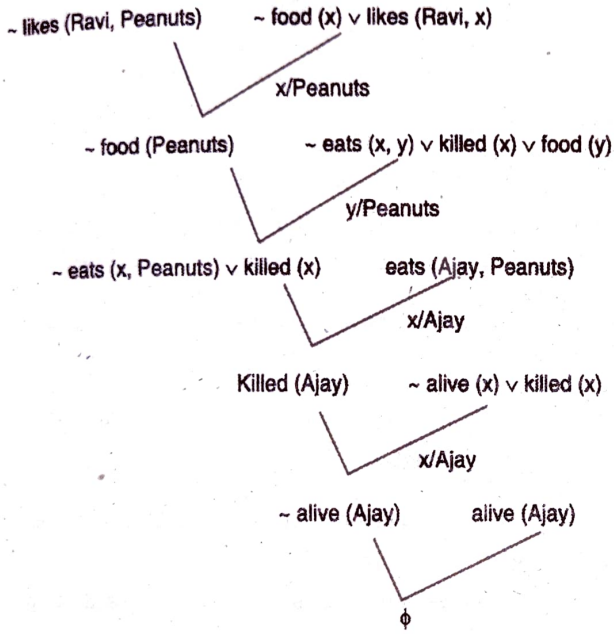
(f) alive (Ajay)

(g) ~ eats (Ajay, x) ∨ eats (Rita, x)

(h) killed (x) ∨ alive (x)

(i) ~ alive (x) ∨ killed (x)

Step 4 : Proof by Resolution

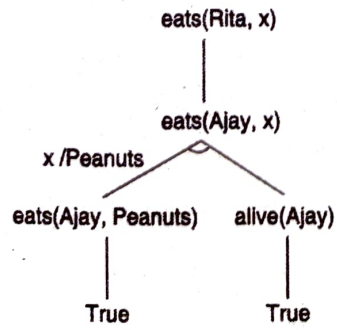


As the result of this resolution is NIL, it means out assumption is wrong. Hence proved that "Ravi likes Peanuts".

To answer : What food Rita eats ?

(B) Proof by backward chaining

(Referring to FOLs of Step 2)



Hence the answer is Rita eats peanuts.

Ex. 3.8.3 : Using a predicate logic convert the following sentences to predicates and prove that the statement "Ram did not jump" is false.

(a) Ram went to temple.

(b) The way to temple is, walk till post box and take left or right road.

(c) The left road has a ditch.

(d) Way to cross the ditch is to jump

(e) A log is across the right road.

(f) One needs to jump across the log to go ahead.

Soln. :

Step 1 : Negate the statement to be proved.

~ jump (Ram)

Step 2 : Converting given statement to FOL

(a) At (Ram, temple)

(b_1) Úx : At (x, temple) ⟶ At (x, PostBox) ∧ take left (x)

(b_2) Úx : At (x, temple) ⟶ At (x, PostBox) ∧ take right (x)

(c) Úx: take left (x) ⟶ cross (x, ditch)

(d) Úx cross (x, ditch) ⟶ jump (x)

(e) Úx : take right (x) ⟶ at (x, log)

(f) Ú x : at (x, log) ⟶ jump (x)

Step 3 : Converting FOLs to CNF

(a) At (Ram, temple)

(b_{11}) ~ At (x, temple) ∨ At (x, PostBox)

(b_{12}) ~ At (x, temple) ∨ take left (x)

(b_{21}) ~ At (x, temple) ∨ At (x, PostBox)

(b_{22}) ~ At (x, temple) ∨ take right (x)

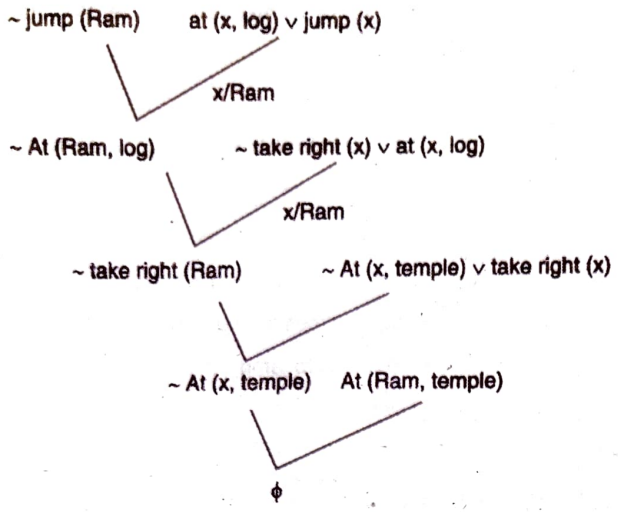
(c) ~ take left (x) ∨ cross (x, ditch)

(d) ~ cross (x, ditch) ∨ jump (x)

(e) ~ take right (x) ∨ at (x, log)

(f) ~ at (x, log) ∨ jump (x)

Step 4 : Proof by Resolution



Hence proved.

Ex. 3.8.4 : Consider following statements.

1. Rimi is hungry.
2. If Rimi is hungry she barks.
3. If Rimi is barking then Raja is angry.

Explain statements in predicate logic. Convert them into CNF form.

Prove that Raja is angry using resolution.

Soln. :

Step 1 : Converting given facts to FOL.

1. Hungry (Rimi)
2. Hungry (Rimi) → barks (Rimi)
3. Barks (Rimi) → angry (Raja)

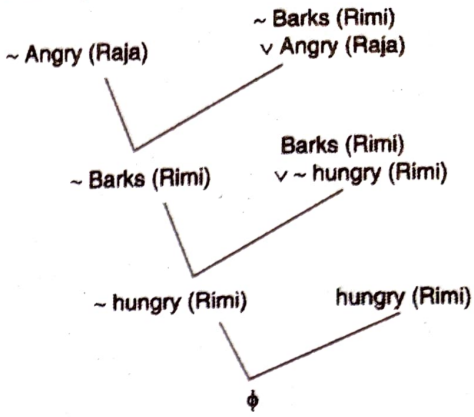
Step 2 : Converting FOL statements to CNF.

1. Hungry (Rimi)
2. ~ hungry (Rimi) ∨ barks (Rimi)
3. ~ barkes (Rimi) ∨ angry (Raja)

Step 3 : Negate the stmt to be proved

- T.P.T. Angry (Raja)
- Negation : ~ Angry (Raja)

Step 4 : Proof by resolution



This shows that our assumption is Wrong. Hence proved that **Raja is Angry.**

Ex. 3.8.5 : Consider following facts.

1. If maid stole the jewelry then butler was not guilty.
2. Either maid stole the jewelry or she milked the cow.
3. If maid milked the cow then butler got the cream.
4. Therefore if butler was guilty then he got the cream.

Prove the conclusion (step 4) is valid using resolution.

Soln. :

Step 1 : Converting given facts to FOL.

1. steal (maid, jwellary) → ~ guilty (butler)
2. steal (mail, jwellary) ∨ milk (maid, cow)
3. silk (maid, cow) → got_Cream (butter)

To prove that

4. guilty (butler) → got_cream (butter)

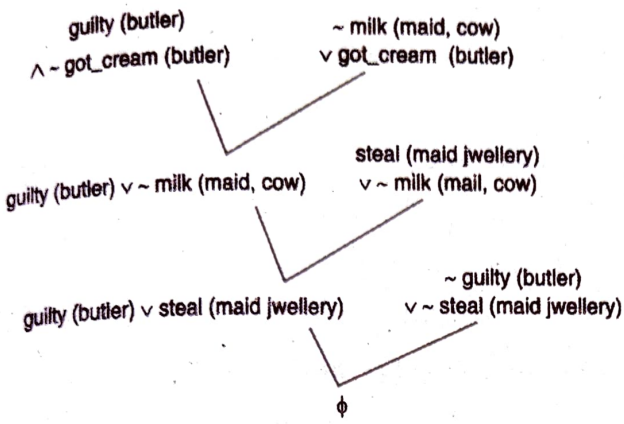
Step 2 : Converting FOL to CNF.

1. ~ steal (maid, jwellary) ∨ ~ guilty (butler)
2. steal (maid, jwellary) ∨ milk (maid, cow)
3. ~ milk (maid, cow) ∨ got_cream (butler)
4. ~ guilty (butler) ∨ got_cream (butler)

Step 3 : Negate the proof sentence

- As sentence 4 is the one to be proved.
- guilty (butler) ∧ ~ got_ cream (butler)

Step 4 : Proof by resolution ∧∨ got_cream (butter)



Hence proved.

Ex. 3.8.6 : Consider following axioms.

All people who are graduating are happy.

All happy people smile.

Someone is graduating.

(i) Represent these axioms in FOL.

(ii) Convert each formula to CNF.

(iii) Prove that someone is smiling using resolution technique. Draw the resolution tree.

Soln. :

Step 1 : Converting axioms to FOL.

(i) $\forall x$: graduating (x) → happy (x).

(ii) $\forall x$: happy (x) → smile (x)

(iii) $\exists x$: graduating (x)

Step 2 : Converting FOL to CNF.

(i) ~ graduating (x) ∨ happy (x)

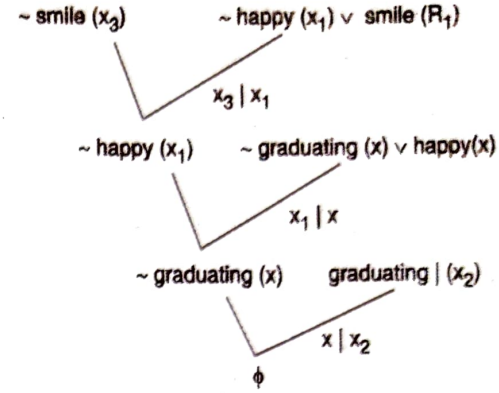
(ii) ~ happy (x_1) ∨ smile (x_1)

(iii) graduating (x_2)

Step 3 : T.P.T. x_3 smile (x_3)

Negating the stmt: ~ smile (x_3)

Step 4 : Proof by resolution



Hence our assumption is wrong.

Hence proved.

Syllabus Topic : Deductive Retrieval

3.9 Deductive Retrieval

3.9.1 Backward Chaining

- For any type of inferencing we have to find a path from start to goal. When we have some data and we make a decision based on this data then the process is called as the **forward channing,** whereas if we have a decision and based on the decision if we fetch the initial data then it is called as **backward chaining.**
- Take an example, "If it is raining then, we will take umbrella". Here, "it is raining" is the data and "we will take umbrella" is a decision. This means that we knew already that it's raining that's why we are going to take umbrella. This process is **forward chaining.**
- Now, take another example, if you are going out and you make a decision of taking umbrella. Then based on this decision it can be guessed that it is raining. Here, "taking umbrella" is a decision based on which we guessed that the data can be "it's raining". This process is **backward chaining.**

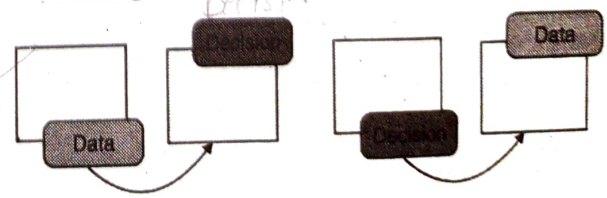


Fig. 3.9.1 : Forward Chaining **Fig. 3.9.2 : Backward Chaining**

- "Forward chaining" is called as a data-driven inference technique and "Backward chaining" is called as a decision-driven or goal-driven inference technique.

Example

Given :

- Rule : human(A) ® mortal(A)
- Data : human(Mandela)

To prove : mortal(Mandela)

☛ **Forward Chaining Solution**

- Human (Mandela) matches Left Hand Side of the Rule. So, we can get A = Mandela based on the above statement we can get : mortal (Mandela).
- Forward chaining is used by the "design expert systems", as it performs operation in a forward direction (i.e. from start to the end).

☛ **Backward Chaining Solution**

It makes use of right hand side matching and backward chaining is used by the "diagnostic expert systems", because it performs operations in a backward direction (i.e. from end to start).

3.9.1(A) Example

- Consider following example. Let us understand how the same example can be solved using both forward and backward chaining.
- Given facts are as follows :
  1. It is a crime for an American to sell weapons to the enemy of America.
  2. Country Nono is an enemy of America.
  3. Nono has some missiles.
  4. All the missiles were sold to Nono by Colonel West.
  5. Missile is a weapon.
  6. Colonel West is American.
- We have to prove that West is a criminal.
- Let's see how to represent these facts by FOL.

1. It is a crime for an American to sell weapons to the enemy nations.
   - American(x) Λ Weapon(y) Λ sell (x, y, z) Λ enemy(z, America) => Criminal (x)
2. Country Nono is an enemy of America.
   - Enemy (Nono, America)
3. Nono has some missiles.
   - Owns (Nono, x)
   - Missile(x)
4. All the missiles were sold to Nono by Colonel West.
   - Missile(x)Λ owns(Nono, x)=> Sell(West, x, Nono)
5. Missile is a weapon.
   - Missile(x)=> weapon(x)
6. Colonel West is American.
   - American (West)

3.9.2 Proof by Forward Chaining

- The proof will start from the given facts. And as we can derive other facts from those, it will lead us to the solution. Please refer to Fig. 3.9.3. As we observe from the given facts we can reach to the predicate Criminal(West).

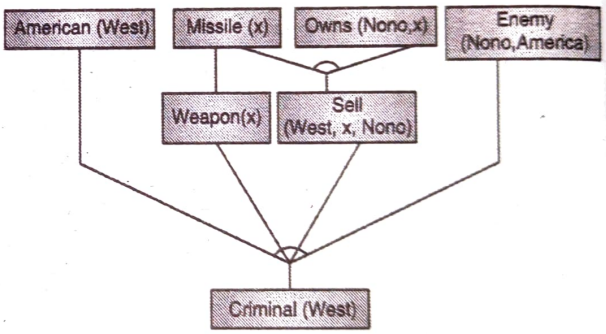


Fig. 3.9.3 : Proof by forward chaining

3.9.3 Proof by Backward Chaining

The proof will start from the fact to be proved. And as we can map it with given facts, it will lead us to the solution. Please refer to Fig. 3.9.4. As we observe, all leaf nodes of the proof are given facts that means "West is Criminal".

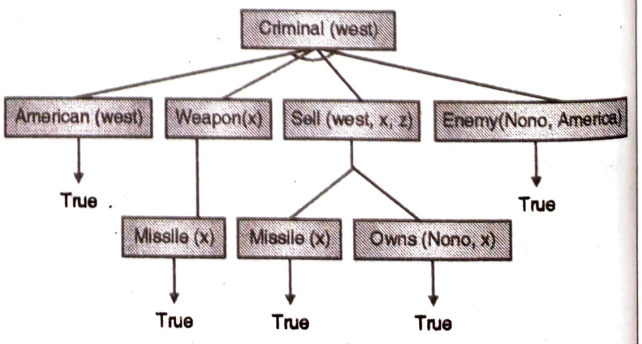


Fig. 3.9.4 : Proof by backward Chaining

Ex. 3.9.1 : Using predicate logic find the course of Anish's liking for the following :

(i) Anish only likes easy courses.

(ii) Computer courses are hard.

(iii) All electronics courses are easy

(iv) DSP is an electronics course.

Soln. :

Step 1 :

Converting given facts to FOL

(i) $\forall x$: course (x) $\wedge$ easy (x) $\rightarrow$ likes (Anish, x)

(ii) $\forall x$: course (x) $\wedge$ computes (x) $\rightarrow$ hard (x)

(iii) $\forall x$: course (x) $\wedge$ electronics (x) $\rightarrow$ easy (x)

(iv) Electronics (DSP)

(v) Course (DSP)

Step 2 : Proof by backward chaining

As we have to find out which course Anish likes. So we will start the proof from the same fact.

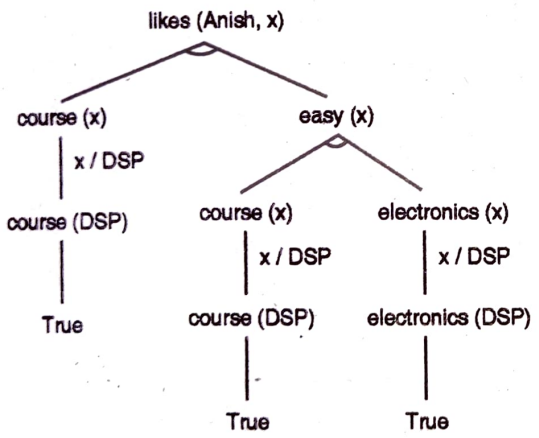


Fig. P. 3.9.1

Syllabus Topic : Second Order Logic

3.10 Second Order Logic

- Second-order logic is an extension of first-order logic where, in addition to quantifiers such as "for every object", one has quantifiers such as "for every property of objects" This augmentation of the language increases its expressive strength, without adding new non-logical symbols, such as new predicate symbols.
- For classical extensional logic, properties can be identified with sets, so that second-order logic provides us with the quantifier "for every set of objects."
- There are two approaches to the semantics of second-order logic. They differ on the interpretation of the phrase "for every set of objects." In the first case, which will be called standard semantics, we are taking for granted certain mathematical concepts. In the second case, which will be called general semantics, much less is being taken for granted. In this case, to be considered valid, a sentence will need to be true under all the allowable meanings of the phrase "for every set of objects."

Syntax

- The language of second-order logic extends the language of first-order logic by allowing quantification of predicate symbols and function symbols.
- As the foregoing example shows, in a second-order language for arithmetic, we can say that the natural numbers are well ordered.
- We know that the well-ordering property is not expressible by any first-order sentence, because the non-standard models of the (first-order) theory of (N; 0, S, <, +, ×) are never well ordered. So going to second-order logic is a genuine extension.
- That is, we can translate some natural-language sentences, such as "The relation < is a well-ordering," into the language of second-order logic that are not translatable into the language of first-order logic.
- For another example, we can (using choice) say that the universe of discourse is infinite by saying that there is a transitive relation on the universe such that every element bears the relation to something, but not to itself:

  $\exists R[\forall x\ \forall y\ \forall z\ (Rxy\ \&\ Ryz \rightarrow Rxz)\ \&\ \forall x[\neg\ Rxx\ \&\ \exists y\ Rxy]$
- Here the second-order quantifier "$\exists R$" expresses the existence of some binary relation on the universe. Because the only predicate symbol, R, is in the scope of this quantifier, the sentence has no predicate symbols open to interpretation. As is well known, no first-order sentence has for its models exactly the infinite structures.

- In more detail, here is what is meant by a second-order language : One starts with a first-order language, and augments it by an unending supply of n-place predicate variables for each positive integer n, and an unending supply of n-place function variables for each positive integer n. (The function variables are can be avoided, but that is another matter.)
- The formation rules for well-formed formulas are the obvious one; in particular universal and existential quantification is allowed for any variable, be it an individual variable, a predicate variable, or a function variable.
- To return to the example of natural numbers, we can express the Peano induction postulate by a second-order sentence:

  $\forall X[X0 \ \& \ \forall y(Xy \rightarrow XSy) \rightarrow \forall y \ Xy]$
- This sentence expresses the idea that X is true of all natural numbers, if it is true of 0 and its truth at some number y guarantees its truth at the successor of y, no matter what set of numbers X might be true of.
- For an example involving the set R of real numbers with its usual ordering, we can express the least-upper-bound property by a second-order sentence:

  $\forall X[\exists y \forall z(Xz \rightarrow z \leq y) \ \&$

  $\exists z \ Xz \rightarrow \exists y \forall y'(\forall z(Xz \rightarrow z \leq y') \leftrightarrow y \leq y')]$
- The foregoing examples are drawn from mathematical situations. There is also the intriguing possibility of natural-language sentences that seem to require second-order formulas for their formalization.
- George Boolos suggested the example, "There are some critics who admire only each other." This sentence asserts the existence of a set of individuals having a certain property; it does not entail, for example, that there are two critics who admire each other and admire no others.

3.10.1 Standard Semantics

- Implicit in the previous section is the concept of truth of a second-order sentence in a structure. Consider a structure M = (A, R, ...) consisting of a non-empty set A serving as the universe of discourse, and some relations and functions on A interpreting the non-logical symbols.
- Then we want to count a second-order sentence of the form $\forall P \ \varphi$ (where P is a k-place predicate variable) as being true in this structure if for every set Q of k-tuples of members of A, we have that φ is true in the structure when P is assigned the relation Q.
- More formally, we need to define inductively what it means for a second-order formula φ to be satisfied in a structure M = (A, R, ...) under an assignment s of objects to the free variables in φ, which will be written $M \models \varphi[s]$. The definition proceeds exactly as in the first-order case, except for the additional clauses for the second-order quantifiers. For a k-place predicate variable P,
- $M \models \forall P \ \varphi[s]$ iff for every k-ary relation Q on A, we have $M \models \varphi[s']$

  where s′ differs from s only in assigning the relation Q to the predicate variable P. (Here "iff" abbreviates "if and only if.") Similarly, for a k-place function variable F,
- $M \models \forall F \ \varphi[s]$ iff for every k-place function G on A, we have $M \models \varphi[s']$

  where s′ differs from s only in assigning the function G to the function variable F. Observe that this definition refers, in the case of a 1-place predicate variable, to all subsets of A, that is, to the entire power set of A. It is this feature that accounts for the extraordinary semantical strength of second-order languages.
- In the case of a second-order sentence σ (i.e., a formula with no free variable), the assignment s is no longer relevant, and we may speak unambiguously of the truth or falsity of σ in the structure M (that is, we can say that M is or is not a model of σ). In particular, the examples in §1 of translations from natural language into the language of second-order logic can now be seen to accomplish their intended purposes. The conjunction of the axioms for a linear ordering and the sentence
- $\exists x \ Px \rightarrow \exists x(Px \ \& \ \forall y(Py \rightarrow (y = x \vee x < y)))$ is true in a structure (A, <) iff the relation < well-orders the set A. The sentence

  $\exists R[\forall x \ \forall y \ \forall z(Rxy \ \& \ Ryz \rightarrow Rxz) \ \& \ \forall x[\neg Rxx \ \& \ \exists y \ Rxy]$

  is true in a structure iff the universe of discourse is an infinite set. This example shows that the compactness theorem does

not hold for second-order logic. If we call the above sentence $\lambda\omega$ and we let λn be the first-order sentence "there are at least n different things in the universe," then the set

$\{\neg\lambda\omega, \lambda 2, \lambda 3, \lambda 4, \ldots\}$

has no model, although each finite subset has a model.

- The conjunction of the Peano postulates

$\forall x \forall y(Sx = Sy \rightarrow x = y)$ and $\forall x(\neg 0 = Sx)$

and the Peano induction postulate

$\forall X[X0 \ \& \ \forall y(Xy \rightarrow XSy) \rightarrow \forall y \ Xy]$

is true in a structure (A, f, e) iff this structure is isomorphic to (N, S, 0), the natural numbers with the successor operation S and distinguished element 0.

- The conjunction of these three sentences provides an example of a sentence that is categorical, that is, it has exactly one model, up to isomorphism. By contrast, a first-order sentence can be categorical only if its one model is finite.
- Similarly, the ordered field of real numbers, (R, 0, 1, +, ×, <), can be characterized up to isomorphism by the first-order axioms for an ordered field, together with the second-order sentence expressing the least-upper-bound property.
- It is well known that any model of these sentences must be isomorphic to the ordered field of real numbers. This example shows that the Löwenheim–Skolem theorem does not hold for second-order logic.
- The preceding examples show that two everyday mathematical structures, (N, S, 0) and (R, 0, 1, +, ×, <) are second-order characterizable. That is, each one has a single second-order axiom of which it is the only model, up to isomorphism. One might ask what other structures might be second-order characterizable.
- Of course, there can be only countably many such structures, up to isomorphism, because each one needs a sentence.
- Next, suppose that in these examples, we existentially quantify all the non-logical symbols (i.e., all the predicate symbols and function symbols). Where $\pi(0, S,)$ is the conjunction of the three Peano postulates, the sentence $\exists x \exists F \pi(x, F)$ is a sentence in the second-order language of equality, that is, it has no non-logical symbols at all.
- A structure for the language of equality consists simply of a non-empty universe of discourse; there are no relations or functions or distinguished elements. Such a structure is of course determined up to isomorphism simply by its cardinality.

3.10.2 General Semantics

- The concept of general semantics for second-order logic avoids any pretense that the power-set operation is a fixed well-understood resource. Instead, the range of the quantifier $\forall X$ must be directly specified.
- By a general pre-structure for a second-order language we mean a structure in the usual sense (a universe of discourse plus interpretations for the non-logical symbols) together with the additional sets.
- For a general pre-structure M, there is a natural way to define what it means for a second-order formula φ to be satisfied in a structure M under an assignment s of objects to the free variables in φ, which again will be written $M \models \varphi[s]$. The second-order quantifiers are now defined to range over the corresponding universe. For a k-place predicate variable P, $M \models \forall P \ \varphi[s]$ iff for every k-ary relation Q in the k-place relation universe, we have $M \models \varphi \ [s']$ where s' differs from s only in assigning the relation Q to the predicate variable P.

Syllabus Topic : Knowledge Representation - Conceptual Dependency, Frames, Semantic Nets

3.11 Knowledge Representation : Conceptual Dependency, Frames, Semantic Nets

- The logical representations are mostly concerned with truth of statements regarding the world. These statements are most generally represented using statements like TRUE or FALSE.
- Logic is successfully used to define ways to infer new sentences from the existing ones. There are certain logics that are used for the representation of information, and range in terms of their expressiveness. There are logic that are more expressive and are more useful in translation of sentences from natural languages into the logical ones. There are several logics that are widely used :

1. **Propositional logic :** These are restricted kinds that make use of propositions (sentences that are either true or false but not both) which can be either true or false. Proposition logic is also known as propositional calculus, sentential calculus or boolean algebra.

   All propositions are either true or false,

   Example

   (i) Leaves are green (ii) Violets are blue.

Sentence	Truth Value	Proposition
Sky is blue	true	yes
Roses are red	true	yes
2 + 2 = 5	false	yes

2. **First Order Predicate Logic :** These are much more expressive and make use of variables, constants, predicates, functions and quantifiers along with the connectives explained already in previous section.
3. Higher order predicate logic.
4. **Fuzzy logic :** These indicate the existence of in between ness or fuzziness in all logics.
5. **Other logic :** These include multiple valued logic, modal logics and temporal logics.

- One of the widest used methods to represent knowledge is to use production rules, it is also known as IF-THEN rules.

Examples

IF condition THEN action
IF premise THEN conclusion
IF proposition p1 and proposition p2 are true
THEN proposition p3 is true

- Some of the benefits of IF-THEN rules are that they are modular, each defining a relatively small and, at least in principle, independent piece of knowledge. New rules may be added and old ones deleted usually independently of other rules.
- Production rules are simple but powerful forms of representing knowledge, they provide flexibility for combining procedural and declarative representations in a unified manner.
- The major advantage of production rules are that they are modular, independent of other rules with the provision for addition new rules and deleting older ones.
- **Semantic networks :** These represent knowledge in the form of graphical networks, since graphs are easy to be stored inside programs as they are concisely represented by nodes and edges.
- A semantic network basically comprises of *nodes* that are named and represent concepts, and labelled *links* representing relations between concepts. Nodes represent both types and tokens.
- Example, the semantic network in Fig. 3.11.1 expresses the knowledge to represent the following data :
  - Tom is a cat.
  - Tom caught a fish.
  - Tom is grey in color.
  - Tom is owned by Sam.
  - Tom is a Mammal.
  - Fish is an Animal.
  - Cats love Milk.
  - All mammals are animals.

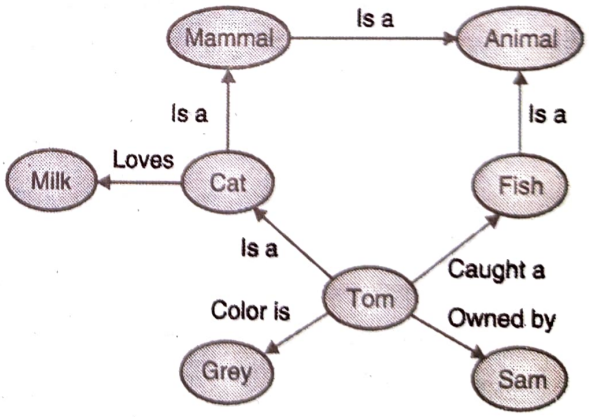


Fig. 3.11.1

- **Conceptual Graph :** It is a recent scheme used for semantic network, introduced by John Sowa, has a finite, connected, bipartite graph. The nodes represent either concepts or conceptual relations. It differs from the previous method that it does not use labelled arcs.
- **Example :** Ram, Laxman and Bharat are Brothers or cat color is grey can be represented as shown in Fig. 3.11.2.

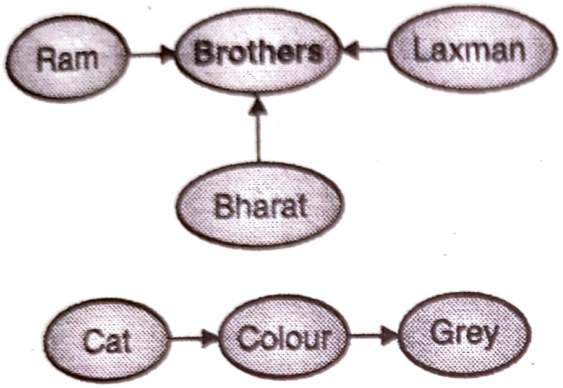


Fig. 3.11.2

- **Frames :** This concept was introduced by Marvin Minsky in 1975. They are mostly used when the task becomes quite complex and needs more structured representation. More structured the system becomes more would be the requirement of using frames which would prove beneficial. Generally frames are record like structures that consists of a collection of slots or attributes and the corresponding slot values.
- Slots can be of any size and type. The slots have names and values (subfields) called as facets. Facets can have names or numbers too.
- A simple frame is shown in the Fig. 3.11.2 for a person Ram,
  - (Ram)
  - (PROFESSION (VALUE professor))
  - (AGE(VALUE 50))
  - (WIFE(VALUE sita))
  - (CHILDREN(VALUE luv kush))
  - (ADDRESS (STREET(VALUE 4C gb road)))
  - CITY(VALUE banaras))
  - (STATE(VALUE mh))
  - (ZIP(VALUE400615))

❑❑❑