CHAPTER : FOUR
Knowledge representation (KR)

Knowledge

Knowledge and Its Types

- ✓ First let's understand what's "knowledge"? Some scholars define knowledge as "Understanding of a subject area", but in various subject areas it's well-focused as "knowledge domain" like Medical Domain, Engineering Domain, Business Domain, etc.

Knowledge includes facts about the real world entities and the relationship between them

Knowledge-based Systems (KBSs) are useless without the ability to represent knowledge.

we can categorize these knowledge types broadly as follows

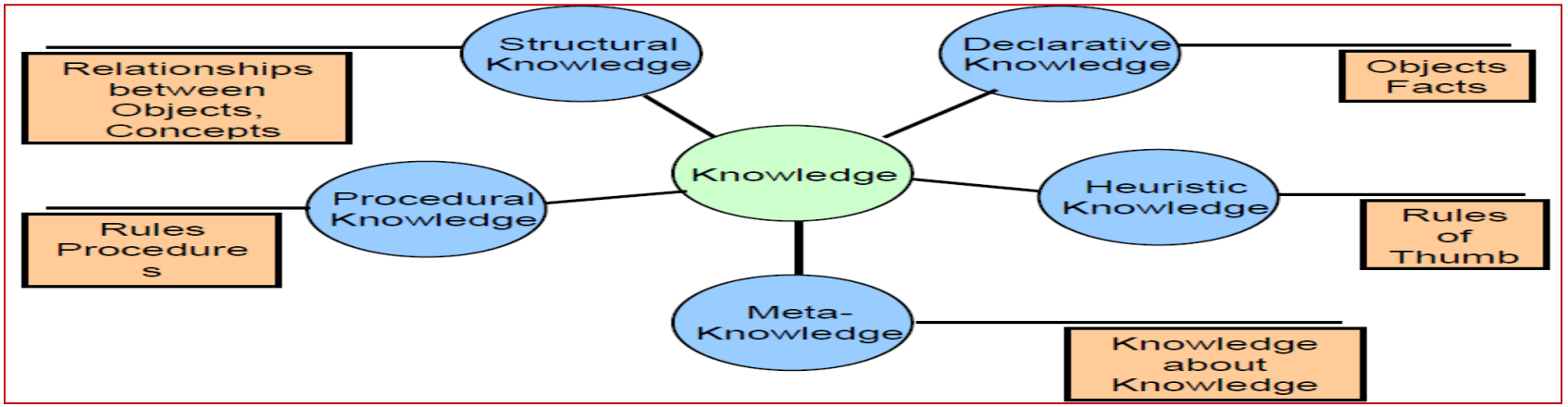
i) **Procedural Knowledge:** It describes how to do things, provides a set of directions of how to perform certain tasks. For example, how to drive a car.

ii) **Declarative Knowledge:** It describes objects, rather than processes or what is known about a situation. For example, it is sunny today and cherries are red.

iii) **Meta Knowledge:** It's knowledge about knowledge. For example, to diagnose someone's medical condition knowledge of Blood Pressure is more important than the Eye Color.

iv) Heuristic Knowledge: It's Rule-of-Thumb, for example, "if I start seeing shops, I am close to the market.". Heuristic knowledge is sometimes called ***shallow knowledge***.

v) Structural Knowledge: Describes structures and their relationships. For example, how the various parts of a car fit together to make the car. So it could be knowledge structures in terms of concepts, sub concepts and objects.



Why Knowledge is important ?

- It enables to:
  - Automate reasoning
  - Discover new facts.
  - Deduce new facts that follow from the KB
  - Answer users queries
  - Make quality decisions - select courses of actions, etc.
- Hence, there is a need to represent knowledge to ease the development of an intelligent system.

– **Towards Representation**

- Thinking about knowledge representation there are multiple approaches and schemes, it can be in ***Pictures and Symbols*** (it's how the earliest humans represented knowledge when sophisticated linguistic systems had not yet evolved), ***Graphs and Networks*** and ***Numbers***.

1. Pictures

- Knowledge can be represented by pictures, for example relationships between individuals in a family. We could also use a series of pictures to store procedural knowledge like how to boil an egg. Thus, we can easily see that pictures are best suited for recognition tasks and for representing structural information.
- Since pictures provide a high-level view of a concept to be obtained they are useful for human understanding, but translating the useful information of pictorial representations in to computers is not as straight forward or *not very easily*, because computers cannot interpret pictures directly without complex reasoning. So. See Figure it's a pictorial representation of a knowledge of some family relationship.

Figure : Pictorial Representation of Family Relationships

- ✓ This kind of representation makes sense readily to humans, but if we give this picture to a computer, it would not have an easy time figuring out the relationships between the individuals or even figuring out how many individuals are there in the picture.
- ✓ For instance, computers might need complex computer vision algorithms to understand pictures.

2. Graphs and Networks

Graphs and networks allow relationships between objects/entities to be incorporated, for example in the below Figure a graph is used to show family relationships. This representation highlights family relationship and its more direct.

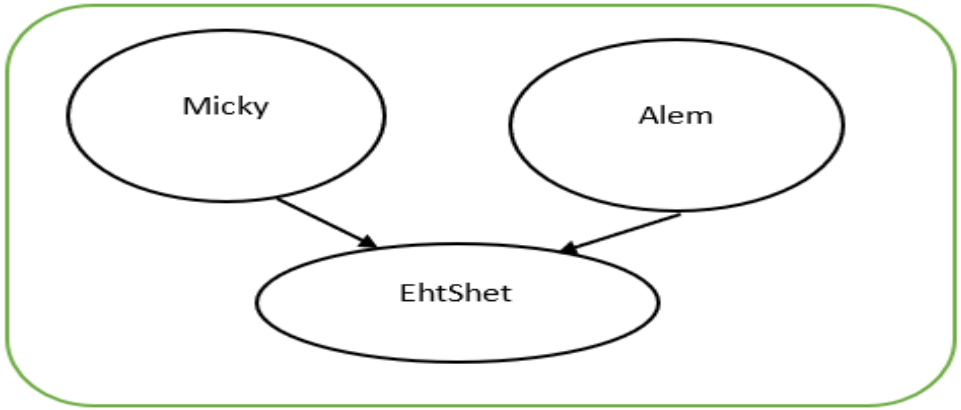


Figure : Graph Representation of Family Relationships

In words we can describe the above family relationship as follows: -

- Micky is EhtShet's Father
- Alem is EhtShet's Mother
- EhtShet is Micky and Alem's Daughter

We can also represent procedural knowledge using graphs, as shown in Figure below it is a knowledge about “how to start a car”.

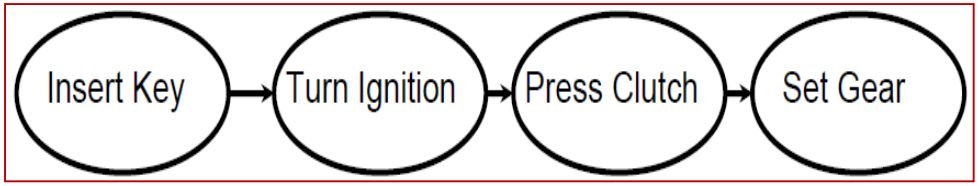


Figure :Graph Representation for Procedural Knowledge

3. Numbers

- ✓ Numbers are an integral part of knowledge representation used by humans and they can be easily translated into computers. Because, eventually every representation we use gets translated into numbers in the computer’s internal representation.
- ✓ Remember that each knowledge representation scheme has its own strengths and weaknesses.

Formal Knowledge Representation Techniques

- ✓ In the above examples we explored intuitive ways of knowledge representations, but how the formal Knowledge Representation (KR) techniques in AI is discussed next.
- ✓ Each technique is suited to represent a certain type of knowledge, so choosing proper representation is an important aspect because it's related with reasoning.

1. Facts

- ✓ Facts are a basic building blocks of knowledge, or they are atomic units of knowledge. Fact represent declarative knowledge, for example we can declare knowledge about objects.
- ✓ A ***proposition*** is a statement of some given fact, so that each proposition has an associated truth value (that is either ***True*** or ***False***).
- ✓ Thus, in AI we represent facts with such propositions and associated truth values, for example:

- Proposition A: "***It is raining.***"
- Proposition B: "***I have an umbrella.***"
- Proposition C: "***I will go to school.***"

1.1. Types of Facts

A) Single-Valued or ***Multiple–Valued***

•Facts may be single-valued or multi-valued, where each fact (attribute) can take one or more values at the same time.

•For example, an individual can only have one eye color, but may have many cars, so that attribute of cars may contain more than one value.

B) Uncertain Facts

•Sometimes we need to represent facts with uncertain information, for example "***it will probably be sunny today***".

•We may choose to store numerical certainty values to tell that how much uncertainty there is in the fact.

C) Fuzzy Facts

•Fuzzy facts are ambiguous in nature, for example "***the book is heavy or light***", because it is unclear what heavy means or it is a subjective description.

•While defining fuzzy facts, we use certainty factor values to specify value of "truth".

D) Object-Attribute-Value (OAV) Triplets

- Object-Attribute-Value Triplets or OAV triplets are a type of fact composed of three parts (object, attribute and value).
- Such facts are used to assert a particular property of some object, for example in the fact "***Ali's eye color is brown.***" Ali is the Object, Eye Color is an Attribute and its Value is Brown.
- Again in the fact "Ahmed's son is Ali." Object is Ahmed, Attribute is Son and Value is Ali (Figure describes these OAV Triplets diagrammatically).

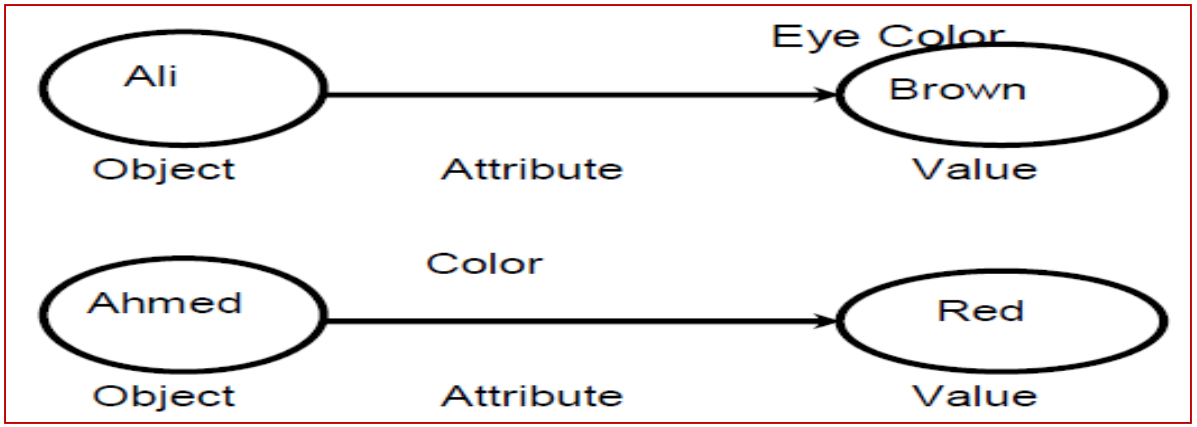


Figure : Example of OAV Triplets

2. Rules

- Rules are another form of knowledge representation and we can define rules as follows.
- "***A knowledge structure that relates some known information to other information that can be concluded or inferred to be true.***"

1. Components of a Rule

- A Rule consists of two components: -

a) Antecedent or Premise (which is the IF part)

b) Consequent or Conclusion (which is the THEN part)

- Example: In the rule "***If it is raining, then I will not go to school.***"

  The Premise is: "***It is raining***"

  The Conclusion is: "***I will not go to school***"

2. Compound Rules

- We can also use connectives like ***AND*** (Conjunctions) and **OR** (Disjunctions) to join multiple premises or antecedents, look examples below.
- "IF it is raining **AND** I have an umbrella, THEN I will go to school."
- "IF it is raining **OR** it is snowing, THEN I will not go to school."

Semantic Networks

- ✓ Semantic networks are graphs with Nodes representing objects and Arcs representing relationships between objects.
- ✓ Various types of relationships maybe defined using semantic networks, but the two most common types of relationships are the IS-A (Inheritance Relation) and the HAS (Ownership Relation).
- ✓ Let's consider an example that demonstrates how knowledge is represented in a semantic network, as shown in Figure

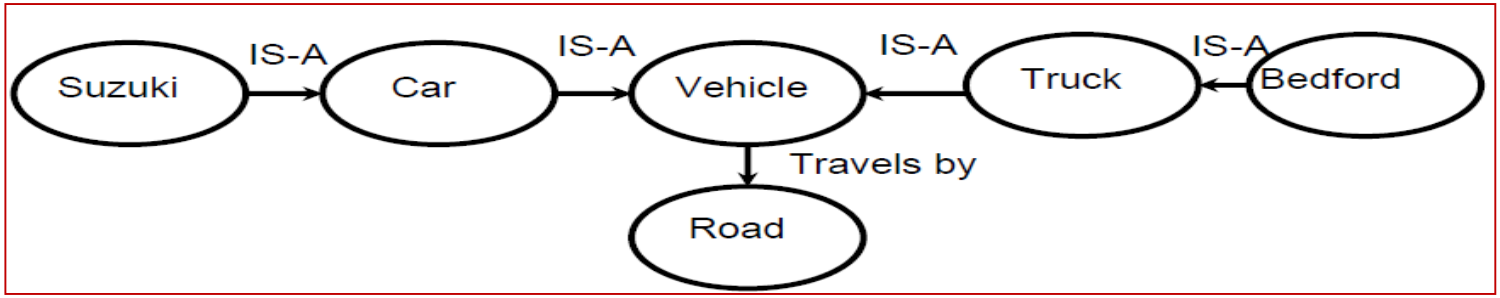


Figure : Vehicle Semantic Network

Network Operation

As shown below to infer new information from semantic networks, we can ask questions from nodes.

- Ask node Vehicle: "How do you travel?"
  - ➢This node looks at arc and replies: Road

- Ask node Suzuki: “How do you travel?”
  - This node does not have a link to travel therefore it asks other nodes linked by the IS-A link.
  - Asks node Car (because of IS-A relationship)
  - Asks node Vehicle (IS-A relationship)
  - Node Vehicle Replies: Road

Frames

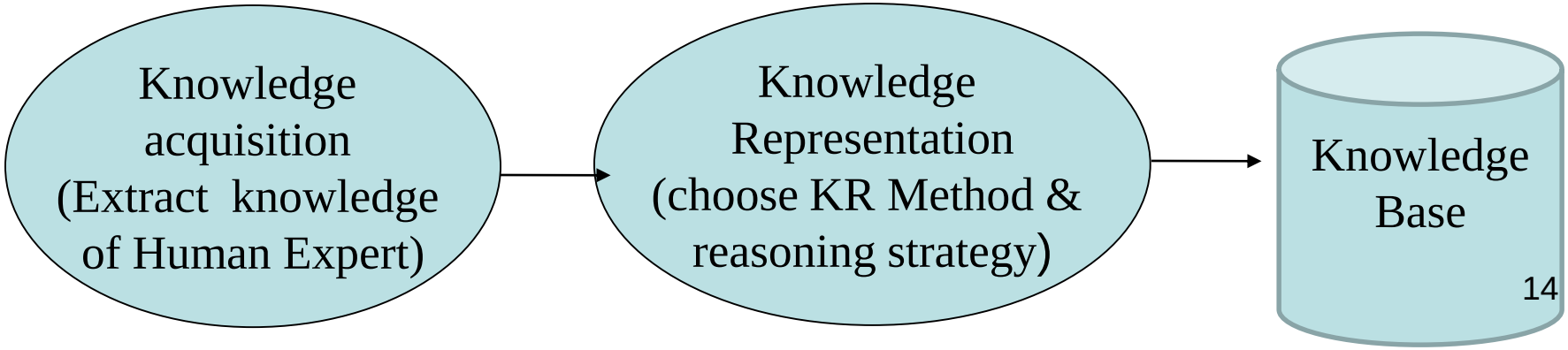
- we can define frames as “data structures for representing stereotypical knowledge of some concept or object”.
- So that a frame is like a schema as we would call it in a database design, they were developed from semantic networks and later evolved into our modern-day Classes and Objects.
- For example, Figure : is a frame representation of student.

Figure : Example of Frame Representation

The various components within the frame are called slots, or example Name is a slot.

Knowledge engineering (KE)

- KE is the process of building a knowledge base through extracting the knowledge from the human expert.
- Knowledge engineering is the process of
  - –Extracting the knowledge from the human expert.
  - –Choose knowledge representation formalism
  - –Choose reasoning and problem solving strategy.
- A knowledge engineer is someone who investigates a particular domain, determines what concepts are important in that domain, and creates a formal representation of the objects and relations in the domain.
  - –A KE has to decide what objects and relations are worth representing, and which relations hold among which objects



The Three main tasks of KE

- **Knowledge acquisition:** The knowledge engineer interview the real human experts to be educated about the domain and to elicit the required knowledge, in a process called knowledge acquisition
- **Knowledge Representation** techniques such as logic are a powerful tool for KR and reasoning. However, such techniques consists of only the syntax, semantics and proof theory.
  - –KR techniques do not offer any guidance as to what facts should be expressed, nor what vocabulary should be used to express them
- **Knowledge base** is used to store a set of facts and rules about the domain expressed in a suitable representation language
  - – Each individual representation are called sentences
  - – Sentences are expressed in a (formal) **knowledge representation (KR) language**

LOGIC AND REASONING

Logic

- ✓ One of the prime activities of human intelligence is reasoning.
- ✓ The activity of reasoning involves construction, organization and manipulation of statements to arrive at new conclusions.
- ✓ Thus, logic can be defined as a scientific study of the process of reasoning and the system of rules and procedures that help in the reasoning process. There are two logic types: -
  - ➢ Propositional Logic (PL)
  - ➢ Predicate Logic or First-Order Predicate Logic (FOPL)

Propositional Logic (PL)

- ✓ Propositional Logic (PL) is the simplest form of logic.
- ✓ It is made of statements and we call each statement as proposition.
- ✓ In propositional logic a proposition takes only two values, that is either TRUE or FALSE.

For example, consider the following statements: -

Statement 1:Gondar was former capital city of Ethiopia.

Statement 2: Sun rises in the west.

Statement 3: What is your name?

- ❖ First two statements are propositions with Statement 1 having a value of TRUE and Statement 2 having a value of FALSE.
- ❖ Third statement is not a proposition, because it can't take a value TRUE or FALSE.

As shown below, usually we can assign a symbolic variable to represent a proposition.

P: It is raining

Q: I carry an umbrella

There are two kinds of propositions, these are **Atomic** or **Simple** Propositions and **Compound** or **Molecular** Propositions

- ✓ Compound Propositions are formed by combining one or more atomic propositions using a set of logical connectives. For example, "Addisu is member of ITD staff." ITD is an atomic proposition. But "Addisu is member of ITD staff and he is also member of CG chair." is a compound proposition, because it is formed from two atomic propositions.
- ✓ When looking at the real-world problems compound propositions are more useful than atomic propositions.
- ✓ The logical connectives that are generally used to form compound propositions are given below in Table.

Name	Connective	Symbols
Conjunction	AND	∧
Disjunction	OR	∨
Negation	NOT	¬
Conditional	If … then (IMPLIES)	→
Bi-conditional	If and only if (IFF)	↔

Table : Commonly Propositional Logic Connectives

- ✓ In addition to the connectives listed above in Table , ***Parentheses*** (left and right), ***Braces*** (left and right), the ***Period*** and delimiters are also used for punctuation.
- ✓ For example, the compound sentence "**It is raining and the wind is blowing.**" could be represented as **R∧B** where **R** stand for propositions "**It is raining.**" and **B** stand for "**The wind is blowing.**".

- ✓ If we write **R∨B** we mean "**It is raining or the wind is blowing or both**.".

Let's sum up the above discussion with the following points.

- Some examples of Atomic or Simple Propositions are: -
  - ✓ It is raining.
  - ✓ My home is painted silver.
  - ✓ Our PM has a PhD in Computer Science.
  - ✓ People live on the moon.
- Examples of Compound Propositions are: -
  - ✓ It is raining and the wind is blowing.
  - ✓ Addis Ababa is capital of Ethiopia and electricity is available 24 hours in Ethiopia.
  - ✓ If you study hard you will be rewarded.
- Most often we use capital letters to stand for propositions (sometimes followed by digits), and we use the special symbols T and F for the values true and false, respectively.

Syntax of Propositional Logic

The syntax of PL is defined recursively as follows: -

- **T** and **F** are **formulas**.
- If **P** and **Q** are **formulas**, the following are formulas: -
  - **¬P**
  - **P ∧ Q**
  - **P ∨ Q**
  - **P → Q**
  - **P ↔ Q**

All formulas are generated from a finite number of the above operations.
An example of a compound formula is: -

((P ∧ Q) → (R ∨ ¬S))

When there is no chance of ambiguity, we will omit parentheses, for example if we omit the parentheses from the above statement, we can get the following: -

(P ∧ Q) → (R ∨ ¬S)

Omitting the parentheses, given the connectives their precedence from highest to lowest is:-

¬, ∧, ∨, →, and ↔

Propositional logic (PL) sentences

- A sentence is made by linking prepositional symbols together using logical connectives.
  - There are atomic and complex sentences.
  - Atomic sentences consist of propositional symbol (e.g. P, Q, TRUE, FALSE)
  - Complex sentences are combined by using **connectives** or **parenthesis:**
  - while S and T are atomic sentences, $S \vee T$, $(S \vee T)$, $(S \wedge T)$, $(S \rightarrow T)$, and $(S \leftrightarrow T)$ are complex sentences.

Examples: Given the following sentences about the "weather problem" convert them into **PL sentences:**

- "It is humid.": Q
- "If it is humid, then it is hot" : $Q \rightarrow P$
- "If it is hot and humid, then it is raining": $(P \wedge Q) \rightarrow R$

Exercise

Examples: Convert the following **English sentences to Propositional logic**

Let A = Lectures are active and R = Text is readable, P = Kebede will pass the exam, then represent the following:

- the lectures are ***not*** active: $\neg A$
- the lectures are active **and** the text is readable: $A \wedge R$
- either the lectures are active **or** the text is readable: $A \vee R$
- **if** the lectures are active, **then** the text is not readable: $A \rightarrow \neg R$
- the lectures are active **if and only if** the text is readable: $A \leftrightarrow R$
- if the lectures are active, then if the text is not readable, Kebede will not pass the exam: $A \rightarrow (\neg R \rightarrow \neg P)$

Semantics of Propositional Logic

- ✓ The semantics or meaning of a sentence is just the values true or false, that is an assignment of a truth value to the sentence.
- ✓ An interpretation for a sentence or group of sentences is an assignment of a truth value to each propositional symbol.

For example, consider the statement $(P \wedge \neg Q)$.

- ✓ One interpretation assigns true (T) to P and false (F) to Q. A different interpretation assigns T to P and T to Q.
- ✓ Clearly, there are four distinct interpretations for this sentence.

Table below summarizes the semantic rules where T and T' denote any true statements, F and F' denote any false statements and A (either true or false) is any statement.

Rule Number	True Statements	False Statements
1	T	F
2	¬F	¬T
3	T ∧ T’	F ∧ A
4	T ∨ A	A ∧ F
5	A ∨ T	F ∨ F’
6	A → T	T → F
7	F → A	T ↔ F
8	T ↔ T’	F ↔ T
9	F ↔ F’	

Table : Semantics Rules for Statements

•We can now find the meaning of any statement for a given interpretation, for example let an interpretation assigns **T** to **P**, **F** to **Q** and **F** to **R** in the following statement: -

• **((P ∧ ¬Q) → R) ∨ Q**

Application of Rule 2 then gives **¬Q** as **T**, Rule 3 gives **(P ∧ ¬Q)** as **T**, Rule 6 gives **(P ∧ ¬Q) → R** as **F**, and Rule 5 gives the statement value as **F** (false).

•A clear meaning of the logical propositions can be arrived at by constructing appropriate truth tables for the compound propositions.

A	**B**	**¬A**	**¬B**	**A ∨ B**	**A ∧ B**	**A → B**	**A ↔ B**
T	T	F	F	T	T	T	T
F	T	T	F	T	F	T	F
T	F	F	T	T	F	F	F
F	F	T	T	F	F	T	T

Table : Truth Tables for Logical Connectives

Properties of Statements

- Some properties of statements are stated below: -

I. Satisfiable: A statement is satisfiable if there is some interpretation for which it is true.

- Example is **P**.

II. Contradiction: A sentence is contradictory (unsatisfiable) if there is no interpretation for which it is true.

- For example, **P ∧ ¬P** is contradiction since every interpretation result in a value of false for **(P ∧ ¬P)**.

III. Tautology (Valid): A sentence is valid if it is true for every interpretation.

- For example, **P ∨ ¬P** is valid since every interpretation result in a value of true for **(P ∨ ¬P)**.
- Remember that a valid sentence is also called tautology.

IV. Equivalence: Two sentences are equivalent if they have the same truth value under every interpretation.

- For example, **P** and **¬(¬P)** are equivalent since they have the same truth values under every interpretation.

Terminology

- **Valid sentence:** A sentence is **valid sentence** or **tautology** if and only if it is True under all possible interpretations in all possible worlds.

  Example: "It's raining or it's not raining." ($R \vee \neg R$).

- **Satisfiable**: A sentence is satisfiable if and only if there is some interpretations in some world for which the sentence is True.

  Example: "It is raining or it is humid". R v Q, R

- **Unsatisfiable:** A sentence is unsatisfiable (inconsistent sentence or self-contradiction) if and only if it is not satisfiable, i.e. a sentence that is False under all interpretations. The world is never like what it describes.

  Example: "It's raining and it's not raining." $R \wedge \neg R$

Inference Rules

- Inference rules of PL provide means to perform logical ***proofs*** or ***deductions***, here below are few inference rules.

1) Modus Ponens

- From **P** and **P → Q** we can infer **Q**, this rule can be written as follows: -

- **P**
- **P → Q**
- **∴ Q**

- For example
  - ✓ Given two statements

- "***Adamu is a father.***"
- "***Adamu is a father → Adamu has a child.***"
  - ✓ We can conclude that "***Adamu has a child.***".

1) Chain Rule

- From **P → Q** and **Q → R** we can infer **P → R**, this rule can be written as: -

- **P → Q**
- **Q → R**
- **∴ P → R**

1) Transposition

- From **P → Q** we can infer **¬Q → ¬P**, this rule can be written as follows: -

- **P → Q**
- **∴ ¬Q → ¬P**

First-Order Predicate Logic (FOPL)

- First-Order Logic (FOL) is expressive enough to concisely represent any kind of situation that are expressed in natural language.
- FOL is type of PC (Predicate calculus)
- Propositional logic works fine in situations where the result is either TRUE or FALSE, but not both.
- Expressiveness is one of the requirements for any serious representation scheme, furthermore PL does not permit us to make generalized statements about classes of similar objects.
- For example, from the following two statements, it should be possible to conclude that ***"kebede must take Calculus course."***.
- ***"All students in Computer Science must take Calculus."***
- ***"kebede is a Computer Science student."***

As stated, it is not possible to conclude in PL that ***kebede*** must take ***Calculus*** since the second statement does not occur as part of the first one.

- FOPL was developed by logicians to extend the expressiveness of PL.
- It is a generalization of PL that permits reasoning about world objects as relational entities as well as classes of objects.

1. *Syntax of FOPL*

- The symbols and rules of combination permitted in FOPL are defined as follows: -

I. Connectives

- There are five connectives symbols, these are: -

- ¬ **Not (Negation)**
- ∧ **And (Conjunction)**
- ∨ **Or (Disjunction)**
- → **Implication**
- ↔ **Equivalence (If-and-only-if)**

II. Quantifiers

- Predicate calculus allows us to use quantifiers for statements in order to refer ***Some*** or ***All*** objects from some set of objects. In basic predicate calculus we use two types of logical quantifiers, these are the Universal Quantifies and the Existential Quantifier.

***A) <u>The Universal Quantifier</u>* (∀)**

- The symbol for the universal quantifier is ∀, it is read as "***for every***" or "***for all***" and used in formulae to assign the same truth value to all variables in the domain.
- For example, in the domain of numbers, we can say **(∀X) (X + X = 2X)**.
- It means that "***For every x (where x is a number), x + x = 2x is true.***".

***B) <u>Existential</u> <u>Quantifier</u>* (∃)**

- The symbol for the existential quantifier is ∃, it is read as "***there exists***" or "***for some***" or "***for at least one***" or "***there is one***". It is used in formulae to say that something is true for at least one value in the domain.
- For example, for domain of persons, ***(∃X) (Person(X) ∧ Father(X, Alemu))***.
- This can be read as "***There exists some person, X who is Alemu's father.***".

III. Constants

- Constants are fixed-value terms that belongs to a given domain of discourse. Constants are used to name specific objects or properties, for example Mule, Tomi, Blue, Ball, etc.

IV. Predicates

•A fact or proposition is divided into two parts: -

i)Predicate: It is the assertion of the proposition.

ii)Argument: It is the object of the proposition.

•For example, the proposition "***Addisu likes Tibs.***" can be represented in predicate logic as ***Likes(Addisu, Tibs)***, where ***Likes*** is the predicate and ***Addisu*** and ***Tibs*** are the arguments.

V. Variables

•Variables are used to a represent general class of objects/properties, for example in the predicate **Likes(X, Y)**, **X** and **Y** are variables that assume the values **X=Addisu** and **Y=Tibs**.

VI. Functions

•Function symbols denote relations defined on a domain, for example symbols **f**, **g** and **words** represent functions.

VII. Formulae

•Formulas combine predicates and quantifiers to represent information, see Figure below that illustrate these symbols using an example.

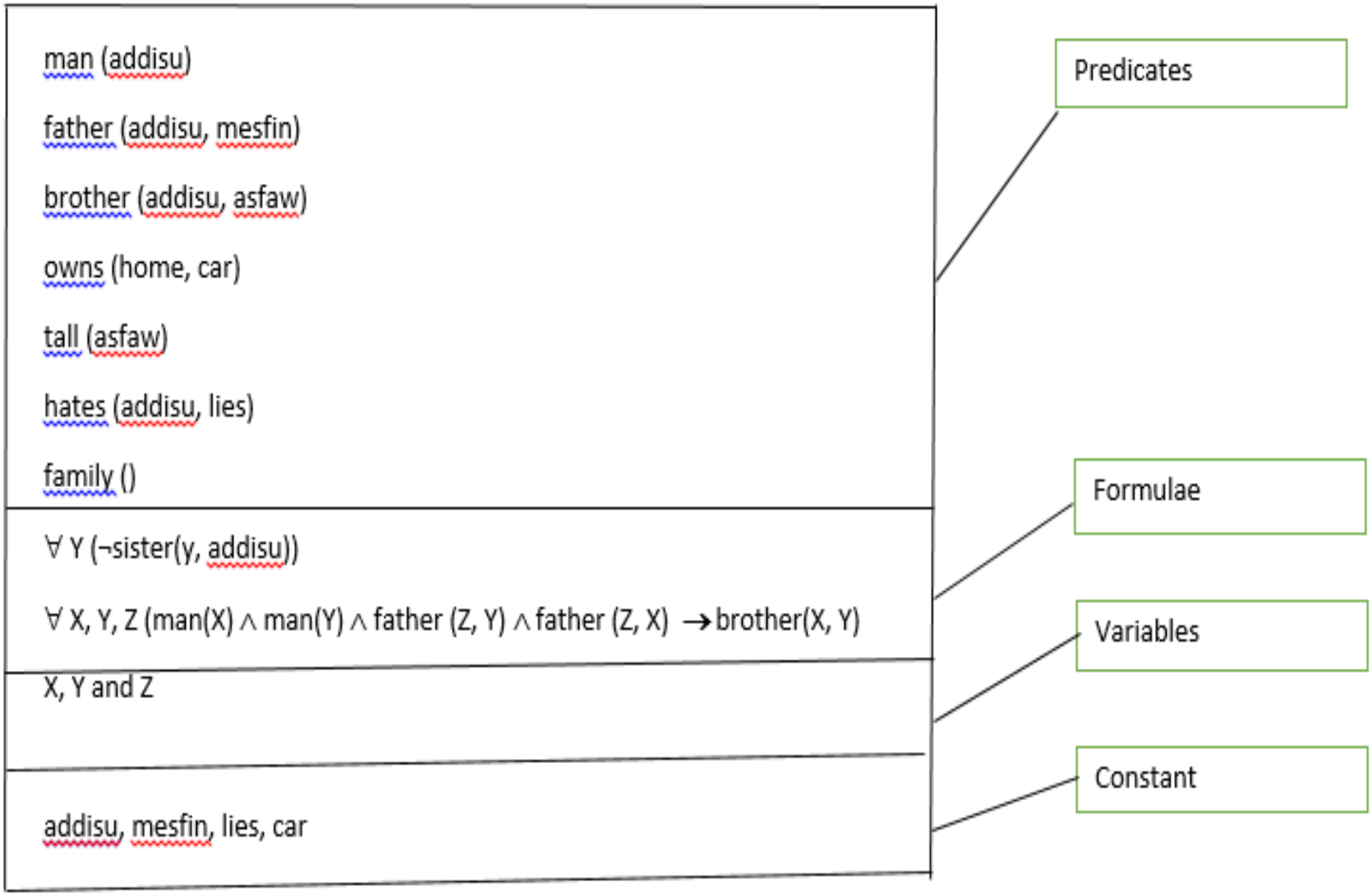


Figure : Example of Formulae

- The predicate section outlines the known facts about the situation in the form of predicates, that is predicate name and its arguments.
- So, **man(addisu)** means that **addisu** is a **man**, **hates(addisu, lies)** means that **addisu** hates **lies**.
- The formulae section outlines formulae that use universal quantifiers and variables to define certain rules.

- **∀Y (¬sister(Y, addisu))** says that there exist
- s no **Y** such that **Y** is the **sister** of **addisu**, that addisu has no sister.
- Similarly,**∀X, Y, Z (man(X) ∧ man(Y) ∧ man(Z) ∧ father(Z, Y) ∧ father(Z, X) → brother(X, Y))** means that if there are three men, **X**, **Y** and **Z**, and **Z** is the father of both **X** and **Y**, then **X** and **Y** are bothers. This expresses the rule for the two individuals being brothers.

- Suppose we want to represent the following statements in FOPL form: -
- E1: All employees earning Birr 14000 or more per year pay taxes.
- E2: Some employees are sick today.
- E3: No employee earns more than the president
- To represent such expressions in FOPL, we must define abbreviations for the predicates and functions. We might for example, define the following:
- E(X) for X is an employee.
- P(X) for X is president.
- I(X) for the income of X (lower case denotes a function).
- GE(U, V) for U is greater than V.
- S(X) for X is sick today.
- T(X) for X pays taxes.
- Using the above abbreviations, we can represent E1, E2 and E3 as
- E1: $\forall X((E(X) \wedge GE(I(X), 14000)) \rightarrow T(X)$
- E2: $\exists X\ (E(X) \rightarrow S(X))$
- E3: $\forall X \exists Y\ ((E(X) \wedge P(Y)) \rightarrow \neg GE(I(X), I(Y)))$

Reasoning

- So far in this chapter we've looked at knowledge representation and logic, next is about mechanisms to reason on the knowledge once we have represented it using some logical scheme.
- ***What's Reasoning?***
  - Reasoning is the process of deriving logical conclusions from given facts.
  - According to Durkin definition reasoning is "***the process of working with knowledge, facts and problem-solving strategies to draw conclusions***".
- In this section we'll look at six types of reasoning: -

1. Deductive Reasoning

- As the name implies, Deductive Reasoning is based on deducing new information from logically related known information. A deductive argument offers assertions that lead automatically to a conclusion, for example: -
  - **If there is dry wood, oxygen and a spark, there will be a fire**.
    - ✓Given: **There is dry wood, oxygen and a spark/flash**.
    - ✓We can deduce: **There will be a fire**.
  - Given: **All men are mortal. Socrates is a man.**
    - ✓We can deduce: **Socrates is mortal.**

2. Inductive Reasoning

- Inductive reasoning is based on forming, or inducing a “generalization” from a limited set of observations, for example: -

- Observation: **All the crows that I have seen in my life are black**.
- Conclusion: **All crows are black**

- ***Comparison of Deductive and Inductive Reasoning***
- We can compare deductive and inductive reasoning using an example. We conclude what will happen when we let a ball go using both each type of reasoning in turn.

- The inductive reasoning is as follows: -

- **By experience, every time I have let a ball go, it falls downwards. Therefore, I conclude that the next time I let a ball go, it will also come down.**

- The deductive reasoning is as follows: -

- **I know Newton's Laws. So, I conclude that if I let a ball go, it will certainly fall downwards.**

- Thus, the essential difference is that inductive reasoning is ***based on experience*** while deductive reasoning is ***based on rules***, hence the latter will always be correct.

3. Abductive Reasoning

- Deduction is exact in the sense that deductions follow in a logically provable way from the axioms. Abduction is a form of deduction that allows for plausible inference, that is the conclusion might be wrong, for example: -
  - Implication: **She carries an umbrella if it is raining.**
  - Axiom: **She is carrying an umbrella.**
  - Conclusion: **It is raining.**
- This conclusion might be false, because there could be other reasons that she is carrying an umbrella, for example she might be carrying it to protect herself from the sun.

4. Analogical Reasoning

- Analogical reasoning works by drawing analogies between two situations, looking for similarities and differences. For example, when you say driving a truck is just like driving a car, by analogy you know that there are some similarities in the driving mechanism, but you also know that there are certain other distinct characteristics of each.

5. Common-Sense Reasoning

- Common-sense reasoning is an informal form of reasoning that uses rules gained through experience or what we call rules-of-thumb. It operates on heuristic knowledge and heuristic rules.

6. Non-Monotonic Reasoning

- Monotonic reasoning is used when the facts of the case are likely to change after some time, for example: -
  - Rule: **IF the wind blows, THEN the curtains sway.**
  - When the wind stops blowing, the curtains should sway no longer.
- In non-monotonic reasoning, we have a “truth maintenance system”.
- It keeps track of what caused a fact to become true.
- If the cause is removed, that fact is removed also.

Inference

- Inference is the process of deriving new information from known information.
- In the domain of AI, the component of the system that performs inference is called an inference engine.
- We will look at inference within the framework of "logic", which we introduced earlier.
- Logic, which we introduced earlier, can be viewed as a formal language. As a language, it has the following components: Syntax, Semantics and Proof Systems.

Syntax

- Syntax is a description of valid statements, the expressions that are legal in that language.
- We have already looked at the syntax of two types of logic systems called propositional logic and predicate logic.
- The syntax of proposition gives us ways to use propositions, their associated truth value and logical connectives to reason.

Semantics

- Semantics pertain to what expressions mean, e.g. the expression 'the cat drove the car' is syntactically correct, but semantically non-sensible.

Proof Systems

- A logic framework comes with a proof system, which is a way of manipulating given statements to arrive at new statements.
- The idea is to derive ‘new’ information from the given information.
- A proof is a sequence of statements aiming at inferring some information.
- While doing a proof, we usually proceed with the following steps:

☞ Begin with initial statements, called premises of the proof (or knowledge base).

☞ Use rules, i.e. apply rules to the known information.

☞ Add new statements, based on the rules that match.

☞ Repeat the above steps until you arrive at the statement you wished to prove.

Rules of Inference

- Rules of inference are logical rules that you can use to prove certain things.

As you look at the rules of inference, try to figure out and convince yourself that the rules are logically sound, by looking at the associated truth tables.

- The rules we will use for propositional logic are ***Modus Ponens (MP)***, ***Modus Tollens (MT)***, ***And-Introduction*** ($\wedge_i$) and ***And-Elimination*** ($\wedge_e$).

I. Modus Ponens (MP)

- Modus Ponens means "affirming method". In our discussion of logic, anything that is written down in a proof is a statement that is true.
- Modus Ponens says that if you know that **A** implies **B** is true, and you know **A** to be true, you can automatically say that **B** is true.
    - $\mathbf{A \rightarrow B}$
    - **A**
    - $\therefore$ **B**

II. Modus Tollens (MT)

- Modus Tollens says that from "**A** implies **B**" and "not **B**" you can conclude

- ". In other words, if A implies **B** is true and **B** is known to be not true, then **A** could not have been true. Had **A** been true, **B** would automatically have been true due to the implication.
    - **A → B**
    - **¬B**
    - **∴ ¬A**

III. <u>And-Introduction</u> ($\wedge_i$)

- And-introduction says that from "**A**" and from "**B**" you can conclude "**A** and **B**". That seems pretty obvious, but is a useful tool to know upfront.
    - **A**
    - **B**
    - **∴ A ∧ B**

IV. <u>And-Elimination</u> ($\wedge_e$)

- And-elimination says that from "**A** and **B**" you can conclude "**A**". It also says that from "**A** and **B**" you can conclude "B".
    - **A ∧ B**
    - **∴ A**

Example: Proofing Using Inference Rules

Proof E from the following given three premises, by applying the above inference rules.

Premise 1: $(A \vee B) \rightarrow (C \quad \neg D)$

Premise 2: $\neg E \rightarrow D$

Premise 3: $A \quad B$

Proof Steps

$A \vee B) \rightarrow (C \wedge \neg D)$	Given Premise
$\neg E \rightarrow D$	Given Premise
$A \vee B$	Given Premise
$(C \wedge \neg D)$	Apply MP to 1 and 3
$\neg D$	Apply $\wedge_e$ to 4
$\neg (\neg E)$	Apply MP to 2 and 5
E	Proved

FOL: Basic elements

– Two standard quantifiers:

• Universal - ∀ - (for all, for every) **and** Existential - ∃ - (there exists, some)

FOL represents objects and relations between objects, variables, and quantifiers in addition to propositions

• **Universal Quantifiers**: makes statements about every object

∀<***variables***> <***sentence***>

–Everyone at AAU is smart:

$\forall_x$ At(x,AAU) ⇒ Smart(x)

–All cats are mammals:

$\forall_x$ cat(x) ⇒ mammal(x)

• $\forall_x$ sentence *P* is true iff *P* is true with *x* being each possible object in the given universe

–The above statement is equivalent to the **conjunction**

At(Jonas, AAU) ⇒ Smart(Jonas) ∧

At(Rawad, AAU) ⇒ Smart(Rawad) ∧

...

• A common mistake to avoid

–Typically, ⇒ is the main connective with ∀

–**Common mistake**: the use of ∧ as the main connective with ∀:

$\forall_x$ At(x,AAU) ∧ Smart(x) is true if "Everyone is at AAU & everyone is smart"

Existential quantification

- Makes statements about some objects in the universe

  $\exists$ <***variables***> <***sentence***>

  – Someone at AAU is smart:

  $\exists_x$ At(x,AAU) $\wedge$ Smart(x)

  – Spot has a sister who is a cat:

  $\exists_x$ sister(spot,x) $\wedge$ cat(x)

- $\exists_x$ sentence *P* is true iff *P* is true with *x* being some possible objects

  – The above statement is equivalent to the **disjunction**

  At(Jonas, AAU) $\wedge$ Smart(Jonas) $\vee$

  At(Alemu, AAU) $\wedge$ Smart(Alemu) $\vee$

  …..

- Common mistake to avoid

  – Typically, $\wedge$ is the main connective with $\exists$

  – Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

  $\exists_x$ At(x,AAU) $\Rightarrow$ Smart(x) is true if there is anyone who is not at AAU

Nested quantifiers

- $\forall_{x,y}$ parent(x,y) $\Rightarrow$ child(y,x)
  - for all x and y, if x is the parent of y then y is the child of x.
- $\exists_x \forall_y$ Loves(x,y)
  - There is a person who loves everyone in the given world
- $\forall_y \exists_x$ Loves(x,y)
  - Everyone in the given universe is loved by at least one person

Properties of quantifiers

- $\forall_x \forall_y$ is the same as $\forall_y \forall_x$
- $\exists_x \exists_y$ is the same as $\exists_y \exists_x$
- $\exists_x \forall_y$ is not the same as $\forall_y \exists_x$

- **Quantifier duality**: each can be expressed using the other, using negation ($\neg$)

  $\forall_x$ Likes(x,icecream) $\quad\quad$ $\neg\exists_x \neg$Likes(x,icecream)

  - Everyone likes ice cream means that there is nobody who dislikes ice cream

  $\exists_x$ Likes(x,cake)

  - There is someone who likes cake means that there is no one who dislikes cake $\quad\quad$ $\neg\forall_x \neg$Likes(x,cake)

Example

- Can have several quantifiers, e.g.,

  $\forall_x \exists_y$ loves(x, y)

  $\forall_x$ handsome(x) $\Rightarrow \exists_y$ loves(y, x)

Represent the following in FOL:

- Everything in the garden is lovely → $\forall$x in(x, garden) $\Rightarrow$ lovely(x)
- Everyone likes ice cream → $\forall$x likes(x, icecream)
- Peter has some friends → $\exists$y friends(y, Peter)
- John plays the piano or the violin

  plays(john, piano) v plays(john, violin)
- Some people like snakes -- $\exists$x(person(x) Λ likes(x, snakes))
- Winston did not write Hamlet – $\neg$write(winston, hamlet)
- Nobody wrote Hamlet – $\neg\exists$x write(x, hamlet)

Unification/union

- Unification is an algorithm for determining the substitutions needed to make two expressions match

  – Unification is a "pattern matching" procedure that takes two atomic sentences as input, and returns the **most general unifier,** i.e., a shortest length substitution list that makes the two literals match.

  – **E.g**: To make, say p(X, X) and p(Y, Z) match, subst(X/ Y) or subst(X/ Z)

- Note: It is possible to substitute two variables with the same value, but not the same variables with different values.

- **Example**

Sentence 1	Sentence 2	Unifier
group(x, cat(x), dog(Bill))	group(Bill, cat(Bill), y)	{x/Bill, y/dog(Bill)}
group(x, cat(x), dog(Bill))	group(Bill, cat(y), z)	{x/Bill, y/Bill, z/dog(Bill)}
group(x, cat(x), dog(Jane))	group(Bill, cat(y), dog(y))	Failure

Cont…

- A variable can never be replaced by a term containing that variable. For example, x/f(x) is illegal.
- Unification and inference rules allows us to make inferences on a set of logical assertions. To do this, the logical database must be expressed in an appropriate form
- A substitution α unfies atomic sentences p and q if pα = qα

p	q	α
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y, Abe)	{y/John, x/Abe}
Knows(John,x)	Knows(y,Mother(y))	{y/John, x/Mother(John)}
Knows(John,x)	Knows(x,Abe)	Fail

Unify rule: Premises with known facts apply unifier to conclusion

Example. if we know q and Knows(John,x) → Likes(John,x)
then we conclude Likes(John,Jane)
Likes(John,Abe)
Likes(John,Mother(John))

Inference Mechanisms

- Inference is a means of interpretation of knowledge in the KB to reason and give advise to users query.
- There are two inference strategies to control and organize the steps taken to solve problems:
  - **Forward chaining**: also called data-driven chaining
    - It starts with facts and rules in the KB and try to draw conclusions from the data
  - **Backward chaining**: also called goal-driven chaining
    - It starts with possible solutions/goals and tries to gather information that verifies the solution

Properties of Good KB

- A KB should be:
  - Clear
  - Correct
  - Expressive
  - Concise
  - Context-insensitive and
  - Effective.
  - The relations that matter should be defined, and the irrelevant details should be suppressed