Knowledge Representation

Definition and Importance of Knowledge

- **Knowledge**: Information that is organized and processed to be useful and actionable. In artificial intelligence, knowledge enables systems to make decisions, solve problems, and perform tasks intelligently.
- **Importance**: Proper knowledge representation is crucial for the efficiency and effectiveness of AI systems. It allows machines to mimic human understanding, reason about the world, and interact meaningfully with their environment.

Knowledge Representation

Introduction

- Human beings are good at understanding, reasoning and interpreting knowledge.
- And using this knowledge, they are able to perform various actions in the real world. But how do machines perform the same?

What is Knowledge Representation?

- Knowledge Representation in AI describes the representation of knowledge.
- Basically, it is a study of how the beliefs, intentions, and judgments of an intelligent agent can be expressed suitably for automated reasoning.
- One of the primary purposes of Knowledge Representation includes modeling intelligent behavior for an agent.

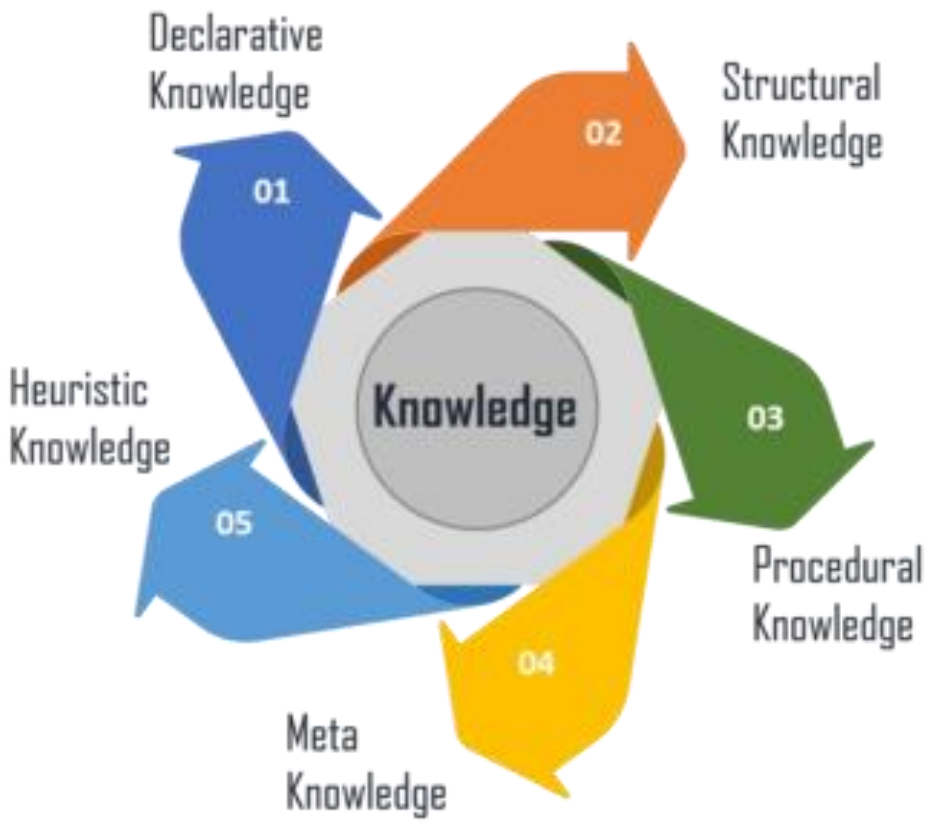
What is Knowledge Representation?

- Knowledge Representation and Reasoning (KR, KRR) represents information from the real world for a computer to understand and then utilize this knowledge to solve complex real-life problems like communicating with human beings in natural language.
- Knowledge representation in AI is not just about storing data in a database, it allows a machine to learn from that knowledge and behave intelligently like a human being.

What is Knowledge Representation?

- The different kinds of knowledge that need to be represented in AI include:
  - Objects
  - Events
  - Performance
  - Facts
  - Meta-Knowledge
  - Knowledge-base

Types of Knowledge



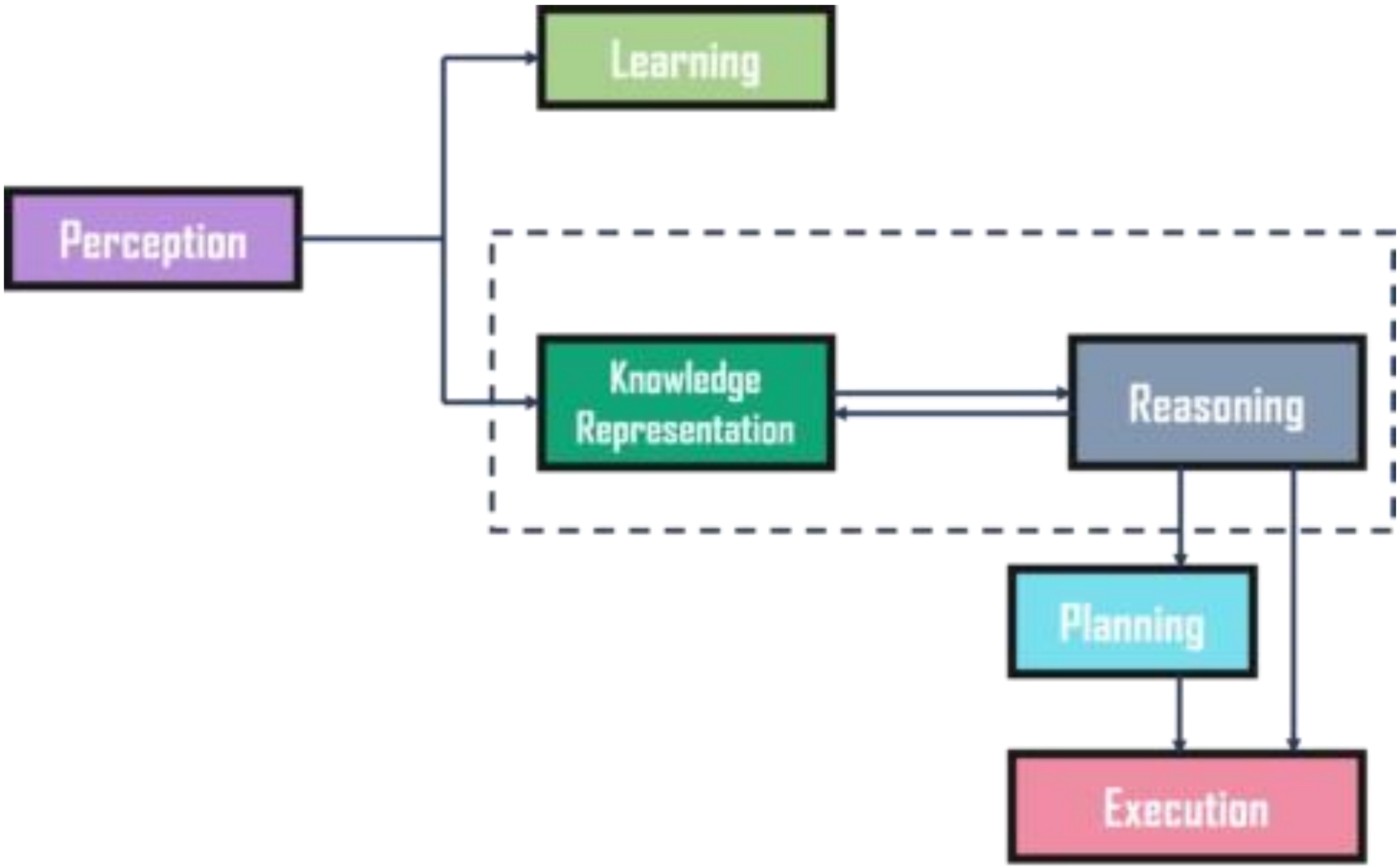
Types of Knowledge

- Declarative Knowledge – It includes concepts, facts, and objects and expressed in a declarative sentence.
- Structural Knowledge – It is a basic problem-solving knowledge that describes the relationship between concepts and objects.
- Procedural Knowledge – This is responsible for knowing how to do something and includes rules, strategies, procedures, etc.
- Meta Knowledge – Meta Knowledge defines knowledge about other types of Knowledge.
- Heuristic Knowledge – This represents some expert knowledge in the field or subject.

Cycle of Knowledge Representation in AI

- Artificial Intelligent Systems usually consist of various components to display their intelligent behavior. Some of these components include:
  - Perception
  - Learning
  - Knowledge Representation & Reasoning
  - Planning
  - Execution

Example:


Learning
Perception
Knowledge Representation
Reasoning
Planning
Execution

- In the real world, knowledge plays a vital role in intelligence as well as creating artificial intelligence.
- It demonstrates the intelligent behavior in AI agents or systems.
- It is possible for an agent or system to act accurately on some input only when it has the knowledge or experience about the input.

Knowledge Representation in AI

Logical Representation

- Logical representation is a language with some definite rules which deal with propositions and has no ambiguity in representation.
- It represents a conclusion based on various conditions and lays down some important communication rules.
- Also, it consists of precisely defined syntax and semantics which supports the sound inference.
- Each sentence can be translated into logics using syntax and semantics.

Logical Representation

Syntax	Semantics
• It decides how we can construct legal sentences in logic. • It determines which symbol we can use in knowledge representation. • Also, how to write those symbols.	• Semantics are the rules by which we can interpret the sentence in the logic. • It assigns a meaning to each sentence.

Logical Representation

- Advantages:
  - Logical representation helps to perform logical reasoning.
  - This representation is the basis for the programming languages.
- Disadvantages:
  - Logical representations have some restrictions and are challenging to work with.
  - This technique may not be very natural, and inference may not be very efficient.

Propositional logic

- **Logical constants**: true, false
- **Propositional symbols**: P, Q, S, ... (**atomic sentences**)
- Wrapping **parentheses**: (…)
- Sentences are combined by **connectives**:

  $\wedge$ ...and [conjunction]

  $\vee$ ...or [disjunction]

  $\Rightarrow$...implies [implication / conditional]

  $\Leftrightarrow$..is equivalent [biconditional]

  $\neg$ ...not [negation]
- **Literal**: atomic sentence or negated atomic sentence

Examples of PL sentences

- P means "It is hot."
- Q means "It is humid."
- R means "It is raining."
- $(P \wedge Q) \rightarrow R$
  "If it is hot and humid, then it is raining"
- $Q \rightarrow P$
  "If it is humid, then it is hot"
- A better way:
  Hot = "It is hot"
  Humid = "It is humid"
  Raining = "It is raining"

Propositional logic (PL)

- A simple language useful for showing key ideas and definitions
- User defines a set of propositional symbols, like P and Q.
- User defines the **semantics** of each propositional symbol:
  - P means "It is hot"
  - Q means "It is humid"
  - R means "It is raining"
- A sentence (well formed formula) is defined as follows:
  - A symbol is a sentence
  - If S is a sentence, then $\neg$S is a sentence
  - If S is a sentence, then (S) is a sentence
  - If S and T are sentences, then (S $\vee$ T), (S $\wedge$ T), (S $\rightarrow$ T), and (S $\leftrightarrow$ T) are sentences
  - A sentence results from a finite number of applications of the above rules

Truth tables

And

p	q	$p \cdot q$
T	T	T
T	F	F
F	T	F
F	F	F

Or

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

If . . . then

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Not

p	$\sim p$
T	F
F	T

Truth tables II

The five logical connectives:

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
False	*False*	*True*	*False*	*False*	*True*	*True*
False	*True*	*True*	*False*	*True*	*True*	*False*
True	*False*	*False*	*False*	*True*	*False*	*False*
True	*True*	*False*	*True*	*True*	*True*	*True*

A complex sentence:

P	H	$P \vee H$	$(P \vee H) \wedge \neg H$	$((P \vee H) \wedge \neg H) \Rightarrow P$
False	*False*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*
True	*False*	*True*	*True*	*True*
True	*True*	*True*	*False*	*True*

First Order Predicate Logic (FOPL)
or
First Order Logic (FOL)

Introduction

- **First Order Predicate Logic (FOPL)**, also known as **First-Order Logic (FOL)**, is a formal system used in logic, mathematics, and computer science to express statements about objects, their properties, and their relationships.
- It extends propositional logic by introducing quantifiers, variables, and predicates, which makes it more expressive.

Components of First-Order Predicate Logic

1. **Constants**: Represent specific objects or entities in the domain.
   - Example: `John`, `Paris`, `3`
2. **Variables**: Represent general objects or entities.
   - Example: `x`, `y`, `z`
3. **Predicates**: Represent properties of objects or relationships between objects.
   - Example: `Loves(x, y)` (x loves y), `IsTall(x)` (x is tall)
4. **Functions**: Map objects to other objects.
   - Example: `Father(x)` (father of x)

Components of First-Order Predicate Logic

5. **Quantifiers**:
    - **Universal Quantifier (∀)**: Indicates that a statement is true for all objects.
        - Example: ∀x (IsHuman(x) → Mortal(x)) (All humans are mortal)
    - **Existential Quantifier (∃)**: Indicates that there exists at least one object for which the statement is true.
        - Example: ∃x (Loves(x, John)) (Someone loves John)
6. **Logical Connectives**:
    - **AND (∧)**: Conjunction
    - **OR (∨)**: Disjunction
    - **NOT (¬)**: Negation
    - **IMPLIES (→)**: Implication
    - **IFF (↔)**: Biconditional

Components of First-Order Predicate Logic

7. **Equality**: Represents that two objects are the same.
   - Example: `x = y`
8. **Domain of Discourse:** The set of all objects that the variables can represent.

Syntax of FOPL

A **well-formed formula (WFF)** is constructed using the components of FOPL. Examples of WFFs include:

- Atomic Formula: `P(x, y)`
- Complex Formula: `∀x ∃y (P(x) ∧ Q(x, y)) → R(y)`

Semantics of FOPL

- The **meaning** of FOPL statements depends on the interpretation of:
  1. **Domain**: The set of objects under consideration.
  2. **Assignment**: Maps variables to objects in the domain.
  3. **Interpretation**: Assigns meanings to constants, predicates, and functions.

Example

Problem Statement:

"All humans are mortal. Socrates is a human. Therefore, Socrates is mortal."

Representation in FOPL:

1. Domain: All living beings.
2. Predicates:
   - `Human(x)`: x is a human.
   - `Mortal(x)`: x is mortal.
3. Constants:
   - `Socrates`: A specific individual.

Example

FOPL Statements:

1. $\forall x\ (Human(x) \rightarrow Mortal(x))$ (All humans are mortal)
2. Human(Socrates) (Socrates is a human)
3. $\therefore$ Mortal(Socrates) (Socrates is mortal)

Applications

- **Artificial Intelligence (AI)**: Knowledge representation and reasoning.
- **Database Queries**: Query languages like SQL.
- **Theorem Proving**: Automated reasoning systems.
- **Linguistics**: Syntax and semantics of natural languages.
- **Mathematics**: Formal proofs and foundations.

Semantic Network Representation

- Semantic networks work as an alternative of predicate logic for knowledge representation. In Semantic networks, you can represent your knowledge in the form of graphical networks.
- This network consists of nodes representing objects and arcs which describe the relationship between those objects. Also, it categorizes the object in different forms and links those objects.
- This representation consist of two types of relations:
  - IS-A relation (Inheritance)
  - Kind-of-relation

Semantic Network (Propositional Net)

- KR techinique used for propositional information.
- Labeled directed graph.
- Consists of:
  1. Nodes – circle/ellipse/rectangle

     - represent physical object, concept or situation.
  2. Links –arrow

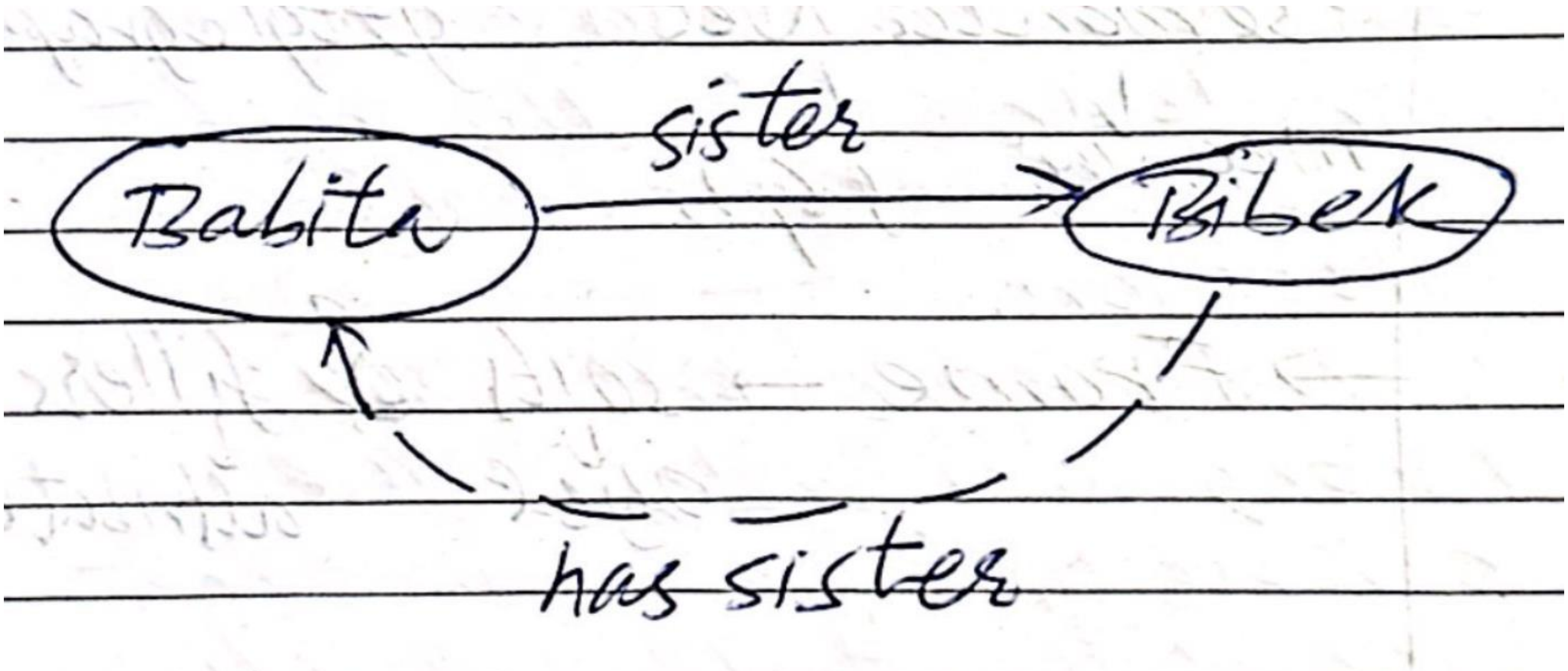
     -to express relationship b/w objects
  3. Link labels – specify particular relations.

Semantic Network (Propositional Net)

- Unless there is a specific evidence to the contrary, it is assumed that all members of a class (category) will inherit all the properties of their super class.
- So semantic n/w allow us to perfrom inheritance reasoning.
- Semantic net allows multiple inheritance. So, an object can belong to more than one category and a cateogry can be subset of more than one another category.

Semantic Network (Propositional Net)

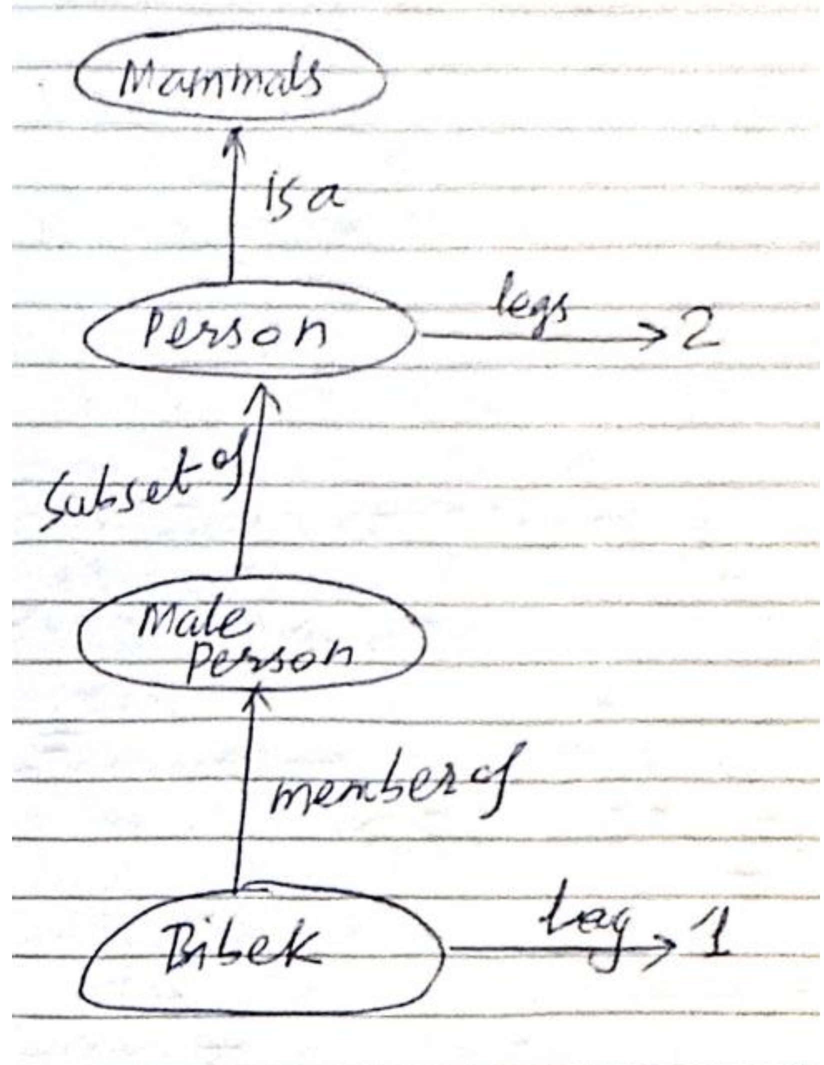
- Inverse Links



Advantages:

- Have the ability to represent default values for categories.

For example: Bibek has 1 leg while he is a person and all perosn have 2 legs. So, person have 2 legs has only default status, which can be overriden by a specific value.



Advantages:

- Convey some meaning in some transparent manner.
- Are simple and easy to understand
- Efficienet in space requirement.

Disadvantages

- Link between object represents only binary relations.

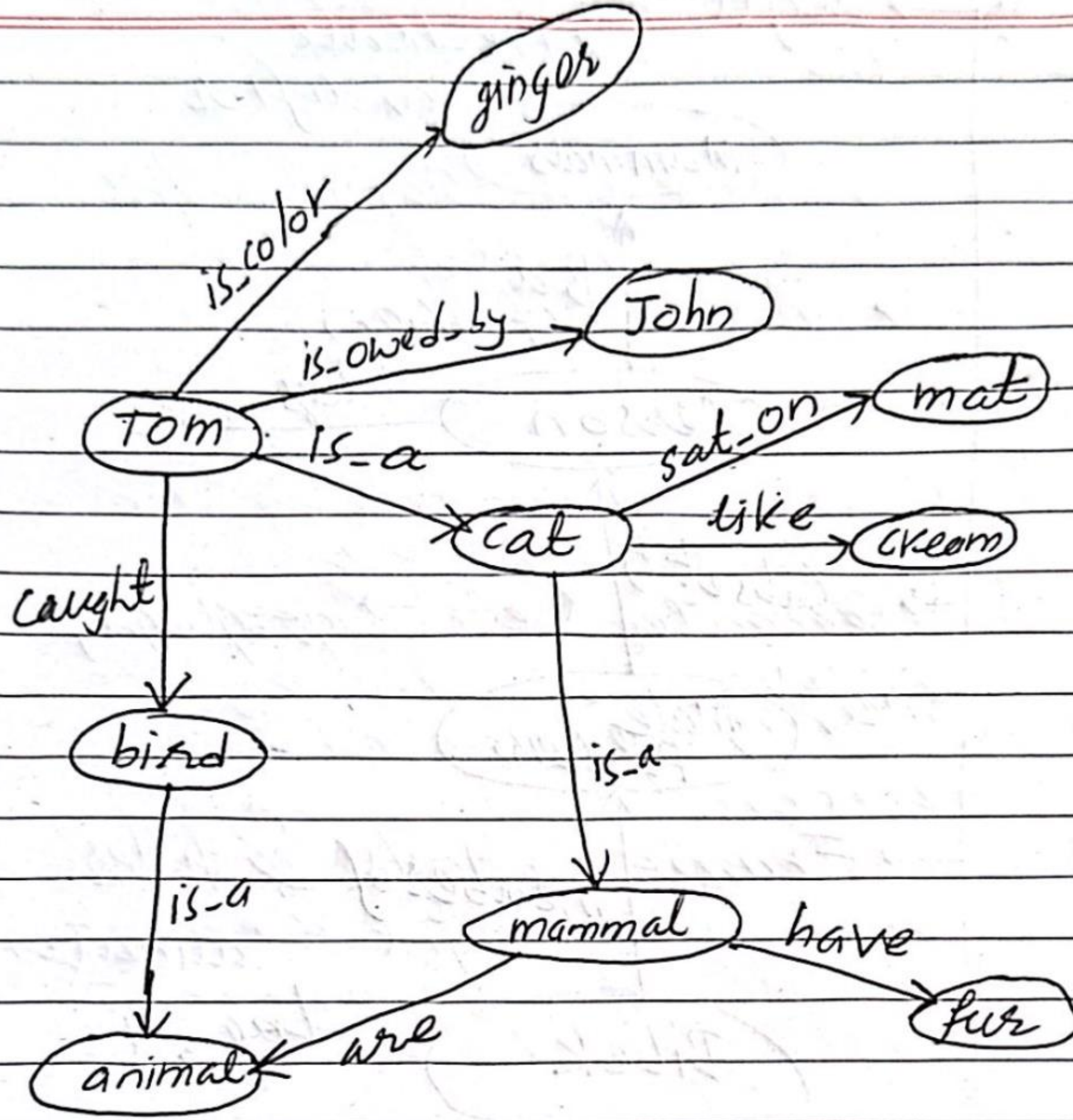
Example: Sentence

Run(Pokhara Express, Pokhara, Begnastal, Today) cannot be asserted directly.

- There is no standard definition of links names.
- Quantified statements are very hard for semantic nets.

Example

- Tom is cat. Tom caught bird. Tom is owned by John. Tom is ginger in color. Cats like cream. The cat sat on the mat. A cat is a mammal. A bird is an animal. All mammals are animals. Mammal have fur.



ginger
is_color
John
is_owed_by
mat
Tom
is_a
sat_on
like
cat
cream
caught
bird
is_a
is_a
mammal
have
fur
are
animal

Example

Statements:

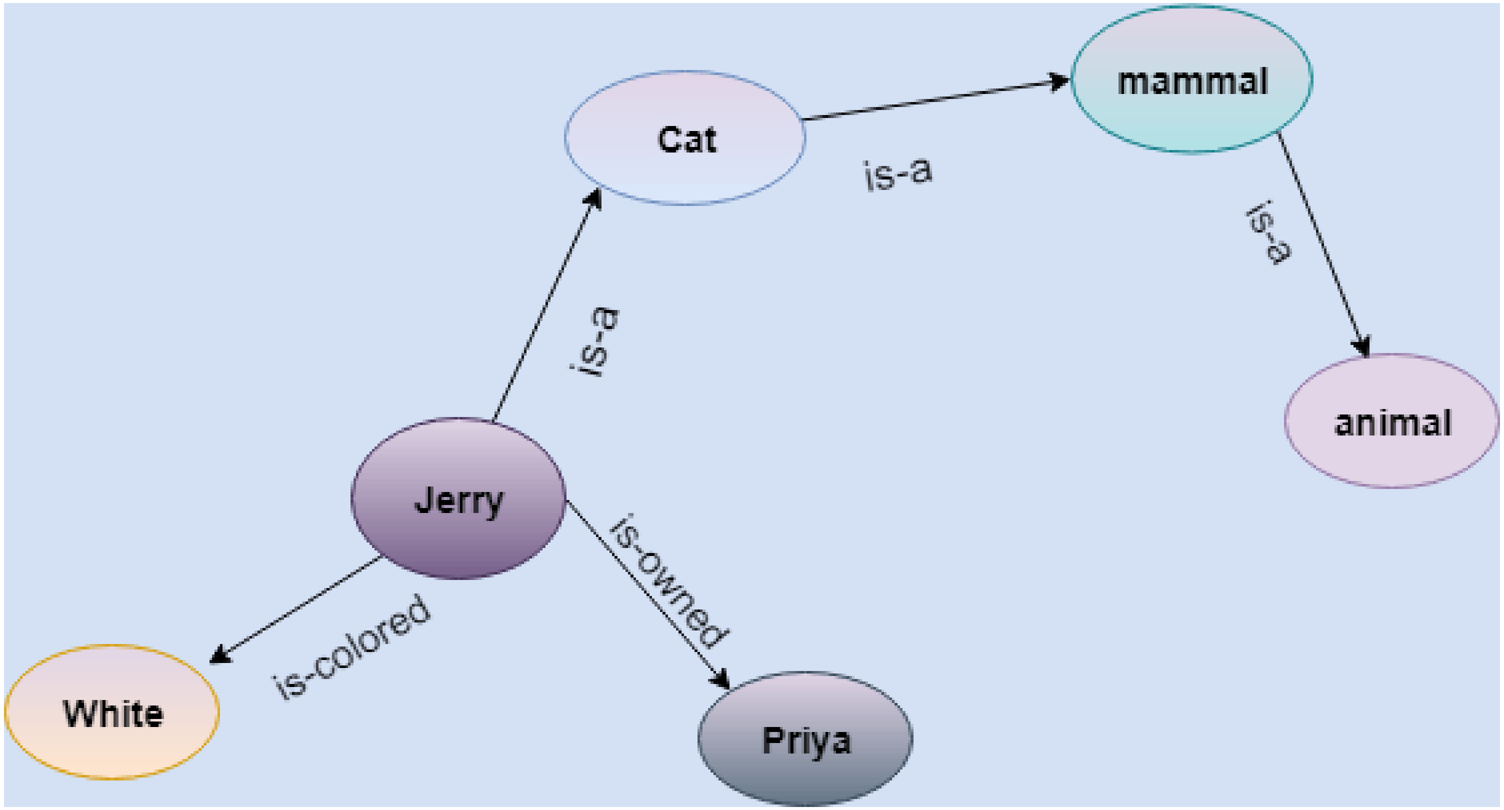
a. Jerry is a cat.

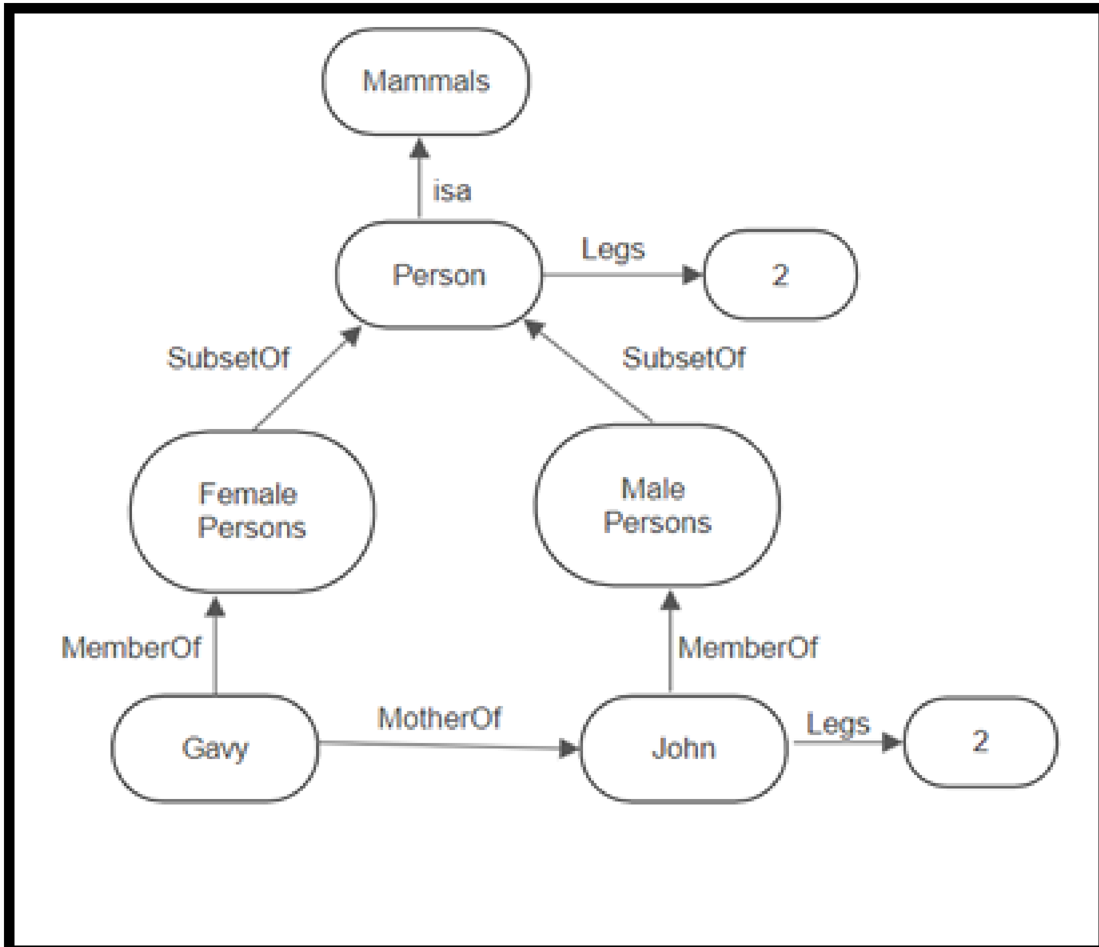
b. Jerry is a mammal

c. Jerry is owned by Priya.

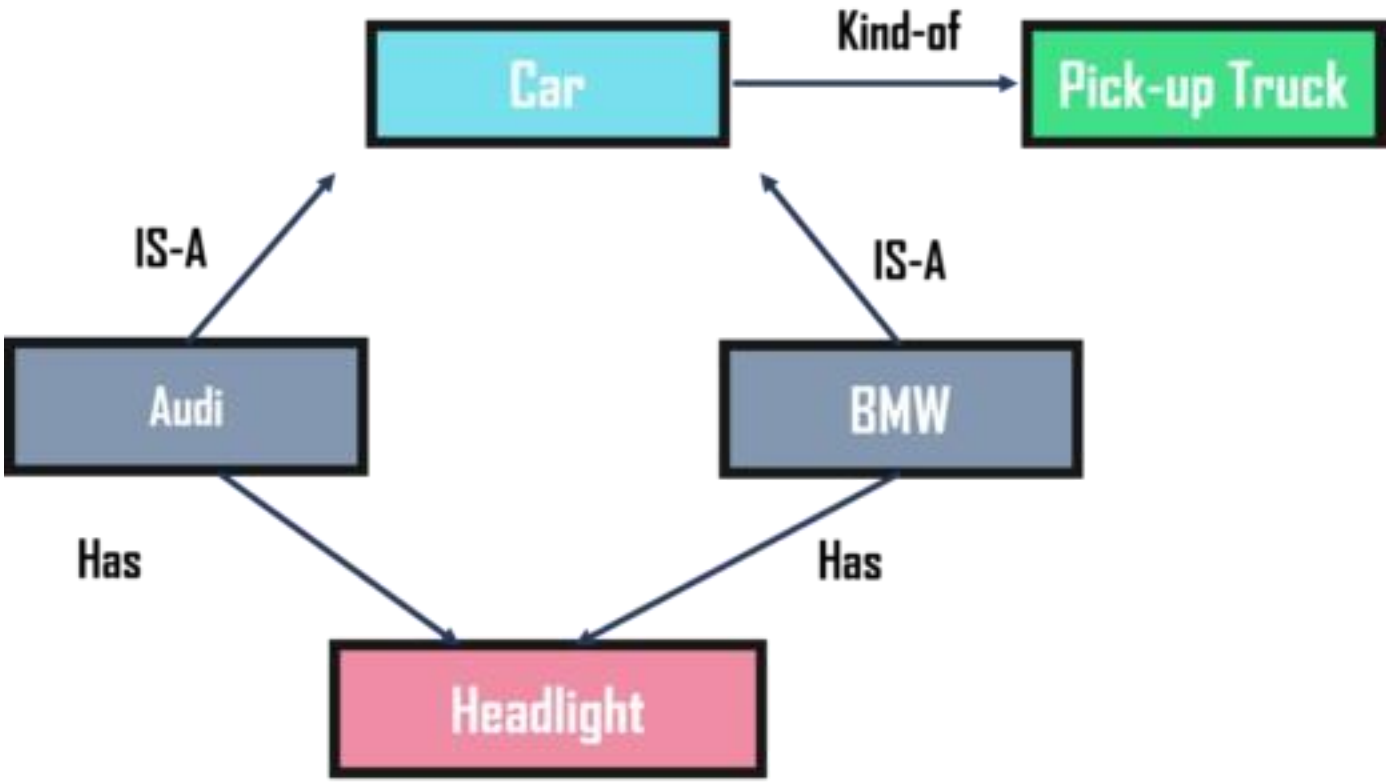
d. Jerry is brown colored.

e. All Mammals are animal.


Cat
mammal
is-a
is-a
is-a
animal
Jerry
is-owned
is-colored
White
Priya


Mammals
isa
Person
Legs
2
SubsetOf
SubsetOf
Female Persons
Male Persons
MemberOf
MemberOf
Gavy
MotherOf
John
Legs
2

Semantic Network Representation



Semantic Network Representation

- Advantages:
  - Semantic networks are a natural representation of knowledge.
  - Also, it conveys meaning in a transparent manner.
  - These networks are simple and easy to understand.
- Disadvantages:
  - Semantic networks take more computational time at runtime.
  - Also, these are inadequate as they do not have any equivalent quantifiers.
  - These networks are not intelligent and depend on the creator of the system.

Frame Representation

- A frame is a record like structure that consists of a collection of attributes and values to describe an entity in the world.
- These are the AI data structure that divides knowledge into substructures by representing stereotypes situations.
- Basically, it consists of a collection of slots and slot values of any type and size.
- Slots have names and values which are called facets.

Frames:

- Frames are record like structure that have slots & slot values for an entity.

- A slot in a frame specify a characteristic of the entity which the frame represents.
- A slot in a frame contains information as attribute-value pairs, default values etc.

Frames:

Example 1:

Employee Details

```
Raj Sharma(
        (Profession        (Value    Manager))
        (EmpID             (Value    100213))
        (Address           (Value    Delhi))
)
```

Frames:

Example 2:

"Tweety is a yellow bird having wings to fly"

```
Tweety(
        (Species          (Value   bird))
        (color            (Value   yellow))
        (Activity         (Value   Fly))
)
```

Frame Representation

- Advantages:
  - It makes the programming easier by grouping the related data.
  - Frame representation is easy to understand and visualize.
  - It is very easy to add slots for new attributes and relations.
  - Also, it is easy to include default data and search for missing values.
- Disadvantages:
  - In frame system inference, the mechanism cannot be easily processed.
  - The inference mechanism cannot be smoothly proceeded by frame representation.
  - It has a very generalized approach.

Production Rules

- In production rules, agent checks for the condition and if the condition exists then production rule fires and corresponding action is carried out.
- The condition part of the rule determines which rule may be applied to a problem. Whereas, the action part carries out the associated problem-solving steps. This complete process is called a recognize-act cycle.
- The production rules system consists of three main parts:
  - The set of production rules
  - Working Memory
  - The recognize-act-cycle

Production Rules

- Advantages:
  - The production rules are expressed in natural language.
  - The production rules are highly modular and can be easily removed or modified.
- Disadvantages:
  - It does not exhibit any learning capabilities and does not store the result of the problem for future uses.
  - During the execution of the program, many rules may be active. Thus, rule-based production systems are inefficient.

Steps to Convert a First-Order Predicate Logic (FOPL) Sentence to Conjunctive Normal Form (CNF)

- Converting FOPL to CNF is essential for automated reasoning systems such as resolution in logic programming or theorem proving.
- A formula is in **CNF** if it is a **conjunction (AND)** of one or more **clauses**, where each clause is a **disjunction (OR)** of **literals**.
- Example in CNF:

$$(A \lor \neg B) \land (C \lor D \lor \neg E)$$

Step-by-Step Process to Convert FOPL to CNF

Step	Description
1. **Eliminate implications and biconditionals**	Replace P → Q with ¬P ∨ Q and P ↔ Q with (¬P ∨ Q) ∧ (¬Q ∨ P)
2. **Move NOT (¬) inward** using De Morgan's Laws	Push negation down to atomic level
3. **Standardize variables**	Rename variables so that each quantifier has a unique variable
4. **Move quantifiers to the front (Prenex form)**	Move all ∀ and ∃ to the front of the expression
5. **Skolemization**	Eliminate existential quantifiers by introducing Skolem constants/functions
6. **Drop universal quantifiers**	All variables are assumed universally quantified in CNF
7. **Distribute ∨ over ∧**	Convert to conjunctive normal form using distributive laws

Steps to convert FOL to CNF

- Eliminate implication
- Standardize variable
- move negation inwards
- skolemization
- Orop universal Quatifier

Eliminate implication

$$\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta$$

$$\alpha \leftrightarrow \beta \equiv (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$$

Standardize variable

$\exists(x)\ smile(x)$	$\exists(x)\ smile(x)$
$\exists(x)\ Graduating(x)$	$\exists(y)\ Graduating(y)$
$\forall(x)\ happy(x)$	$\forall(z)\ happy(z)$

- move negation inwards

$$\neg(\forall(x)\, P(x)) \equiv \exists(x)\, \neg P(x)$$

$$\neg(\exists(x)\, P(x)) \equiv \forall(x)\, \neg P(x)$$

$$\neg(\alpha \vee \beta) \equiv \neg\alpha \wedge \neg\beta$$

$$\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$$

$$\neg(\neg\alpha) \equiv \alpha$$

- Skolemization (Remove Existential Quantifier) and Replace it by Skolem contant

$\exists(x)$ smile(x)

$\exists(y)$ Graduating(y)

After Skolemization

Smile(A)

Graduating(B)

Drop universal Quantifier

$\forall x \ (\text{smile}(x))$

$\forall y \ \text{Graduating}(y)$

After Droping

$\text{Smile}(x)$

$\text{Graduating}(y)$

- All people who are graduating are happy
- All happy people smile
- Some one is Graduating

i) Convert to FOL

ii) Convert FOL to CNF

iii) Prove that "Is someone Smiling?" using resolution

iv) Draw Resolution tree

Convert to FOL

- All people who are graduating are happy

- All happy people smile

- Someone is Graduating

we need to prove that is Someone smiling?

Convert to FOL

- All people who are graduating are happy

  $\forall x\,(\text{Graduating}(x) \rightarrow \text{happy}(x))$

- All happy people smile

  $\forall x\,(\text{happy}(x) \Rightarrow \text{smile}(x))$

- Someone is Graduating

  $\exists x\ \text{Graduating}(x)$

we need to prove that is Someone smiling?

$\neg\,\exists x\ \text{smile}(x)$

Eliminate Implication $\alpha \to \beta \equiv \neg\alpha \vee \beta$

$\forall x\, [\neg Graduating(x) \vee Happy(x)]$

$\forall x\, [\neg Happy(x) \vee Smile(x)]$

$\exists x\; Graduating(x)$

$\neg \exists x\; Smile(x)$

$\forall x\; (Graduating(x) \to Happy(x)]$

$\forall x\; (Happy(x) \to Smile(x))$

$\exists x\; Graduating(x)$

$\neg \exists x\; Smile(x)$

Convert FOL to CNF

Eliminate Implication $\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta$

$\forall x\, [\neg \mathrm{Graduating}(x) \vee \mathrm{Happy}(x)]$

$\forall x\, [\neg \mathrm{Happy}(x) \vee \mathrm{Smile}(x)]$

$\exists x\ \mathrm{Graduating}(x)$

$\neg \exists x\ \mathrm{Smile}(x)$

Standardize variables Apart

$\forall x\, [\neg \mathrm{Graduating}(x) \vee \mathrm{Happy}(x)]$

$\forall y\, [\neg \mathrm{Happy}(y) \vee \mathrm{Smile}(y)]$

$\exists z\ \mathrm{Graduating}(z)$

$\neg \exists w\ \mathrm{smile}(w)$

Move Negation Inwards

$\forall x\,[\neg Graduating(x) \vee Happy(x)]$

$\forall y\,[\neg Happy(y) \vee smile(y)]$

$\exists z\; Graduating(z)$

$\forall w\; \neg smile(w)$

Skolemization:

$\forall x\,[\neg Graduating(x) \vee Happy(x)]$

$\forall y\,[\neg Happy(y) \vee smile(y)]$

$Graduating(A)$

$\forall w\; \neg smile(w)$

Drop Universal Quantifier

$\neg Graduating(x) \vee Happy(x)$

$\neg Happy(y) \vee Smile(y)$

$Graduating(A)$

$\neg Smile(w)$

now the sentences are in CNF

Resolution tree:

- IF Fact 'F' is to be Proved then it start with ¬F
It contradicts all the other rule in KB
The Process stop when it returns Null clause

Drop Universal Quantifier

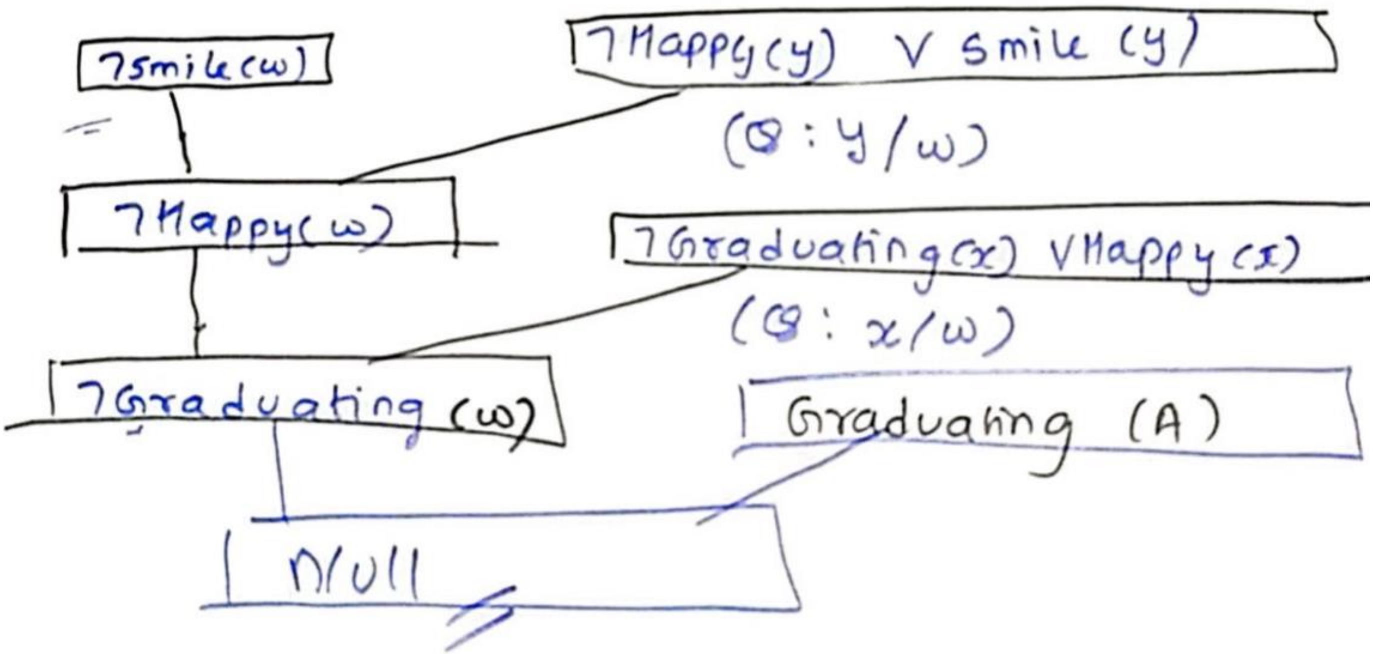
$\neg Graduating(x) \vee Happy(x)$

$\neg Happy(y) \vee Smile(y)$

$Graduating(A)$

$\neg Smile(w)$

Resolution tree



Hence someone is smiling

Convert the following FOPL sentence to CNF:

$$\forall x(\exists y\ (Loves(x, y)) \rightarrow Happy(x))$$

Conversion Steps

Step 1: Eliminate implication

$$\forall x(\neg \exists y\ Loves(x,y) \vee Happy(x))$$

Step 2: Move NOT inward (De Morgan)

$$\forall x(\forall y\ \neg Loves(x,y) \vee Happy(x))$$

Step 3: Standardize variable (Already okay)

Step 4: Already in prenex form

$$\forall x \forall y(\neg Loves(x, y) \vee Happy(x))$$

Step 5: Skolemization (no ∃ left, skip)

Step 6: Drop universal quantifiers (assumed in CNF)

$$\neg Loves(x, y) \vee Happy(x)$$

Step 7: Already in CNF
Final CNF:

$$\neg Loves(x, y) \vee Happy(x)$$

Convert the following FOPL sentence to CNF:

$$\forall x(\exists y\ (Loves(x, y)) \rightarrow Happy(x))$$

Conversion Steps

Example:

Forward Chaining

- It is a form of reasoning which starts with atomic sentences in the knowledge base and applies inerence rule in the forward direction to extract more data until a goal is reached.

How Forward Chaining Works:

- **Initial Facts:** Start with a set of known facts.
- **Inference Rules:** Have a set of rules that describe how new facts can be inferred from existing facts. These rules are typically in the form of "if-then" statements.
- **Match and Apply:** Check which rules can be applied to the known facts. If the condition (the "if" part) of a rule is satisfied by the known facts, the action (the "then" part) is executed.
- **Add New Facts:** The result of the rule application is added to the set of known facts.
- **Repeat:** Continue this process iteratively until no more rules can be applied or a specific goal is achieved.

Properties:

- It moves from bottom to top.
- It the process of making conclusion based on known facts of data, by starting from initial state and reach a goal state.
- Forward chaining apporach is also called as data-driven as we reach to the goal using available data.
- It is commonly used in expert system.

Characteristics of Forward Chaining:

- **Data-Driven:** It starts from known data and uses inference rules to extract more data.
- **Exploratory:** It is useful in situations where all possible consequences of the known data need to be explored.
- **Dynamic:** New facts can be continually added, which may trigger new rules and lead to new inferences.

Applications of Forward Chaining:

- **Expert Systems:** Used in systems that mimic the decision-making abilities of a human expert.
- **Production Systems:** Employed in systems where a set of rules determines actions to be taken based on conditions.
- **Diagnostic Systems:** Used in systems that diagnose problems based on observed symptoms and known facts.

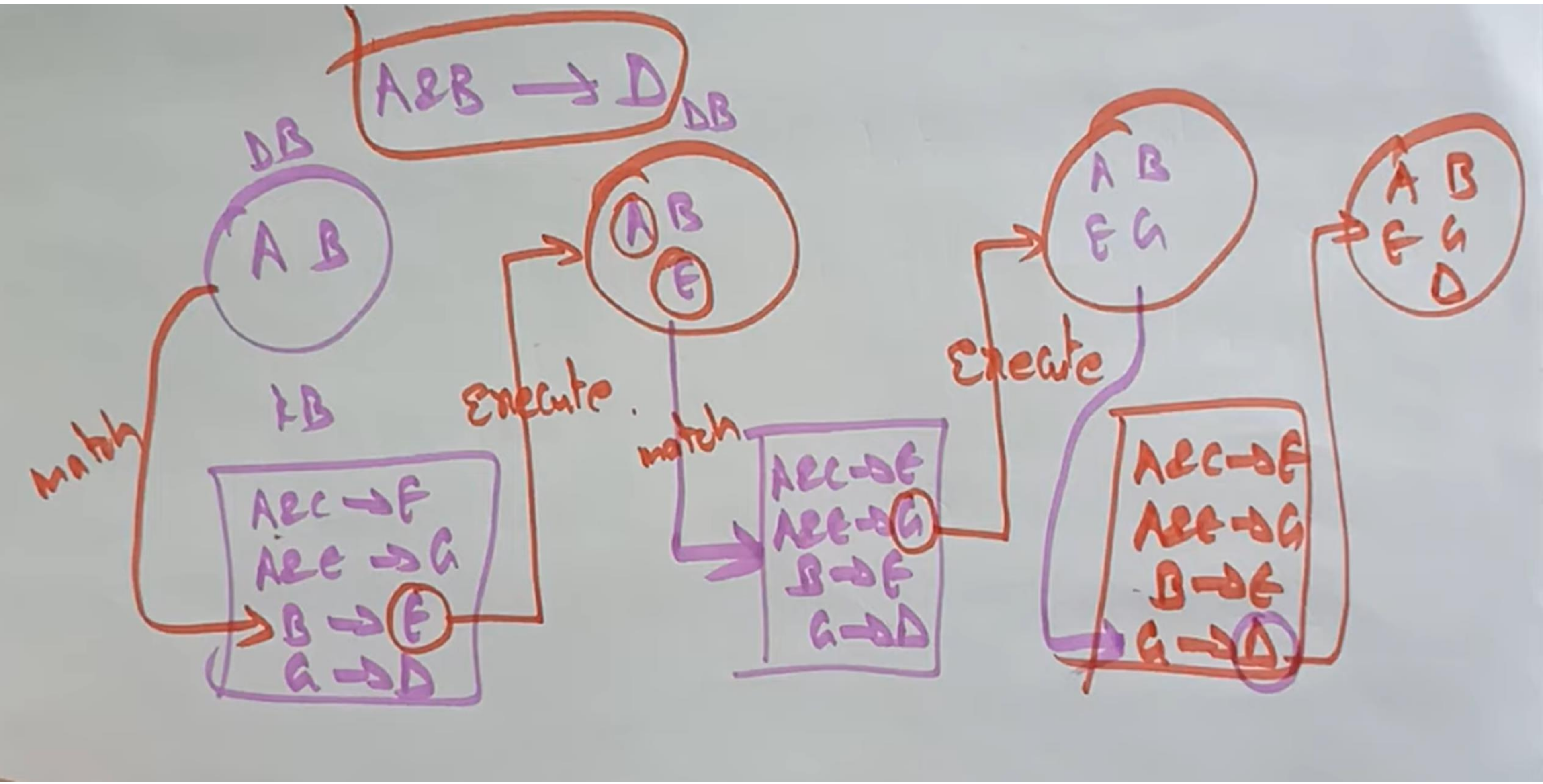
Example of Forward Chaining:

- Consider a simple expert system for diagnosing animal types:
- **Initial Facts:**
  - The animal has feathers.
  - The animal can fly.
- **Inference Rules:**
  - If the animal has feathers, then it is a bird.
  - If the animal can fly and it is a bird, then it is a flying bird.
  - If the animal is a flying bird, then it might be an eagle.

In this example.

1. Start with the facts: "The animal has feathers" and "The animal can fly."
2. Apply the first rule: "If the animal has feathers, then it is a bird."
3. Now, the set of known facts includes: "The animal is a bird."
4. Apply the second rule: "If the animal can fly and it is a bird, then it is a flying bird."
5. Now, the set of known facts includes: "The animal is a flying bird."
6. Apply the third rule: "If the animal is a flying bird, then it might be an eagle."
7. Now, the set of known facts includes: "The animal might be an eagle."

Example.


A&B → D
DB
DB
A B
KB
A&C → F
A&E → G
B → E
G → D
match
Execute
A B
E
match
A&C → E
A&E → G
B → E
G → D
Execute
A B
E G
A&C → E
A&E → G
B → E
G → D
A B
E G
D

Backward Chaining

- A backward chaining is a form of reasoning, which starts with the goal and works backward, chaining through rules to find known facts support the goal.
- Backward chaining is another method used in artificial intelligence (AI) for inference, particularly in expert systems and rule-based systems

How Backward Chaining Works:

- **Goal:** Start with a specific goal or hypothesis that you want to prove.
- **Inference Rules:** Have a set of rules that describe how goals can be achieved. These rules are typically in the form of "if-then" statements.
- **Match and Trace Back:** Determine what conditions (the "if" part of a rule) must be true for the goal (the "then" part) to be satisfied.
- **Subgoals:** If the conditions are not already known facts, treat them as new subgoals and apply the same process to them.
- **Repeat:** Continue this process iteratively, breaking down each goal into smaller subgoals, until the initial facts are reached or the goal is determined to be unachievable.

Properties:

- It is known as top down approach.
- It is based on modus ponens inference rule.
- In backward chaining, the goal is broken into sub goal or sub goal to prove the facts true.
- It is called as goal driven approach, as a list of goal decides which rules are selected and used.
- It is used in game theory, automated theorem proving tools, inference engines, proof asistants, and various AI application.
- It mostly used a depth first search strategy for proof.

Characteristics of Backward Chaining:

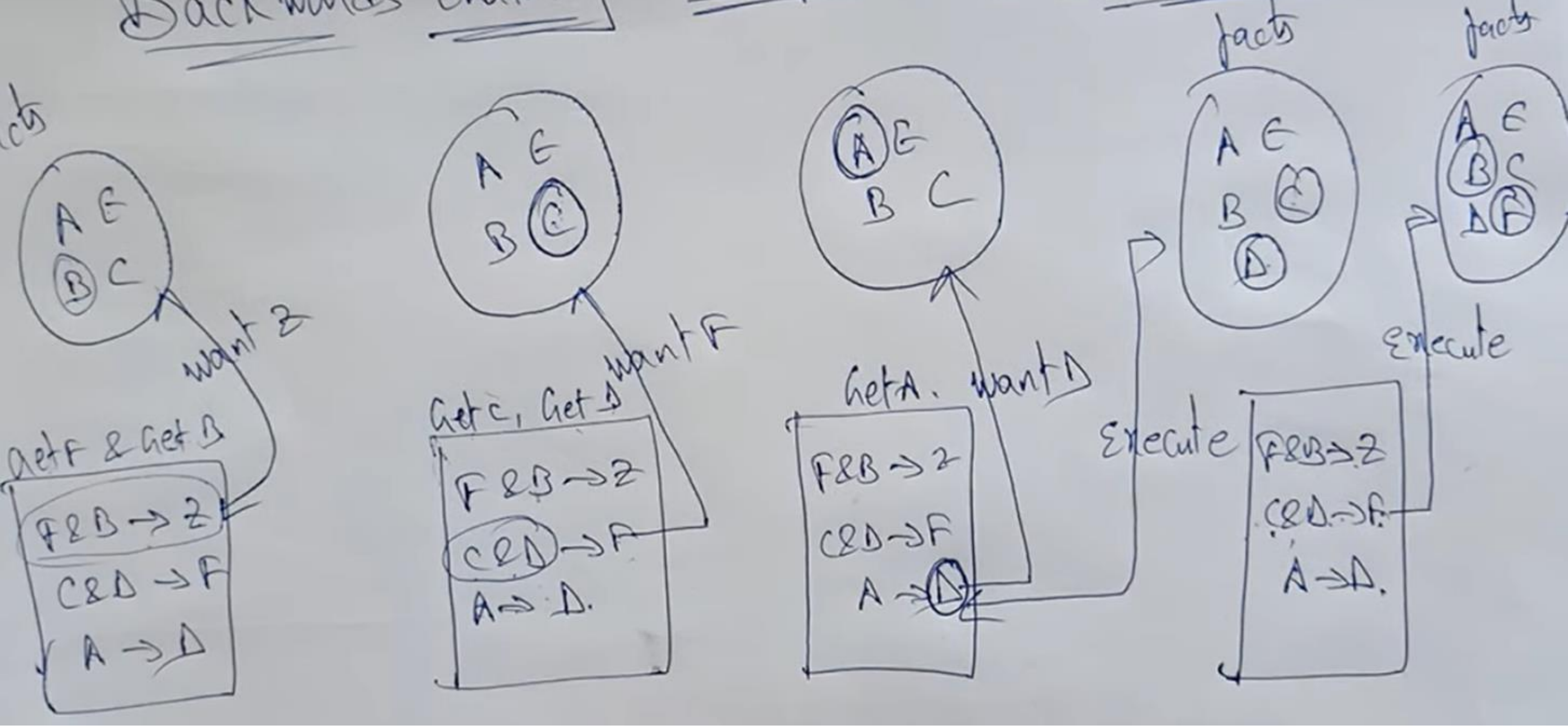
- **Goal-Driven:** It starts from a specific goal and works backward to find the necessary conditions to achieve that goal.
- **Focused:** It is useful in situations where a specific conclusion or diagnosis needs to be verified.
- **Efficient:** It only explores the rules relevant to the specific goal, potentially reducing the amount of computation compared to forward chaining.

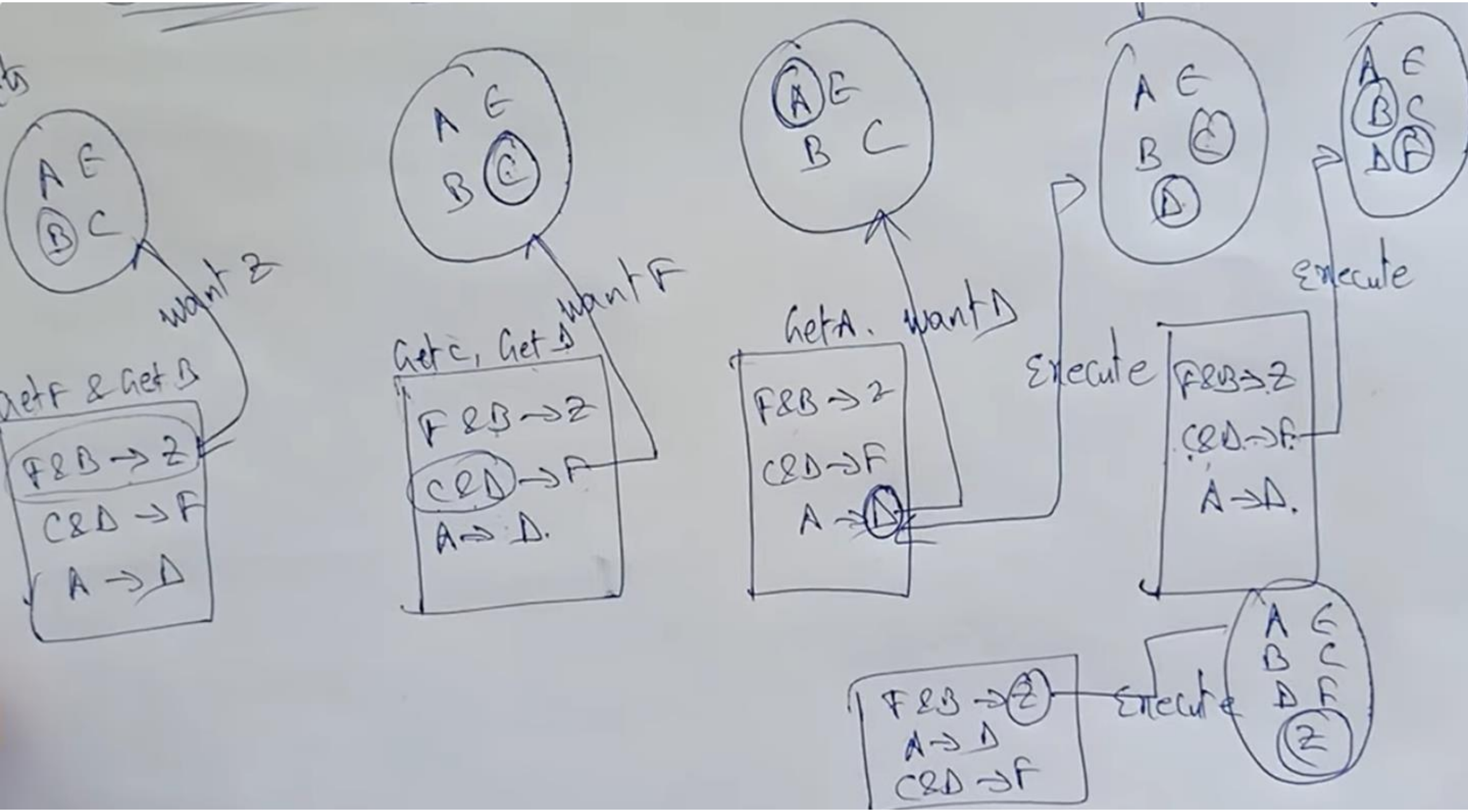
Applications of Backward Chaining:

- **Expert Systems:** Used in systems that mimic the decision-making abilities of a human expert, especially for diagnostic and troubleshooting purposes.
- **Theorem Proving:** Employed in automated reasoning systems that prove mathematical theorems by working backward from the theorem to be proved.
- **Query Systems:** Used in databases and information retrieval systems where specific queries need to be answered based on available data.

Backwards chaining Example

~~Get~~ Goal state: Z




A E
B C
want Z
Get F & Get B
F&B → Z
C&D → F
A → D
A E
B C
want F
Get C, Get D
F&B → Z
C&D → F
A → D.
A E
B C
Get A.
want D
F&B → Z
C&D → F
A → D
Execute
A E
B C
D
F&B → Z
C&D → F
A → D.
Execute
A E
B C
D F
F&B → Z
A → D
C&D → F
Execute
A E
B C
D F
Z

Representation Requirements

- A good knowledge representation system must have properties such as:
  - Representational Accuracy: It should represent all kinds of required knowledge.
  - Inferential Adequacy: It should be able to manipulate the representational structures to produce new knowledge corresponding to the existing structure.
  - Inferential Efficiency: The ability to direct the inferential knowledge mechanism into the most productive directions by storing appropriate guides.
  - Acquisitional efficiency: The ability to acquire new knowledge easily using automatic methods.

Approaches to Knowledge Representation in AI

- Simple Relational Knowledge
  - It is the simplest way of storing facts which uses the relational method. Here, all the facts about a set of the object are set out systematically in columns.
  - Also, this approach of knowledge representation is famous in database systems where the relationship between different entities is represented.
  - Thus, there is little opportunity for inference.

Approaches to Knowledge Representation in AI

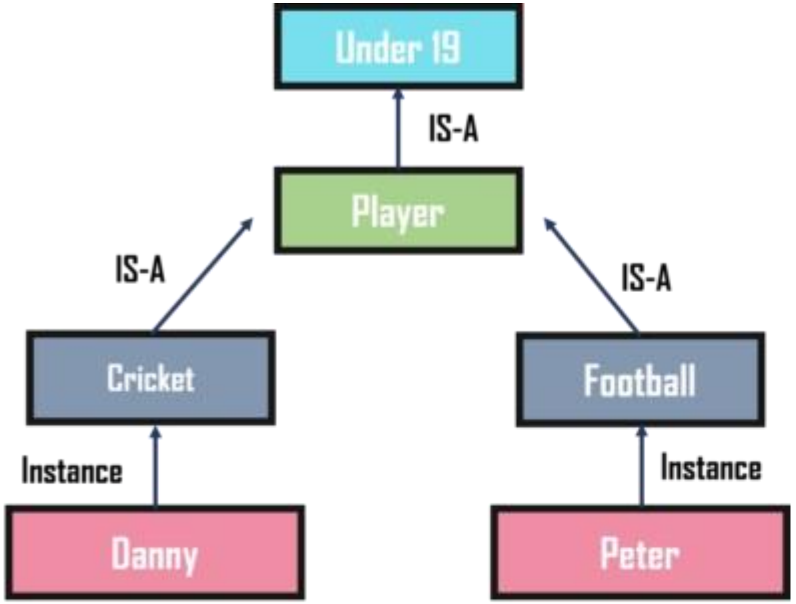
Name	Age	Emp ID
John	25	100071
Amanda	23	100056
Sam	27	100042

Approaches to Knowledge Representation in AI

- Inheritable Knowledge
  - In the inheritable knowledge approach, all data must be stored into a hierarchy of classes and should be arranged in a generalized form or a hierarchal manner.
  - Also, this approach contains inheritable knowledge which shows a relation between instance and class, and it is called instance relation.
  - In this approach, objects and values are represented in Boxed nodes.

Approaches to Knowledge Representation in AI

- Inheritable Knowledge



Approaches to Knowledge Representation in AI

- Inferential Knowledge
- The inferential knowledge approach represents knowledge in the form of formal logic. Thus, it can be used to derive more facts. Also, it guarantees correctness.
- Example:

  Statement 1: John is a cricketer.

  Statement 2: All cricketers are athletes.

  Then it can be represented as;

  – Cricketer(John)
  – ∀x = Cricketer (x) – – – - > Athelete (x)s

Issues in Knowledge Representation

- The fundamental goal of knowledge Representation is to facilitate inference (conclusions) from knowledge.
- The issues that arise while using KR techniques are many. Some of these are explained below.
- Important Attributed:
  - Any attribute of objects so basic that they occur in almost every problem domain?
  - There are two attributed “instance” and “isa”, that are general significance. These attributes are important because they support property inheritance.

Issues in Knowledge Representation

- Relationship among attributes:
  - Any important relationship that exists among object attributed?
  - The attributes we use to describe objects are themselves entities that we represent.
  - The relationship between the attributes of an object, independent of specific knowledge they encode, may hold properties like:
- Inverse – This is about consistency check, while a value is added to one attribute. The entities are related to each other in many different ways.

Issues in Knowledge Representation

- Existence in an isa hierarchy —
  - This is about generalization-specification, like, classes of objects and specialized subsets of those classes, there are attributes and specialization of attributes.
  - For example, the attribute height is a specialization of general attribute physical-size which is, in turn, a specialization of physical-attribute.
  - These generalization-specialization relationships are important for attributes because they support inheritance.

Issues in Knowledge Representation

- Technique for reasoning about values –
  - This is about reasoning values of attributes not given explicitly.
  - Several kinds of information are used in reasoning, like, height: must be in a unit of length, Age: of a person cannot be greater than the age of person's parents.
  - The values are often specified when a knowledge base is created.

Issues in Knowledge Representation

- Single valued attributes –
  - This is about a specific attribute that is guaranteed to take a unique value.
  - For example, a baseball player can at time have only a single height and be a member of only one team.
  - KR systems take different approaches to provide support for single valued attributes.

Issues in Knowledge Representation

- Choosing Granularity:
  - At what level of detail should the knowledge be represented?
  - Regardless of the KR formalism, it is necessary to know:
    - At what level should the knowledge be represented and what are the primitives?
    - Should there be a small number or should there be a large number of low-level primitives or High-level facts.
    - High-level facts may not be adequate for inference while Low-level primitives may require a lot of storage.

Issues in Knowledge Representation

- Example of Granularity:

  Suppose we are interested in following facts:

  John spotted Sue.

  This could be represented as

  Spotted (agent(John),object (Sue))
- Such a representation would make it easy to answer questions such are:

  Who spotted Sue?
- Suppose we want to know:

  Did John see Sue?
- Given only one fact, we cannot discover that answer.
- We can add other facts, such as

  Spotted(x, y) -> saw(x, y)
- We can now infer the answer to the question.

Issues in Knowledge Representation

- Set of objects:
- How should sets of objects be represented?
- There are certain properties of objects that are true as member of a set but not as individual;
- Example: Consider the assertion made in the sentences:

  "there are more sheep than people in Australia", and

  "English speakers can be found all over the world."
- To describe these facts, the only way is to attach assertion to the sets representing people, sheep, and English.

Issues in Knowledge Representation

- The reason to represent sets of objects is: if a property is true for all or most elements of a set, then it is more efficient to associate it once with the set rather than to associate it explicitly with every elements of the set.
- This is done,
  - in logical representation through the use of universal quantifier, and
  - in hierarchical structure where node represent sets and inheritance propagate set level assertion down to individual.

Issues in Knowledge Representation

- Finding Right structure:
  - Given a large amount of knowledge stored in a database, how can relevant parts are accessed when they are needed?
  - This is about access to right structure for describing a particular situation.
  - This requires, selecting an initial structure and then revising the choice.

Issues in Knowledge Representation

- While doing so, it is necessary to solve following problems:
  - How to perform an initial selection of the most appropriate structure.
  - How to fill in appropriate details from the current situations.
  - How to find a better structure if the one chosen initially turns out not to be appropriate.
  - What to do if none of the available structures is appropriate.
  - When to create and remember a new structure.