CHAPTER – 5

Representing Knowledge Using Rules

Procedural Knowledge

- **Definition**: Procedural knowledge is *knowing how* to do something. It refers to the knowledge encoded in procedures, algorithms, or step-by-step instructions.
- **Nature**: Implicit; it's often hard to express in words. Think of it like muscle memory or a set of instructions in code.
- **Examples**:
    - How to ride a bicycle.
    - A sorting algorithm in a program.
- **In AI**: Represented as scripts, production rules, or functions in a program. Used in expert systems and production systems.
- **Characteristics**:
    - Executable instructions.
    - Often involves control flow (e.g., loops, conditions).
    - Focused on “how” to achieve a goal.

Declarative Knowledge

- **Definition**: Declarative knowledge is *knowing that* something is true. It represents facts and relationships between facts.
- **Nature**: Explicit; easy to communicate.
- **Examples**:
    - Paris is the capital of France.
    - If it rains, the ground gets wet.
- **In AI**: Expressed using logic (e.g., first-order logic), semantic networks, or frames.
- **Characteristics**:
    - Focused on “what” is known, not “how” to use it.
    - Suitable for reasoning and querying.
    - Easier to modify and update.

Comparison Table:

Feature	Procedural Knowledge	Declarative Knowledge
Focus	How to do something	What is known
Represented As	Algorithms, Procedures	Facts, Rules
Modifiability	Hard to update	Easy to update
Use in AI	Production systems, rule engines	Logic-based systems, databases
Control over process	High	Low

2. Logic Programming

What is Logic Programming?

- A style of programming based on formal logic.
- Programs are written as a set of logical statements and rules.
- The system uses inference to derive conclusions from facts.

Core Language: Prolog

- **Prolog** (Programming in Logic) is the most well-known logic programming language.
- Syntax involves **facts**, **rules**, and **queries**.
- Example:

parent(john, mary).

parent(mary, susan).

grandparent(X, Y) :- parent(X, Z), parent(Z, Y).

Query: grandparent(john, susan)? → Returns true.

Key Concepts:

- **Facts**: Base knowledge (e.g., cat(tom)).
- **Rules**: Implications using logical inference (e.g., animal(X) :- cat(X)).
- **Queries**: Questions asked to the system (e.g., ?- animal(tom)).

Advantages:

- Separation of logic from control.
- Natural fit for problem domains involving reasoning.
- High-level abstraction for knowledge representation.

Applications:

- Expert systems.
- Natural language processing.
- Theorem proving.
- AI planning.

3. Forward Versus Backward Reasoning

Forward Reasoning (Data-driven inference)

- Starts with **known facts** and applies rules to infer new facts.
- Continues until a goal is reached or no new facts can be derived.

Example:

Rules:

1. If A, then B

2. If B, then C

Known fact: A

Forward chaining:

A → B → C

Used When:

- You have lots of data and want to discover conclusions.
- You want to simulate the behavior of an expert system.

AI Example:

- Used in **production systems**, e.g., CLIPS, Jess.

Backward Reasoning (Goal-driven inference)

- Starts with a **goal** and works backward to see if the known facts support it.
- Checks if rules can satisfy the goal by recursively proving required sub-goals.

Example:

Goal: Prove C

Check if there's a rule for C → Rule: If B, then C

Check if B is true → Rule: If A, then B

Known fact: A

→ A → B → C

Used When:

- The goal is known, and you want to verify if it's derivable.
- You want to limit the search space to only relevant facts.

AI Example:

- Used in **Prolog**, **theorem provers**, diagnostic systems.

Comparison Table:

Feature	**Forward Reasoning**	**Backward Reasoning**
Direction	From facts to conclusions	From goal to required facts
Triggered By	New data	A specific query or goal
Efficiency	Good for generating data	Good for specific goal search
Use Case	Monitoring systems, automation	Diagnosis, problem-solving