Procedural Knowledge

Procedural or imperative knowledge clarifies how to perform a certain task. It lays down the steps to perform. For example, - how to copy an array using javascript in a procedural way.

Example

```
var a=[1, 2, 3, 4, 5];
var b=[];
for(var i=0;i < a.length;i++) {
   b.push(a[i]);
}
console.log(b);
```

Output

```
[1, 2, 3, 4, 5]
```

Declarative Knowledge

Declarative or functional knowledge clarifies what to do to perform a certain task. It lays down the function to perform. For example, - how to copy an array using javascript in a declarative way.

Example

```
var a=[1, 2, 3, 4, 5];
var b=a.map(function(number){
   return number*1
});
console.log(b);
```

Output

```
[1, 2, 3, 4, 5]
```

The prior difference between them is that the **declarative** representation just specifies the knowledge but not the mechanism to implement the knowledge. On the contrary, the **procedural** representation provides the control information along with the knowledge.

❑ Procedural Knowledge (imperative)

- ✓ Procedural Knowledge is the knowledge of how to perform or how to operate or how to do something.
- ✓ It's a representation in which the control information that is necessary to use the knowledge is embedded in the knowledge itself.
- ✓ For example: computer programs, directions, and recipes; these indicate specific use or implementation.
- ✓ Example: How to cook vegetable or how to prepare a particular dish is procedural knowledge.
- ✓ Moreover, Knowledge encoded in some procedures, small programs that know how to do specific things and how to proceed.
- ✓ So in procedural representation it says how they will examined.
- ✓ Procedural knowledge includes: rules, strategies, agendas, procedures

❑ Procedural Knowledge (imperative)

- ✓ For example: consider the following

man (marcus).
man (caesar).
person (cleo).
∀x: man(X) → person(X).

- ✓ Now, try to answer the question. ?-person(Y).
- ✓ The knowledge base justifies any of the following answers.

Y=marcus
Y=caesar
Y=cleo

- ✓ We get more than one value that satisfies the predicate.
- ✓ If only one value needed, then the answer to the question will depend on the order in which the assertions examined during the search for a response.
- ✓ If the assertions declarative then they do not themselves say anything about how they will be examined. In case of procedural representation, they say how they will examine.

❑ Declarative Knowledge (descriptive)

- ✓ Declarative knowledge refers to facts or information stored in the memory, that is considered static in nature. Declarative knowledge, also referred to as conceptual, propositional or descriptive knowledge, describes things, events, or processes; their attributes; and their relation to each other.
- ✓ Declarative knowledge includes: concepts, objects, facts.
- ✓ In declarative knowledge, a statement in which knowledge is specified, but how the knowledge is to be put is not given.
- ✓ So to use declarative representation, we must have a program that explains what is to do with the knowledge and how.
- ✓ It is more general than procedural knowledge
- ✓ Example 1: The first step in cooking a vegetable is chopping it.
- ✓ Example 2: To prepare a dish one needs to gather its ingredients.

❑ Difference between Procedural and Declarative Knowledge

Procedural Knowledge	Declarative Knowledge
Knowledge about " how to do something". For e.g: (i) To determine Peter or Robert is older, first find their age. (ii) Procedure to harvest a crop	Knowledge about "that something is true or false". For e.g: (i) Peter is older than Robert . (ii) Knowledge of the month when a crop should be harvested
In procedural representation they say how they will be examined.	In declarative, they does not say how they will be examined.
Focuses on tasks that must be performed to reach a particular objective or goal.	Refers to representation of objects and events.

BASIS FOR COMPARISON	PROCEDURAL KNOWLEDGE	DECLARATIVE KNOWLEDGE
Basic	Includes the knowledge of how a particular thing can be accomplished.	Includes the basic knowledge about something.
Alternate name	Interpretive knowledge	Descriptive knowledge
Stated by	Direct application to the task and difficult to articulate formally.	Declarative sentences and easily articulated.
Popularity	Less common	Generally used
Ease of sharing the knowledge	Hard to communicate	Can be easily shared, copied, processed and stored.
Taken from	Experience, action, and subjective insight.	Artifact of some type as a principle, procedure, process and concepts.
Nature	Process oriented	Data-oriented
Feature	Debugging is difficult	Validation is quite simple

A. Forward Chaining

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

The Forward-chaining algorithm starts from known facts, triggers all rules whose premises are satisfied, and add their conclusion to the known facts. This process repeats until the problem is solved.

Properties of Forward-Chaining:

- It is a down-up approach, as it moves from bottom to top.
- It is a process of making a conclusion based on known facts or data, by starting from the initial state and reaches the goal state.
- Forward-chaining approach is also called as data-driven as we reach to the goal using available data.
- Forward -chaining approach is commonly used in the expert system, such as CLIPS, business, and production rule systems.

B. Backward Chaining:

Backward-chaining is also known as a backward deduction or backward reasoning method when using an inference engine. A backward chaining algorithm is a form of reasoning, which starts with the goal and works backward, chaining through rules to find known facts that support the goal.

Properties of backward chaining:

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

Following is the difference between the forward chaining and backward chaining:

- Forward chaining as the name suggests, start from the known facts and move forward by applying inference rules to extract more data, and it continues until it reaches to the goal, whereas backward chaining starts from the goal, move backward by using inference rules to determine the facts that satisfy the goal.
- Forward chaining is called a **data-driven** inference technique, whereas backward chaining is called a **goal-driven** inference technique.
- Forward chaining is known as the **down-up** approach, whereas backward chaining is known as a **top-down** approach.
- Forward chaining uses **breadth-first search** strategy, whereas backward chaining uses **depth-first search** strategy.
- Forward and backward chaining both applies **Modus ponens** inference rule.
- Forward chaining can be used for tasks such as **planning, design process monitoring, diagnosis, and classification**, whereas backward chaining can be used for **classification and diagnosis tasks**.
- Forward chaining can be like an exhaustive search, whereas backward chaining tries to avoid the unnecessary path of reasoning.

S. No.	Forward Chaining	Backward Chaining
1.	Forward chaining starts from known facts and applies inference rule to extract more data unit it reaches to the goal.	Backward chaining starts from the goal and works backward through inference rules to find the required facts that support the goal.
2.	It is a bottom-up approach	It is a top-down approach
3.	Forward chaining is known as data-driven inference technique as we reach to the goal using the available data.	Backward chaining is known as goal-driven technique as we start from the goal and divide into sub-goal to extract the facts.
4.	Forward chaining reasoning applies a breadth-first search strategy.	Backward chaining reasoning applies a depth-first search strategy.
5.	Forward chaining tests for all the available rules	Backward chaining only tests for few required rules.
6.	Forward chaining is suitable for the planning, monitoring, control, and	Backward chaining is suitable for diagnostic, prescription, and debugging application.

7.	Forward chaining can generate an infinite number of possible conclusions.	Backward chaining generates a finite number of possible conclusions.
8.	It operates in the forward direction.	It operates in the backward direction.
9.	Forward chaining is aimed for any conclusion.	Backward chaining is only aimed for the required data.

✓Forward v/s Backward Reasoning

✓Factors that influence whether to choose forward or backward reasoning :

✓Are there **more possible start states or goal states**? We would like to go from smaller set of states to larger set of states.

✓In which direction is the **branching factor** (the average number of nodes that can be reached directly from a single node) greater? We would like to proceed in the direction with the lower branching factor.

✓Will the program be **asked to justify its reasoning process** to the user? It so, it is important to proceed in the direction that corresponds more closely with the way user will think.

✓What kind of **event is going to trigger a problem-solving episode**? If it is the arrival of a new fact , forward reasoning should be used. If it a query to which response is desired, use backward reasoning.

Difference Between Forward and Backward Reasoning in AI

Forward Reasoning

- It is a data-driven task.
- It begins with new data.
- The object is to find a conclusion that would follow.
- It uses an opportunistic type of approach.
- It flows from incipient to the consequence.
- The inference engine searches the knowledge base with the given information depending on the constraints.
- The precedence of these constraints have to match the current state.
- The first step is that the system is given one or more constraints.
- The rules are searched for in the knowledge base for every constraint.
- The rule that fulfils the condition is selected.
- Every rule can produce new condition from the conclusion which is obtained from the invoked one.
- New conditions can be added, and are processed again.
- The step ends if no new conditions exist.
- It may be slow,
- It follows top-down reasoning.

Backward Reasoning

- It is a goal driven task.
- It begins with conclusions that are uncertain.
- The objective is to find the facts that support the conclusions.
- It uses a conservative type of approach.
- It flows from consequence to the incipient.
- The system helps choose a goal state, and reasons in a backward direction.
- First step is that the goal state and rules are selected.
- Sub-goals are made from the selected rule, which need to be satisfied for the goal state to be true.
- The initial conditions are set such that they satisfy all the sub-goals.
- The established states are matched to the initial state provided.
- If the condition is fulfilled, the goal is the solution.
- Otherwise the goal is rejected.
- It tests less number of rules.
- It provides small amount of data.
- It follows bottom-up reasoning technique.
- It contains less number of initial goals and has large number of rules.
- It is based on the decision fetched by the initial state.
- It is also known as a decision-driven or goal-driven inference technique.

Logic Programming

Logic Programming is the combination of two words, logic and programming. Logic Programming is a programming paradigm in which the problems are expressed as facts and rules by program statements but within a system of formal logic. Just like other programming paradigms like object oriented, functional, declarative, and procedural, etc., it is also a particular way to approach programming.

To understand how a problem can be solved in logic programming, we need to know about the building blocks − Facts and Rules −

Facts

Actually, every logic program needs facts to work with so that it can achieve the given goal. Facts basically are true statements about the program and data. For example, Delhi is the capital of India.

Rules

Actually, rules are the constraints which allow us to make conclusions about the problem domain. Rules basically written as logical clauses to express various facts. For example, if we are building any game then all the rules must be defined.

Logic Programming- Prolog

Prolog is a logic programming language. It has important role in artificial intelligence. Unlike many other programming languages, Prolog is intended primarily as a declarative programming language. In prolog, logic is expressed as relations (called as Facts and Rules). Core heart of prolog lies at the **logic** being applied. Formulation or Computation is carried out by running a query over these relations.

Key Features :

1. Unification : The basic idea is, can the given terms be made to represent the same structure.
2. Backtracking : When a task fails, prolog traces backwards and tries to satisfy previous task.
3. Recursion : Recursion is the basis for any search in program.

Advantages :

1. Easy to build database. Doesn't need a lot of programming effort.
2. Pattern matching is easy. Search is recursion based.
3. It has built in list handling. Makes it easier to play with any algorithm involving lists.

Disadvantages :

1. LISP (another logic programming language) dominates over prolog with respect to I/O features.
2. Sometimes input and output is not easy.

A Declarative and a Procedural Representation

$\forall x : pet(x) \wedge small(x) \rightarrow apartmentpet(x)$
$\forall x : cat(x) \vee dog(x) \rightarrow pet(x)$
$\forall x : poodle(x) \rightarrow dog(x) \wedge small(x)$
$poodle(ftujfy)$

A Representation in Logic

```
apartmentpet(X) :- pet(X), small(X).
pet(X) :- cat(X).
pet(X) :- dog(X).
dog(X) :- poodle (X) .
small(X) :- poodle(X).
poodle(fluffy).
```

A Representation in PROLOG

Answering Questions in PROLOG

```
apartmentpet(X) :- pet(X), small(X).
pet(X) :- cat(X).
pet(X) :- dog(X).
dog(X) :- poodle (X) .
small(X) :- poodle(X).
poodle(fluffy).

?- apartmentpet(X).

?- cat(fluffy).

?- cat (mittens)
```

9. Which form is called as a conjunction of disjunction of literals?

a) Conjunctive normal form

b) Disjunctive normal form

c) Normal form

d) All of the mentioned

Answer: A

The proposition $(P \Rightarrow Q) \wedge (Q \Rightarrow P)$ is a

1. tautology
2. contradiction
3. contingency
4. absurdity

Option 3 : contingency

Concept:

- A compound proposition that is always true for all possible truth values of the propositions is called a tautology.
- A compound proposition that is always false is called a contradiction.
- A proposition that is neither a tautology nor contradiction is called a contingency.

-

$(P \Rightarrow Q) \wedge (Q \Rightarrow P) = P \Leftrightarrow Q$ (It is always contingency)

Lifted inference rules require finding substitutions that make different logical expressions looks identical.

a) Existential Instantiation

b) Universal Instantiation

c) *Unification*

d) Modus Ponen

Which is used to compute the truth of any sentence?
a) Semantics of propositional *logic*
b) Alpha-beta pruning
c) First-order logic
d) Both Semantics of propositional logic & Alpha-beta pruning

Which condition is used to cease the growth of forward chaining?

a) Atomic sentences

b) Complex sentences

c) No further inference

d) All of the mentioned

Answer: c Explanation: Forward chain can grow by adding new atomic sentences until no further inference is made.

3. Which algorithm will work backward from the goal to solve a problem?

a) Forward chaining

b) Backward chaining

c) Hill-climb algorithm

d) None of the mentioned

Answer: b

Explanation: Backward chaining algorithm will work backward from the goal and it will chain the known facts that support the proof.

. What will backward chaining algorithm will return?

a) Additional statements

b) Substitutes matching the query

c) Logical statement

d) All of the mentioned

Answer: b

Explanation: It will contains the list of goals containing a single element and returns the set of all substitutions satisfying the query.

.. How can be the goal is thought of in backward chaining algorithm?

a) Queue

b) List

c) Vector

d) Stack

Answer: d

Explanation: The goals can be thought of as stack and if all of them us satisfied means, then current branch of proof succeeds.

. Which algorithm are in more similar to backward chaining algorithm?

a) Depth-first search algorithm

b) Breadth-first search algorithm

c) Hill-climbing search algorithm

d) All of the mentioned

Answer: a

Explanation: It is depth-first search algorithm because its space requirements are linear in the size of the proof.

. Which is a refutation complete inference procedure for propositional logic?

a) Clauses

b) Variables

c) Propositional resolution

d) Proposition

Answer: c

Explanation: Propositional resolution is a refutation complete inference procedure for propositional logic.

. What kind of clauses are available in Conjunctive Normal Form?

a) Disjunction of literals

b) Disjunction of variables

c) Conjunction of literals

d) Conjunction of variables

Answer: a

Explanation: First-order resolution requires the clause to be in disjunction of literals in Conjunctive Normal Form.

. At which state does the propositional literals are complementary?

a) If one variable is less

b) If one is the negation of the other

c) All of the mentioned

d) None of the mentioned

Answer: b

Explanation: Propositional literals are complementary if one is the negation of the other.

5. What will happen if two literals are identical?

a) Remains the same

b) Added as three

c) Reduced to one

d) None of the mentioned

Answer: c

Explanation: Propositional factoring reduces two literals to one if they are identical.

When the resolution is called as refutation-complete?

a) Sentence is satisfiable

b) Sentence is unsatisfiable

c) Sentence remains the same

d) None of the mentioned

Answer: b Explanation: Resolution is refutation-complete, if a set of sentence is unsatisfiable, then resolution will always be able to derive a contradiction.