**I. HORN CLAUSE**

A*clause* (i.e., a [*disjunction*](https://mathworld.wolfram.com/Disjunction.html)*of*[*literals*](https://mathworld.wolfram.com/Literal.html)) is called a ***Horn clause*** if it contains at most one [positive literal](https://mathworld.wolfram.com/PositiveLiteral.html). Horn clauses are usually written as

|  |
| --- |
| L_1,...,L_n=>L(=¬L_1 v ... v ¬L_n v L) |

or

|  |
| --- |
| L_1,...,L_n=>(=¬L_1 v ... v ¬L_n), |

where n>=0 and L is the only [positive literal](https://mathworld.wolfram.com/PositiveLiteral.html).

A [***definite clause***](https://mathworld.wolfram.com/DefiniteClause.html) is a Horn clause that has [exactly one](https://mathworld.wolfram.com/ExactlyOne.html) [positive literal](https://mathworld.wolfram.com/PositiveLiteral.html). A Horn clause without a [positive literal](https://mathworld.wolfram.com/PositiveLiteral.html) is called a [***goal***](https://mathworld.wolfram.com/Goal.html).

Horn clauses express a subset of statements of [first-order logic](https://mathworld.wolfram.com/First-OrderLogic.html). Programming language Prolog is built on top of Horn clauses. Prolog programs are comprised of [definite clauses](https://mathworld.wolfram.com/DefiniteClause.html) and any question in Prolog is a [goal](https://mathworld.wolfram.com/Goal.html)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Disjunction form** | **Implication form** | **Read intuitively as** |
| **Definite clause** | ¬*p* ∨ ¬*q* ∨ ... ∨ ¬*t* ∨ *u* | *u* ← *p* ∧ *q* ∧ ... ∧ *t* | assume that, if *p* and *q* and ... and *t* all hold, then also *u* holds |
| **Fact** | *u* | *u* | assume that *u* holds |
| **Goal clause** | ¬*p* ∨ ¬*q* ∨ ... ∨ ¬*t* | *false* ← *p* ∧ *q* ∧ ... ∧ *t* | show that *p* and *q* and ... and *t* all hold |

In the non-propositional case, all variables in a clause are implicitly universally quantified with the scope being the entire clause. Thus, for example:

¬ *human*(*X*) ∨ *mortal*(*X*)

stands for:

∀X( ¬ *human*(*X*) ∨ *mortal*(*X*) ) as ¬a∨b= a →b

which is logically equivalent to:

∀X ( *human*(*X*) → *mortal*(*X*) )

**II. Forward and Backward Chaining**

AI can help us solve numerous problems of varying complexities. One such type of problem is the case where **one has to predict outcomes using the given pool of knowledge.**  Here, the **knowledge base** is given and using **logical rules and reasoning**, one has to **predict the outcome**.

These problems are usually solved using Inference Engines, which utilize their two special modes: **Backward Chaining**and **Forward Chaining**.

**Inference Engine:**

An Inference Engine is a tool of Artificial Intelligence that is used as a component of the system to deduce new information from a **knowledge base** using **logical rules and reasoning**. The first-ever Inference Engines were a part of **expert systems** in AI. As previously stated, Inference Engines predict outcomes with the already existing pool of data, comprehensively analyzing it and using logical reasoning to predict the outcomes.

This same process would be repeated as **new facts** would be discovered and this would make the inference engine trigger additional rules for its findings. After some runs of the inference engine, it was noticed that Inference Engines works in one of the two ways, either based on goals or based on facts, which later came to be known as **forwarding chaining**and **backward chaining**.

***Forward chaining*** comes with known facts and iterates the process to find new facts while ***backward chaining starts with goals*** and works backwards to determine what conditions would be required to achieve the given goals.

## Examples regarding Inference Rules

Let’s take a look at some simple examples to help you differentiate between both sets of inference rules.

**Inference Rules**

**(i) Deductive inference rule:**

Forward Chaining: Conclude from “A” and “A implies B” to “B”.

A

A -> B

B

Example:

It is raining.

If it is raining, the street is wet.

The street is wet.

(ii) **Abductive inference rule**:

Backward Chaining: Conclude from “B” and “A implies B” to “A”.

B

A -> B

A

Example:

The street is wet.

If it is raining, the street is wet.

It is raining.

## Forward Chaining

As a data-driven as well as bottom-up logic approach, forward chaining starts from known facts and conditions, then progresses towards logical conclusion using **if-then** statements. Then these conditions and rules are applied to the problem until no further applicable situations are left or the limit has been reached. Forward Chaining searches for any solutions and can come up with an infinite number of possible conclusions.

The Forward-thinking approach is used in AI to help an AI agent solve logical problems by inspecting the data from the previous learning’s and then coming to a conclusion full of solutions. That’s not all, Forward Chaining might as well be used to explore the available information or answer a question or solve a problem. Forward chaining is extensively used to break down a long and complex logical approach by attaching each step once the previous one is completed. This way, ***it goes from beginning to the end with relative ease.***

### Steps for working of Forwarding Chaining

1. Step 1: We start from the already stated facts, and then, we’ll subsequently choose the ***facts that do not have any implications*** at all.
2. Step 2: Now, we will ***state those facts that*** can be inferred from available facts with satisfied premises.
3. Step 3: In step 3 we can **check** the given statement that needs to be checked and check whether ***it is satisfied with the substitution which infers all the previously stated facts. Thus we reach our goal.***

**Backward Chaining**

It is a logical process of determining unknown facts from known solutions by moving backwards from known solutions to determine the initial conditions and rules.

The Backward Chaining approach is used in AI to find the conditions and rules because of which a particular logical result or conclusion was reached. Real-life applications of Backward Chaining include use to find information regarding conclusions and solutions in reverse engineering practices as well as **game theory applications**.

Some other applications of Backward Chaining include automated theorem proving tools, inference engines, proof assistants and othe**r**[**artificial intelligence applications**](https://www.mygreatlearning.com/blog/business-applications-for-artificial-intelligence-and-machine-learning/).

### Steps of working for Backward Chaining

1. Step 1. In the first step, we’ll take the ***Goal Fact*** and from the goal fact, ***we’ll derive other facts*** that we’ll prove true.
2. Step 2: We’ll derive other facts from goal facts that satisfy the rules
3. Step 3: At step-3, we will extract further fact which infers from facts inferred in step 2.
4. Step 4: We’ll repeat the same until we get to a certain fact that satisfies the conditions.

## Difference between Forward Chaining and Backward Chaining

|  |  |  |
| --- | --- | --- |
| **S No** | **Forward Chaining** | **Backward Chaining** |
| 1. | It starts from known facts extract more data unit it reaches to the goal using inference rule | It starts from the goal and works backward through inference rules to find the required facts that support the goal. |
| 2. | Bottom-up Approach | Top-Down Approach |
| 3. | Known as Data-driven approach as we use given data to reach the goals | Known as goal-driven approach because we use the goal given to reach the facts that support the goals |
| 4 | Applies a breadth-first search strategy | Applies a depth-first search strategy |
| 5 | Tests for all the available rules | Only tests for certain given and selected rules |
| 6 | Suitable for planning, monitoring, control, and interpretation application. | Suitable for diagnostic, prescription, and debugging application. |
| 7. | Can generate infinite number of possible conclusions | Can generate a finite number of possible concluding facts and conditions |
| 8. | Operates in Forward Direction | Operates in Backward Direction |
| 9 | Forward Chaining is aimed for any conclusion. | Backward chaining is aimed for only the required data. |