What is Forward Chaining and Backward Chaining? Explain with example.

**Forward chaining** is one of the two main methods of reasoning when using an [inference engine](https://en.wikipedia.org/wiki/Inference_engine) and can be described [logically](https://en.wikipedia.org/wiki/Logically) as repeated application of [*modus ponens*](https://en.wikipedia.org/wiki/Modus_ponens). Forward chaining is a popular implementation strategy for [expert systems](https://en.wikipedia.org/wiki/Expert_system), [business](https://en.wikipedia.org/wiki/Business_rules_engine) and [production rule systems](https://en.wikipedia.org/wiki/Production_system_(computer_science)). The opposite of forward chaining is [backward chaining](https://en.wikipedia.org/wiki/Backward_chaining).

Forward chaining starts with the available [data](https://en.wikipedia.org/wiki/Data) and uses inference rules to extract more data (from an end user, for example) until a [goal](https://en.wikipedia.org/wiki/Goal) is reached. An [inference engine](https://en.wikipedia.org/wiki/Inference_engine) using forward chaining searches the inference rules until it finds one where the [antecedent](https://en.wikipedia.org/wiki/Antecedent_(logic)) (**If** clause) is known to be true. When such a rule is found, the engine can conclude, or infer, the [consequent](https://en.wikipedia.org/wiki/Consequent) (**Then** clause), resulting in the addition of new [information](https://en.wikipedia.org/wiki/Information) to its data.[[1]](https://en.wikipedia.org/wiki/Forward_chaining#cite_note-1)

Inference engines will [iterate](https://en.wikipedia.org/wiki/Iteration#Computing) through this process until a goal is reached.

For example, suppose that the goal is to conclude the color of a pet named Fritz, given that he croaks and eats flies, and that the [rule base](https://en.wikipedia.org/wiki/Rule_base) contains the following four rules:

1. **If** X croaks and X eats flies - **Then** X is a frog
2. **If** X chirps and X sings - **Then** X is a canary
3. **If** X is a frog - **Then** X is green
4. **If** X is a canary - **Then** X is yellow

Let us illustrate forward chaining by following the pattern of a computer as it evaluates the rules. Assume the following facts:

* Fritz croaks
* Fritz eats flies

With forward reasoning, the inference engine can derive that Fritz is green in a series of steps:

1. Since the base facts indicate that "Fritz croaks" and "Fritz eats flies", the antecedent of rule #1 is satisfied by substituting Fritz for X, and the inference engine concludes:

Fritz is a frog

2. The antecedent of rule #3 is then satisfied by substituting Fritz for X, and the inference engine concludes:

Fritz is green

The name "forward chaining" comes from the fact that the inference engine starts with the data and reasons its way to the answer, as opposed to [backward chaining](https://en.wikipedia.org/wiki/Backward_chaining), which works the other way around. In the derivation, the rules are used in the opposite order as compared to [backward chaining](https://en.wikipedia.org/wiki/Backward_chaining). In this example, rules #2 and #4 were not used in determining that Fritz is green.

Because the data determines which rules are selected and used, this method is called [data-driven](https://en.wikipedia.org/wiki/Data_driven), in contrast to [goal-driven](https://en.wikipedia.org/wiki/Goal-oriented) [backward chaining](https://en.wikipedia.org/wiki/Backward_chaining) inference. The forward chaining approach is often employed by [expert systems](https://en.wikipedia.org/wiki/Expert_system), such as [CLIPS](https://en.wikipedia.org/wiki/CLIPS).

One of the advantages of forward-chaining over backward-chaining is that the reception of new data can trigger new inferences, which makes the engine better suited to dynamic situations in which conditions are likely to change

**Backward chaining** (or **backward reasoning**) is an [inference](https://en.wikipedia.org/wiki/Inference) method that can be described (in lay terms) as working backward from the goal(s). It is used in [automated theorem provers](https://en.wikipedia.org/wiki/Automated_theorem_prover), [inference engines](https://en.wikipedia.org/wiki/Inference_engine), [proof assistants](https://en.wikipedia.org/wiki/Proof_assistant) and other [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) applications.[[1]](https://en.wikipedia.org/wiki/Backward_chaining#cite_note-1)

In [game theory](https://en.wikipedia.org/wiki/Game_theory), its application to (simpler) [subgames](https://en.wikipedia.org/wiki/Subgame) in order to find a solution to the game is called [backward induction](https://en.wikipedia.org/wiki/Backward_induction). In chess, it is called [retrograde analysis](https://en.wikipedia.org/wiki/Retrograde_analysis), and it is used to generate tablebases for [chess endgames](https://en.wikipedia.org/wiki/Chess_endgame) for [computer chess](https://en.wikipedia.org/wiki/Computer_chess).

Backward chaining is implemented in [logic programming](https://en.wikipedia.org/wiki/Logic_programming) by [SLD resolution](https://en.wikipedia.org/wiki/SLD_resolution). Both rules are based on the [modus ponens](https://en.wikipedia.org/wiki/Modus_ponens) inference rule. It is one of the two most commonly used methods of [reasoning](https://en.wikipedia.org/wiki/Reasoning) with [inference rules](https://en.wikipedia.org/wiki/Inference_rule) and [logical implications](https://en.wikipedia.org/wiki/Logical_consequence) – the other is [forward chaining](https://en.wikipedia.org/wiki/Forward_chaining). Backward chaining systems usually employ a [depth-first search](https://en.wikipedia.org/wiki/Depth-first_search) strategy, e.g.[Prolog](https://en.wikipedia.org/wiki/Prolog).[[2]](https://en.wikipedia.org/wiki/Backward_chaining#cite_note-CheinMugnier2009-2)

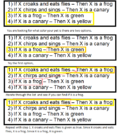
How it works

Backward chaining starts with a list of [goals](https://en.wikipedia.org/wiki/Goal) (or a [hypothesis](https://en.wikipedia.org/wiki/Hypothesis)) and works backwards from the [consequent](https://en.wikipedia.org/wiki/Consequent) to the [antecedent](https://en.wikipedia.org/wiki/Antecedent_(logic)) to see if there is [data](https://en.wikipedia.org/wiki/Data) available that will support any of these consequents.[[3]](https://en.wikipedia.org/wiki/Backward_chaining#cite_note-Norwig_Definition-3) An [inference engine](https://en.wikipedia.org/wiki/Inference_engine) using backward chaining would search the [inference](https://en.wikipedia.org/wiki/Inference) rules until it finds one which has a consequent (**Then** clause) that matches a desired goal. If the antecedent (**If** clause) of that rule is not known to be true, then it is added to the list of goals (in order for one's goal to be confirmed one must also provide data that confirms this new rule).

For example, suppose a new pet, Fritz, is delivered in an opaque box along with two facts about Fritz:

* Fritz croaks
* Fritz eats flies

The goal is to decide whether Fritz is green, based on a [rule base](https://en.wikipedia.org/wiki/Rule_base) containing the following four rules:

[](https://en.wikipedia.org/wiki/File:Backward_Chaining_Frog_Color_Example.png)

An Example of Backward Chaining.

1. **If** X croaks and X eats flies – **Then** X is a frog
2. **If** X chirps and X sings – **Then** X is a canary
3. **If** X is a frog – **Then** X is green
4. **If** X is a canary – **Then** X is yellow

With backward reasoning, an inference engine can determine whether Fritz is green in four steps. To start, the query is phrased as a goal assertion that is to be proved: "Fritz is green".

1. Fritz is substituted for X in rule #3 to see if its consequent matches the goal, so rule #3 becomes:

**If** Fritz is a frog – **Then** Fritz is green

Since the consequent matches the goal ("Fritz is green"),the rules engine now needs to see if the antecedent ("If Fritz is a frog") can be proved. The antecedent therefore becomes the new goal:

Fritz is a frog

2. Again substituting Fritz for X, rule #1 becomes:

**If** Fritz croaks and Fritz eats flies – **Then** Fritz is a frog

Since the consequent matches the current goal ("Fritz is a frog"), the inference engine now needs to see if the antecedent ("If Fritz croaks and eats flies") can be proved. The antecedent therefore becomes the new goal:

Fritz croaks and Fritz eats flies

3. Since this goal is a conjunction of two statements, the inference engine breaks it into two sub-goals, both of which must be proved:

Fritz croaks

Fritz eats flies

4. To prove both of these sub-goals, the inference engine sees that both of these sub-goals were given as initial facts. Therefore, the conjunction is true:

Fritz croaks and Fritz eats flies

therefore the antecedent of rule #1 is true and the consequent must be true:

Fritz is a frog

therefore the antecedent of rule #3 is true and the consequent must be true:

Fritz is green

This derivation therefore allows the inference engine to prove that Fritz is green. Rules #2 and #4 were not used.

Note that the goals always match the affirmed versions of the consequents of implications (and not the negated versions as in [modus tollens](https://en.wikipedia.org/wiki/Modus_tollens)) and even then, their antecedents are then considered as the new goals (and not the conclusions as in [affirming the consequent](https://en.wikipedia.org/wiki/Affirming_the_consequent)) which ultimately must match known facts (usually defined as consequents whose antecedents are always true); thus, the inference rule which is used is [modus ponens](https://en.wikipedia.org/wiki/Modus_ponens).

Because the list of goals determines which rules are selected and used, this method is called [goal-driven](https://en.wikipedia.org/wiki/Goal-oriented), in contrast to [data-driven](https://en.wikipedia.org/wiki/Data_driven) [forward-chaining](https://en.wikipedia.org/wiki/Forward_chaining) inference. The backward chaining approach is often employed by [expert systems](https://en.wikipedia.org/wiki/Expert_systems).

Programming languages such as [Prolog](https://en.wikipedia.org/wiki/Prolog), [Knowledge Machine](https://en.wikipedia.org/wiki/Knowledge_Machine) and [ECLiPSe](https://en.wikipedia.org/wiki/ECLiPSe) support backward chaining within their inference engines