Forward and Backward Chaining

DEFINITION

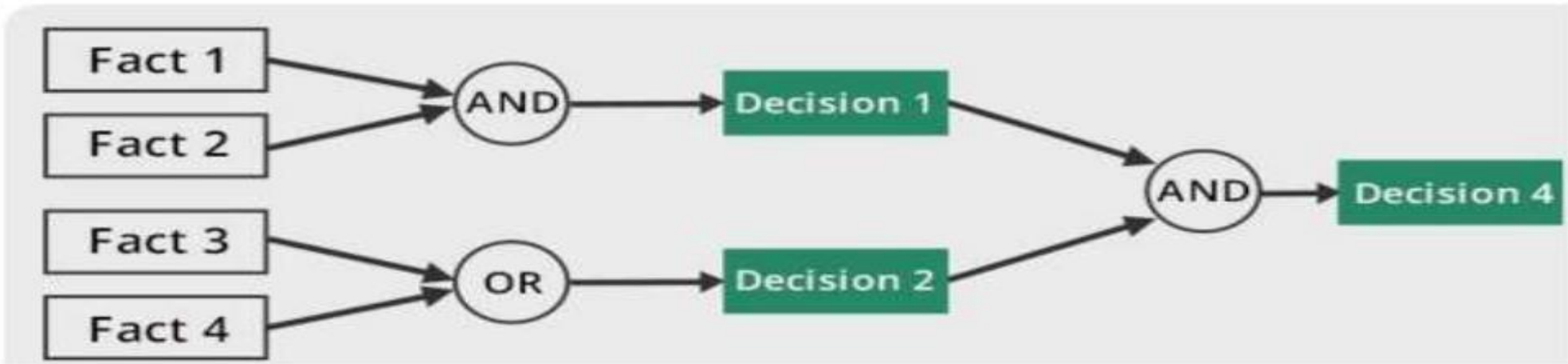
- Forward chaining is a data driven method of deriving a particular goal from a given knowledge base and set of inference rules
- Inference rules are applied by matching facts to the antecedents of consequence relations in the knowledge base
- The application of inference rules results in new knowledge (from the consequents of the relations matched), which is then added to the knowledge base

Forward Chaining

It is a strategy of an expert system to answer the question, **"What can happen next?"**

Here, the interface engine follows the chain of conditions and derivations and finally deduces the outcome. It considers all the facts and rules, and sorts them before concluding to a solution.

This strategy is followed for working on conclusion, result, or effect. For example, prediction of share market status as an effect of changes in interest rates.



FORWARD CHAINING (Ben Coppin)

- In Forward chaining, the system starts from a set of facts, and a set of rules, and tries to find a way of using those rules and facts to deduce a conclusion or come up with a suitable course of action.
- This is known as data-driven reasoning because the reasoning starts from a set of data and ends up at the goal, which is the conclusion.

Forward Chaining Mechanism

- When applying forward chaining, the first step is to take the facts in the fact database and see if any combination of these matches all the antecedents of one of the rules in the rule database.
- When all the antecedents of a rule are matched by facts in the database, then this rule is triggered
- Usually, when a rule is triggered, it is then fired, which means its conclusion is added to the facts database.
- If the conclusion of the rule that has fired is an action or a recommendation, then the system may cause that action to take place or the recommendation to be made

FORWARD CHAINING EXAMPLE

Rule 1

IF on first floor and button is pressed on first floor

THEN open door

Rule 2

IF on first floor

AND button is pressed on second floor

THEN go to second floor

Rule 3

IF on first floor

AND button is pressed on third floor

THEN go to third floor

Rule 4

IF	on second floor
AND	button is pressed on first floor
AND	already going to third floor
THEN	remember to go to first floor later

Let us imagine that we start with the following facts in our database:

Fact 1

At first floor

Fact 2

Button pressed on third floor

Fact 3

Today is Tuesday

- Now the system examines the rules and finds that Facts 1 and 2 match the antecedents of Rule 3. Hence, Rule 3 fires, and its conclusion “Go to third floor” is added to the database of facts.
- Presumably, this results in the elevator heading toward the third floor. Note that Fact 3 was ignored altogether because it did not match the antecedents of any of the rules.
- Now let us imagine that the elevator is on its way to the third floor and has reached the second floor, when the button is pressed on the first floor. The fact “Button pressed on first floor” Is now added to the database, which results in Rule 4 firing.

- Now let us imagine that later in the day the facts database contains the following information:
  Fact 1
  At first floor
  Fact 2
  Button pressed on second floor
  Fact 3
  Button pressed on third floor

- In this case, two rules are triggered—Rules 2 and 3. In such cases where there is more than one possible conclusion, conflict resolution needs to be applied to decide which rule to fire

Discussion

- Forward chaining applies a set of rules and facts to deduce whatever conclusions can be derived, which is useful when a set of facts are present, but you do not know what conclusions you are trying to prove
- In some cases, forward chaining can be inefficient because it may end up proving a number of conclusions that are not currently interesting
- In such cases, where a single specific conclusion is to be proved, backward chaining is more appropriate

FORWARD CHAINING (ANOTHER EXAMPLE) "WEATHER FORECASTING SYSTEM"

Suppose we have developed the following rules for our weather forecasting system,

Rule I

If we suspect temperature is less than 20^{o}
AND there is humidity in the air
Then there are chances of rain

Rule II

If Sun is behind the clouds
AND air is very cool.
Then we suspect temperature is less than 20^{o}.

Rule III **If** air is very heavy
Then there is humidity in the air.

- Suppose we have been given the following facts,
  a) Sun is behind the clouds.
  b) Air is very heavy and cool.
- Problem: Using Forward chaining try to conclude that there are chances of rain.

First Pass

Rule, premise	Status	Working Memory
1, 1 we suspect temperature is less than 20°	Unknown	a) Sun is behind the clouds. b) Air is very heavy and cool.
1, 2 there is humidity in the air	Unknown	a) Sun is behind the clouds. b) Air is very heavy and cool.
2, 1 Sun is behind the clouds	True	a) Sun is behind the clouds. b) Air is very heavy and cool.
2,2 air is very cool.	True, fire rule	a) Sun is behind the clouds. b) Air is very heavy and cool. c) We suspect temperature is less than 20°

Second Pass

Rule, premise	Status	Working Memory
1, 1 we suspect temperature is less than 20°	True	a) Sun is behind the clouds. b) Air is very heavy and cool. c) We suspect temperature is less than 20°
1, 2 there is humidity in the air	Unknown	a) Sun is behind the clouds. b) Air is very heavy and cool. c) We suspect temperature is less than 20°
3, 1 air is very heavy	True, fire rule	a) Sun is behind the clouds. b) Air is very heavy and cool. c) We suspect temperature is less than 20° d) there is humidity in the air

Third Pass

Rule, premise	Status	Working Memory
1, 1 we suspect temperature is less than 20°	True	a) Sun is behind the clouds. b) Air is very heavy and cool. c) We suspect temperature is less than 20° d) there is humidity in the air
1, 2 there is humidity in the air	True, fire rule	a) Sun is behind the clouds. b) Air is very heavy and cool. c) We suspect temperature is less than 20° d) there is humidity in the air **e) there are chances of rain**

So we have deduced there are chances of rain.

CONFLICT RESOLUTION

- In a situation where more than one conclusion can be deduced from a set of facts, there are a number of possible ways to decide which rule to fire (i.e., which conclusion to use or which course of action to take)

Example:

IF	you're bored
AND	you've no cash
THEN	go to a friend's place.

IF	you're bored
AND	You've no cash
THEN	go to a park.

CONFLICT RESOLUTION STRATEGIES

1. Fire the first rule in sequence.
2. Assign rule priorities (by importance).
3. More specific rules are preferred over more general rules.(e.g. a rule having 5 IF’s(handle more info) will be preferred over one having 3 IF’s)
4. Prefer rules whose premises are added more recently (time stamping)
5. Parallel strategy (create view points)

Example

For example, consider the following set of rules:

IF it is cold

THEN wear a coat

IF it is cold

THEN stay at home

IF it is cold

THEN turn on the heat

If there is a single fact in the fact database, which is "it is cold," then clearly there are three conclusions that can be derived. In some cases, it might be fine to follow all three conclusions, but in many cases the conclusions are incompatible (for example, when prescribing medicines to patients).

In one conflict resolution method, rules are given priority levels, and when a conflict occurs, the rule that has the highest priority is fired, as in the following example:

IF patient has pain

THEN prescribe painkillers priority 10

IF patient has chest pain

THEN treat for heart disease priority 100

Here, it is clear that treating possible heart problems is more important than just curing the pain.

- An alternative method is the longest-matching strategy. This method involves firing the conclusion that was derived from the longest rule.
- For example:

IF patient has pain
THEN prescribe painkiller

IF patient has chest pain
AND patient is over 60
AND patient has history of heart conditions
THEN take to emergency room

- Here, if all the antecedents of the second rule match, then this rule's conclusion should be fired rather than the conclusion of the first rule because it is a more specific match.

- A further method for conflict resolution is to fire the rule that has matched the facts most recently added to the database.
- In some case, it may be that the system fires one rule and then stops (as in medical diagnosis), but in many cases, the system simply needs to choose a suitable ordering for the rules (as when controlling an elevator) because each rule that matches the facts needs to be fired at some point

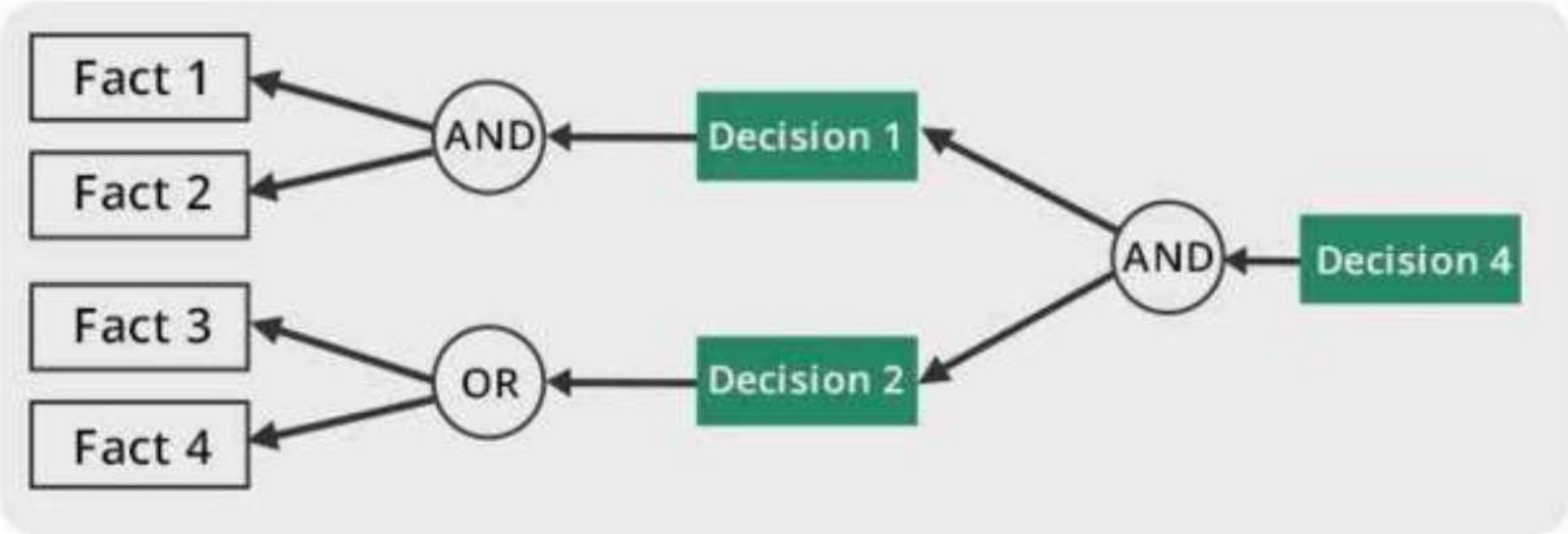
BACKWARD CHAINING

- Backward chaining is a goal driven method of deriving a particular goal from a given knowledge base and set of inference rules
- Inference rules are applied by matching the goal of the search to the consequents of the relations stored in the knowledge base
- In backward chaining, we start from a conclusion, which is the hypothesis we wish to prove, and we aim to show how that conclusion can be reached from the rules and facts in the database.
- The conclusion we are aiming to prove is called a goal, and so reasoning in this way is known as goal-driven reasoning

Backward Chaining

With this strategy, an expert system finds out the answer to the question, **"Why this happened?"**

On the basis of what has already happened, the interface engine tries to find out which conditions could have happened in the past for this result. This strategy is followed for finding out cause or reason. For example, diagnosis of blood cancer in humans.



- The benefit in this method is particularly clear in situations where the first state allows a very large number of possible actions.
- In this kind of situation, it can be very inefficient to attempt to formulate a plan using forward chaining because it involves examining every possible action, without paying any attention to which action might be the best one to lead to the goal state.
- Backward chaining ensures that each action that is taken is one that will definitely lead to the goal, and in many cases this will make the planning process far more efficient

BACKWARD CHAINING EXAMPLE "WEATHER FORECAST SYSTEM"

Suppose we have developed the following rules for our weather forecasting system,

Rule I

If we suspect temperature is less than 20^{o}
AND there is humidity in the air
Then there are chances of rain

Rule II

If Sun is behind the clouds
AND air is very cool.
Then we suspect temperature is less than 20^{o}.

Rule III **If** air is very heavy
Then there is humidity in the air.

- Suppose we have been given the following facts,
  a) Sun is behind the clouds.
  b) Air is very heavy and cool.
- Problem: Using Backward chaining try to conclude that there are chances of rain.

Step	Description	Working Memory
1	Goal "There are chances of rain." Not in Working Memory.	
2	Find rules with our goal "There are chances of rain" in conclusion: It is in Rule 1.	
3	Now see if Rule 1, premise 1 is known "we suspect temperature is less than 20^0".	
4	This is conclusion of rule 2. So going to Rule 2. The premise 1 of rule 2 is "Sun is behind the clouds".	
5	This is primitive. We ask from user Response: Yes	Sun is behind the clouds.

6	See if Rule 2, premise 2 is known "Air is very cool".	
7	This is also primitive. We ask its Response: Yes. Both conditions of Rule 2 are met so Fire rule 2	Sun is behind the clouds. Air is very cool. **We suspect temperature is less than 20^0.**
8	So Rule 1 premise 1 is in working memory, coming to Rule 1, premise 2 "There is humidity in the air"	Sun is behind the clouds. Air is very cool. **We suspect temperature is less than 20^0.**
9	This is conclusion of Rule 3. So see if Rule 3, premise 1 is known "Air is very heavy".	Sun is behind the clouds. Air is very cool. **We suspect temperature is less than 20^0.**

10	This is primitive so asking from user Response: Yes. Fire rule	Sun is behind the clouds. Air is very cool. We suspect temperature is less than 20^0. **There is humidity in the air.**
11	Now Rule 1 premise 1 and 2 both are in working memory so fire Rule 1.	Sun is behind the clouds. Air is very cool. Air is very heavy. We suspect temperature is less than 20^0. There is humidity in the air. **There are chances of rain.**

Forward-chaining	Backward-chaining
Starts with the initial facts.	Starts with some hypothesis or goal.
Asks many questions.	Asks few questions.
Tests all the rules.	Tests some rules.
Slow, because it tests all the rules.	Fast, because it tests fewer rules.
Provides a huge amount of information from just a small amount of data.	Provides a small amount of information from just a small amount of data.
Attempts to infer everything possible from the available information.	Searches only that part of the knowledge base that is relevant to the current problem.
Primarily data-driven	Goal-driven
Uses input; searches rules for answer	Begins with a hypothesis; seeks information until the hypothesis is accepted or rejected.
Top-down reasoning	Bottom-up reasoning
Works forward to find conclusions from facts	Works backward to find facts that support the hypothesis
Tends to be breadth-first	Tends to be depth-first
Suitable for problems that start from data collection, e.g. planning, monitoring, control	Suitable for problems that start from a hypothesis, e.g. diagnosis
Non-focused because it infers all conclusions, may answer unrelated questions	Focused; questions all focused to prove the goal and search as only the part of KB that is related to the problem
Explanation not facilitated	Explanation facilitated
All data is available	Data must be acquired interactively (i.e. on demand)
A small number of initial states but a high number of conclusions	A small number of initial goals and a large number of rules match the facts
Forming a goal is difficult	Easy to form a goal