Knowledgebase Representation

Production Systems

Production systems

- The human mental process is internal, and it is too complex to be represented as an algorithm
- However, most experts are capable of expressing their knowledge in the form of rules for problem solving.
- ***Production systems*** (also known as Rule-based systems) are the simplest form of artificial intelligence.
- The term rule in AI, can be defined as an IF-THEN structure that relates given information or facts in the IF part to some action in the THEN part.
- A rule provides some description of how to solve a problem.
- Rules are relatively easy to create and understand.

Production systems

- Any rules consists of two parts: the IF part, called the antecedent (premise or condition) and the THEN part called the consequent (conclusion or action)

  IF antecedent

  THEN consequent.

- ***Example*:**

  IF the season is winter

  THEN it is cold.

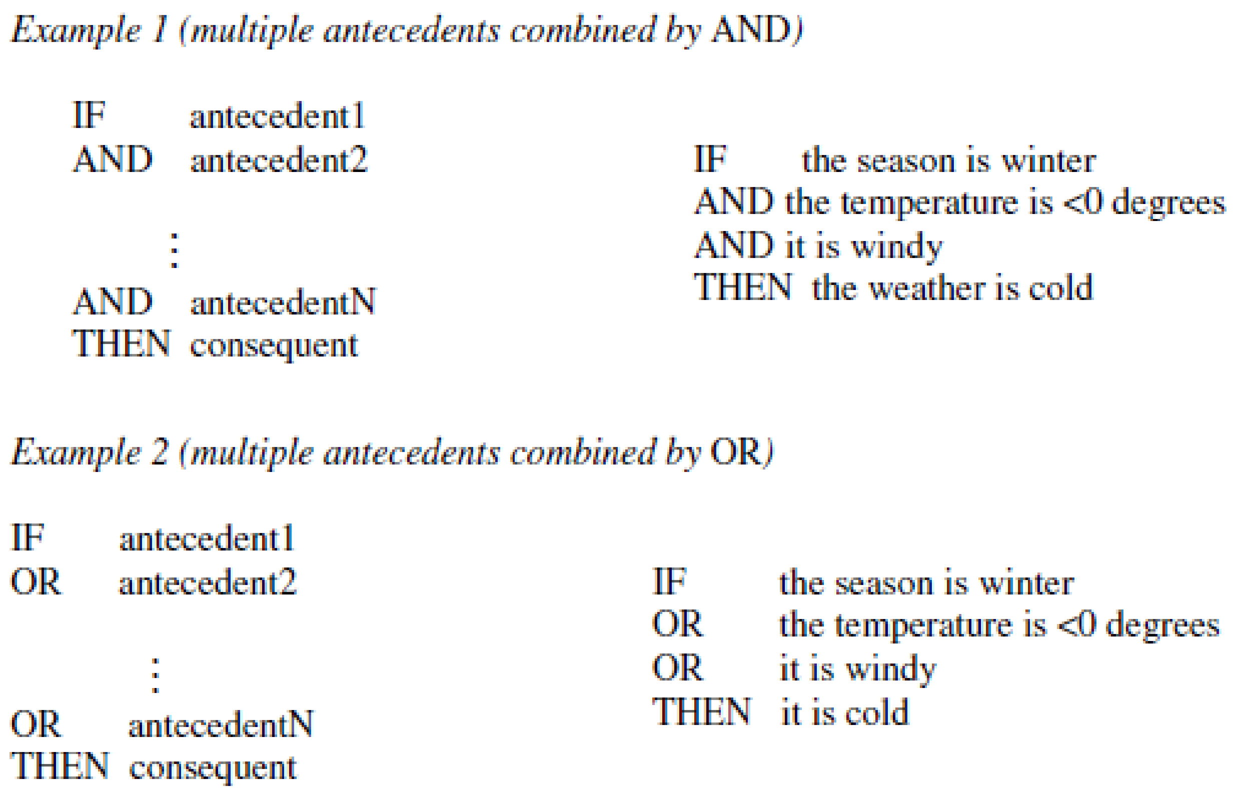
Rules

- *The antecedent of a rule has two parts:*
- *object (also called a linguistic object);*
- *value of the linguistic object.*
- The operator identifies the object and assigns the value. Operators (such as: is, are, is not, are not) are used to assign a symbolic value to a linguistic object.
- Production systems can also used mathematical operators to define an object as numerical and assign it to the numerical value

```
IF          'age of the customer' < 18
AND         'cash withdrawal' > 1000
THEN            'signature of the parent' is required
```

Rules

- A rule can have multiple antecedents joined by the keywords AND (conjunction), OR (disjunction).



Example 1 (multiple antecedents combined by AND*)*

```
IF      antecedent1
AND     antecedent2
          ⋮
AND     antecedentN
THEN    consequent
```

```
IF      the season is winter
AND the temperature is <0 degrees
AND it is windy
THEN  the weather is cold
```

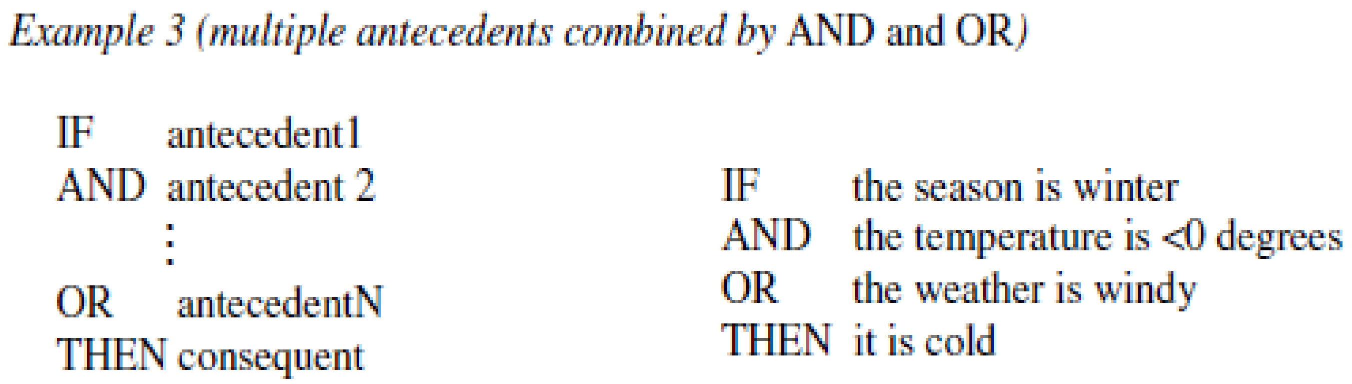
Example 2 (multiple antecedents combined by OR*)*

```
IF      antecedent1
OR      antecedent2
          ⋮
OR      antecedentN
THEN    consequent
```

```
IF      the season is winter
OR      the temperature is <0 degrees
OR      it is windy
THEN    it is cold
```

Rules

- A rule also can have a combination of both.



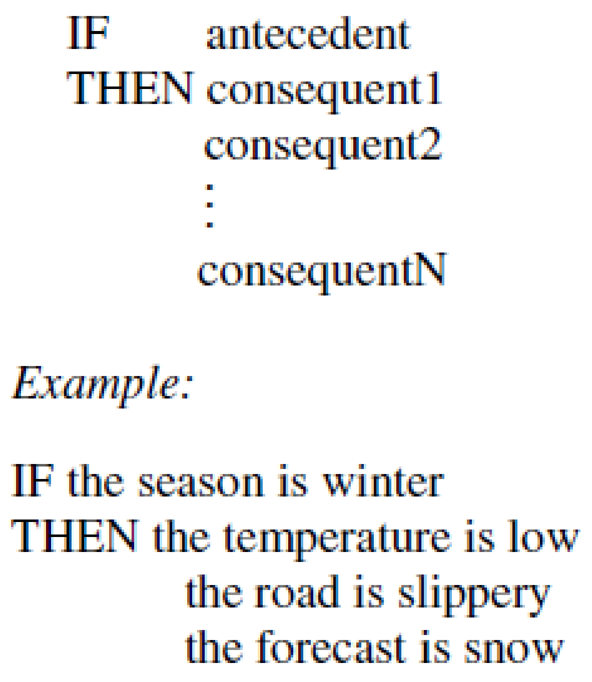
Example 3 (multiple antecedents combined by AND and OR*)*

```
IF    antecedent1
AND   antecedent 2
          ⋮
OR    antecedentN
THEN  consequent
```

```
IF    the season is winter
AND   the temperature is <0 degrees
OR    the weather is windy
THEN  it is cold
```

Rules

- *The consequent can also have multiple clauses, for instance:*



IF antecedent
THEN consequent1
consequent2
⋮
consequentN

Example:

IF the season is winter
THEN the temperature is low
the road is slippery
the forecast is snow

Rules

Based on the conclusion or the consequent of a rule, rules can express:

- **Relation:**

  IF x >0

  THEN x is positive

- **Recommendation:**

  IF it is rainy

  THEN take an umbrella

- **Directive:**

  IF Phone battery signal

  AND phone battery empty

  THEN charge the phone

- **Heuristic:**

  IF phone light is off

  THEN battery is flat

production systems components

The production system consists of *production rules*, *working memory* and *recognize-act control*.

- Production rules: Condition part of the rule is used to determine when the rule may be applied, while the action parts defines the associated problem-solving step.
- *Working Memory*: It contains a description of the current state of the problem-solving.
- *Recognize-act control:* It implements search allowing the production system to move towards a goal within the set of rules.

Reasoning

- Reasoning is the way in which rules are combined to derive new knowledge.
- Reasoning is how humans work with knowledge, facts and problem-solving strategies to draw conclusions.
- There are two main ways in which rules are executed and this conducts to the existence of two main rule systems:
- *Forward chaining* systems: A forward chaining system starts with the initial facts and keep using the rules to draw new conclusions (or take certain actions) given those facts.
- *Backward chaining* systems: A backward chaining system starts with some hypothesis (or goal) to prove and keep looking for rules that would allow concluding that hypothesis, by setting new sub-goals to prove as the process advances.

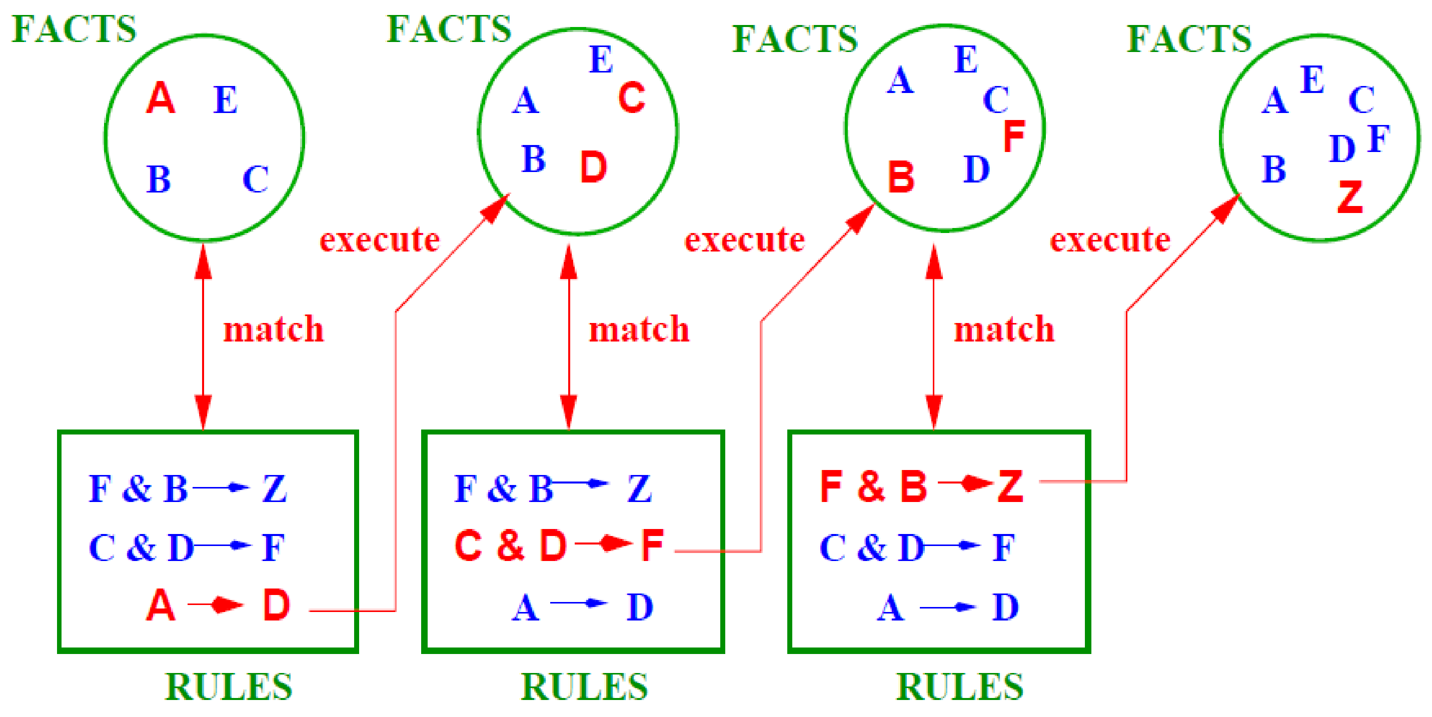
Forward Chaining Systems

- The forward chaining works as follow: given a certain set of facts in the working memory, use the rules to generate new facts until the desired goal is reached.

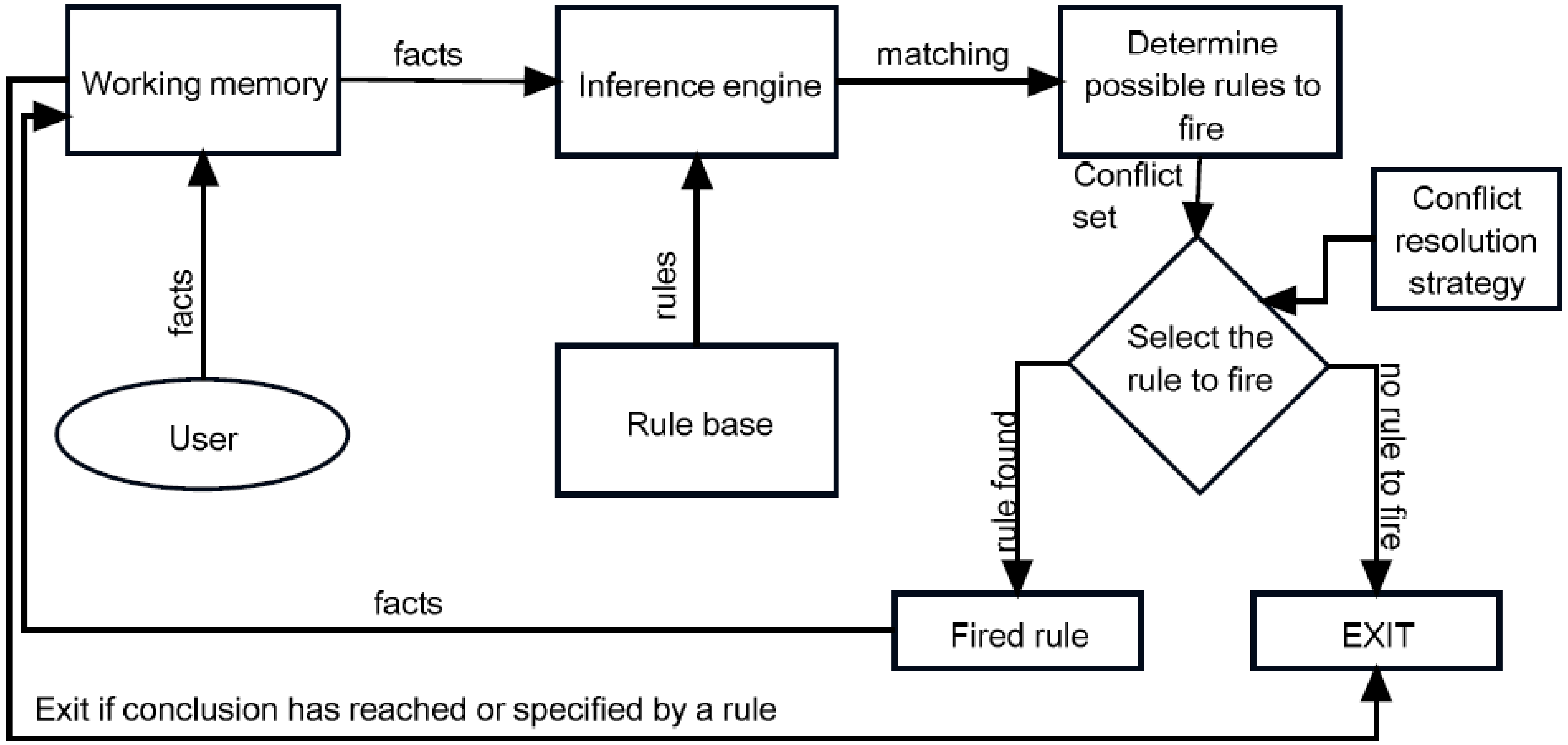
1. ***Match*** the IF part of each rule against facts in working memory.
2. If there is more than one rule that could be used (more than one rule which fires), *select* which one to apply by using **conflict resolution**
3. ***Apply* the rule**. If new facts are obtained add them to working memory.
4. *Stop* (or *exit*) when the conclusion is added to the working memory or if there is a rule which specifies to end the process.

Forward Chaining Systems

Forward chaining systems are primarily data-driven



Forward Chaining Systems



Forward chaining systems

Example

Consider the following expert systems whose database consists of the facts A, B, C, D, E and whose knowledge base is given by the rules below:

- Rule 1: **IF** A is true AND C is true **THEN** B is true
- Rule 2: **IF** C is true AND D is true **THEN** F is true
- Rule 3: **IF** C is true AND D is true AND E is true **THEN** X is true
- Rule 4: **IF** A is true AND B is true AND X is true **THEN** Y is true
- Rule 5: **IF** D is true AND Y is true **THEN** Z is true

Example

- The database consisting of **facts** A, B, C, D and E and with the knowledge base consisting of the 5 given rules. We will now show how forward chaining can be applied to this system to reach the **conclusion Z**.
- If multiple rules can fire at a time, then we will choose to fire the first of the rules, which was not fired before.
- Let us now follow step by step the forward chaining as presented above.

Cycle 1

1. Match the IF part of each rule against facts in working memory. The following rules can be selected:

- Rule 1: IF A AND C THEN B (since both A and C are in the database);
- Rule 2: IF C AND D THEN F (since both C and D are in the database);
- Rule 3: IF C AND D AND E THEN X (since all C, D and E are in the database)
- Rules 4 and 5 cannot be selected because their IF part cannot be matched (X in the case of Rule 4 and Y in the case of Rule 5 respectively are not in the database at this moment).

2. *T*here are 3 rules that can be used: Rule 1, Rule 2 and Rule 3. And we will always select the first one, which can be applied and was not applied earlier. Thus, Rule 1 will be the rule fired first.

Cycle 1

3. Apply the rule. If new facts are obtained add them to working memory. At this step we are applying the Rule 1:

- Rule 1: IF A AND C THEN B
- The consequent of this rule is B is true. But B is already in the database, so no new facts are obtained by applying this rule.

4. Stop (or exit) when the conclusion is added to the working memory or if there is a rule which specifies to end the process.

- Our conclusion is Z and was not reached yet, so we will go again to the first step.

Cycle 2

1. At this time, the rules which can be selected are the same: Rule 1, Rule 2 and Rule 3.

2. Since Rule 1 already fired, the next selected rule is Rule 2.

3. Rule 2 is fired: Rule 2: IF C AND D THEN F.

- A new fact if obtained, F, which is not already in the database, so F will be added to the database which is now: A, B, C, D, E, F.

4. Conclusion Z was not reached yet, so we will start the process again.

Cycle 3

1. Rules which can fire now are still the same: Rule 1, Rule 2, Rule 3.

2. Since rules Rule 1 and Rule 2 have been already used, the only remaining one to be selected is Rule 3:

- Rule 3: **IF** C is true AND D is true AND E is true **THEN** X is true

3. Rule 3 is fired, and a new fact is obtained, X, which is not in the database, so will be added.

4. X is not the conclusion, so we should still restart the process.

Cycle 4

1. The database contains now the facts A, B, C, D, E, F, and X. Thus, the first 4 rules can be matched at this point.

2. Since the first 3 rules have been already used, the only one which can be selected to fire is Rule 4:

- Rule 4: IF A AND B AND X THEN Y.

3. Rule 4 is fired and a new fact, Y, is added to the database.

4. Still, Y is not the conclusion, so we have to continue.

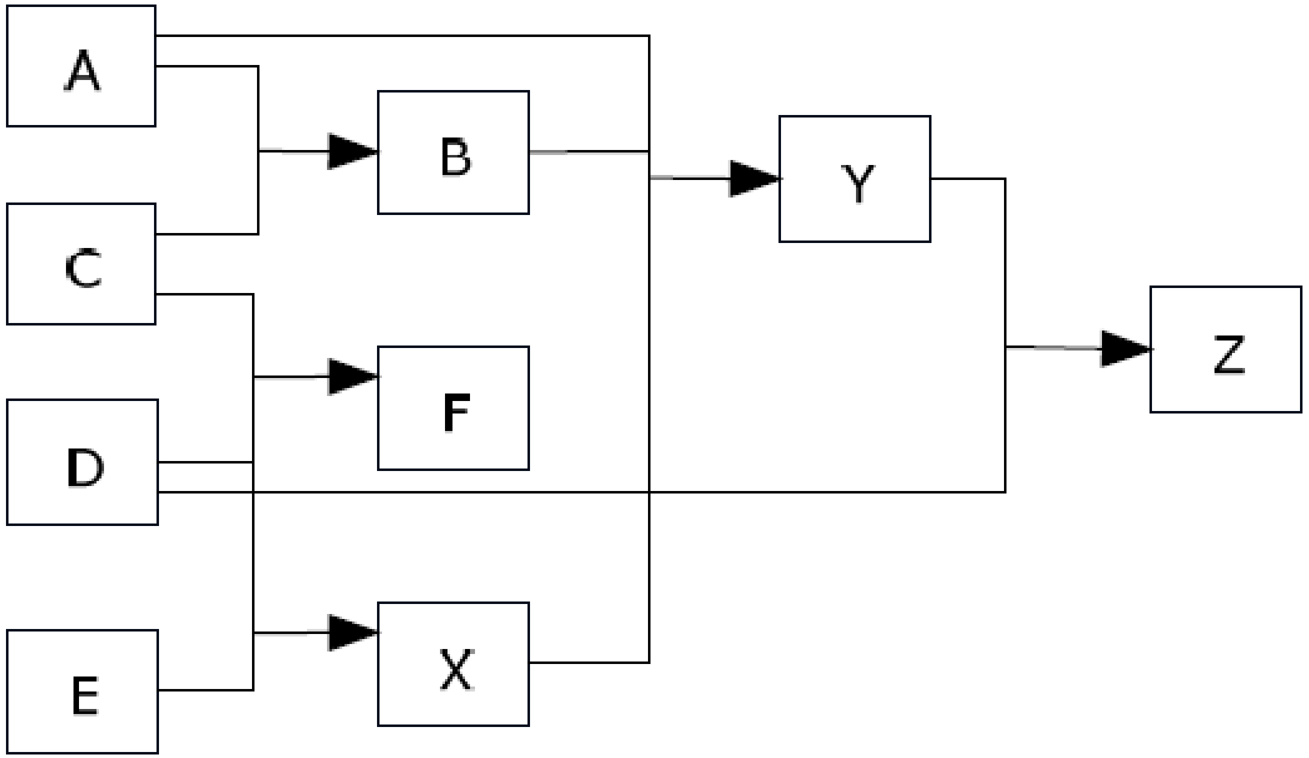
Cycle 5

1. All the 5 rules match the IF condition.
2. The only remaining rule to use is Rule 5:

- Rule 5: IF D AND Y THEN Z.

3. Rule 5 is fired and a new fact, Z, is obtained.
4. Z represents our conclusion so the process may stop here.

An inference chain example



Backward Chaining Systems

- The backward chaining systems work backwards from a hypothesized goal, attempting to prove it by linking the goal to the initial facts. To backward chain from a goal in the working memory the inference engine must follow the steps:

1. Select rules with conclusions matching the goal.

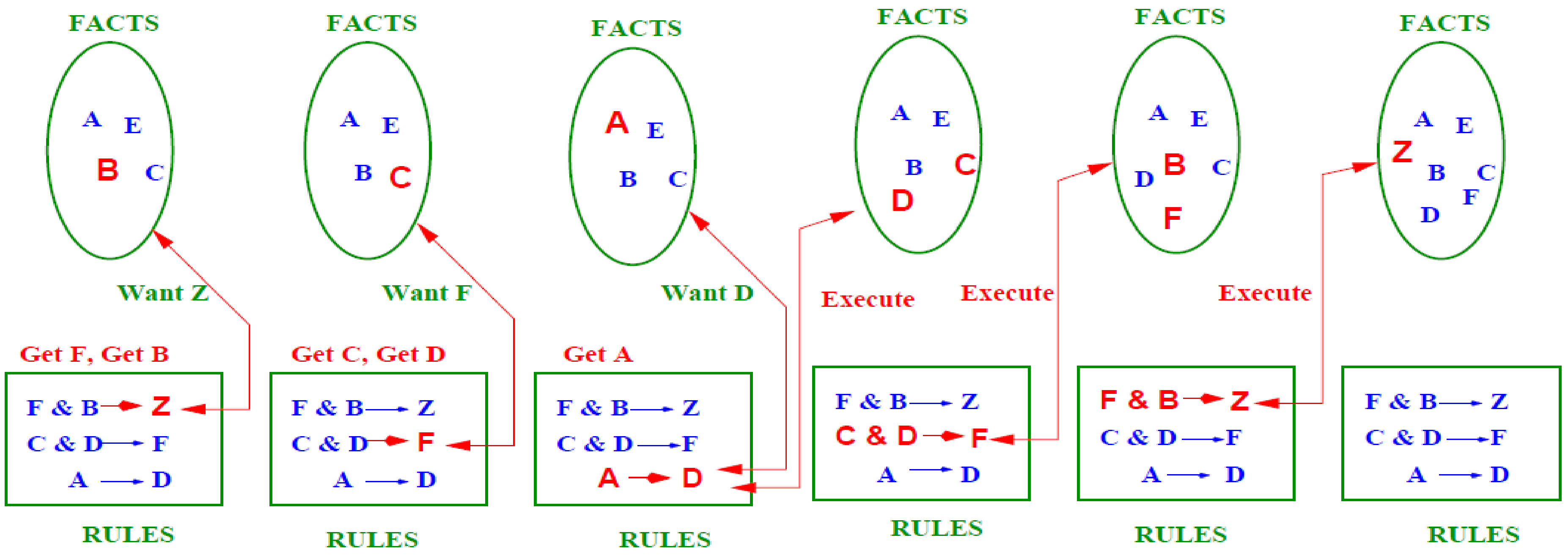
2. Replace the goal by the rule's premises. These become sub-goals.

3. Work backwards until all sub-goals are known to be true. This can be achieved either:

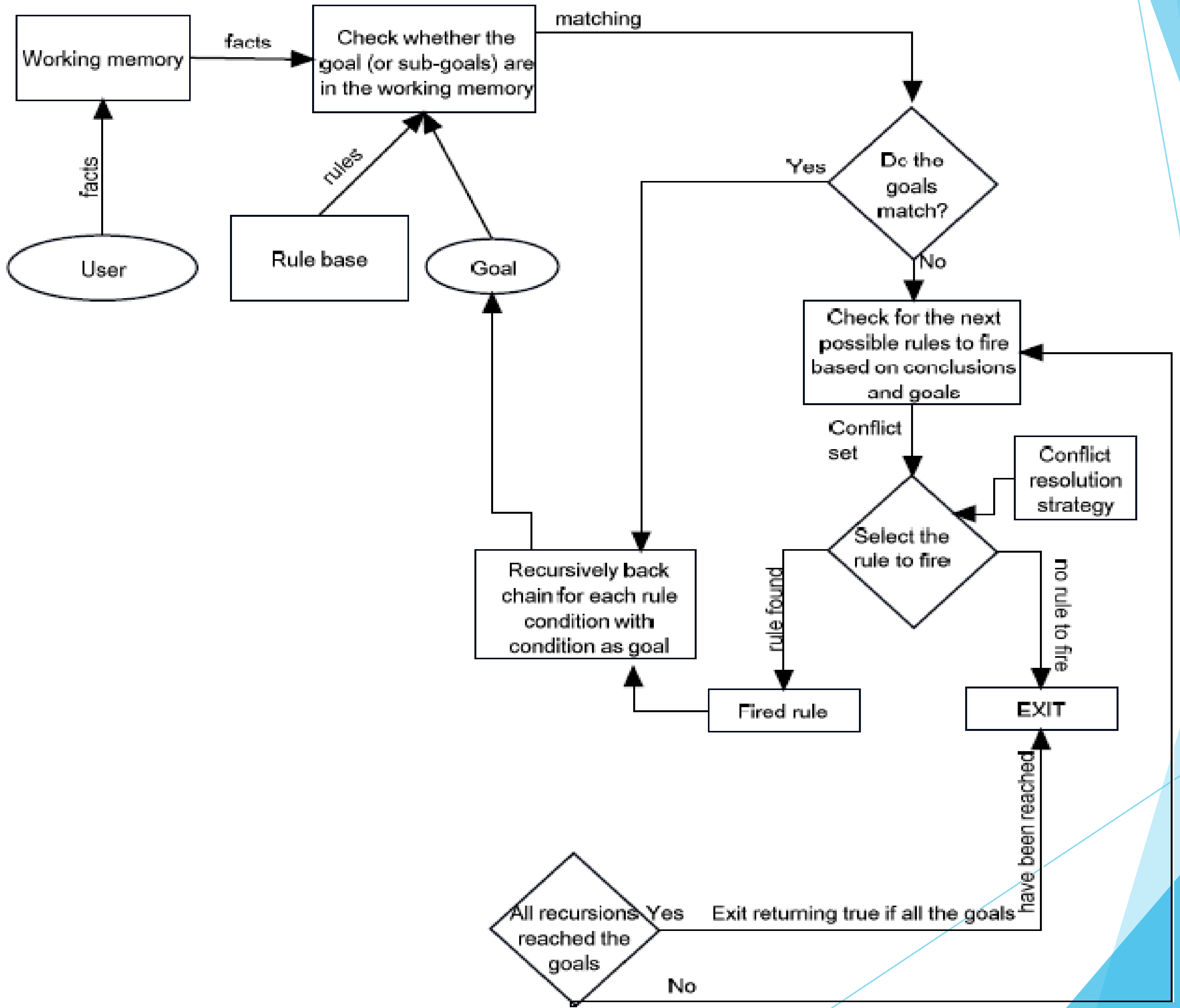
- They are facts (in the working memory) or
- They can be conducted from other rules.

Backward Chaining Systems

Backward chaining systems are goal-driven inference.



Backward Chaining Systems


Working memory
facts
Check whether the goal (or sub-goals) are in the working memory
matching
facts
rules
User
Rule base
Goal
Yes
Do the goals match?
No
Check for the next possible rules to fire based on conclusions and goals
Conflict set
Conflict resolution strategy
Select the rule to fire
rule found
no rule to fire
Recursively back chain for each rule condition with condition as goal
Fired rule
EXIT
have been reached
All recursions reached the goals
Yes
Exit returning true if all the goals
No

Example

Let us now consider the same example as for forward chaining: the database consists of the facts A, B, C, D, E and whose knowledge base is given by the rules below:

- Rule 1: **IF** A is true AND C is true **THEN** B is true
- Rule 2: **IF** C is true AND D is true **THEN** F is true
- Rule 3: **IF** C is true AND D is true AND E is true **THEN** X is true
- Rule 4: **IF** A is true AND B is true AND X is true **THEN** Y is true
- Rule 5: **IF** D is true AND Y is true **THEN** Z is true

Step1

1. Select rules with conclusions matching the goal.
- The only rule with conclusion matching the goal is Rule 5.

2. Replace the goal by the rule's premises. These become sub-goals.
- D is in the database but we don’t have Y. So, the first sub-goal is Y.

3. Work backwards until all sub-goals are known to be true.
- We don’t have all sub-goals as true, so we back-chain again.

Step2

1. Select rules with conclusions matching the goal.

- Our goal is Z but our sub-goal is Y. Rule 4 has Y as conclusion.

2. Replace the goal by the rule's premises. These become sub-goals.

- Among the Rule’s 4 premises we have A and B in the database but we don’t have X. Thus, our current sub-goal is X

3. Work backwards until all sub-goals are known to be true.

- We don’t have all sub-goals as true, so we back-chain again.

Step 3

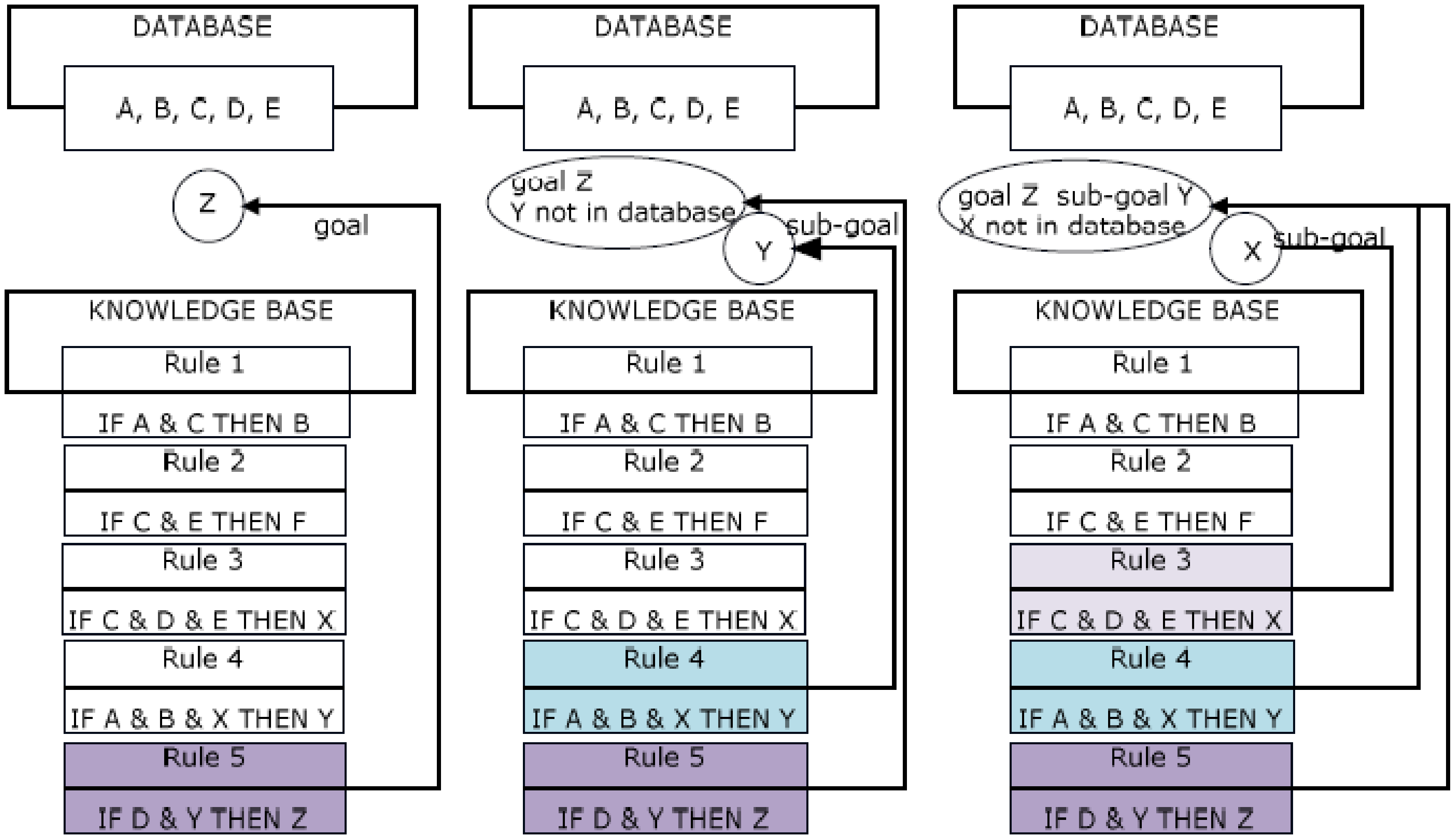
1. Select rules with conclusions matching the goal. Our goal is Z but our sub-goals are Y and X. Most recent one is X. Rule 3 has X as conclusion.

2. Replace the goal by the rule's premises. These become sub-goals. Rule’s 3 premises are C, D and E and we can find all of them in the database. Thus, our sub-goal X can be obtained by first firing Rule 3.

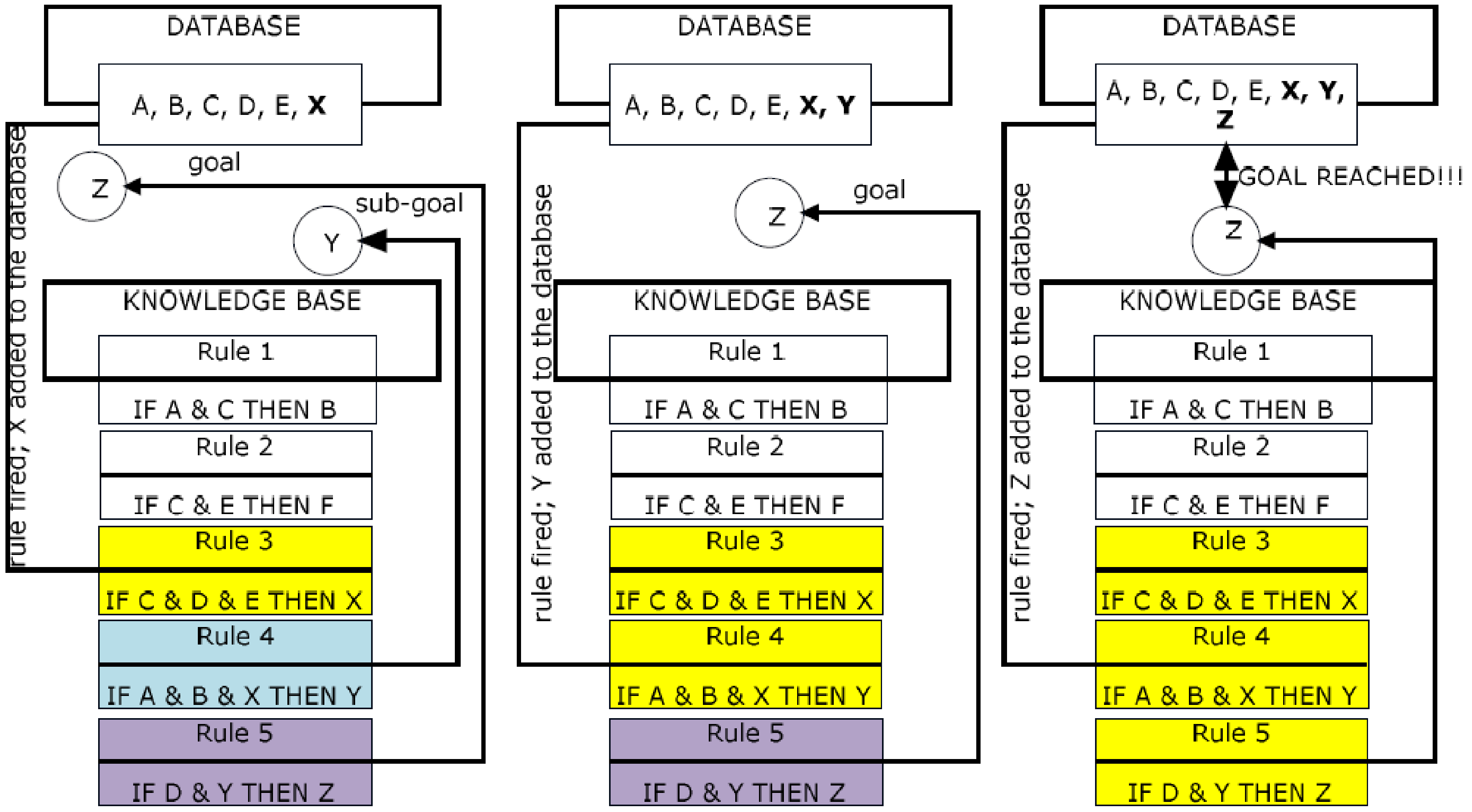
3. Work backwards until all sub-goals are known to be true. We obtained one of the sub-goals. We don’t have all sub-goals as true, so we recursively back-chain to obtain the other sub-goals. First on the agenda is Y.

- Going back to Step 2, we now have all the premises for Rule 4. Thus, rule 4 can fire and Y is obtained and added to the database.
- Going back to Step 1, we now have all the premises for Rule 5 whose conclusion is our desired goal. Thus, we can fire Rule 5 and obtain the goal Z.

Backward Chaining Example



Backward Chaining Example



Example 2

Consider the following expert systems whose database consists of the facts

h1, h2 and whose knowledge base is given by the rules below:

- R1: **IF h** is true **THEN Friend1(X)** is true
- R2: **IF a** is true AND **b** is true **THEN Friend1(X)** is true
- R3: **IF c(X)** is true **THEN Friend2(X)** is true
- R4: **IF not(d)** is true **THEN a** is true
- R5: **IF d** is true **THEN b** is true
- R6: **IF e** is true **THEN b** is true
- R7: **IF not(e)** is true **THEN c(2)** is true
- R8: **IF h1** is true AND **h2** is true **THEN d** is true
- R9: **IF h2** is true AND **h3** is true **THEN e** is true

Prove or disprove the goal friend1(X) & friend2(X) by using backward and forward chaining.

Solution – Backward chaining

Start with the first goal friend1(X):

Matching friend1(X):

R1: **IF h** is true **THEN Friend1(X)** is true (false)

R2: **IF a** is true AND **b** is true **THEN Friend1(X)** is true

Matching a :

R4: **IF not(d)** is true **THEN a** is true

Matching d

R8: **IF h1** is true AND **h2** is true **THEN d** is true

h1 is true and h2 is true then d is true then R8 is true

Return to R4:

R4: **IF not(d)** is true **THEN a** is true

d is true then not(d) become false so a is false then R4 is false

Return to R2:

a is false then Friend1(X) is false.

Solution – Backward chaining

The second goal friend2(X):

Matching friend2(X):

R3: **IF c(X)** is true **THEN Friend2(X)** is true

    Matching c(X) :

    R7: **IF not(e)** is true **THEN c(2)** is true

        Matching e

        R9: **IF h2** is true AND **h3** is true **THEN e** is true

        h2 is true and h3 is false then e is false then R9 is false

    Return to R7:

    R7: **IF not(e)** is true **THEN c(2)** is true

    e is false, then not(e) become true

    So, c(2) is true then R7 is true

Return to R3:

R3: **IF c(X)** is true **THEN Friend2(X)** is true

c(X) is true for X=2

then Friend2(X) is true where X=2.

Solution – Forward chaining

Working memory: h1, h2

1. Matching h1,h2:

R8: **IF h1** is true AND **h2** is true **THEN d** is true

Working memory: h1, h2, d

2. Matching h1,h2, d:

R5: **IF d** is true **THEN b** is true

Working memory: h1, h2, d, b

3. No rules can be fired now (R8,R5 fired before) so Neglect NOT

add not(e) to the working memory because not(e) is true (doesn’t exist in memory)

Working Memory : h1,h2,d,b, not(e)

R7: **IF not(e)** is true **THEN c(2)** is true

Working Memory : h1,h2,d,b, not(e) and c(2)

Solution – Forward chaining

4. Matching h1,h2,d,b, not(e) and c(2)

R3: **IF c(X)** is true **THEN Friend2(X)** is true

C(2) is true so Friend2(x) will be added to the memory with x=2

Forward Chaining or Backward Chaining? Which One Should Apply?

- **Forward chaining is generally suggested if:**

• All or most of the data is given in the problem statement.

• There exist many potential goals but only a few of them are achievable in a particular problem instance.

• It is difficult to formulate a goal or hypothesis.

- **Backward chaining is suggested** in the flowing situations:

• A goal or hypothesis is given in the problem statement or can be easily formulated.

• There are many rules that match the facts, producing a large number of conclusions - choosing a goal prunes the search space.

• Problem data are not given (or easily available) but must be acquired as necessary (in certain systems).

Forward Chaining versus Backward Chaining

- **Advantages of forward chaining**

Simple

- **Disadvantages of forward chaining**

1. Many rules may be applicable at each stage. How should we choose which one to apply next at each stage?
2. The whole process is not directed towards a goal. How do we know when to stop applying the rules?

- **Advantages of Backward chaining**

The search is goal directed, so we only apply the rules that are necessary to achieve the goal.

- **Disadvantages of Backward chaining**

A goal has to be known.

Conflict Resolution

- Let us consider three simple rules for crossing a road.:

  Rule 1: IF the ‘traffic light’ is green THEN the action is go

  Rule 2: IF the ‘traffic light’ is red THEN the action is stop

  Rule 3: IF the ‘traffic light’ is red THEN the action is go
- We have two rules, Rule 2 and Rule 3, with the same IF part.
- Both of them can be set to fire when the condition part is satisfied.
- These rules represent a conflict set
- The inference engine must determine which rule to fire from such a set.
- A method for choosing a rule to fire when more than one rule can be fired in a given cycle is called conflict resolution

Conflict Resolution : strategies

- There are several different strategies such as :

1. *First applicable:* If the rules are in a specified order, firing the first applicable one is the easiest way to control the order in which rules fire.

  - Suppose we have the following rules in the knowledge base:

Rule 1: IF color is yellow THEN fruit is apple;

Rule 2: IF color is yellow AND shape is long THEN fruit is banana

Rule 3: IF shape is round THEN fruit is apple.

  - And the database consisting of **yellow (color)** and **round (shape).**
  - Then two rules can fire: Rule 1 and Rule 3 and the order will be Rule 1 first and then Rule 3.

Conflict Resolution : strategies

2. Most Specific: This strategy is based on the number of conditions of the rules.

- From the conflict set, the rule with the most conditions is chosen. This is based on the assumption that a specific rule process more information than a general one.

- Rule 1: IF the weather is cold THEN the season is winter
- Rule 2: IF the weather is cold AND the temperature is low AND the wind is blushing AND the forecast is snow THEN the season is winter.
- **Among the two rules, most significant one is Rule 2. Thus, Rule 2 is selected if the most specific strategy is used.**

Conflict Resolution : strategies

3. Least Recently Used: Each of the rules has a time or step associated, which marks the last time it was used. This maximizes the number of individual rules that are fired at least once. If all rules are needed for the solution of a given problem, this is a perfect strategy.

- Rule 1: IF color is yellow [28.02.2020, 13:45]

  THEN fruit is apple;
- Rule 2: IF color is yellow [01.03.2020, 12:00]

  AND shape is long

  THEN fruit is banana
- Rule 3: IF shape is round [05.03.2020, 20:00]

  THEN fruit is apple.
- **In this example, the Rule 1 is the least recently introduces, thus this will be the one selected to fire.**

Conflict Resolution : strategies

4.Highest priority: In the case of this strategy, each rule is given a ‘weight,’ which specifies how much it should be considered over the alternatives. The rule with the most preferable outcomes is chosen based on this weight.

- Rule 1: IF color is yellow 30%

  THEN fruit is apple;
- Rule 2: IF color is yellow 30%

  AND shape is long

  THEN fruit is banana
- Rule 3: IF shape is round 40%

  THEN fruit is apple.
- **In this example Rule 3 will be selected because it is having the highest weight among all the three rules.**

Conflict Resolution : strategies

5. Most recently entered : Fire the rule that uses the data most recently entered in the database.

This method relies on time tags attached to each fact in the database. In the conflict set, the expert system first fires the rule whose antecedent uses the data most recently added to the database.