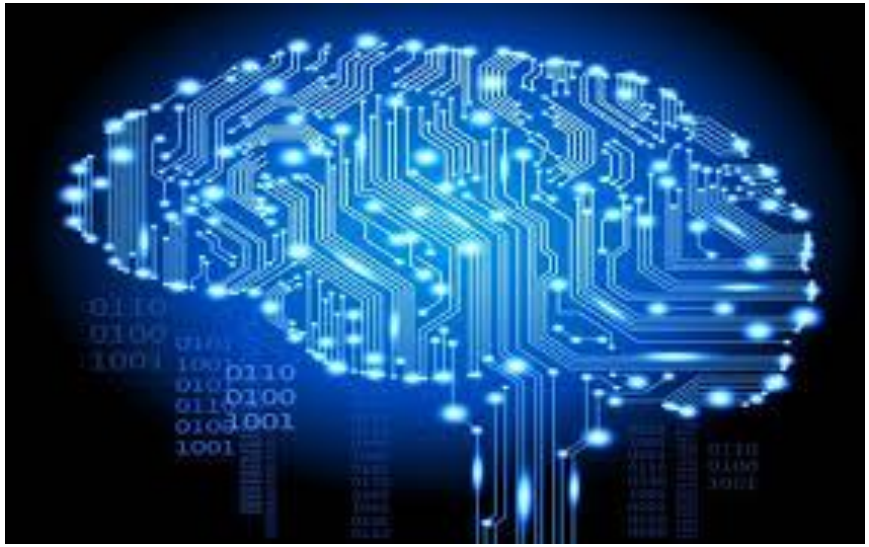
Knowledge Representation & Reasoning (KR&R)

Mahit Kumar Paul
Assistant Professor, Dept. of CSE
RUET, Rajshahi-6204
mahit.cse@gmail.com

What is Knowledge

Knowledge is defined as the body of facts and principles accumulated by human-kind or the acts, fact, or state of knowing. [This definition is far from complete]

Reference(P2-P5**):** AI and Expert Systems
by Dan. W. Patterson



What is Knowledge…

- Let we say something like "**John knows that … …**" we fill in the blank with a simple
  - ✓ "John knows that *Mary will come to the party*,"
  - ✓ "John knows that *Spain won the Euro*"

knowledge is a relation between a knower and a *proposition*

Knower : John

Proposition: *Mary will come to the party*, *Spain won the Euro*

Propositions are abstract entities that can be *true* or *false*, *right* or *wrong*.

What is Knowledge…

- More elaborately, knowledge is closely related to

  -Understanding

  -Learning

  -Thinking

  -Remembering

  -Reasoning

Types of Knowledge

- Declarative knowledge – Passive knowledge expressed as statements of fact about world.

  Example: personnel data in database, color of flowers.

- Procedural knowledge - Compiled knowledge related to the performance of some task

  Example: riding bicycle, solution of algebraic problem.

- Heuristic knowledge - Special type of knowledge used by human to solve complex problem. A rules of thumb is used to make good judgment in it.

  Example: Chess game

- Epistemology knowledge - Study of the nature of knowledge.
- Meta knowledge - Knowledge about knowledge.

  Example: knowledge about how an expert system makes decision.

Representation

- **Representation** is a relationship between two domains, where the first is meant to "stand for" or take the place of the second.
- Usually, the first domain, the representor, is more concrete, immediate, or accessible in some way than the second.

Example: *If* the robot turns left *Then* it will face obstacles

Knowledge representation, is the field of study concerned with using formal symbols to represent a collection of propositions believed by some agent.

Reasoning

In general, it is the formal manipulation of the symbols representing a collection of believed propositions to produce representations of new ones.

If it rains Then weather will be cool.

If weather is cool Then I will have sleep.

Now,

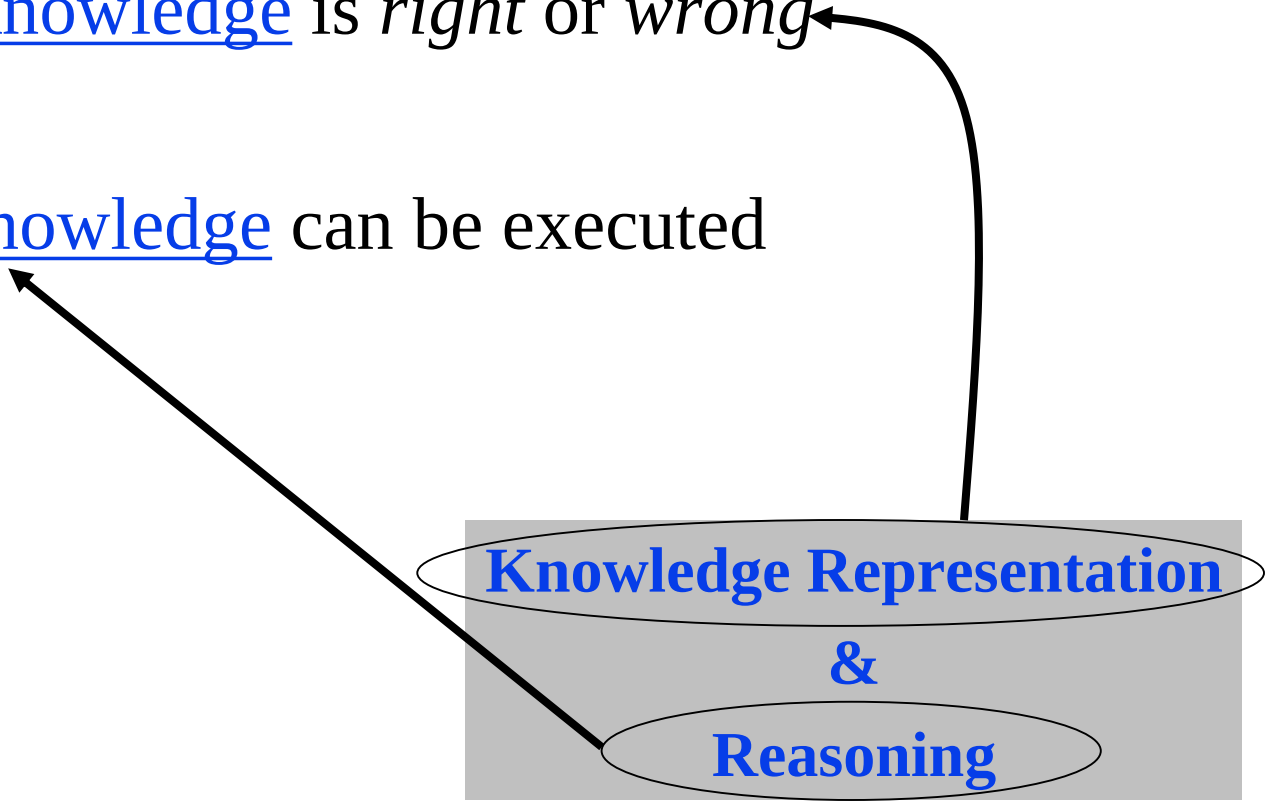
If it rains Then I will have sleep.

We would call this form of reasoning ***logical inference*** because the final sentence represents a logical conclusion of the propositions represented by the initial ones

Reasoning is a form of calculation, not unlike arithmetic, but over symbols standing for propositions rather than numbers.

KR&R

- Declarative knowledge is *right* or *wrong*
- Procedural knowledge can be executed



Knowledge Representation

- Knowledge is represented in a computer in the form of rules. Consists of an **IF part and THEN part**.
- IF part lists a set of conditions in some logical combination.
- If the IF part of the rule is satisfied; consequently, the THEN part can be concluded.
- Chaining of IF-THEN rules to form a line of reasoning.
- **Forward chaining** (facts driven)
- **Backward chaining** (goal driven)

Examples

- IF tomorrow is exam THEN study today
- IF robot turns left THEN it will face obstacles
- IF cgpa $\geq$ 3.00 THEN it is first class.

Knowledge Base (KB) [1,2]

- A **knowledge base** is a special kind of database for knowledge management. It provides the means for the computerized collection, organization, and retrieval of knowledge.
- **Within the knowledge base,** the programmer expresses information about a problem to be solved.
- Often this information is declarative, i.e. the programmer states some **facts, rules,** or **relationships** without having to be concerned with the detail of how and when that information should be applied.

Knowledge Base (KB)…

- These latter details are determined by the **inference engine**, which uses the knowledge base as a data file.

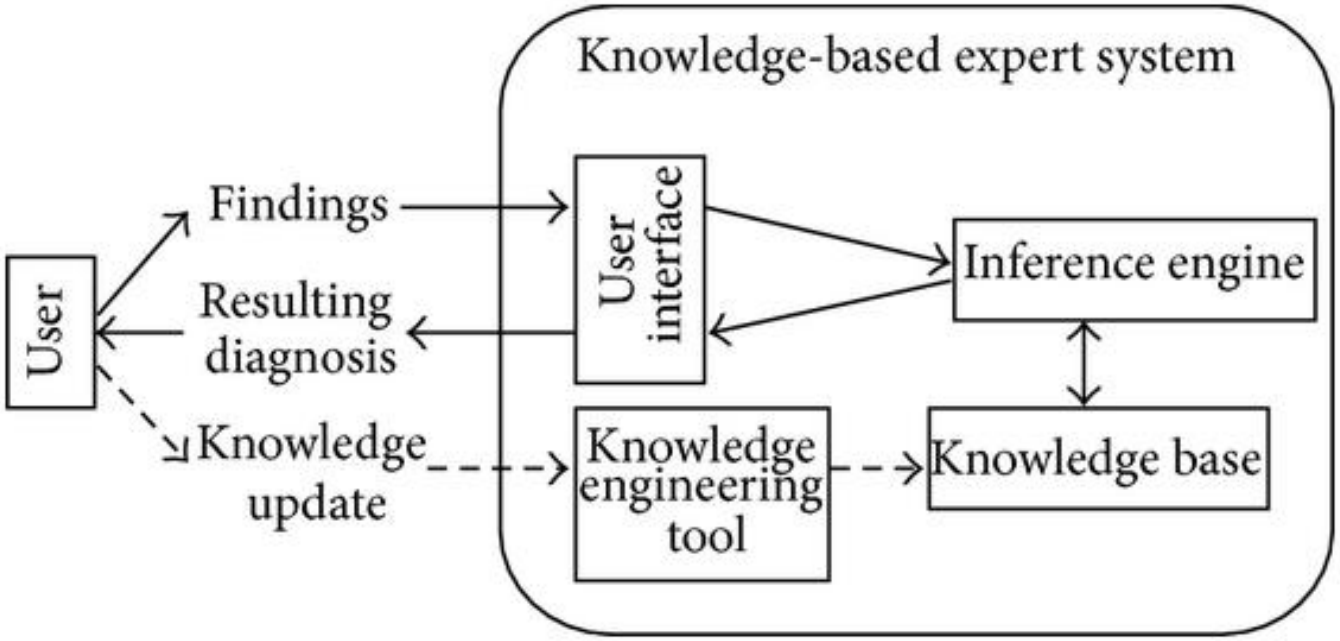


Figure : Typical structure of a knowledge-based expert system [3].

Working Memory [2,4]

- The working memory of a rule-based system is a store of information used by the system to decide which of the condition-action rules is able to be fired.
- The contents of the working memory when the system was started up would normally include the input data - e.g. the patient's symptoms and signs in the case of a medical diagnosis system. Subsequently, the working memory might be used to store intermediate conclusions and any other information inferred by the system from the data using the condition-action rules.
- Working memory is important for reasoning and the guidance of decision-making and behavior.

Inference Engine [2]

- An inference engine **tries to derive answers** from a **knowledge base.**
- A **kind of program** to manipulate the rules - for example to decide which ones are **ready to fire** i.e. which ones have conditions that match the contents of working memory. The program that does this is called an inference engine, because in many rule-based systems, the task of the system is to infer something, e.g. a diagnosis, from the data using the rules.
- An **inference engine** is a computer program that applies artificial intelligence to try to obtain answers or responses to queries from a knowledge base.

Inference Engine…

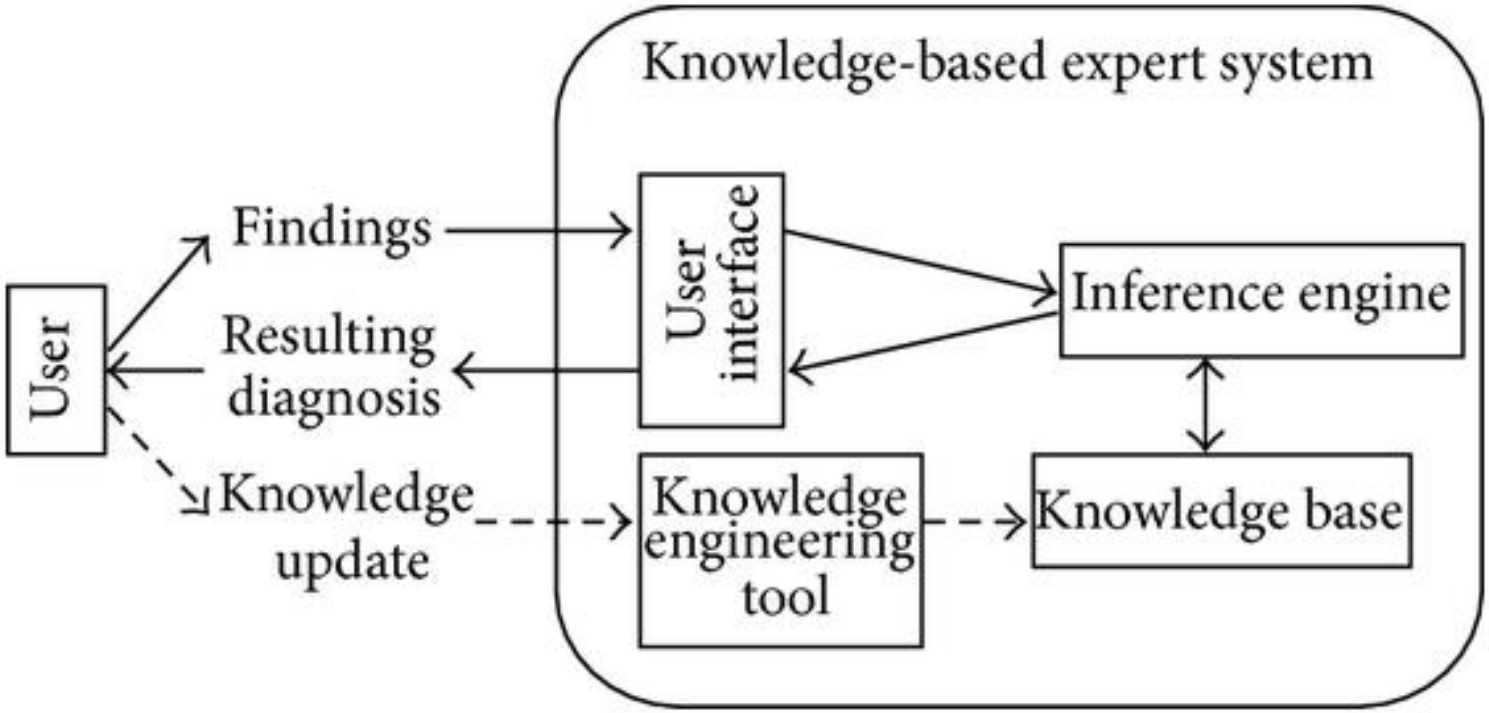


Figure : Typical structure of a knowledge-based expert system [3].

Match-Resolve-Act-Cycle[4]

The match-resolve-act cycle is what the inference engine does.

loop
match conditions of rules with contents of *working memory*
if no rule matches **then** stop
resolve *conflicts*
act (i.e. perform conclusion part of rule)
end loop

Forward & Backward Chaining[2,4]

The two main methods for reasoning when using an Inference Engine are:

- Forward chaining
- Backward chaining

Forward Chaining (FC)

- Forward chaining is a way of utilizing a set of **condition-action rules**. In this mode of operation, a rule-based system is data-driven.

- It starts with the available data and uses **inference rules** to extract more data until a **goal** is reached.

- An **inference engine** using forward chaining searches the inference rules until it finds one where the **antecedent** (**If** clause) is known to be true. When such a rule is found, the engine can conclude, or infer, the **consequent** (**Then** clause), resulting in the addition of new information to its data in **working memory**.

Forward Chaining…

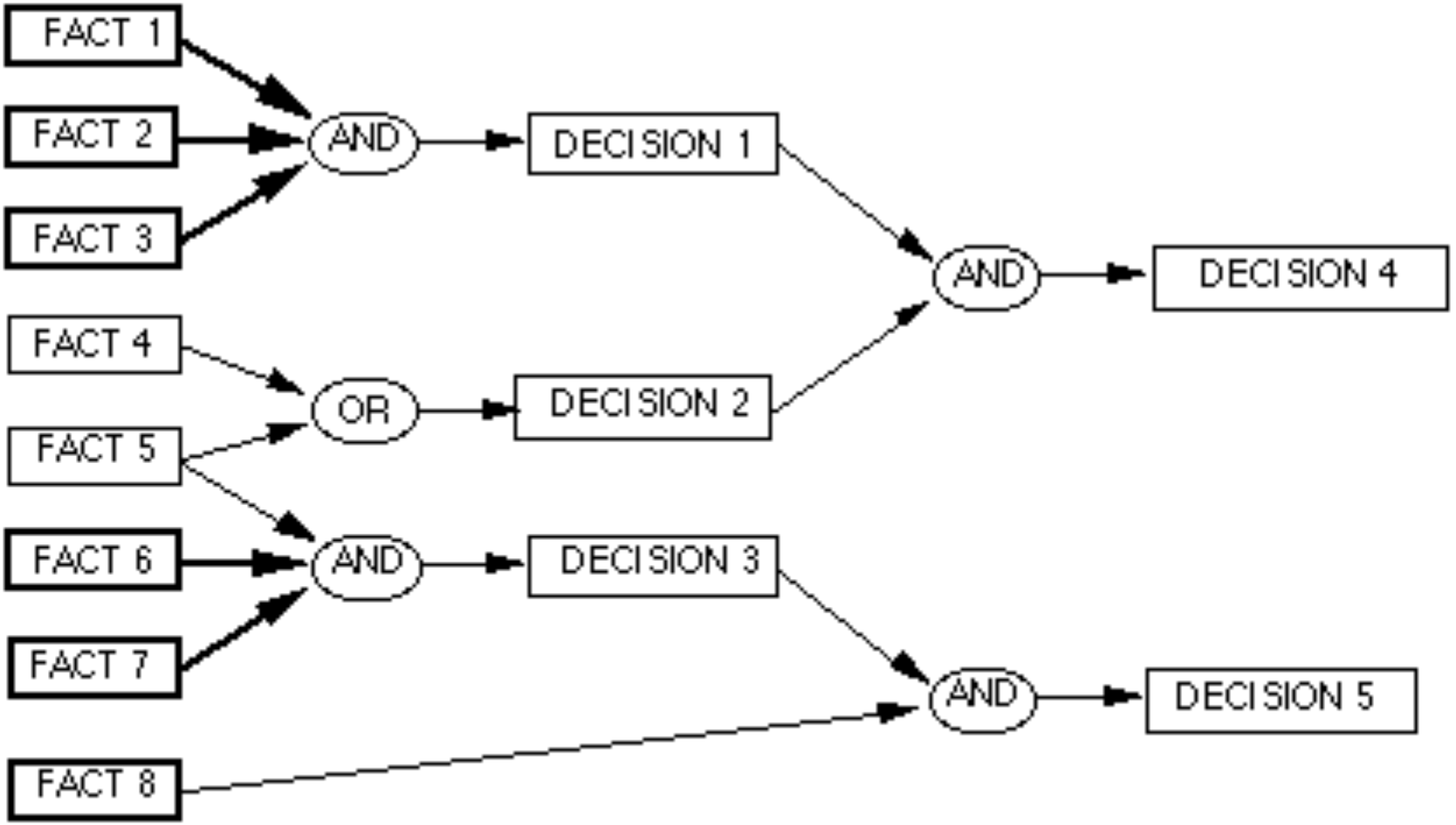


Fig : Forward chaining

Forward Chaining 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 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

Forward Chaining Example…

Let, assume the following facts:

1. Fritz croaks
2. 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

Backward Chaining (BC)

- Backward chaining starts with a list of goals (or a hypothesis) and works backwards from the consequent to the antecedent to see if there is data available that will support any of these consequents.

- An inference engine using backward chaining would search the inference rules until it finds one which has a consequent (**Then** clause) that matches a desired goal.

Backward Chaining…

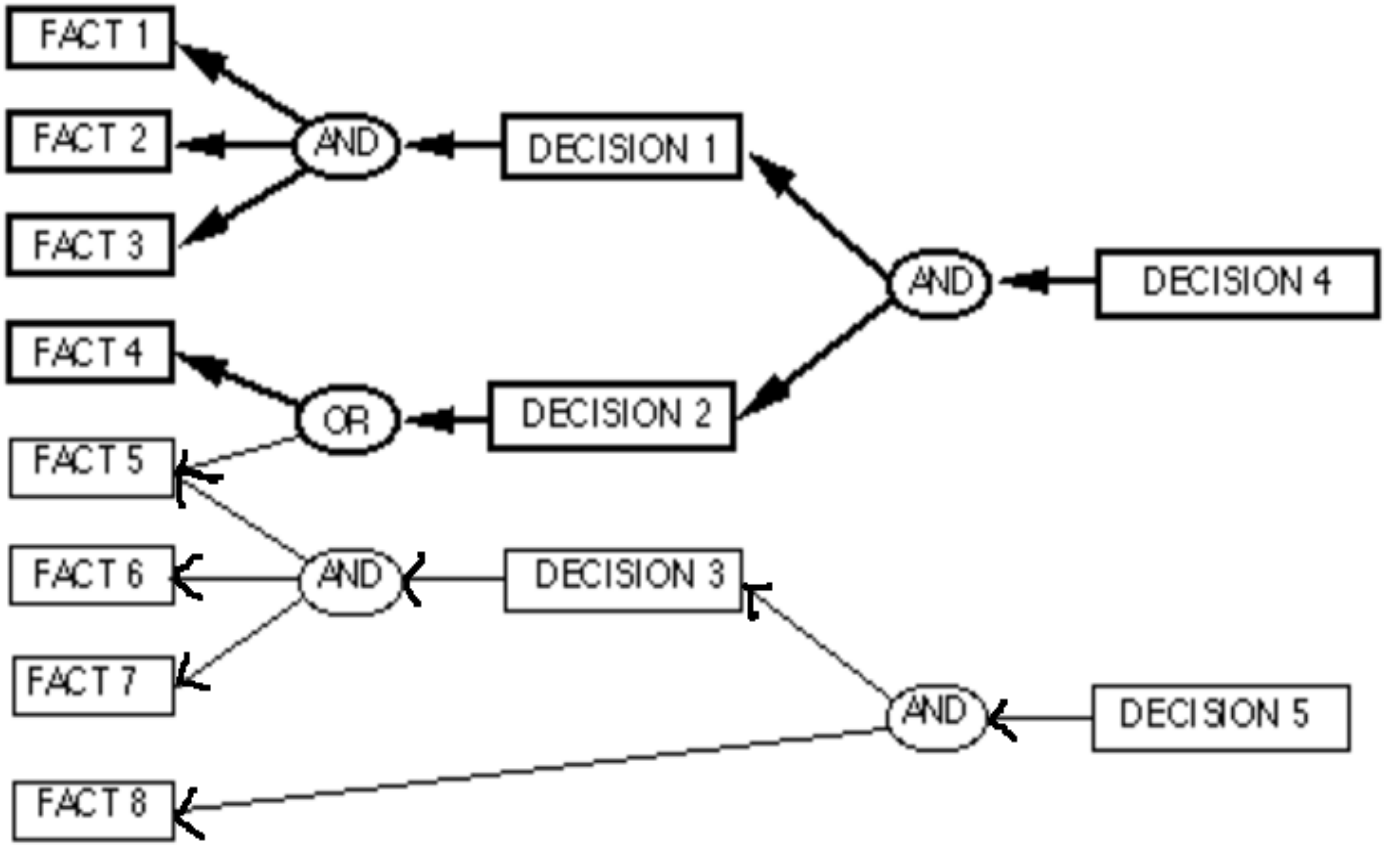


Fig : Backward chaining

Backward Chaining Example

Suppose we have to decide whether **Fritz** is green based on the rule base:

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

Backward Chaining Example…

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

Backward Chaining Example…

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

Backward Chaining Example…

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

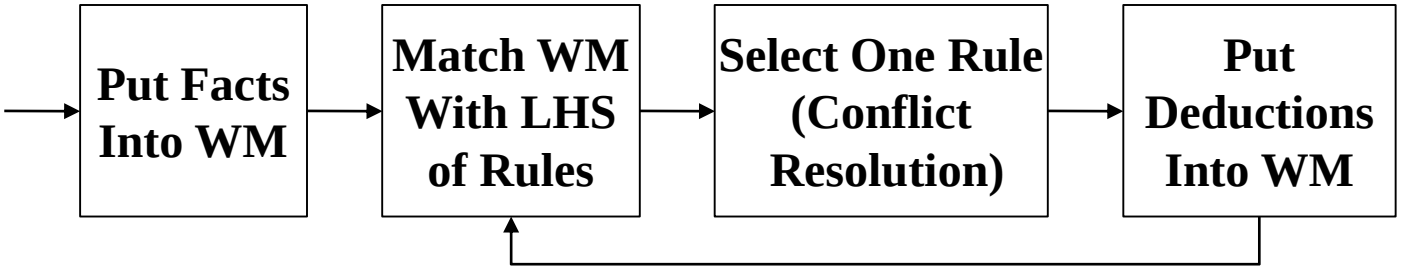
FC versus BC

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

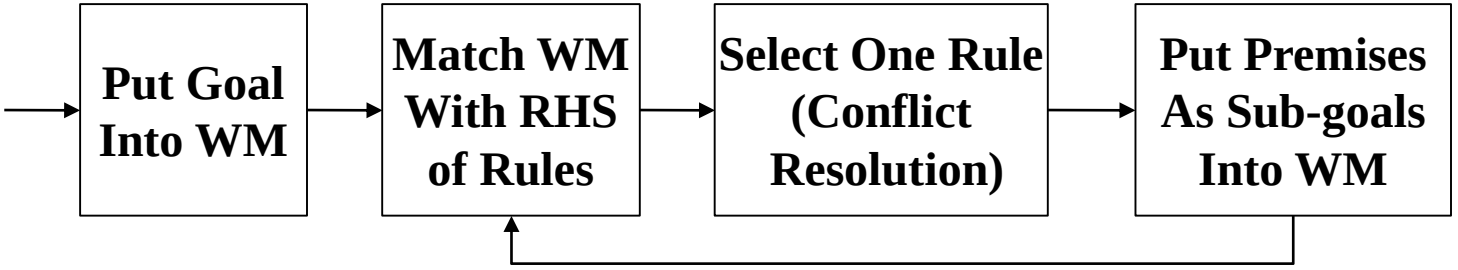
Conflict Resolution

- **Conflict resolution** strategies are used in production systems in **artificial intelligence**, such as in rule-based expert systems, to help in choosing **which production rule to fire**.
- The need for such a strategy arises when the conditions of two or more rules are satisfied by the currently known facts.

Conflict Resolution [5]...



Forward Chaining



Backward Chaining

- What to do if there is more than 1 matching rule in each inference cycle?

Why Conflict Resolution Important?

- Decisions made at the conflict resolution stage is crucial because they can dramatically **affect the solution reached** and **how quickly it is found**.

Conflict Set

- The **conflict set** is defined as the set of pairs of the following form and we **need to chose one** to fire.

{⟨Production rule, matching working memory elements ⟩}

- **Two types of rule set**
  - ✓ Deterministic rule set
    - ➢ At most 1 rule matched at each cycle (i.e. 0 or 1)
  - ✓ Non-deterministic rule set
    - ➢ More than 1 matching rule
    - ➢ Inference engine must select one rule to apply using its conflict resolution strategy

Conflict Resolution Strategies [5,6]

- General conflict resolution strategies.
- Problem specific conflict resolution strategies.

General Conflict Resolution Strategies [5,6]

Most common ***general conflict resolution strategies*** are:

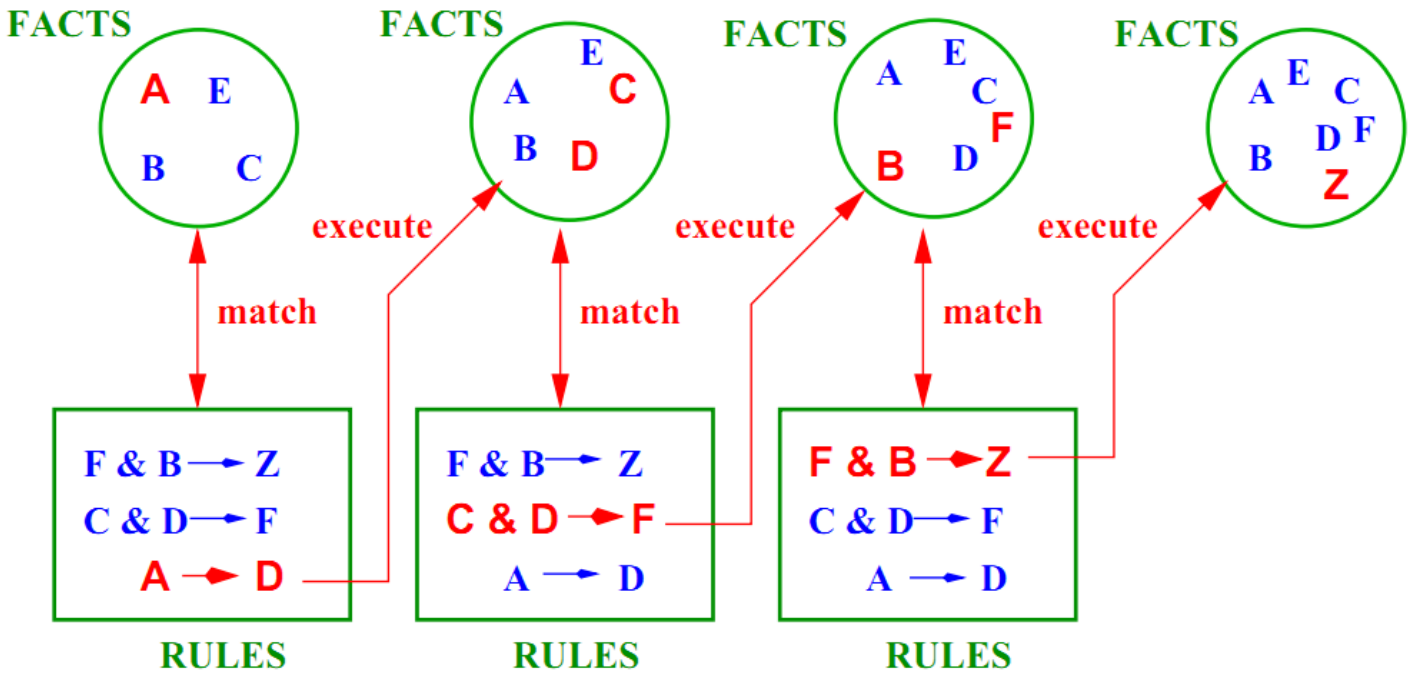
1. Textual Order

   - Assume that rules are ordered (linearly).
   - Most important rules should be placed early in the Knowledge Base.
   - Fire the first matching rule

General Conflict Resolution Strategies [5,6]...

2. Recency

- When two or more rules could be chosen, favor the one that matches the most recently added facts. Because these are most likely to describe the current situation.



General Conflict Resolution Strategies [5,6]...

3. Specificity

- More specific rules are preferred to more general rules
  - ✓ Rules with greater number of conditions
  - ✓ Rules with fewer variables are more specific
  - ✓ More instantiated by the WM are more specific
- More specific rules should be applied earlier because they use more data and so can be used for special cases or exceptions to general rules

General Conflict Resolution Strategies [5,6]...

Example: Consider the following set of rules:

R1:	IF:	engine does not turn AND battery is not flat
	THEN:	ask user to test starter motor
R2:	IF:	there is no spark
	THEN:	ask user to check the points
R3:	IF:	engine turns AND engine does not start
	THEN:	ask user to check the spark
R4:	IF:	engine does not turn
	THEN:	ask user to check the battery
R5:	IF:	battery is flat
	THEN:	ask user to charge battery AND EXIT

General Conflict Resolution Strategies [5,6]...

Example:

Now, If the initial facts are "engine does not turn" and "battery is not flat", the conflict set is:

{⟨ R1, engine does not turn, battery is not flat ⟩, ⟨ R4, engine does not turn ⟩}

General conflict resolution strategy: **Specificity** would work well here.

Problem Specific Conflict Resolution Strategies [5,6]

1. Extra Conditions

- Simply add extra conditions to the rules to avoid the conflicts.
- These extra conditions can be related to the **inference strategies**, E.g.
  - ✓ To what is currently being searched for
  - ✓ To what rule applications tend to be most useful.

Problem Specific Conflict Resolution Strategies [5,6]

Example: In previous example Rule R1 can be modified as:

R1: IF: *haven't already tested starter motor*
AND engine does not turn
AND battery is not flat
THEN: ask user to test starter motor

Problem Specific Conflict Resolution Strategies [5,6]…

2. Meta-Rules

- Adds Meta-Rules to the original rule; i.e. adds rules with rule.
- This will increase the overall number of rules, but it does separate the factual and heuristic knowledge and makes the knowledge base easier to maintain.

Problem Specific Conflict Resolution Strategies [5,6]…

Example: In previous example , rule R1 can be supplemented with a meta-rule as follows:

RULE R1:

IF: engine does not turn
AND battery is not flat
THEN: ask user to test starter motor

META-RULE M1:

IF: *haven't already tested starter motor*
THEN: *select R1*

Choice of Conflict Resolution Strategy

- Determine how the ES (Expert System) searches in the problem space
  - ✓ how quick to find the solution
  - ✓ if the system finds a solution
    - ➢ and if found, which solution
- Should be chosen to fit your application
- Desirable properties
  - ✓ stability
    - ➢ follow a line of reasoning
  - ✓ sensitivity
    - ➢ react quickly to any change in WM

References

[1] http://what-when-how.com/artificial-intelligence/knowledge-based-systems-artificial-intelligence/
[2] http://www.ijmlc.org/vol5/492-A14.pdf
[3] https://www.hindawi.com/journals/ijr/2014/672714/fig2/
[4] http://www.cse.unsw.edu.au/~billw/cs9414/notes/kr/rules/rules.html
[5] http://www.csc.villanova.edu/~matuszek/spring2011/syllabus2011.html
[6] https://www.cs.bham.ac.uk/~jxb/IAI/w7.pdf

THANKS