# Lecture 7:

# Reasoning with Rules and Cases

Dr. Roman V Belavkin

BIS4435

## OVERVIEW

- Rule–based ES
    1. Facts and rules
    2. The architecture of rule–based ES
    3. Reasoning
- Case–based ES
    1. Cases and similarity
    2. The architecture of case–based ES
    3. Operation
- Comparison of two approaches

# DECLARATIVE AND PROCEDURAL KNOWLEDGE

In rule–based approach, the symbolic knowledge is divided into two types:

**Declarative** : these are propositions describing facts, objects or events that are true (i.e. happening or that we observe).

**Procedural** : these are logical rules (inferences) that we can use to reason using the facts and make decisions.

There is evidence that human brain learns and remembers these types of knowledge is somewhat different ways. Rules (procedural knowledge) are usually harder to describe and forget.

## FACTS (PROPOSITIONS)

- Facts usually are represented in a form of logical *propositions*:

  Barney is a dog

  Dog has four legs

  4 is a sum of 2 and 2

- Facts can be `True` or `False`.

## RULES (IMPLICATIONS)

- Rules have a form:

  IF *condition* THEN *action*

  **Example**

  IF *Barney is a dog* THEN *Barney has four legs*

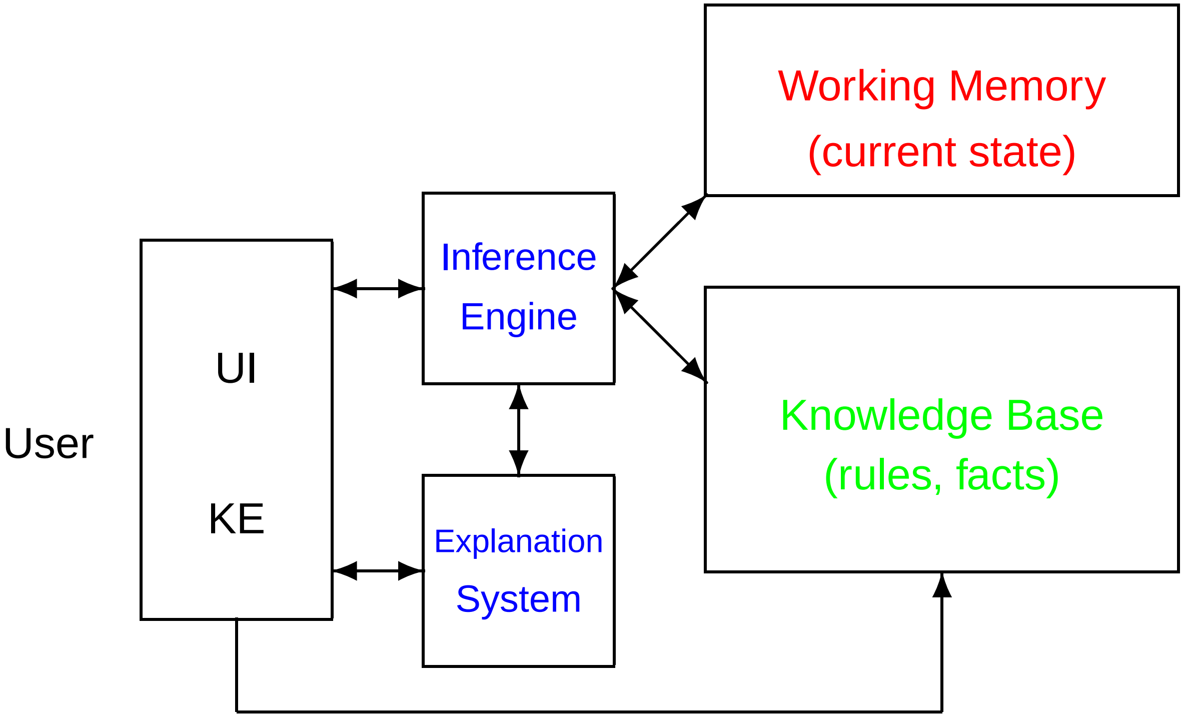
  IF *the price is high* THEN *sell*

- Condition can be a fact or a collection of facts. Action can be adding another fact, asking a question, etc.

- The IF... part is called the *left–hand–side* (or the *antecedent*, the *premise*) of a rule. The THEN... part is called the *right–hand–side* (or the *consequent*) of a rule.

# ARCHITECTURE OF ES

Main components of a classical rule–based expert system:



Knowledge base, working memory, inference engine, explanation system, user interface and knowledge base editor.

# MAIN COMPONENTS OF ES

**Knowledge Base** contains all the knowledge of ES in a form of rules (procedural knowledge) and facts (declarative knowledge). Can be compared with a hard disk drive of PC, or long–term memory of the human brain.

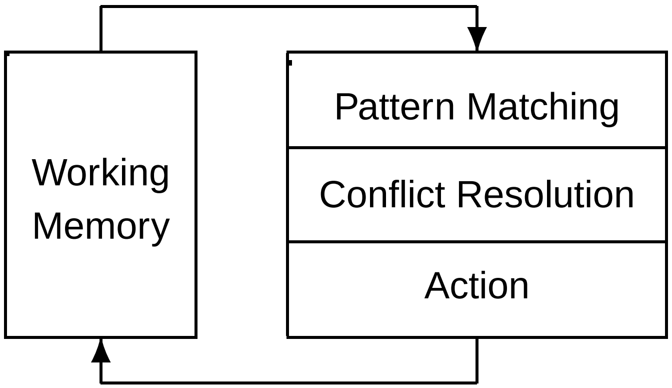
**Working Memory** contains only the facts describing the current state of a problem. Can be compared with RAM of PC, or short–term memory of the human brain.

**Inference Engine** implements the reasoning process. In brief, it finds rules in the knowledge base that correspond to the contents of working memory and applies them to the problem.

# RECOGNISE–ACT CYCLE

Reasoning in rule–based systems occurs in cycles.

Each cycle consists of three stages:



**Pattern Matching** : (recognition) the contents of the working memory is compared with the rules. All rules that *match* the current problem state are selected into a *conflict set*.

**Conflict Resolution** : a single rule from the conflict set is selected.

**Action** : the action part of the selected rule is performed. It may result in changing the working memory.

# REASONING DIRECTIONS

Rule–based systems can use two directions for reasoning during problem solving:

**Forward** (data–driven): Start $\longrightarrow$ Goal

**Backward** (goal–driven): Start $\longleftarrow$ Goal

- The corresponding types of reasoning are called *forward* and *backward chaining* respectively.

## EXAMPLE: WEATHER FORECAST ES

| | | | | |
|---|---|---|---|---|
| 1 | IF | *cyclone* | THEN | *clouds* |
| 2 | IF | *anticyclone* | THEN | *clear sky* |
| 3 | IF | *pressure is low* | THEN | *cyclone* |
| 4 | IF | *pressure is high* | THEN | *anticyclone* |
| 5 | IF | *arrow is down* | THEN | *pressure is low* |
| 6 | IF | *arrow is up* | THEN | *pressure is high* |

## FORWARD CHAINING

| Cycle | Working Memory | Conflict set | Rule fired |
|---|---|---|---|
| 0 | arrow is up | 6 | 6 |
| 1 | arrow is up, pressure is high | 6, 4 | 4 |
| 2 | arrow is up, pressure is high, anticyclone | 6, 4, 2 | 2 |
| 3 | arrow is up, pressure is high, anticyclone, clear sky | 6, 4, 2 | Halt |

The reasoning starts from the available data (data–driven), and the working memory is matched against the left–hand–side of the rules.

## BACKWARD CHAINING

| Cycle | Working Memory | Conflict set | Rule fired |
|---|---|---|---|
| 0 | clear sky | 2 | 2 |
| 1 | clear sky, anticyclone | 2, 4 | 4 |
| 2 | clear sky, anticyclone, pressure is high | 2, 4, 6 | 6 |
| 3 | clear sky, anticyclone, pressure is high, arrow is high | 2, 4, 6 | Halt |

The reasoning starts from the goal state (goal–driven), and the working memory is matched against the right–hand–side of the rules.

## COMPLEX PATTERNS

- The condition part of a rule may have several facts connected by logical operators NOT, AND, OR:

  IF (*arrow is down* AND *clear sky*) THEN *broken*<br>
  OR (*arrow is up* AND *clouds*) *barometer*

- Such a complex proposition (pattern) is just a more complex fact.
- A rule will match the working memory only if the whole condition (i.e. the whole proposition) is `True`.
- Logical rules for *negation* (NOT), *conjunction* (AND) and *disjunction* (OR) are used to evaluate the condition.

# LOGICAL RULES (TRUTH TABLES)

To check whether a proposition is true we should refer to truth tables:

| $x$ | NOT $x$ |
|---|---|
| T | F |
| F | T |

| $x_1$ | $x_2$ | $x_1$ AND $x_2$ |
|---|---|---|
| F | F | F |
| T | F | F |
| F | T | F |
| T | T | T |

| $x_1$ | $x_2$ | $x_1$ OR $x_2$ |
|---|---|---|
| F | F | F |
| T | F | T |
| F | T | T |
| T | T | T |

## SIMPLIFICATION OF RULES

Complex rules can be represented by several smaller rules.

- Disjunctive rules (with OR) can be broken down into several:

IF *arrow is down* AND *clear sky* THEN *broken barometer*

IF *arrow is up* AND *clouds* THEN *broken barometer*

- Sometimes it is useful to create intermediate facts:

IF a AND (b OR c) THEN d ⟶

IF b THEN p<br>
IF c THEN p<br>
IF a AND p THEN d

## CONFLICT RESOLUTION

Sometimes, several rules can match the working memory, but only one has to be selected (hence the conflict). There are several strategies to deal with conflict resolution:

**Refraction** : once the rule has fired, it is not used again.

**Recency** : use the rule that has been used recently in such situation.

**Specificity** : use the rule with the more specific condition (more facts).

**Priority** : assign priority to rules (i.e. rank, utility, probability, cost, etc) and choose the one with the highest priority.

**Parallel** : fire all rules with separate lines of reasoning.

## META–RULES

- There maybe rules in the system that control execution of other rules. They are called the *meta–rules*.

  IF *broken barometer* THEN *use barometer diagnosis rules*

- Meta–rules can be used for conflict resolution, for directing the search in the knowledge base, for using the type of reasoning and so on.

- Meta–rules can improve the system's performance.

# OTHER COMPONENTS OF ES

**User interface** may provide interaction facilities, where ES asks the user some questions. The answers are interpreted into facts in working memory and used in reasoning.

**Knowledge base editor** gives the possibility to examine and edit the knowledge base.

Rule–based ES can explain their reasoning by showing the trace of the rules applied to solve the problem. This is useful because human can understand the logics behind the solution.

## LIMITATIONS OF RULE–BASED SYSTEMS

| | | | |
|---|---|---|---|
| IF | $x$ *has wings* | THEN | $x$ *is a bird* |
| IF | $x$ *is a bird* | THEN | $x$ *can fly* |

How about an ostrich?

- It is hard to describe all the cases by rules.
- The knowledge is very task dependent and cannot adapt easily.
- The knowledge lacks deeper understanding of the problem.

There are other types of ES: model–based, case–based.

# CASE–BASED REASONING SYSTEMS

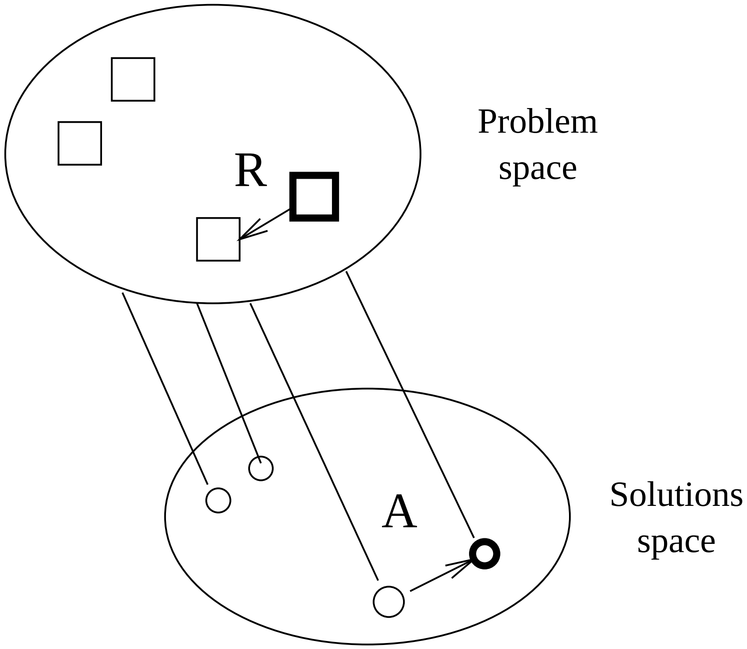
- Instead of facts and rules, CBR systems use *cases* and their entire *solutions*.

| Case: | Age | Gender | M. Income (£ K) | M. Expenses (£ K) | Home owner | Credit score |
|---|---|---|---|---|---|---|
| 1 | 21 | 0 | 2 | 1 | 0 | 3 |
| 2 | 18 | 1 | 1 | 2 | 0 | 1 |
| 3 | 50 | 1 | 6 | 2 | 1 | 5 |
| 4 | 23 | 0 | 3 | 1 | 1 | 4 |
| 5 | 40 | 1 | 3 | 2 | 0 | 2 |

- Cases can have quite complex descriptions using symbolic and numerical values

- CBR is based on the concepts of *similarity* and *analogy*. Similarity is used to find similar cases, and analogy is used to find solutions for similar cases.

# PROBLEM (CASE) and SOLUTION SPACES



**R** : retrieval of a similar case from problem space

**A** : adaptation of a solution from solution space

## OPERATION CYCLE of CBR

In general, can be described using four **RE**s:

- **RE**trieve the most similar case or cases.
- **RE**use the case(s) to attempt to solve the problem.
- **RE**vise the proposed solution if necessary.
- **RE**tain the new solution as a part of a new case.

# RETRIEVAL and REUSE

To solve a new case, a CBR systems **retrieves** an old similar case.

There are two main methods to retrieve similar cases

**Nearest–neighbour** : is based on comparing the cases using some distance (e.g. Euclidean distance)

$$\|\mathbf{a} - \mathbf{b}\| = \sqrt{(a_1 - b_1)^2 + \cdots + (a_m - b_m)^2}$$

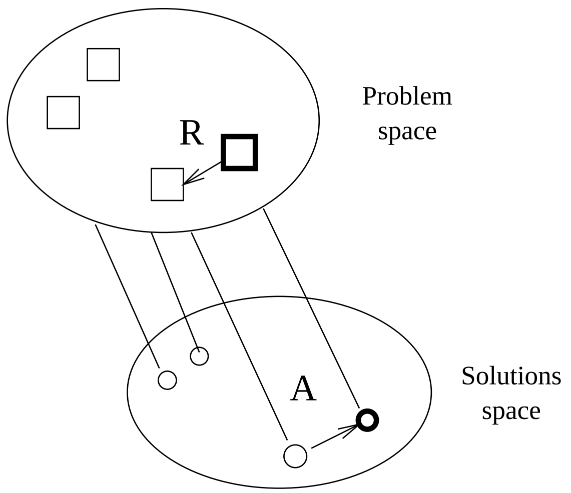
where $\mathbf{a} = (a_1, \ldots, a_m)$ and $\mathbf{b} = (b_1, \ldots, b_m)$ are two cases.

**Inductive retrieval** : is based on learning which feature of the case carries the most useful information to predict the solution (highest information gain).

The solution of a retrieved case is **reused** to for the new case.

## REVISE and RETAIN

- Because the new case is most likely different from all the old cases, the solution of a retrieved case can be **revised** or **adapted**.
- If the solution was a success, then the new case is **retained** in the database together with the new solution.



## RULE–BASED vs CASE–BASED

- In rule–based systems a solution is achieved through an application of many rules, inference of facts, etc.
- In case–based systems the whole problem definition (case) is compared with similar problems, and the entire solution is applied at once.
- Rule–based systems usually work better for well–defined problems that do not change with time.
- Case–based systems can be used where problems are less understood and are dynamic.