20IS603 Architecture of Intelligent Systems

Symbolic Learning

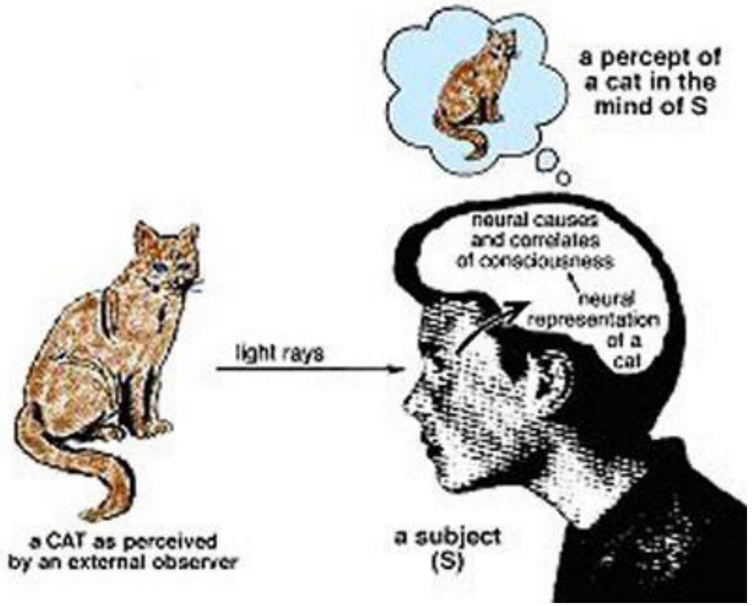
Lecture #12

Why Learning?

- An agent is learning if it improves its performance on future tasks after making observations about the world.
- The designers cannot anticipate all possible situations that the agent might find itself in.
- The designers cannot anticipate all changes over time.
- Sometimes human programmers have no idea how to program a solution themselves.
- Machine learning is concerned with computer programs that automatically improve their performance through experience.

Limitations in Knowledge representation

- Software engineer may not possess the domain expertise - knowledge would need to be obtained from a domain expert through a process of knowledge acquisition - an alternative is for the system to be designed to learn for itself.
- Rules that describe a particular domain may not be completely understood, such as listening to and interpreting a spoken voice
- Even though a domain may be well understood, it may not be expressible explicitly in terms of rules, facts, or relationships
- The problem may change over time.



To have the system learn for itself from a set of example solutions

Classification of Learning

- Supervised learning
- Rote learning - system receives confirmation of correct decisions – during an incorrect decision it is “spoon-fed” with the correct rule.
- Learning from advice - Rather than being given a specific rule that should apply in a given circumstance, the system is given a piece of general advice
- Learning by induction - presented with sets of example data and is told the correct conclusions that it should draw from each.
- Learning by analogy - The system is told the correct response to a similar, but not identical, task - adapt the previous response to generate a new rule applicable to the new circumstances

Classification of Learning

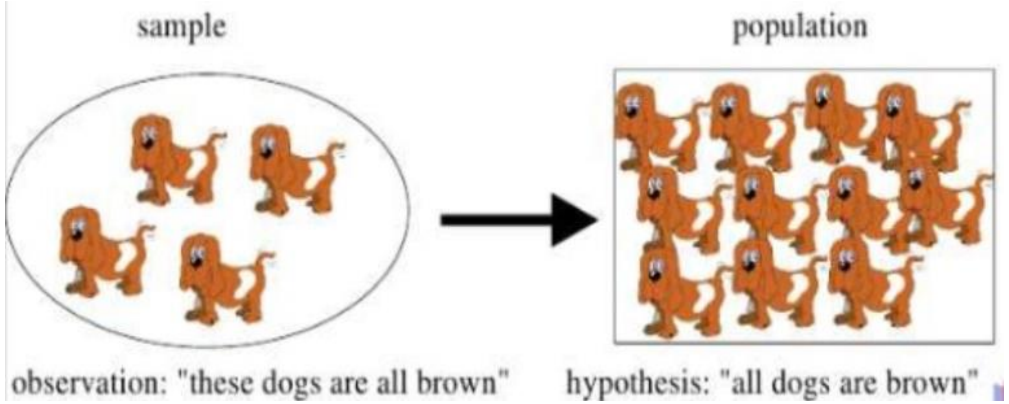
- Explanation-based learning (EBL) - analyzes a set of example solutions and their outcomes to determine why each one was successful or otherwise
- Case-based reasoning (CBR) - Any case about which the system has reasoned is filed away, together with the outcome, whether it be successful or otherwise. Whenever a new case is encountered, the system adapts its stored behavior to fit the new circumstances.
- Explorative or unsupervised learning - continuously searches for patterns and relationships in the input data, perhaps marking some patterns as interesting and demanding further investigation – Eg: data mining, clusters

Symbolic learning

- Embedding of human knowledge and behavior rules into computer programs
- Formulate and modify rules, facts, and relationships, explicitly expressed in words and symbols.
- Create and modify their own knowledge base
- Focused on learning by induction and case-based reasoning.

Learning by induction

- Learning from observation and earlier knowledge by generalization of rules and conclusions



- The identified and extracted generalized rules come to use in reasoning and problem solving

Learning by induction (2)

- Rule induction involves generating from specific examples a general rule

  `if <general circumstance> then <general conclusion>`

- Based on trial-and-error - said to be an empirical approach - not certain of the accuracy
- The aim of induction is to build rules that are successful as often as possible, and to modify them quickly when they are found to be wrong.
- Whatever is being learned—typically, rules and relationships—should match the positive examples but not the negative ones.

Learning by induction (3)

- Generate an initial prototype rule that can subsequently be refined
- Initial prototype may be a copy of a general-purpose template, or it can be generated by hypothesizing a causal link between a pair of observations

```
rule r5_1
    if status of valve_1 is open
    then flow_rate of valve_1 becomes 0.5.
rule r5_2
    if flow_rate of valve_1 is 0.5
    then status of valve_1 becomes open.
```

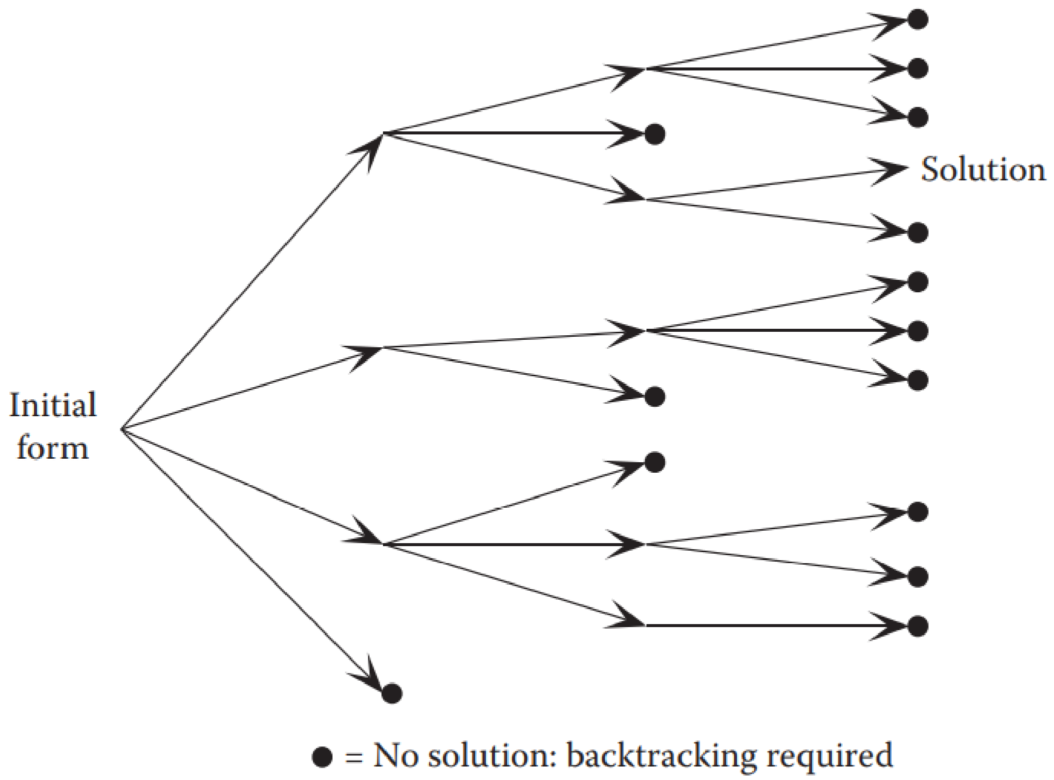
- The prototype rules can then be modified in the light of additional example data or rejected

Learning by induction (4)

- Rule modifications can be classified as either strengthening or weakening.
- Condition is made stronger – specialized - by restricting the circumstances to which it applies
- Condition is made weaker – generalized - by increasing its applicability
- A rule needs to be generalized if it fails to fire for a given set of data
- A rule needs to be specialized if it fires when it should not.

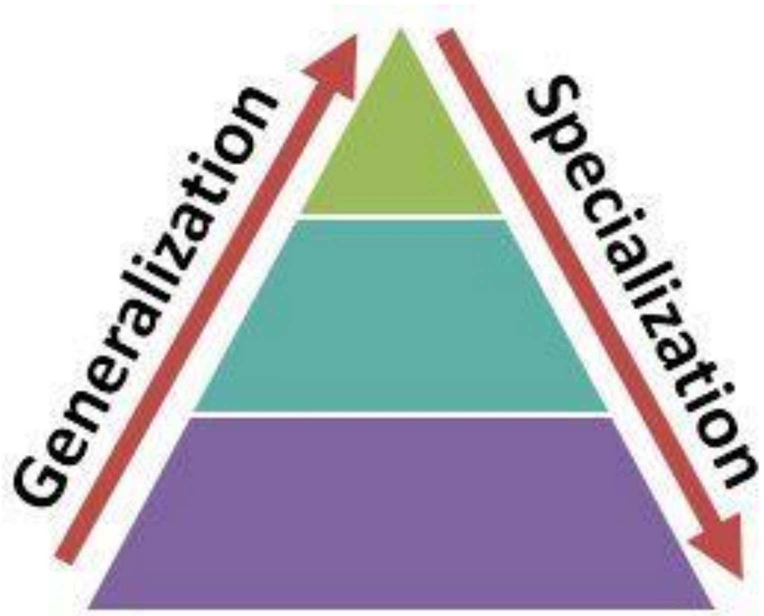
Learning Viewed as a Search Problem

- Finding the correctly modified rule is a search problem
- Each branch in search tree represents a generalization or specialization that would correctly handle the most recently encountered input.
- Subsequent inputs may reveal an incorrect choice - indicated by a dot at the end of a branch.
- The system keep track of its current position in the search tree, as it must backtrack whenever an unsuitable choice is found to have been made.



Techniques for Generalization and Specialization

- Universalization
- Replacing constants with variables;
- Using disjunctions (generalization) and conjunctions (specialization);
- Moving up a hierarchy (generalization) or down it (specialization); and
- Chunking



Techniques for Generalization and Specialization (2)

Universalization

- Inferring a new general rule from a set of specific cases
- Consider the following series of separate scenarios:

```
Status of valve_1 is open and flow_rate of valve_1 is high.
Status of valve_2 is open and flow_rate of valve_2 is high.
Status of valve_3 is open and flow_rate of valve_3 is high.
```

- From these it is possible to induce the following general rule:

```
rule r5_5
     if status of X is open
     then flow_rate of X becomes high
```

Techniques for Generalization and Specialization (3)

Replacing Constants with Variables

- General rules can be generated from more specific ones by replacing constants with local variables

```
rule specific1
  if status of gas_valve_1 is open
  then flow_rate of gas_valve_1 becomes high.

rule specific2
  if status of gas_valve_2 is open
  then flow_rate of gas_valve_2 becomes high.

rule specific3
  if status of gas_valve_3 is open
  then flow_rate of gas_valve_3 becomes high.

rule specific4
  if status of gas_valve_4 is open
  then flow_rate of gas_valve_4 becomes high.

rule specific5
  if status of gas_valve_5 is open
  then flow_rate of gas_valve_5 becomes high.
```

General rule:

```
rule r5_5
  if status of X is open
  then flow_rate of X becomes high.

rule r5_6
  if status of X is open
  then flow_rate of Y becomes high.
```

or:

```
rule r5_7
  if status of X is open
  then Y of X becomes high.
```

Techniques for Generalization and Specialization

Using Conjunctions

- Rules can be made more specific by adding conjunctions to the condition

```
rule r5_5
     if status of X is open
     then flow_rate of X becomes high.
```

- This is modified by strengthening the condition - by use of a conjunction (*and*)

```
rule r5_8
     if status of X is open
     and X is a gas_valve
     then flow_rate of X becomes high.
```

Techniques for Generalization and Specialization

Using Disjunctions

- Rules can be made more general by adding disjunctions to the condition

```
rule r5_5
     if status of X is open
     then flow_rate of X becomes high.
```

- Add a disjunction (*or*) to the condition part of the rule:

```
rule r5_9
     if status of X is open
     and [X is a gas_valve or X is a water_valve]
     then flow_rate of X becomes high.
```

Techniques for Generalization and Specialization

Moving up or down a Hierarchy

- Use of an is-a-kind-of relationship in order to generalize

```
rule r5_8
        if status of X is open
        and X is a gas_valve
        then flow_rate of X becomes high.
```

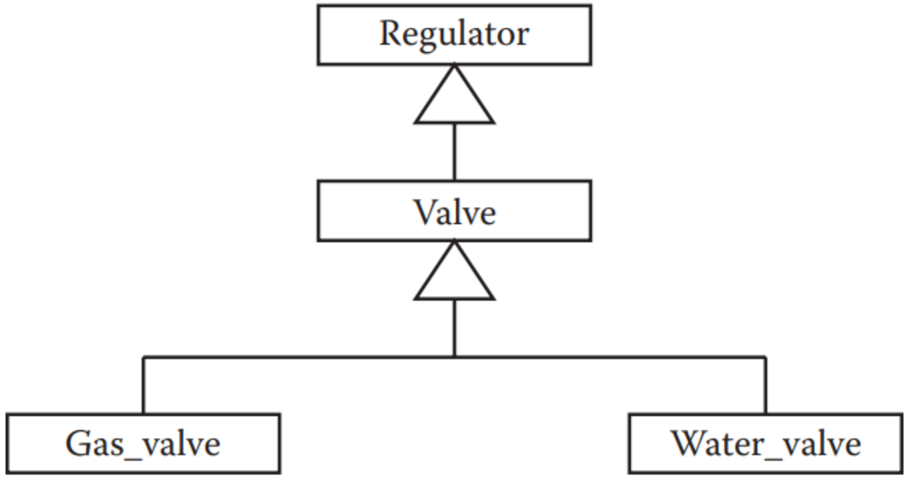
Modified as:

```
rule r5_10
        if status of X is open
        and X is a valve
        then flow_rate of X becomes high.
```

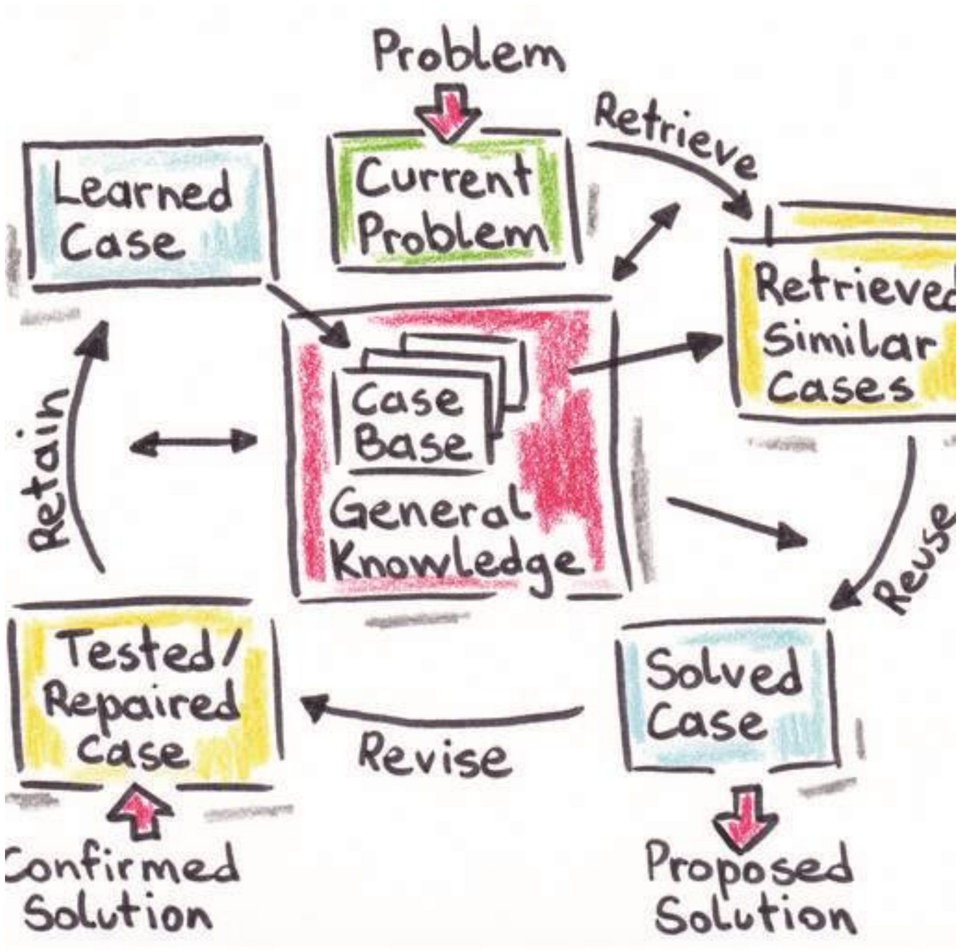
Techniques for Generalization and Specialization

Chunking

- Mechanism for automated learning
- Given an overall goal, every problem encountered along the way can be regarded as a subgoal
- Problems are tackled hierarchically; if a goal cannot be met at one level, it is broken down into subgoal
- The series of rules required to satisfy a subgoal is collapsed down into a single production rule - process of chunking
- The new rule, or chunk, is then stored so that it can rapidly solve the same subgoal



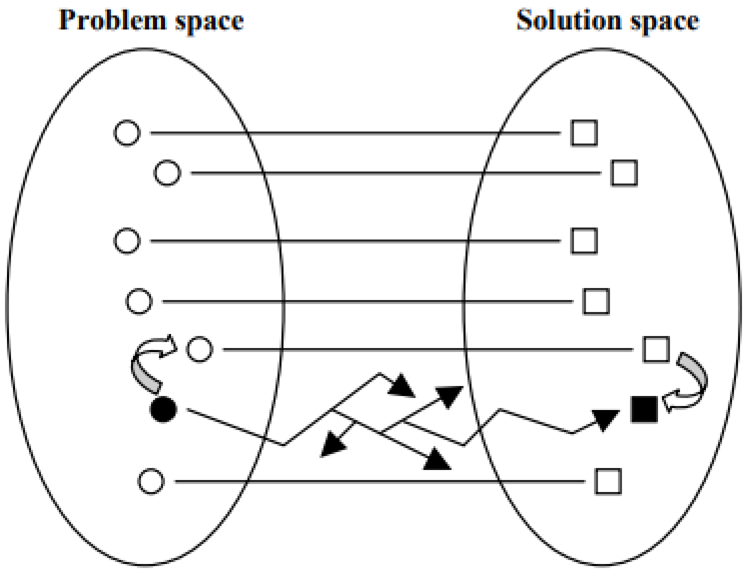
20IS603 Architecture of Intelligent Systems



Case-Based Reasoning

Case-Based Reasoning (CBR)

- To recall previous experience whenever a similar problem arises
- Core assumptions behind CBR - *similar problems have similar solutions*.
- A case-based reasoner solves new problems by adapting solutions that were used to solve old problems.
- Offers a reasoning paradigm that is similar to the way many people routinely solve problems
- CBR is reasoning by remembering: previously solved problems (cases) are used to suggest solutions for novel but similar problems



What is a Case?

- Several features describing a problem
- An outcome or a solution
- Cases can be very rich
  - text, numbers, symbols, plans, multimedia
- Records of real events
- Excellent for justifying decisions
- A case-base is a set of cases.
- Case-bases are usually just flat files or relational databases

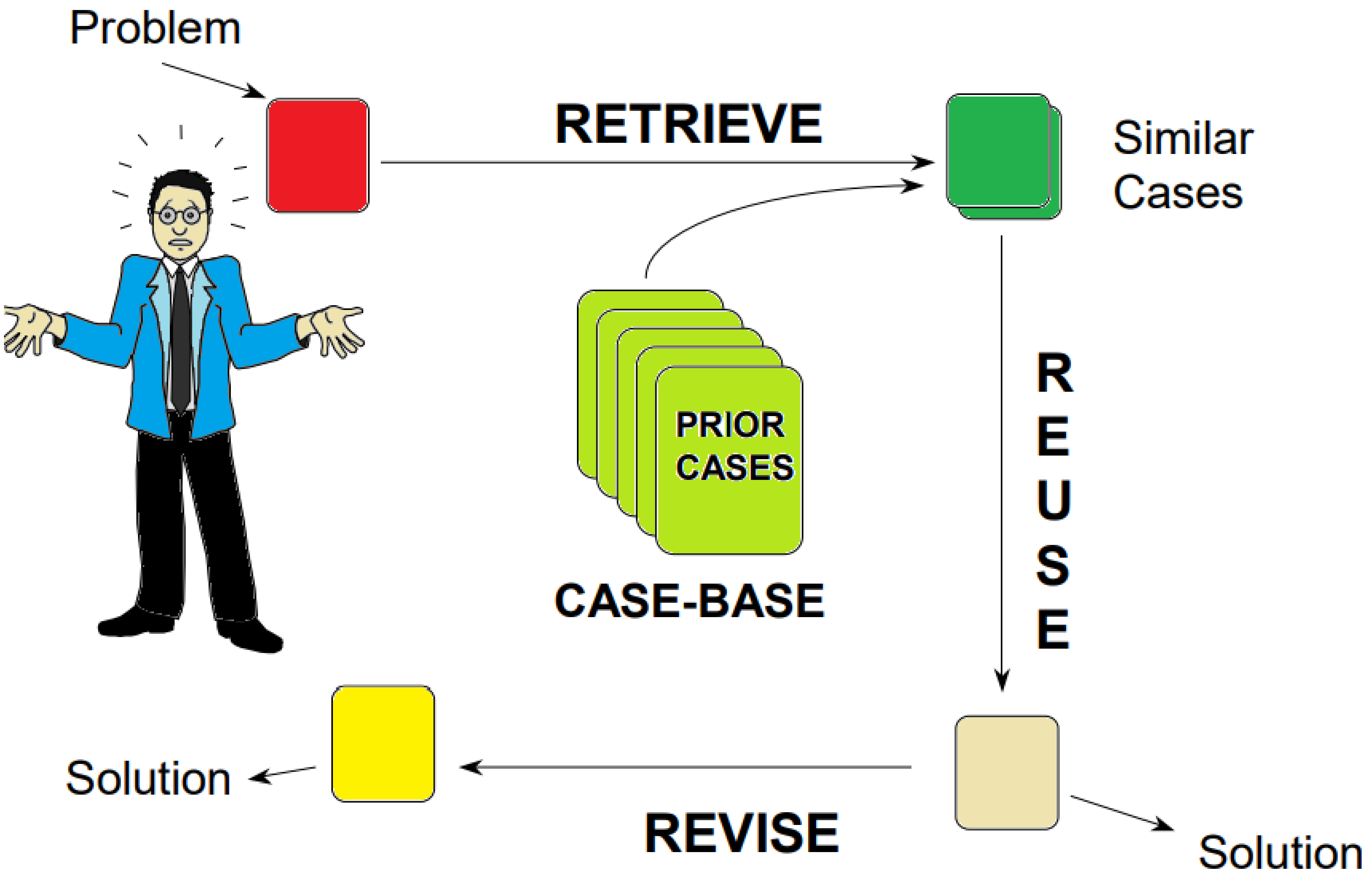
CBR Problem Solver

- Case – previously made and stored experience item
- Case-Base – core of every case-based problem solver - collection of cases
- A case-based problem solver solves new problems primarily by reuse of solutions from the cases in the case-base
- Once similar cases are selected, the solution(s) from the case(s) are adapted to become a solution of the current problem
- When a new (successful) solution to the new problem is found, a new experience is made, which can be stored in the case-base to increase its competence, thus implementing a learning behavior.

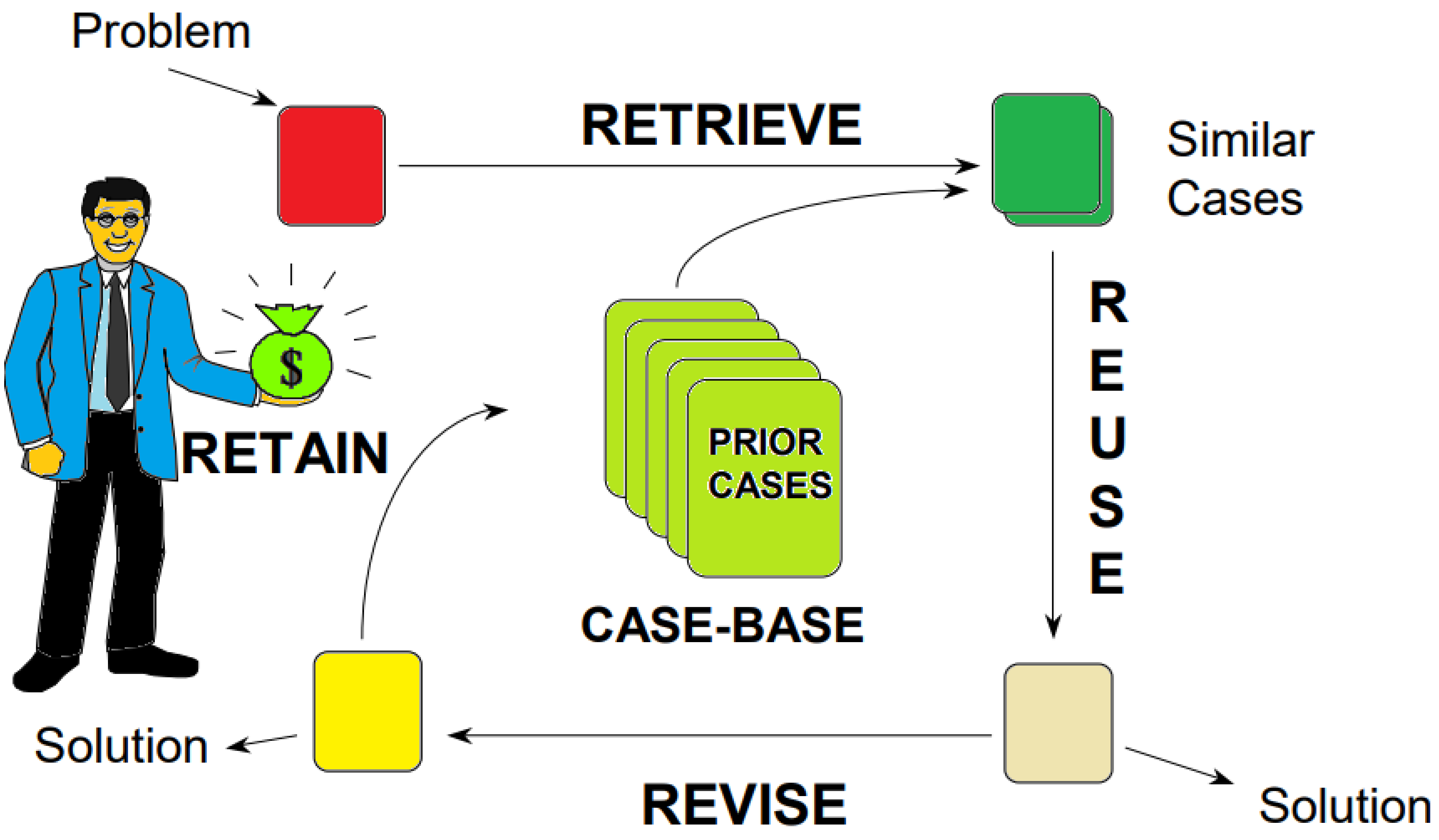
Case-Based Reasoning Cycle

- Consists of 4 sequential steps around the knowledge of the CBR system
  - RETRIEVE the most similar case(s);
  - REUSE the case(s) to attempt to solve the current problem;
  - REVISE the proposed solution if necessary;
  - RETAIN the new solution as a part of a new case

Case-Based Reasoning Cycle (2)

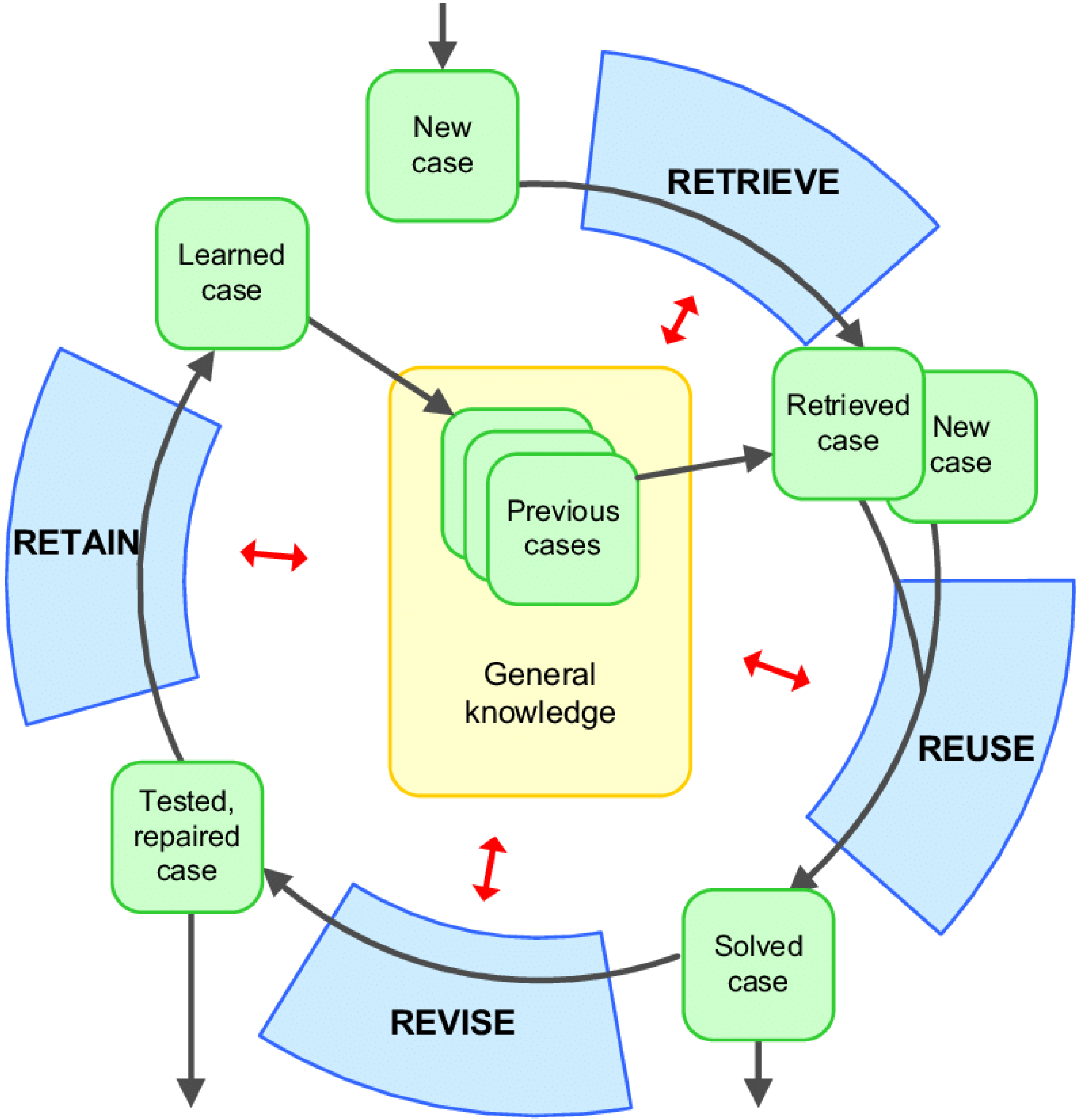


Case-Based Reasoning Cycle (3)



Case-Based Reasoning Cycle (4)

- A new problem is matched against the cases furnishing the case base and one or more similar cases are *retrieved*.
- A solution suggested by the matching cases is then *reused*.
- Unless the retrieved case is a close match, the solution will probably have to be *revised* (adapted) and tested (*evaluated*) for success, producing a new case that can be *retained* ensuing, consequently, *update of the case base*



Application areas of CBR

- help-desk and customer service
- recommender systems in electronic commerce
- knowledge and experience management
- medical applications and applications in image processing
- applications in law, technical diagnosis, design, planning
- applications in the computer games and music domain.

Limitations of CBR

- Handling large case bases
- Cannot handle dynamic problem domains
- Semiautomated operation
- Inefficient storage and retrieval of cases due to noise

Rule-based System & Case-based reasoning - Comparison

Criterion	Rule-based reasoning	Case-based reasoning
Knowledge unit	Rule	Case
Granularity	Fine	Coarse
Knowledge acquisition	Obtaining rules & hierarchies	Obtaining cases & hierarchies
Explanation mechanism	Back trace of rules fired	Precedent cases
Characteristic output	Decision + confidence measure	Decision + Precedent cases