COMP474/6741

Knowledge Bases/Expert Systems

Negnevitsky: Chap.2

Some slides are part of my course at MIT, 2010

Today

- KBS: from Expert systems to KBS
- Characteristics of KBS
- Research Areas involved in KBS
- Necessity of KBS
- Problems of KBS
- Components of KBS
- KBS on production systems
- Cased Based Reasoning
- Other Methodologies
  - Systems based on Neural networks
  - Intelligent Agents and multiagent systems
- Development process of KBS
- Methodologies for problem solving
- Constructive Problem Solving

General Problem Solving

- All the problem solving methods that we have seen are all purpose
- Their basis is an heuristic function that orders the solution paths to explore in order to reach faster the goal
- Expressiveness of heuristic functions is limited
- Only a function can not represent all possible decision during the search of a solution
- The reduction of computational time is limited
- More specific knowledge about the problem allows to make better decisions

Knowledge Based Systems

- We are looking for systems that allow a better use of the domain knowledge
- This knowledge allows to analyze better each step of the resolution process
- These systems can simulate the way an expert solves a problem
- The knowledge of the domain has to be thoroughly formalized
- This knowledge has to be integrated in the solving problem process

From the Expert Systems to KBS

Expert Systems

- The goal is to simulate the solving problem process of human experts
- Their development is based on traditional knowledge engineering techniques
- Mainly implemented with rule production systems
- Closed applications that usually do not use machine learning

Knowledge Based Systems

- The goal is to use domain knowledge to solve problems
- Automatic knowledge acquisition is a part of the knowledge engineering process
- Heterogeneous methodologies and techniques (rules, cases, qualitative models, intelligent agents, emergent computation, ...)
- Adaptive systems with machine learning capabilities

Today

- KBS: from Expert systems to KP
- Characteristics of KBS
- Research Areas involved in KBS
- Necessity of KBS
- Problems of KBS
- Components of KBS
- KBS on production systems
- Cased Based Reasoning
- Other Methodologies
  - Systems based on Neural networks
  - Intelligent Agents and mutiagent systems
- Development process of KBS
- Methodologies for problem solving
- Constructive Problem Solving

Characteristics of KBS

- KBS are used in complex problems where conventional software applications are not enough
- The nature of these problems requires some specific characteristics
  - Flexibility and capability to be applied to different kinds of problems
  - Emulation of rational thinking as problem solving mechanism
  - To be able to manage in a rich environment with a great quantity of information
  - To be able to use symbolic information during the reasoning process
  - Use of natural interfaces as a mean of communication with the user
  - Use of machine learning as method of adaptation

Characteristics of KBS

When building systems with these characteristics:

- Domain knowledge and problem solving knowledge has to be independent of the mechanism that controls the resolution of the problem
- Heuristic knowledge has to be incorporated in the resolution process (incomplete, approximate, non systematic).
- A tight interaction with the user/environment/other systems has to be allowed

Today

- KBS: from Expert systems to KBS
- Characteristics of KBS
- Research Areas involved in KBS
- Necessity of KBS
- Problems of KBS
- Components of KBS
- KBS on production systems
- Cased Based Reasoning
- Other Methodologies
  - Systems based on Neural networks
  - Intelligent Agents and multiagent systems
- Development process of KBS
- Methodologies for problem solving
- Constructive Problem Solving

Research areas involved in KBS

The contribution of many artificial intelligence areas are needed to give to KBS the mentioned characteristics and abilities, among them:

- Knowledge representation (domain knowledge, problem solving knowledge, heuristics, ...)
- Reasoning and inference (Predicate calculus, default reasoning, uncertainty, time, ...)
- Heuristic search/problem solving
- Natural language processing (interfaces)
- Machine learning (automatic domain knowledge acquisition, adaptation, ...)

Today

- KBS: from Expert systems to KBS
- Characteristics of KBS
- Research Areas involved in KBS
- Necessity of KBS
- Problems of KBS
- Components of KBS
- KBS on production systems
- Cased Based Reasoning
- Other Methodologies
  - Systems based on Neural networks
  - Intelligent Agents and multiagent systems
- Development process of KBS
- Methodologies for problem solving
- Constructive Problem Solving

Necessity of KBS

- To have systems that make available the knowledge of highly qualified experts
- To help/train experts/non experts
- To preserve expert knowledge
- To obtain quickly solutions supported by explanations
- To process great volumes of information
- To have systems that can make decisions autonomously

When problems need KBS to be solved?

- The problem is complex enough to justify the development costs
- We have a correct assessment of the size of the problem
- We have available expert knowledge in order to solve the problem
- The problem can be described as a reasoning process
- It is possible to adequately structure the knowledge and the solving process
- Can not be solved by means of traditional techniques
- We have available cooperative experts

Today

- KBS: from Expert systems to KBS
- Characteristics of KBS
- Research Areas involved in KBS
- Necessity of KBS
- Problems of KBS

- Components of KBS
- KBS on production systems
- Cased Based Reasoning
- Other Methodologies
  - Systems based on Neural networks
  - Intelligent Agents and multiagent systems
- Development process of KBS
- Methodologies for problem solving
- Constructive Problem Solving

Problems of KBS

- Brittleness
- The reasoning process is hard to control
- Low reusability of the knowledge and problem solving strategies
- Sometimes it is difficult to integrate machine learning techniques in the system
- To acquire knowledge from the experts is an expensive and hard task
- To validate the soundness and completeness of the system is difficult

Areas of problems that use KBS

- All kinds of problems that use specialized expert knowledge
- Applied to uncountable domains (medicine, engineering, weather forecasting, banking, ...)
- Problems that involve analyzing evidences (interpretation, diagnostics, supervision, prediction, ...)
- Problems that involve building a solution (design, planning, configuration, ...)

Today

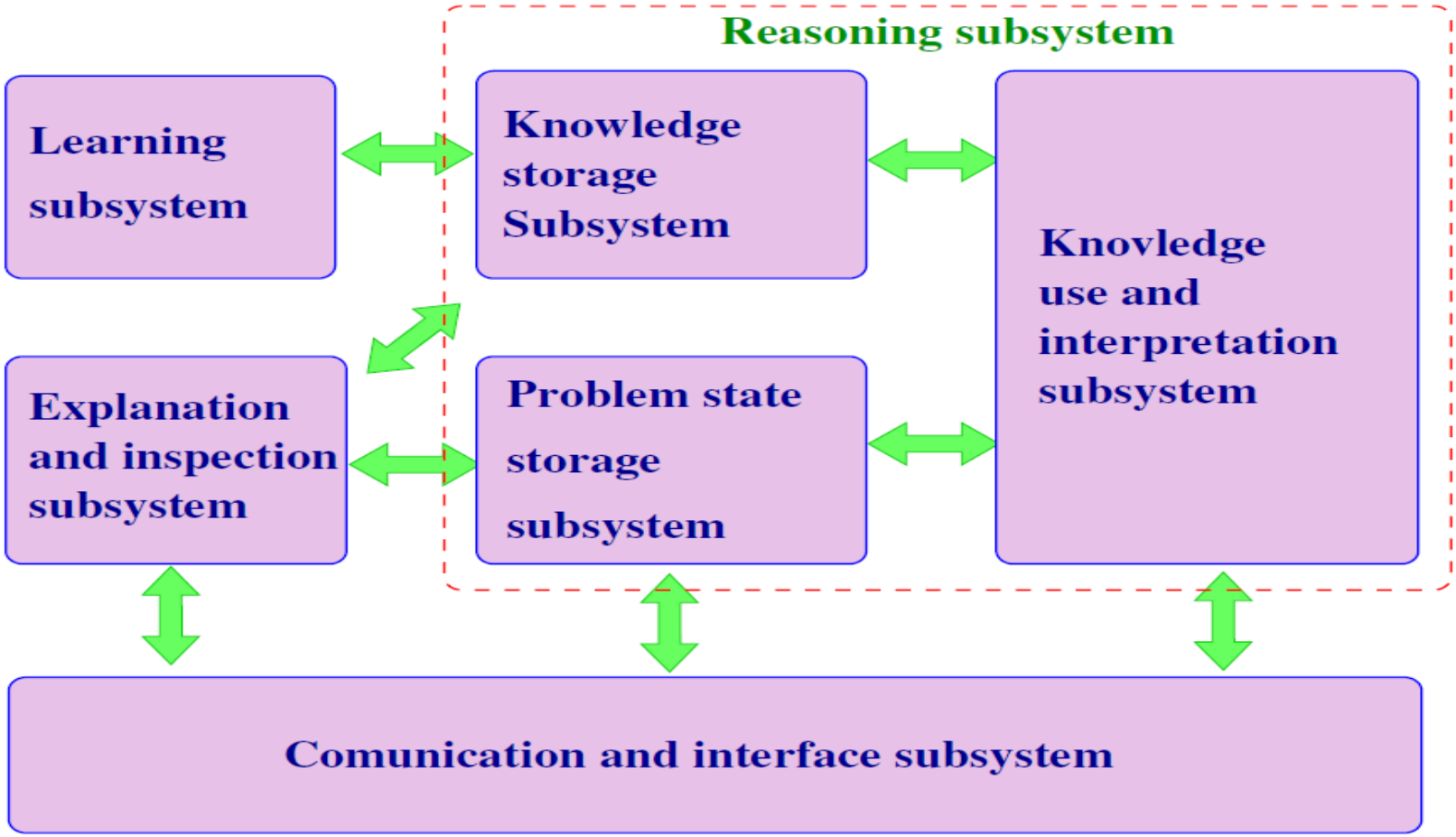
- KBS: from Expert systems to KBS
- Characteristics of KBS
- Research Areas involved in KBS
- Necessity of KBS
- Problems of KBS
- Components of KBS

- KBS on production systems
- Cased Based Reasoning
- Other Methodologies
  - Systems based on Neural networks
  - Intelligent Agents and multiagent systems
- Development process of KBS
- Methodologies for problem solving
- Constructive Problem Solving

Components of KBS

- We have to build systems with specific characteristics:
  - Problem solving using symbolic information
  - Resolution by mean of reasoning and heuristic methods
  - Capacity to explain the resolution process
  - Interactivity (with a user/the environment)
  - Able to adapt to the environment
- A basic set of components is needed
  - Reasoning subsystem
    - Knowledge storage subsystem (Knowledge base)
    - Knowledge use and interpretation subsystem (solving mechanism)
    - Problem state storage subsystem (working memory)
  - Explanation and resolution inspection subsystem
  - Communication and interface subsystem
  - Learning subsystem

Components of KBS



Today

- KBS: from Expert systems to KBS
- Characteristics of KBS
- Research Areas involved in KBS
- Necessity of KBS
- Problems of KBS
- Components of KBS
- KBS on production systems

- Cased Based Reasoning
- Other Methodologies
  - Systems based on Neural networks
  - Intelligent Agents and multiagent systems
- Development process of KBS
- Methodologies for problem solving
- Constructive Problem Solving

KBS on production systems

- Problems are solved using the reasoning mechanism of an inference engine
- The domain knowledge is described by means of an ontology
- The problem solving knowledge is described using production rules or an equivalent formalism

Knowledge storage subsystem

- It will store all the knowledge needed to solve problems in the domain of application
- We can find three kinds of knowledge:
  - **Factual knowledge** (domain objects and their characteristics)
  - **Relational knowledge** (relations among the domain objects)
  - **Conditional knowledge** (Deductive knowledge about the problem)
- The two first kind of knowledge are described in the domain ontology
- The third one represents the knowledge related to the resolution of the problem and is described by production rules

Knowledge storage subsystem: Rules

- Conditional knowledge includes:
  - **Deductive knowledge** (structural): Describes the process of problem solving as a chain of deductions
  - **Knowledge about goals** (strategic): Drives the resolution process to the solution
  - **Causal knowledge** (supportive): Supports explanation of the problem resolution process
- **Rule modules:** Sets of related rules
  - Eases the development and maintaining of the system
  - Increases the efficiency of the reasoning process
  - Allows to implement strategies for the use of the knowledge (meta-knowledge, meta-rules)

Knowledge storage subsystem: Meta-Rules

- They describe high level knowledge about the process of solving the problem
- They allow to take control of the resolution process
  - Activating and deactivating rules/modules
  - Deciding the application order of rules/modules
  - Deciding resolution strategies, handling of exceptions, uncertainty, ...
- This knowledge is more difficult to obtain from the experts

Knowledge use and interpretation subsystem

- Usually is an inference engine
- It applies its execution cycle to solve the problem
  - Detection of applicable rules
  - Selection of the best rule (Domain independent strategy or driven by metaknowledge)
  - Execution of the rule

Problem state storage subsystem

- Stores the problem initial facts and all the facts obtained during the resolution process
- It can store also other kinds of information needed for the control of the resolution process or other subsystems
  - Order of deduction of the facts
  - Use preferences of the facts
  - Rule that produced each fact
  - Rules recently fired
  - Backtracking points
  - ...

Explanation and resolution inspection subsystem

- The possibility to justify the decisions taken during the resolution process gives credibility to the system
- It allows also to detect wrong assumptions or decisions during the process
- A system should be able to answer **why** and **how**
- Different levels of explanation:
  - **Trace:** A list of the steps of the resolution is given
  - **Justification:** The reason why each element that appear in the trace of the resolution is given (reasoning path, questions, facts, preferences, subgoals, ...)

Learning subsystem

- Usually the set of problems that a KBS solves is closed
- In some domains it is necessary to adapt to the environment and to solve new problems
- Learning can happen:
  - During the development process of the KBS: The knowledge engineering process is substituted or complemented by inductive learning methods, building a model of the problem from examples
  - During the resolution process: Incorrect solutions are detected and corrected, new rules that make more efficient the resolution process are learned

Today

- KBS: from Expert systems to KBS
- Characteristics of KBS
- Research Areas involved in KBS
- Necessity of KBS
- Problems of KBS
- Components of KBS
- KBS on production systems
- Cased Based Reasoning

- Other Methodologies
  - Systems based on Neural networks
  - Intelligent Agents and multiagent systems
- Development process of KBS
- Methodologies for problem solving
- Constructive Problem Solving

Cased Based Reasoning (CBR)

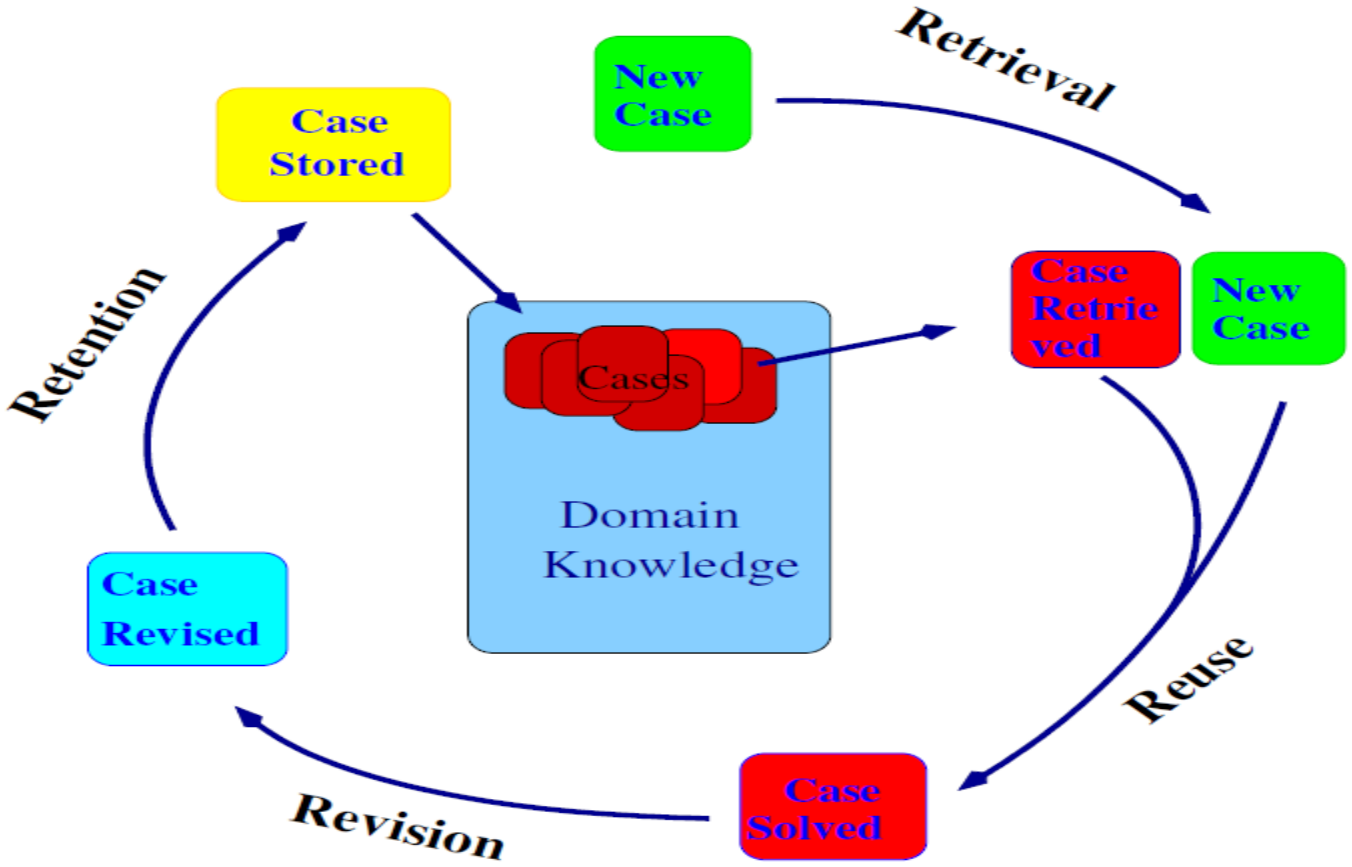
- The solution of a problem is obtained identifying a previous similar solution
- Advantages:
  - The problem of knowledge acquisition is reduced
  - It is easier to maintain/correct/extend the system
  - The solving process is more efficient
  - It allows to obtain explanations closer to the user experience

Execution cycle

It has four phases

1. **Retrieval:** The more similar stored cases are retrieved
2. **Reuse:** The solution of the cases are obtained
3. **Revision:** The retrieved solution is evaluated and adapted
4. **Retention:** The relevance of the new case is assessed and the case is stored if it is an interesting one

Execution cycle



Knowledge storage subsystem

- The knowledge is composed by **cases**
- A case is a complex structure (characteristics, solution)
- The cases are stored in the **case base** (structure, indexing)
- We will have also knowledge about:
  - How to evaluate case similarity
  - How to combine/adapt/retrieve solutions
  - How to evaluate solutions

Knowledge use and interpretation subsystem

- It is based on the execution cycle of Cased Based Reasoning
  - Retrieve the more similar cases from the case base
  - Obtain the solution from the cases
  - Combine/adapt the solution (procedures/reasoning)

Problem state storage subsystem

- Information of the current case
- Most similar cases retrieved
- Reasoning results from evaluating/combining/adapting the solutions

Explanation - Learning

- **Explanation**
  - It is a part of the information of a case
  - This explanation will be complemented by the reasoning process to combine/adapt the solutions
- **Learning**
  - To add new cases (easier than in production systems)
  - The new case has to be different enough (evaluation)
  - Cases could be forgotten (low usage, similar to others)

Today

- KBS: from Expert systems to KBS
- Characteristics of KBS
- Research Areas involved in KBS
- Necessity of KBS
- Problems of KBS
- Components of KBS
- KBS on production systems
- Cased Based Reasoning
- Other Methodologies
  - Systems based on Neural networks
  - Intelligent Agents and multiagent systems
- Development process of KBS
- Methodologies for problem solving
- Constructive Problem Solving

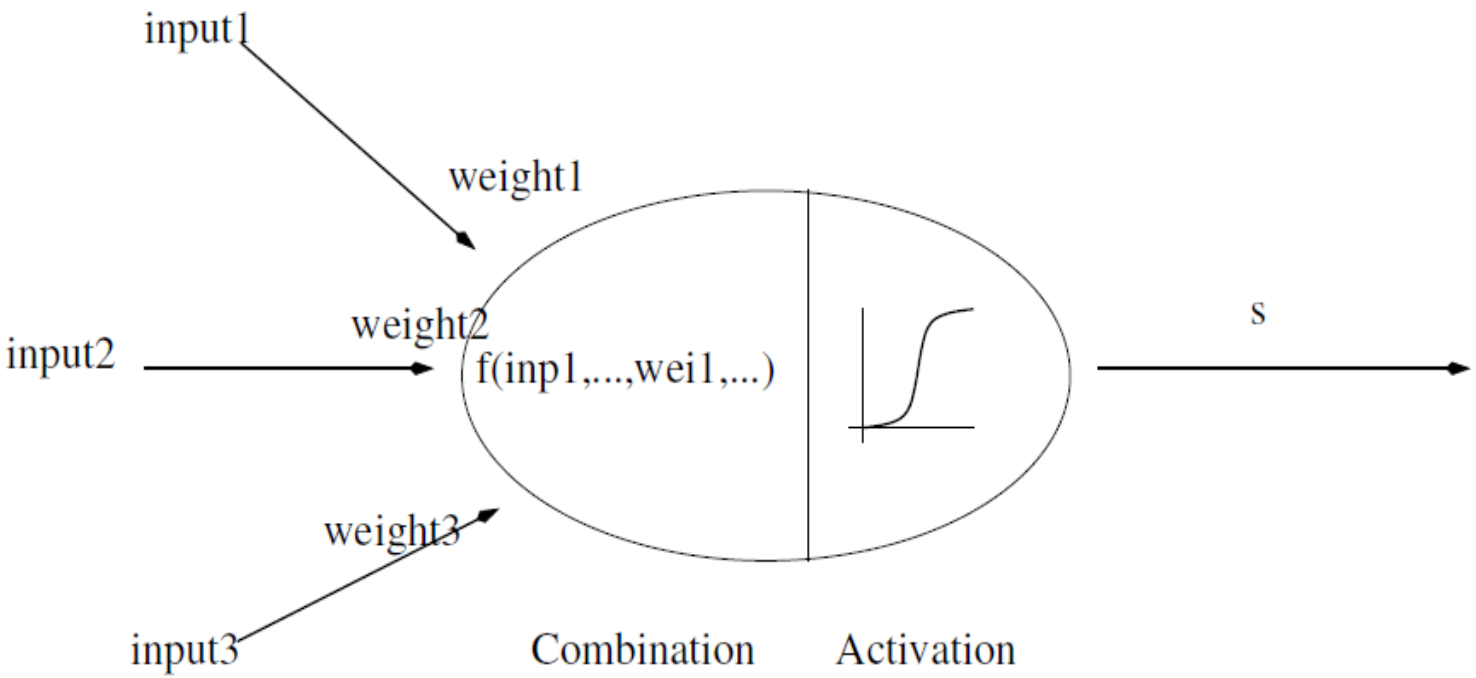
Other Methodologies

- Systems based on neural networks
- Intelligent Agents/Multiagent systems

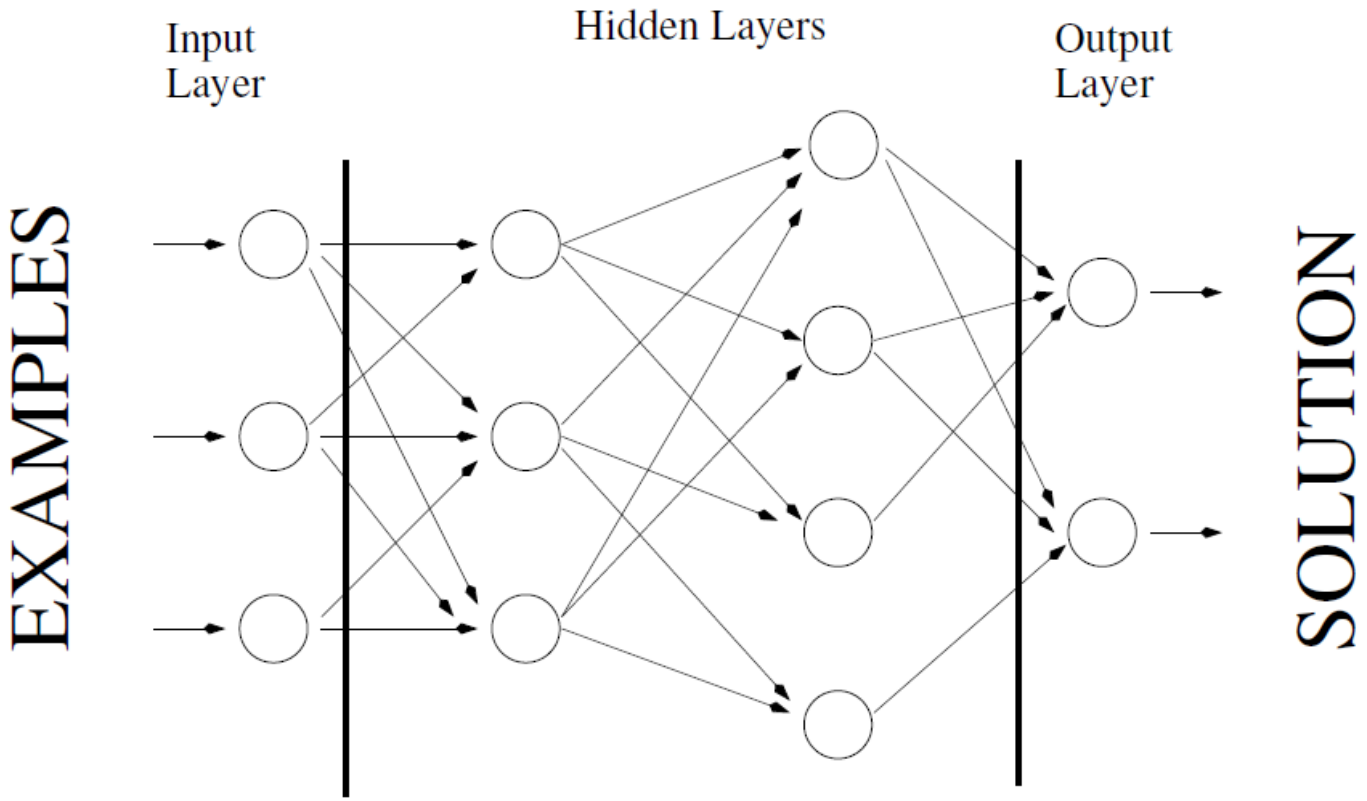
Neural networks

- Conexionist artificial intelligence
- The base element is the neuron (computational element)
- **Neuron:** Inputs, outputs, state, functions for the combination of inputs and state and function to compute the output
- Neurons are organized in networks with layers
- A neural network associates inputs (problem description) to outputs (solution of the problem)
- A neural network has to be trained (using solved problems) in order to learn how to solve the problem (association)

Neural networks

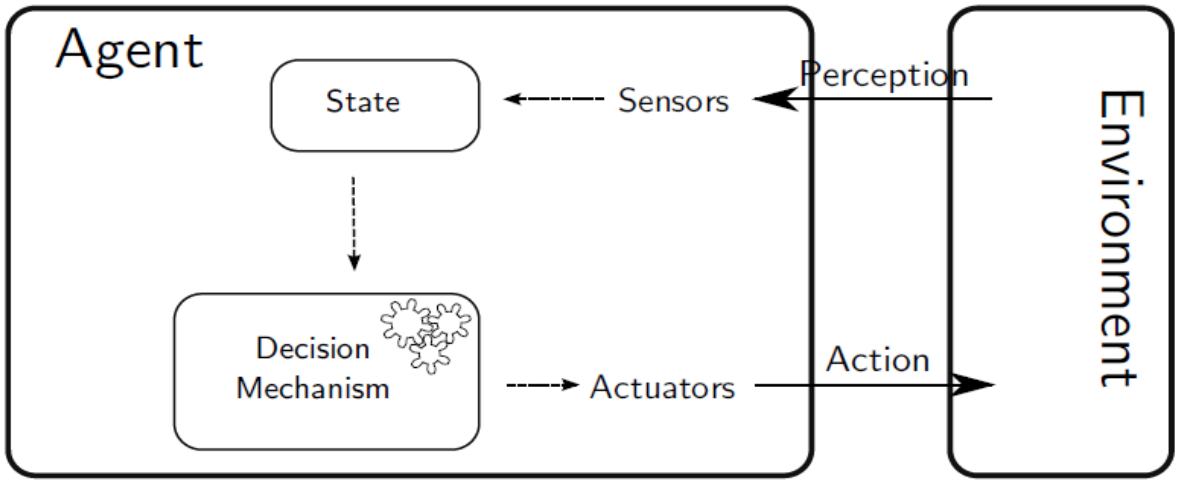


Neural networks



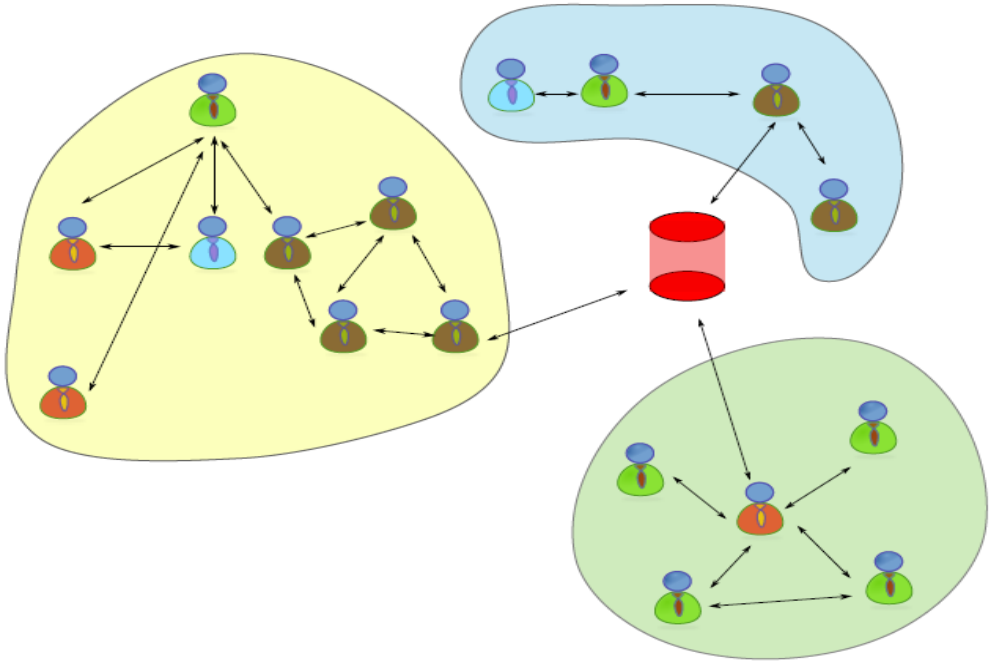
Intelligent Agents/Multiagent systems

- It supposes a collective vision of intelligent systems (instead of monolithic)
- An intelligent agent solves only a *simple* task
- An agent:
  - Obtains information from the environment (peception)
  - Elaborates a decision based on its perceptions and state (reasoning)
  - Performs an action (actuation)



Intelligent Agents/Multiagent systems

- The global problem is solved by cooperation/coordination
- Techniques involved: organization, cooperation, coordination, negotiation, tasks distribution, communication, reasoning about others, ...



Intelligent Agents/Multiagent systems

- Advantages:
  - More flexible systems
  - Reconfiguration/reorganization allow to solve other tasks
    $\Longrightarrow$ Agents act as reusable components
  - Fault tolerance (an agent can be substituted by other)
  - Distributed computing
- Related to:
  - Grid computing/Cloud computing (Management of tasks and resources)
  - Web services (No reasoning capabilities)

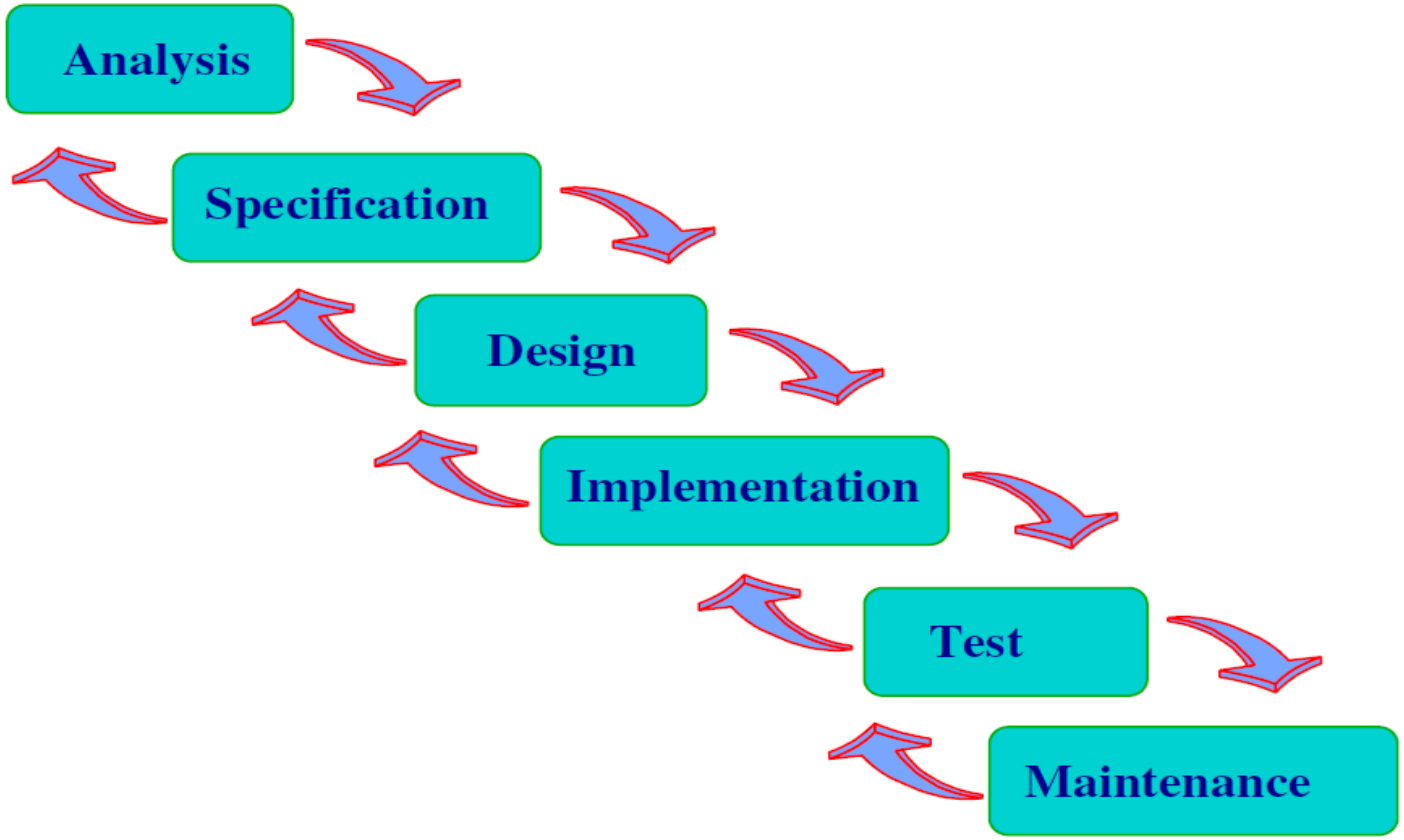
Today

- KBS: from Expert systems to KBS
- Characteristics of KBS
- Research Areas involved in KBS
- Necessity of KBS
- Problems of KBS
- Components of KBS
- KBS on production systems
- Cased Based Reasoning
- Other Methodologies
  - Systems based on Neural networks
  - Intelligent Agents and multiag[illegible]ystems
- Development process of KBS
- Methodologies for problem solving
- Constructive Problem Solving

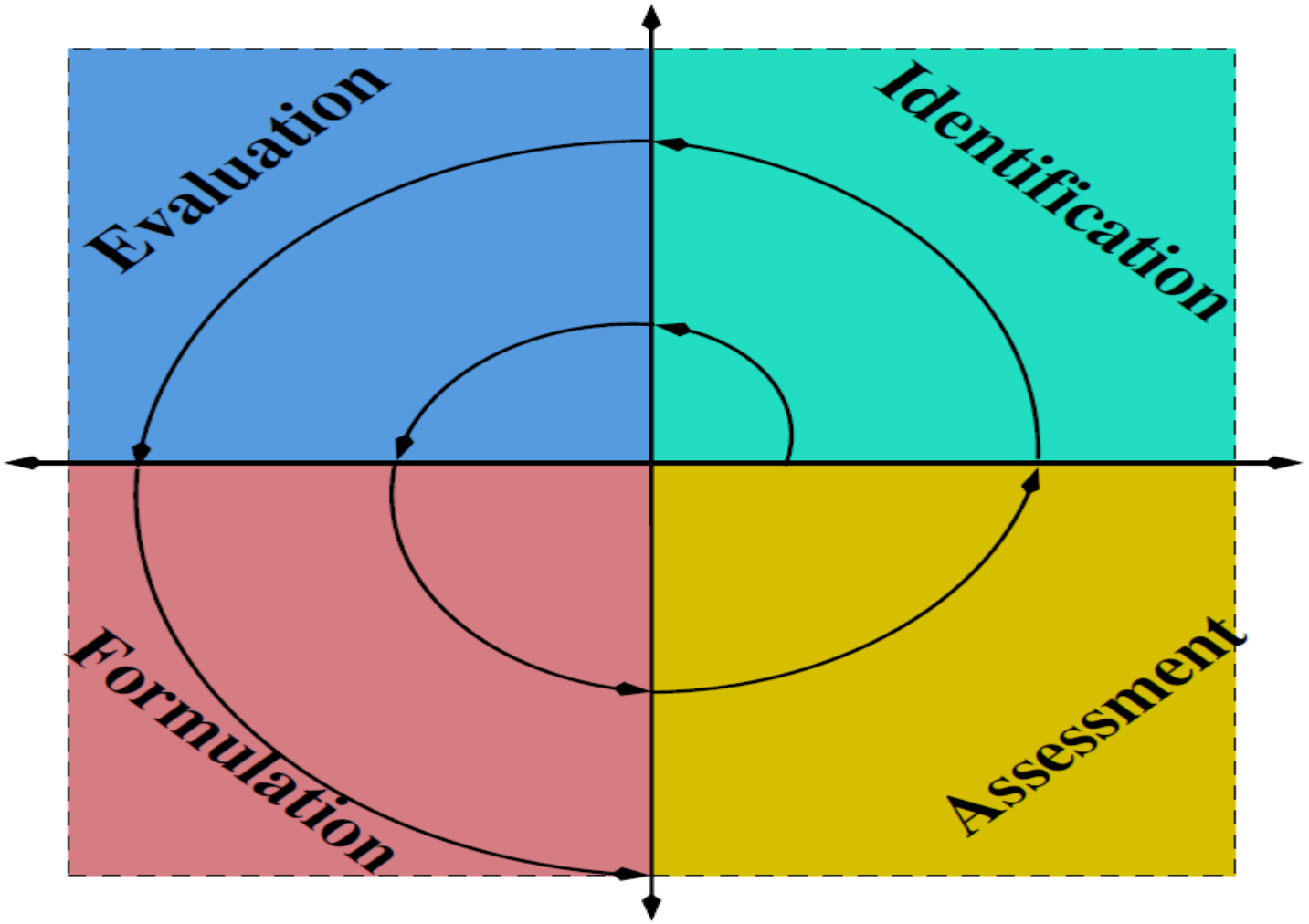
Development of KBS

- The most important phase in the development of a KBS is knowledge elicitation (knowledge extraction))
- This requires the interaction between the **Knowledge Engineer** and the expert
- The methodologies of software engineering have to include this step in their development process
- The methodologies of software engineering have to be adapted to the specific needs of KBS

SI: Waterfall Model



SI: Spiral Model



Particularities of the KBS

- Conventional software systems $\Longrightarrow$ Known and common algorithms
- KBS $\Longrightarrow$ incomplete, uncertain and heuristic knowledge
- Conventional software systems $\Longrightarrow$ It is possible to give an estimate of what knowledge is needed and its volume
- KBS $\Longrightarrow$ It is difficult to know what knowledge will be needed and its volume

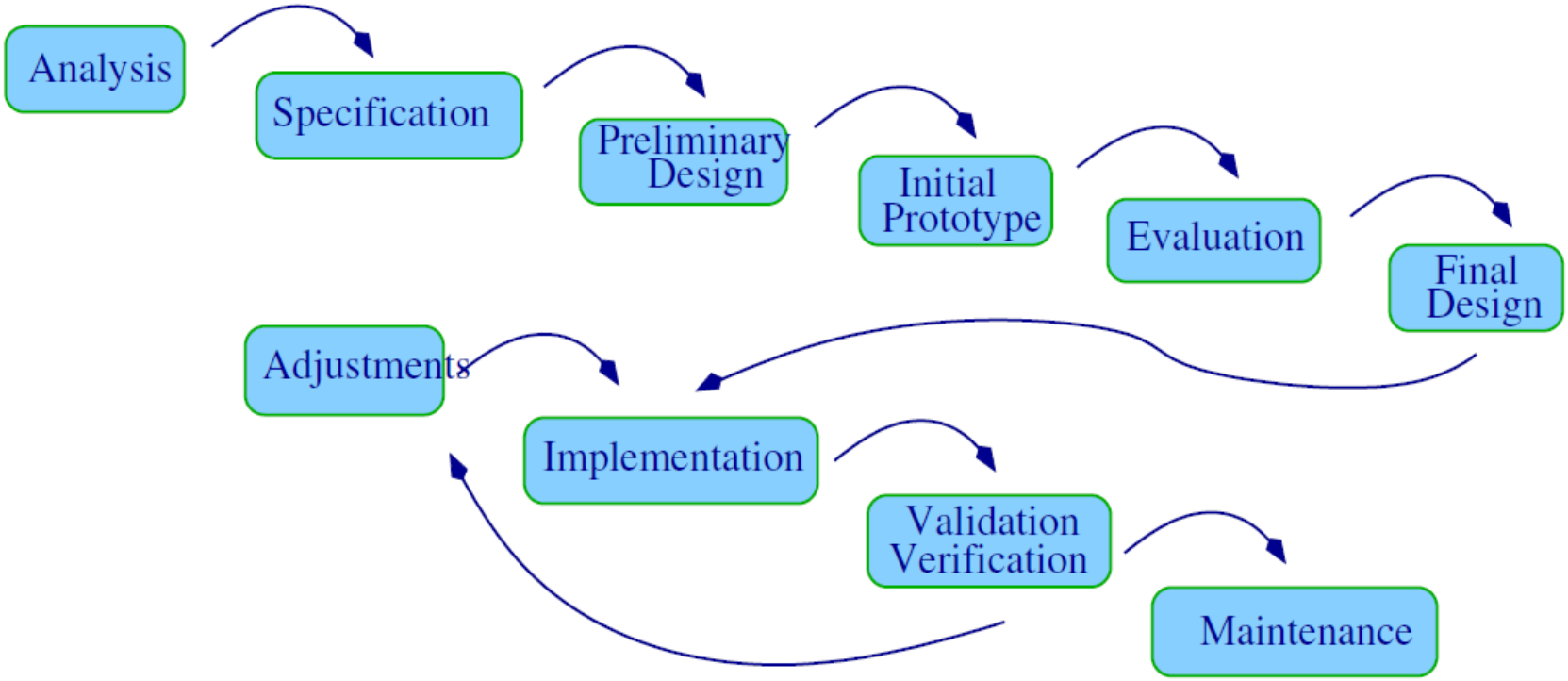
Particularities of the KBS

- It is difficult to obtain a clear design at the early stages of the development
- Wrong initial decisions imply radical changes in the design during the development
- The **knowledge engineer** has to perform the task of knowledge acquisition $\Longrightarrow$ Interviews with the experts
- Problems:
  - The KE has to learn the basic principles of the domain
  - To find a knowledge representation formalism that the expert can understand
  - The experts prefer to reason from particular cases rather than from general definitions
  - It is difficult for the experts to detail explicitly their knowledge (expert's paradox)

Particularities of the KBS

- Solution: **Incremental design and rapid prototyping**
- Goal: To develop a functional prototype that has all the basic functionalities of the system
- The analysis and the specification has to take in account the whole system
- The design and implementation are limited to the initial prototype
- This prototype is completed incrementally
- Advantage: A functional system is always available during all the development process

The life cycle of a KBS



The life cycle of a KBS (I)

1. **Analysis of the problem:** Gather information about the project and assess its viability
2. **Requirements specification:** Determine the goals of the project and the methodologies to achieve them
3. **Preliminary design:** High level decisions about the design (Knowledge representation formalism, tools, sources of knowledge)
4. **Initial prototype and evaluation:** Build a prototype with limited coverage, evaluate the design decisions from the prototype
5. **Final design:** Validate the decisions and propose a design that allows an incremental development

The life cycle of a KBS (II)

6. **Implementation:** Complete the knowledge acquisition process and complete incrementally the initial prototype
7. **Validation and verification:** Test and validate that the system works accordingly to specifications
8. **Design adjustments:** Apply the feedback from the tests (Design changes should be minimal)
9. **Maintaining:** Maintain the system.

Specialized Methodologies

- **CommonKADS**
  - Spiral life cycle and modeling using UML-like formalism
  - Six models are built: Organization, tasks, agents, communication, knowledge and design.
- **MIKE**
  - Spiral life cycle: Knowledge acquisition (acquisition model and knowledge structure model), design, implementation, evaluation.

A simplified methodology

- For developing small KBS a methodology that follows the waterfall model can be applied
  1. **Problem identification**
  2. **Conceptualization**
  3. **Formalization**
  4. **Implementation**
  5. **Validation and Test**

Identification

- We have to assess if the problem is adequate for developing a KBS
  - Is there an algorithmic solution?
  - Are there available knowledge sources?
  - Is the size/goal/complexity of the problem adequate?
- Search and evaluate knowledge sources
- Determine what knowledge is necessary to build the system
- Determine the goals of the system (what do we expect from it?)

Conceptualization

This phase will give us a perspective of the problem from the expert point of view

- We have to:
  - Determine the elements of the domain $\Longrightarrow$ Informal description of the ontology
  - Decompose the problem in subproblems by means of successive refinement, discovering the reasoning blocks
  - Detail the flow of reasoning and the inputs and outputs of each subproblem
  - Detail and distinguish among evidences, hypothesis and actions and discover their relations
- All this information has to be acquired by means of interviews with the experts and from the knowledge sources
- The result will be a semiformal model of the domain, the subproblems and the problem solving methods used by the expert

Formalization

This phase will transform the perspective of the problem from the expert point of view to the knowledge engineer point of view

- Decide the knowledge representation formalism that is more adequate
- Identify the problem search space
- Analyze the subproblems typology and the reasoning blocks and decide what methodologies for problem solving are more adequate
- Analyze the need for the treatment of uncertainty or incomplete information

Implementation

- Build the ontology of the domain
- Embed the problems identified inside the chosen methodologies for problem solving
- Implement the different modules corresponding to each problem using the knowledge acquired
- If an approach based on rapid prototyping is used, the development will start with an initial prototype that will be incrementally completed

Validation and Test

- Some representative cases will be chosen and solved using the system
- These should include cases used to build the systems and also new cases
- If the development is done incrementally this phase will be repeated each implementation cycle
- The validation of a KBS is more complex than conventional software systems

Classification of problems

- The identification of a typology of problems that can be solved using KBS helps to development
- Each type allows to identify:
  - The most common tasks
  - A set of specific problem solving methodologies
  - Adequate methods for knowledge representation and inference
- Two generic tasks will be used to classify the problems that can be solve by KBS:
  - **Analysis tasks:** Interpretation of a system
  - **Synthesis tasks:** Construction of a system

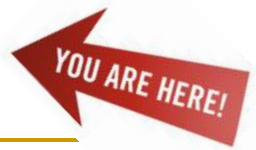
Synthesis - Analysis

Both can be specialized

- Analysis tasks
  - Identification, tells what kind of system we have
    - Monitorization, detects discrepancies on behaviour
    - Diagnostics, explains discrepancies
  - Prediction, tells what kind of output can be expected
  - Control, determines what inputs allow certain outputs
- Synthesis tasks
  - Specification, searchs for the constraints that have to be satisfied
  - Design, generates a configuration of elements following some constraints
  - Assembling, builds a system putting together elements

Today

- KBS: from Expert systems to KBS
- Characteristics of KBS
- Research Areas involved in KBS
- Necessity of KBS
- Problems of KBS
- Components of KBS
- KBS on production systems
- Cased Based Reasoning
- Other Methodologies
  - Systems based on Neural networks
  - Intelligent Agents and multiagent systems
- Development process of KBS
- Methodologies for problem solving
- Constructive Problem Solving



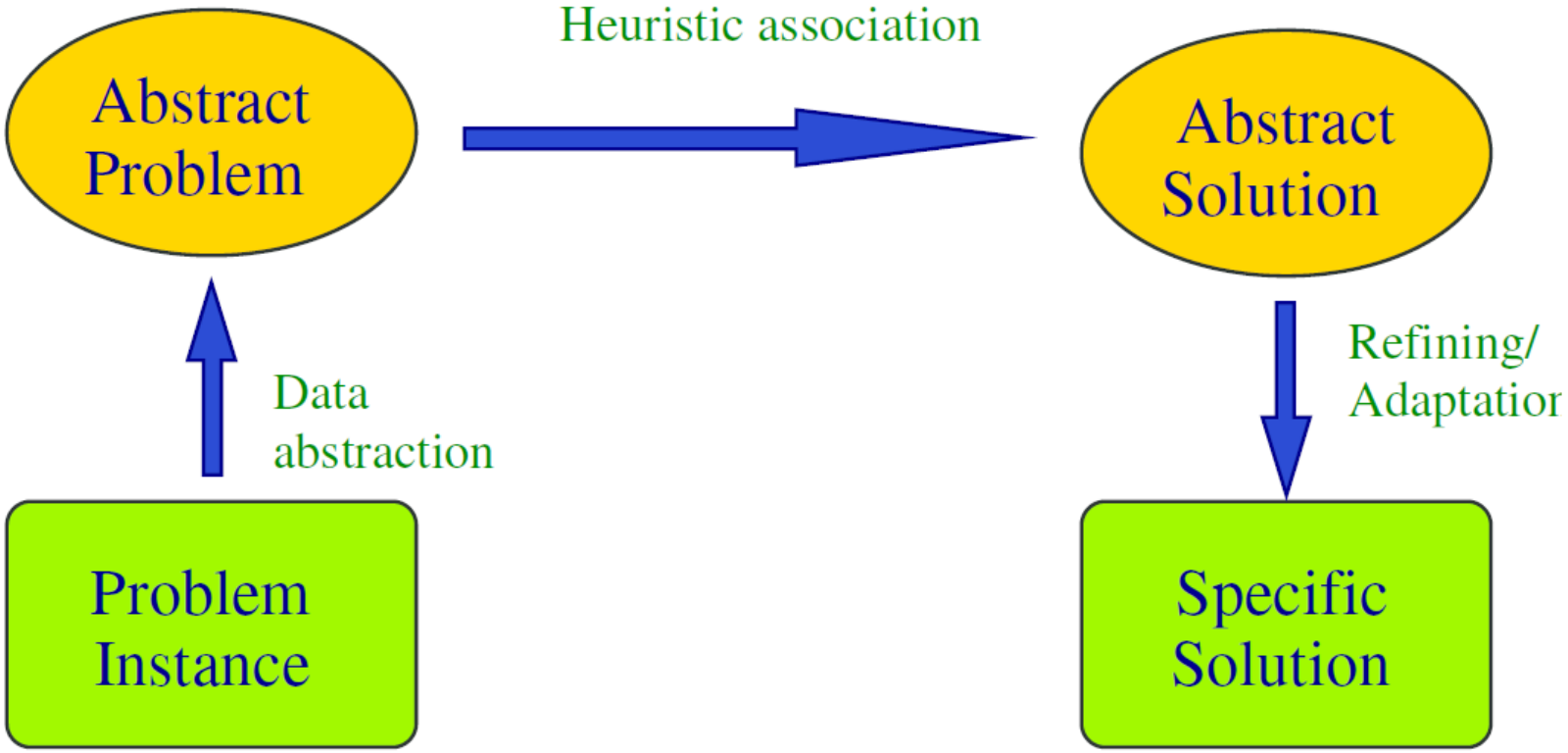
Methodologies for problem solving

- Each one of the generic problems has special characteristics
- There are specific problem solving methodologies for each kind
- We will talk about two methodologies:
  - *Heuristic Classification*
  - *Constructive Problem Solving*

Heuristic classification

- It is used for **analysis** tasks
- The goal is to choose a solution from a limited set of solutions
- Input data is associated with the solutions (simple direct association or using reasoning)
- Three phases:
  1. **Data Abstraction** (Definitional, qualitative, generalization, ...)
  2. **Heuristic Association**
  3. **Refining**

Heuristic classification



Heuristic classification : Knowledge acquisition

- The acquisition of the knowledge to solve problems that use heuristic classification can be done systematically
- Three kinds of concepts can be distinguished:
  - **Hypothesis**: Possible solutions to our problem
  - **Symptoms**: Characteristics that describe the hypothesis
  - **Initial causes**: Initial information about the problem that lead to the Symptoms
- From each set of concepts we need to obtain the set of deductions that go from one to another
- From the initial causes to the symptoms we have the abstraction rules
- From the symptoms to the hypothesis we have the heuristic association rules

Heuristic classification : Knowledge acquisition

- For each set of concepts we have to:
  - See what concepts from the first set (antecedents) are associated with concepts of the second set (consequents)
  - Choose as antecedents of the rules the concepts that are more specific to each consequent (separability)
  - If it is necessary new intermediate concepts should be added to link the consequents and antecedents and to create the needed chains of deduction
  - Observe the confidence of the association between antecedent and consequent (uncertainty)
- If the hypothesis are abstract solutions $\Longrightarrow$ Determine rules of solution refining

Heuristic classification: Example (1)

- We want to develop a KBS to assess bank loan requests for creating a business
- There is a limited set of solutions (accept or decline)
- The goal is to decide, given the client characteristics, if the request is accepted and under what conditions, or if the request is declined
- This is an **analysis** problem that can be solved using **Heuristic Classification**.

Heuristic classification: Example (2)

Lets suppose that a loan request has the following information:

- If the client has bank guarantees.
- If some relative of him guarantees the loan.
- If he has accounts, houses, cars, other properties and their value
- If he has antecedents of unpaid debts
- If he has signed bad checks
- If he has loans already granted
- Kind of business that the client want to create
- Amount of money that he is asking for

Heuristic classification: Example (3)

Determine the set of characteristics that define the abtract problems

- Financial guarantees (Very good, good, normal, bad, very bad)
- Assets
- Fiability of the loan
- Compromise with the client
- Viability of the business

Heuristic classification: Example (4)

Determine the set of abstract solutions

- Decline
- Grant
- Grant but reducing the amount
- Grant and give a preferential interest rate

Heuristic classification: Example (5)

Determine the rules that abstract from the problem data

- if *guarantees* > a million euros or rich uncle then *financial guarantees*=good
- if *guarantees* < 100000 euros then *financial guarantees* bad
- if sum of *assets* < a million then *assets*=bad
- if sum of *assets* > two millions then *assets*=good
- if *bad checks* or *unpaid debts* then *fiability*=very bad
- if *fast food business* or *ice scream business* then *viability*=normal
- if *chain of stores* or *ISP* then *viability*=very good
- if *loans already granted* > a million or *brother of branch director* then *compromise*=good
- ...

Heuristic classification: Example (6)

Determine rules that associate characteristics to solutions

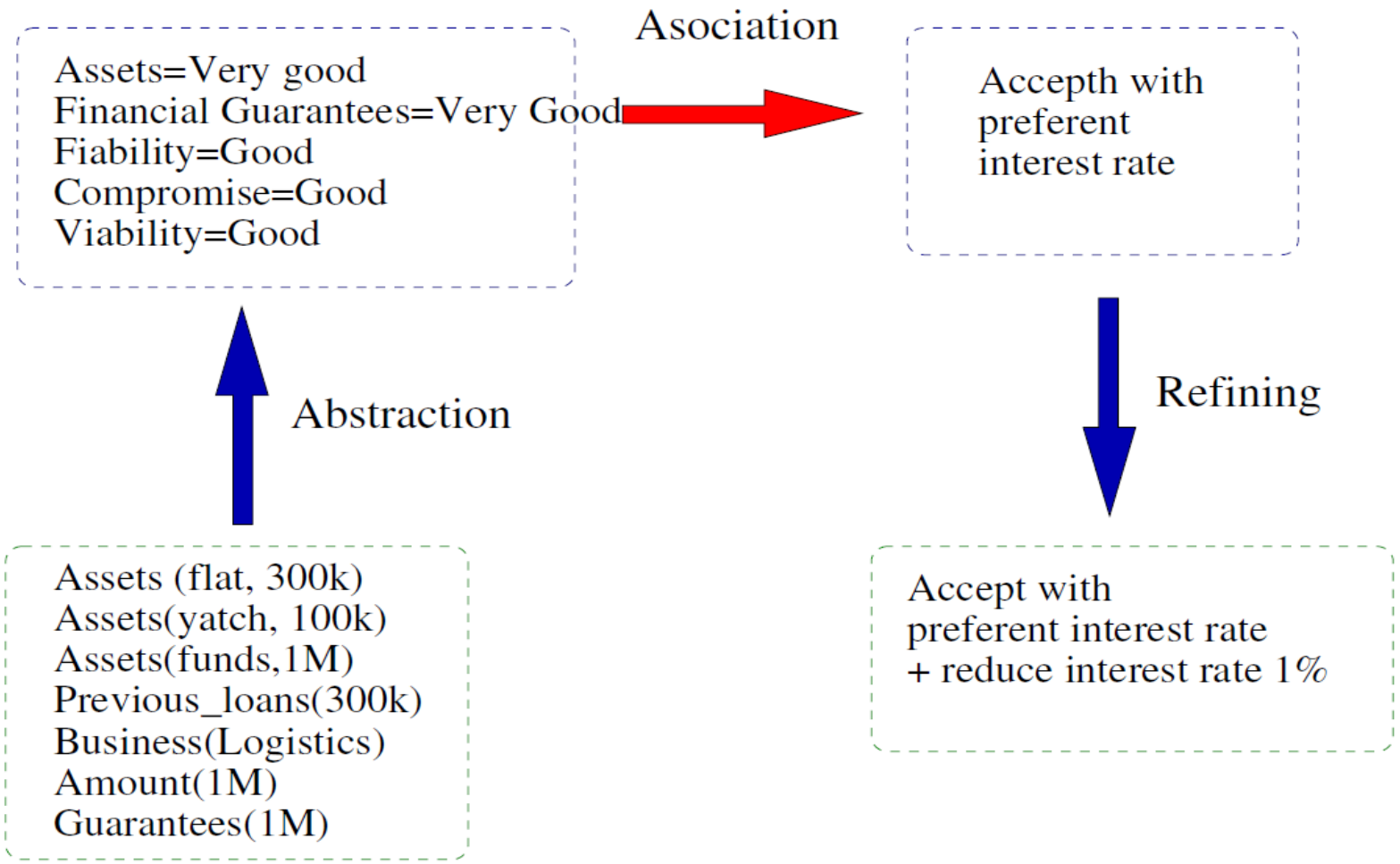
- if *financial guarantees*=normal and *assets*=bad then *decline*
- if *fiability*={bad, very bad} then *decline*
- if *financial guarantees*=normal and *assets*=normal and *viability*=good then *grant but reducing*
- if *financial guarantees*=good and *assets*=normal and *compromise*=normal and *viability*=good then *grant*
- if *financial guarantees*=good and *assets*=good and *compromise*=very good and *viability*=very good then *grant with preferential interest rate*
- ...

Heuristic classification: Example (7)

Determine rules to refine the solutions

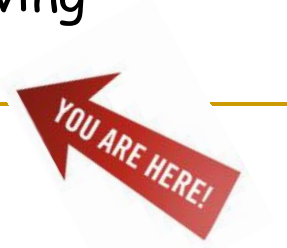
- if *grant but reducing* and *amount* $>$ 500000 euros and *assets* $=$ 500000 euros then *reduce* to 500000 euros
- if *grant with preferential interest rate* and *amount* $>$ a million and *assets* $>$ a million then *reduce* 1 % the interest rate
- if *grant with preferential interest rate* and *brother of branch director* then *reduce* 2 % the interest rate
- ...

Heuristic classification: Example (8)



Today

- KBS: from Expert systems to KBS
- Characteristics of KBS
- Research Areas involved in KBS
- Necessity of KBS
- Problems of KBS
- Components of KBS
- KBS on production systems
- Cased Based Reasoning
- Other Methodologies
  - Systems based on Neural networks
  - Intelligent Agents and multiagent systems
- Development process of KBS
- Methodologies for problem solving
- Constructive Problem Solving



Constructive problem solving

- Used for problems that have an unbounded number of solutions
- The resolution implies to build the solution from a set of elements (actions, components, failures, ...)
- It is applied to **synthesis** tasks
- Heuristic and local search methods can be used but with impracticable time cost

Constructive problem solving

- To build a solution we need knowledge about:
  - The model that describes the structure of the solution
  - The model of behaviour of the components of the solution
  - The actions that allow to build the solution
  - The set of constraints between the components and the solution
  - How to evaluate the decisions about the actions needed to build the solution and about the quality of the solution (complete or incomplete) itself
- The constraints could be:
  - About the configuration of the components (Physical, temporal, ...)
  - About the inputs/outputs/preconditions/postconditions of the actions used to build the solution
  - Interactions among the previous constraints

Methods for Constructive problem solving

- **Propose and apply**: We start from an empty solution. Iteratively an action that allows to extend the current partial solution is selected until the complete solution is achieved
- **Least commitment**: We start from a complete initial solution. Iteratively an action that allow to modify the current solution is selected but guaranteeing that the action will impose the minimal constraints to future actions

Propose and apply

- The search is performed in the space of partial solutions
- We start with and empty or incomplete solution
- Each step extends the solution
- The best action is chosen each iteration
- The search is always inside the space of solutions

Propose and apply

- We need knowledge about:
  - Actions used to solve the problem
  - Constraints and relations among the components of the solution
  - Evaluation of the effects of an action on the solution
  - Evaluation of the quality of the solution
- The process to solve a problem can be done in different ways
  - Sequential resolution (Lots of knowledge are needed to be efficient)
  - Hierarchical problem decomposition (more efficient, but it requires to obtain problem decomposition operators)

Propose and apply: *Algorithm*

1. Initialize the goal: The initial solution is created
2. Propose an action: The possible actions that can be applied on the current solution are selected
3. Prune actions: Actions are pruned using global criteria
4. Evaluate actions: The effects of the actions on the solution are compared and evaluated
5. Select an action: The best action is chosen. If there is no best action backtracking is considered
6. Apply the action: The selected action is applied to the current solution
7. Evaluate the solution: If the current solution is the goal the process stops or another iteration is performed

Least commitment

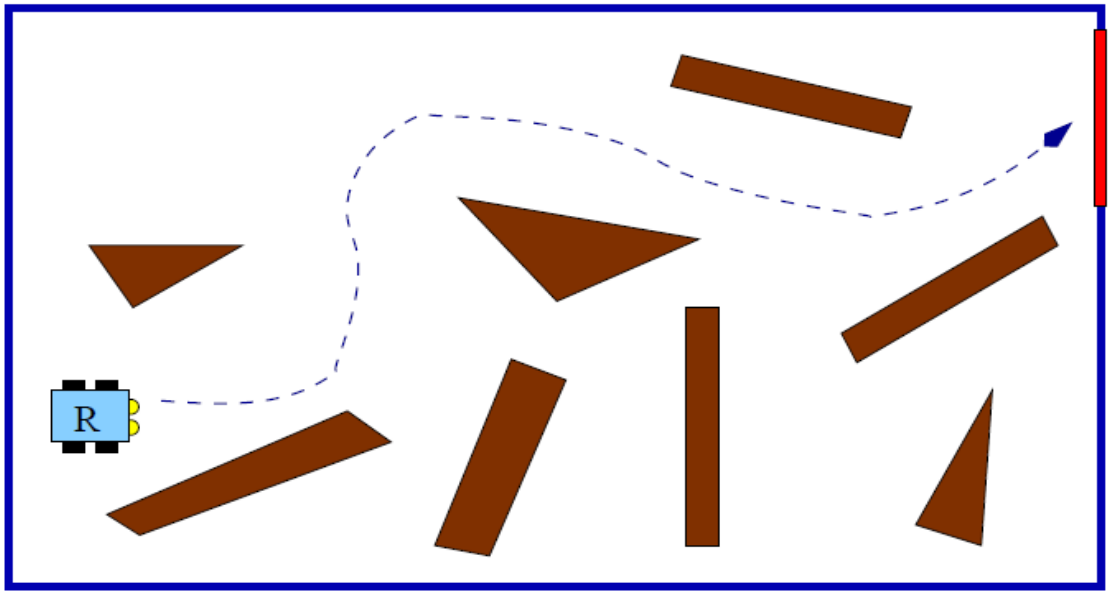
- The space of complete solutions is explored
- The search begins from a solution (it could be a no solution also)
- The solution is modified and corrected
- The choice of actions to apply is defined by the heuristic of least commitment: the minimal modification that impose less future constraints
- The search can pass from the space of solutions to the space of no solutions and viceversa

Least commitment : Algorithm

1. Start with a non optimal solution that satisfies the constraints if possible
2. Modify the solution with an action chosen using the heuristic of least commitment (action that imposes less constraints on the solution)
3. If the modification violates any constraint undo a previous action performing minimal modifications (not necessarily the last action)

Constructive problem solving : Example (1)

- We want to plan the best path for a robot inside a room
- Inside the room there are obstacles that have to be avoided
- We have a set of actions:
  - Move forward or backwards at a specific speed some distance
  - Rotate an specific number of degrees



Constructive problem solving : Example (2)

- Global constraints: Arrive to the exit door, minimal length and time path
- Constraints for the actions: Avoid collisions with obstacles or walls, maintain certain distance to obstacles to maneuver
- Evaluation of the actions:
  - Move: The nearer and faster the robot moves towards the goal the better
  - Rotate: The farther the trajectory of the robot is from the obstacles the better

Today

- KBS: from Expert systems to KBS
- Characteristics of KBS
- Research Areas involved in KBS
- Necessity of KBS
- Problems of KBS
- Components of KBS
- KBS on production systems
- Cased Based Reasoning
- Other Methodologies
  - Systems based on Neural networks
  - Intelligent Agents and multiagent systems
- Development process of KBS
- Methodologies for problem solving
- Constructive Problem Solving

Appendix

Expert Systems

What is an Expert System

"An intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require human expertise for their solutions."

- Professor Edward Feigenbaum, Stanford

"A computer program that embodies the knowledge of an expert."

- Charles Weddle, FSU

Expert Systems

What is an Expert System

- An expert's knowledge is specific to one **problem domain**. A problem domain is medicine, science, engineering, etc.

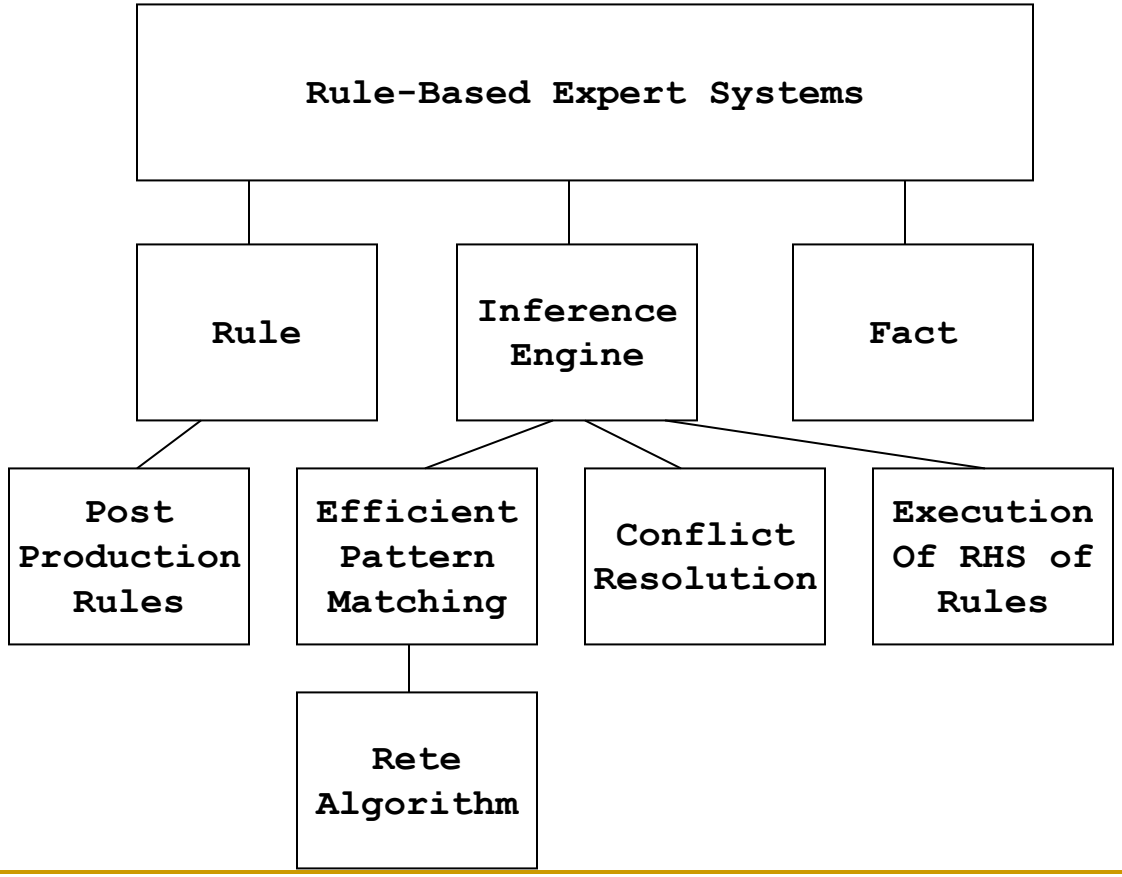
- The expert's knowledge about solving specific problems is called the **knowledge domain**.

- Knowledge is represented in an expert system as **rules**.

- In the knowledge domain that it knows about, the expert system reasons or makes **inferences** to produce the solution of a problem.

Expert Systems

What is an Expert System



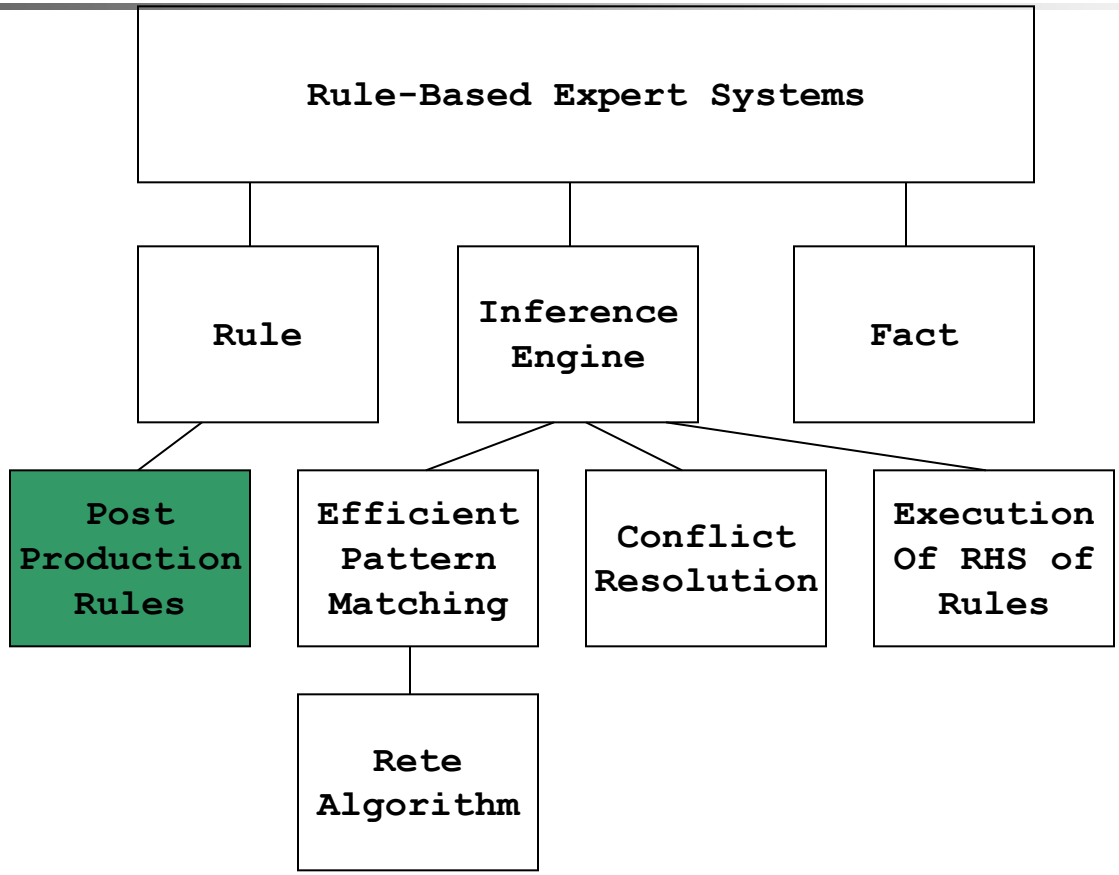
Expert Systems History

Timeline

1943 –	**Post Production rules**
1954 –	**Markov Algorithm for controlling rule execution**
1956 –	Dartmouth Conference; Logic Theorist; Heuristic Search; "AI" Term coined
1957 –	**General Problem Solver (GPS) started (Newel and Simon)**
1958 –	LISP AI Language
1965 –	Work begun on DENDRAL, the first expert system (Feigenbaum, Buchanan, et al.)
1969 –	MACSYMA math expert system (Martin and Moses)
1970 –	Work begins on PROLOG (Colmeraurer, Roussell, et al.)
1971 –	HEARSAY I for speech recognition, *Human Problem Solving* popularizes rules (Newel and Simon)
1973 –	MYCIN expert system for medical diagnosis (Shortliffe, et al.) GUIDON intelligent tutoring (Clancey) TEIRESIAS explanation facility concept (Davis) EMYCIN, first shell (Van Melle, Shortliffe, and Buchanan) HEARSAY II, blackboard model of multiple cooperating experts
1977 –	OPS Expert System Shell (Forgy) used in XCON/R1
1978 –	Work started on XCON/R1 to configure DEC computers (McDermott)
1979 –	**Rete Algorithm for fast pattern matching (Forgy)**
1985 –	**CLIPS expert system tool (NASA)**

Expert Systems History

Post Production Rules



Expert Systems History

Post Production Rules

One of the most Popular types of expert system is the rule-based system. Rules are a type of production system whose origins go back to the 1940's

- Production systems were first used in symbolic logic by Post in 1943.
- Post's basic idea was that any mathematic or logic system is simply a set of rules specifying how to change one string of symbols into another set of symbols.
- That is, given an input string, the antecedent, a production rule could produce a new string, the consequent.

Expert Systems History

Post Production Rules

- The antecedent, a production rule could produce a new string, the consequent.

Antecedent -> Consequent

patient has a fever -> take aspirin

IF patient has a fever THEN take aspirin

- The Post production attaches no meaning to these strings and has no idea what patient, fever, or aspirin represent. A human interprets the strings into meaningful statements in the real world.

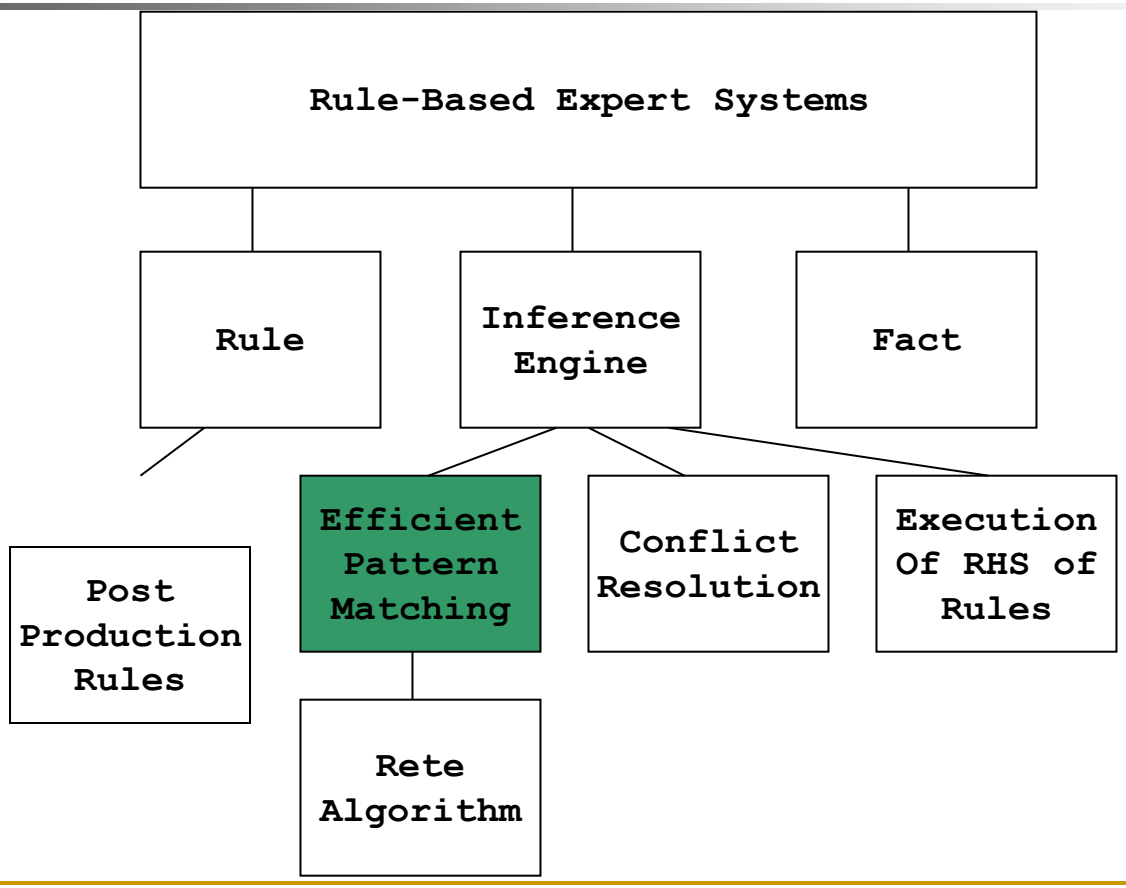
Expert Systems History

Post Production Rules

- *Modular Nature*. This makes it easy to encapsulate knowledge and expand the expert system by incremental development.
- *Explanation Facilities*. It is easy to build explanation facilities with rules because the antecedents of a rule specify exactly what is necessary to activate the rule.
- *Similar to the human cognitive process*. Based on the work of Newel and Simon, rules appear to be a natural way of modeling how humans solve problems. The simple IF-THEN representation of rules makes it easy to explain to experts the structure of the knowledge you are trying to elicit from them.

Expert Systems History

Markov Algorithm



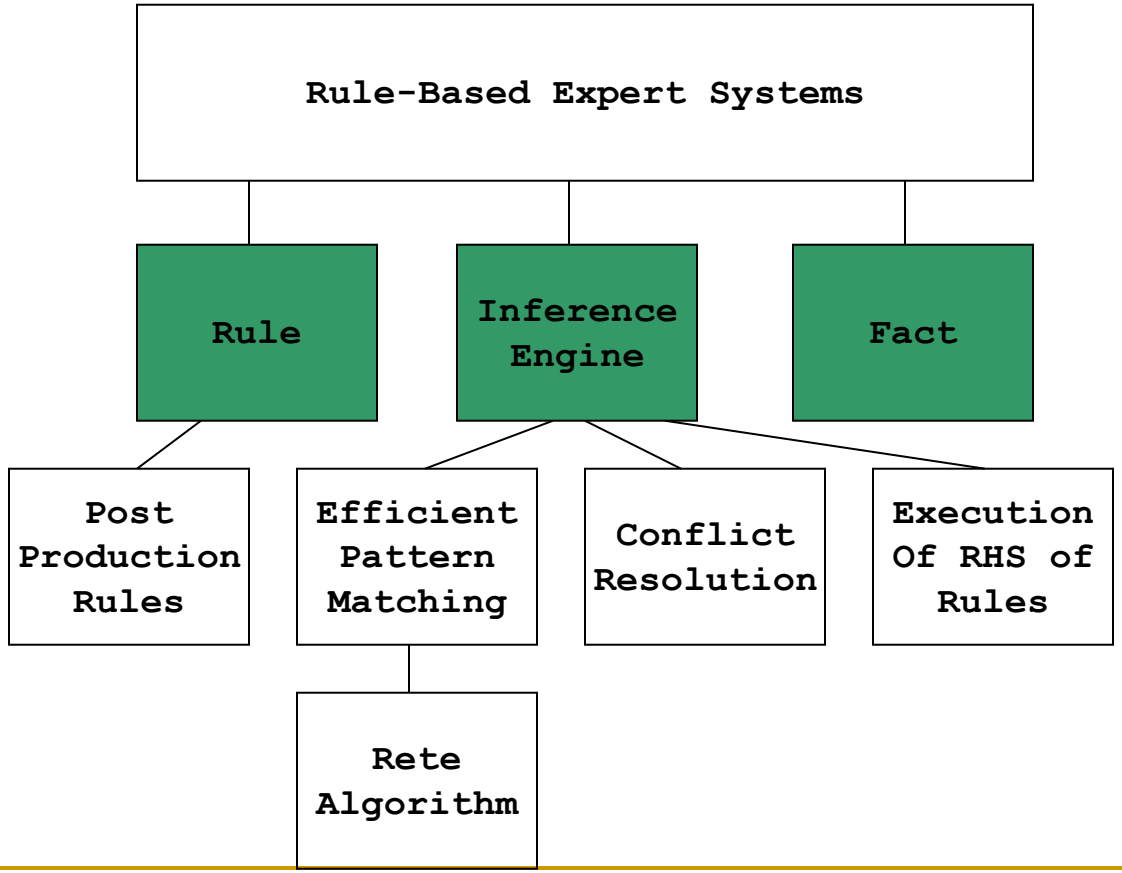
Expert Systems History

Markov Algorithm

- The next advance in applying production rules was made by Markov in 1954 that specified a control structure for production systems. Addressed the issue of lack of control strategy in Post production systems.
- A **Markov Algorithm** is an ordered group of productions that are applied in order of priority to an input string.
- For example, Production foo -> bar, when applied to String 'foobarfoo' produces the new string 'barbarfoo' for the first run.

Expert Systems History

Newel and Simon



Expert Systems History

Newel and Simon

- The **General Problem Solver** by Newel and Simon in the late 1950s, a program with the goal of general problem solving.
- Newel and Simon showed that much of human problem solving or cognition could be expressed by IF-THEN **production rules**.
- A rule corresponds to a small, modular collection of knowledge called a **chunk**.
- A rule corresponds to a small, modular collection called a **chunk**. A grand master chess expert may have 50,000 or more chunks of knowledge about chess patters.

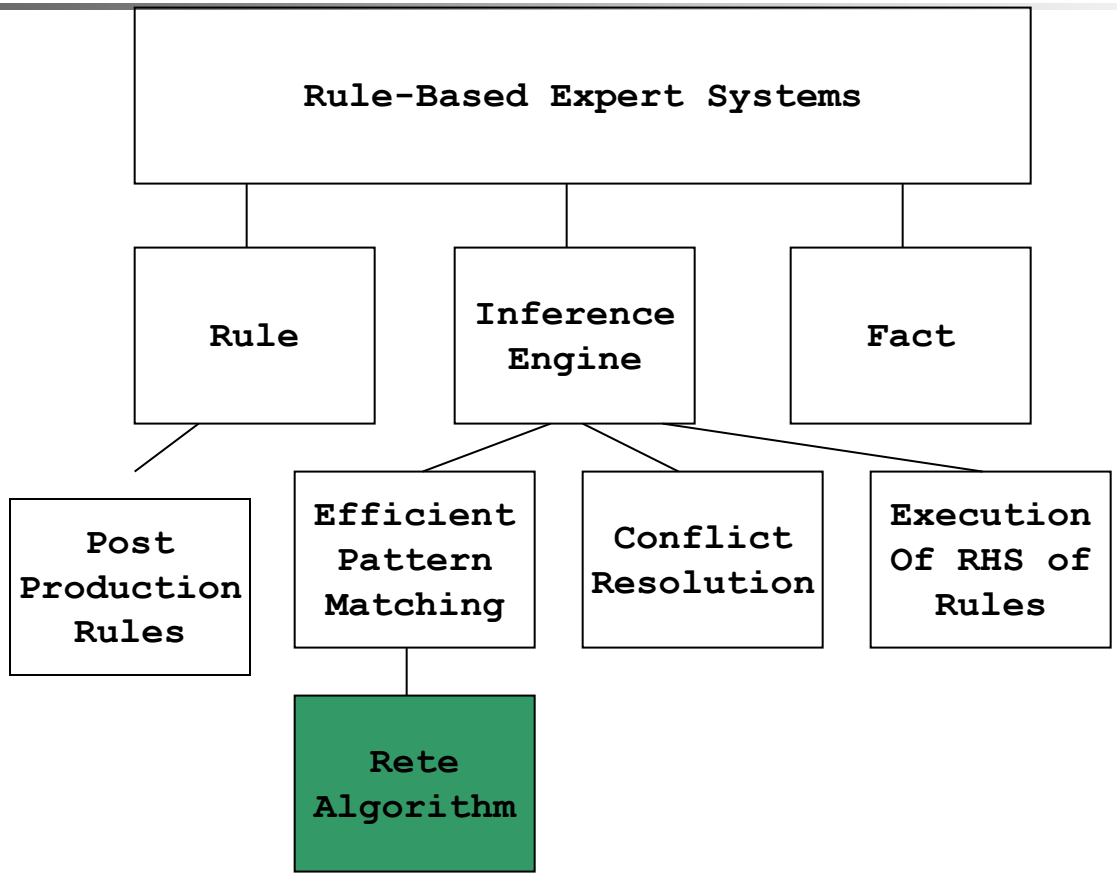
Expert Systems History

Newel and Simon

- Newel and Simon also showed that the other element necessary for human problem solving is a **cognitive processor**.
- The cognitive processor tries to find the rules that will be **activated** by the appropriate stimuli. This is the inference engine in an expert system.
- Newel and Simons' model of human problem solving, which defines long term memory, short term memory and the cognitive processor, is the basis of modern rule-based expert systems.

Expert Systems History

Rete Algorithm



Expert Systems History

Rete Algorithm

- The **Rete algorithm**, developed by Charles L. Forgy at CMU in 1979, is an algorithm that knows about all the rules and can apply any rule without having to try each one sequentially.
- Addresses the issue that the Markov Algorithm is highly inefficient for systems with many rules.
- Real Expert Systems contain thousands of rules and efficiency is of major importance.
- The Rete algorithm is a fast pattern matcher that obtains it's speed by storing information about the rules in a network.
- The Rete algorithm looks only for changes in matches on every cycle.

Expert Systems History

Rete Algorithm

- The Rete algorithm was first built into CMU's OPS (official production system) expert system shell.
- Fast pattern matching algorithms such as Rete completed the foundation for the practical application of expert systems.
- CLIPS uses the Rete Algorithm.

CLIPS

What is CLIPS?

- CLIPS is a multi-paradigm programming language that provides support for rule-based, object-oriented, and procedural programming.
- CLIPS – C Language Integrated Production System.
- Design using the C programming language at NASA/Johnson Space Center with the specific purpose of providing high portability, low cost, and easy integration with external systems.

CLIPS

What is CLIPS?

- The failure to provide expert systems technology within NASA's operational computing constraints could largely be traced to the use of LISP as the base language for nearly all expert system software tools at that time.

- In particular, three problems hindered the use of LISP based expert system tools within NASA:

❖ The low availability of LISP on a wide variety of conventional computers.

❖ The high cost of state-of-the-art LISP tools and hardware.

❖ The poor integration of LISP with other languages (making embedded applications difficult).

CLIPS

What is CLIPS?

- The inference and representation capabilities provided by the rule based programming language of CLIPS are similar to, but more powerful than, those of OPS5 of CMU.
- CLIPS supports only forward chaining rules, it does not support backward chaining rules.
- Originally CLIPS provided support only for rule-based programming. Version 5.0 of CLIPS introduced procedural and object-oriented programming support
- CLIPS is now maintained independently from NASA as public domain software.
- My project uses the rule-based programming capabilities of CLIPS.

CLIPS

CLIPS Constructs

CLIPS Construct – **Facts**

In order to solve a problem, a CLIPS program must have data or information which it can reason. A "chunk" of information in CLIPS is called a **fact**.

For example, a fact in CLIPS;

```
(person (name "John Q. Public")
  (age 23)
  (eye-color blue)
  (hair-color black))
```

CLIPS

CLIPS Constructs

CLIPS Construct – **Deftemplate**

Before facts can be created, CLIPS must be informed of the list of valid slots for a given relational name.

Groups of facts that share the same relational name and contain common information can be described using the **deftemplate** construct.

For example, a deftemplate in CLIPS;

```
(deftemplate person
      (slot name)
      (slot age)
      (slot eye-color)
      (slot hair-color))
```

CLIPS

CLIPS Constructs

CLIPS Construct - **Assert**

Facts can be **Asserted** using deftemplates with the assert keyword.

For example,

```
(assert (person (name "John Q. Public")
                (age 23)
                (eye-color blue)
                (hair-color black))
```

CLIPS

CLIPS Constructs

CLIPS Construct - **Deffacts**

Deffacts construct, allows facts to automatically be asserted instead of having to type them in.

For example,

```
(deffacts people
     (person (name "John Q. Public") (age 23)
             (eye-color blue) (hair-color black))
     (person (name "Jane S. Public") (age 24)
             (eye-color blue) (hair-color blond)))
```

CLIPS

CLIPS Constructs

CLIPS Construct – **Defrule**

Defrule construct, In order to accomplish useful work, an expert system must have rules as well as facts.

For example,

IF the emergency is a fire
THEN the response is to activate the sprinkler system

CLIPS

CLIPS Constructs

CLIPS Construct – **Defrule**

```
(deftemplate emergency (slot type))
(deftemplate response (slot action))

(defrule fire-emergency
      (emergency (type fire))
=>
      (assert (response (action activate-sprinklers))))
```

General defrule format;

```
(defrule <rule_name>
      <patterns>*
=>
      <actions>*
```

Conclusion

It's Over!

- Expert Systems are a practical application of AI that has a deep past that draws from the work of many people.
- Early expert systems were expensive and difficult to build. With the advent of expert system tools like clips, expert systems can be built for much less cost and in less time.
- Procedural based programming languages like Java RULE!
- CADDIE the golfing expert system, WILL make you a better golfer!