Expert Systems

PROFESSOR DARIAN M. ONCHIS

HTTPS://STAFF.FMI.UVT.RO/~DARIAN.ONCHIS/

EUROPEAN LABORATORY FOR LEARNING AND INTELLIGENT SYSTEMS (ELLIS)

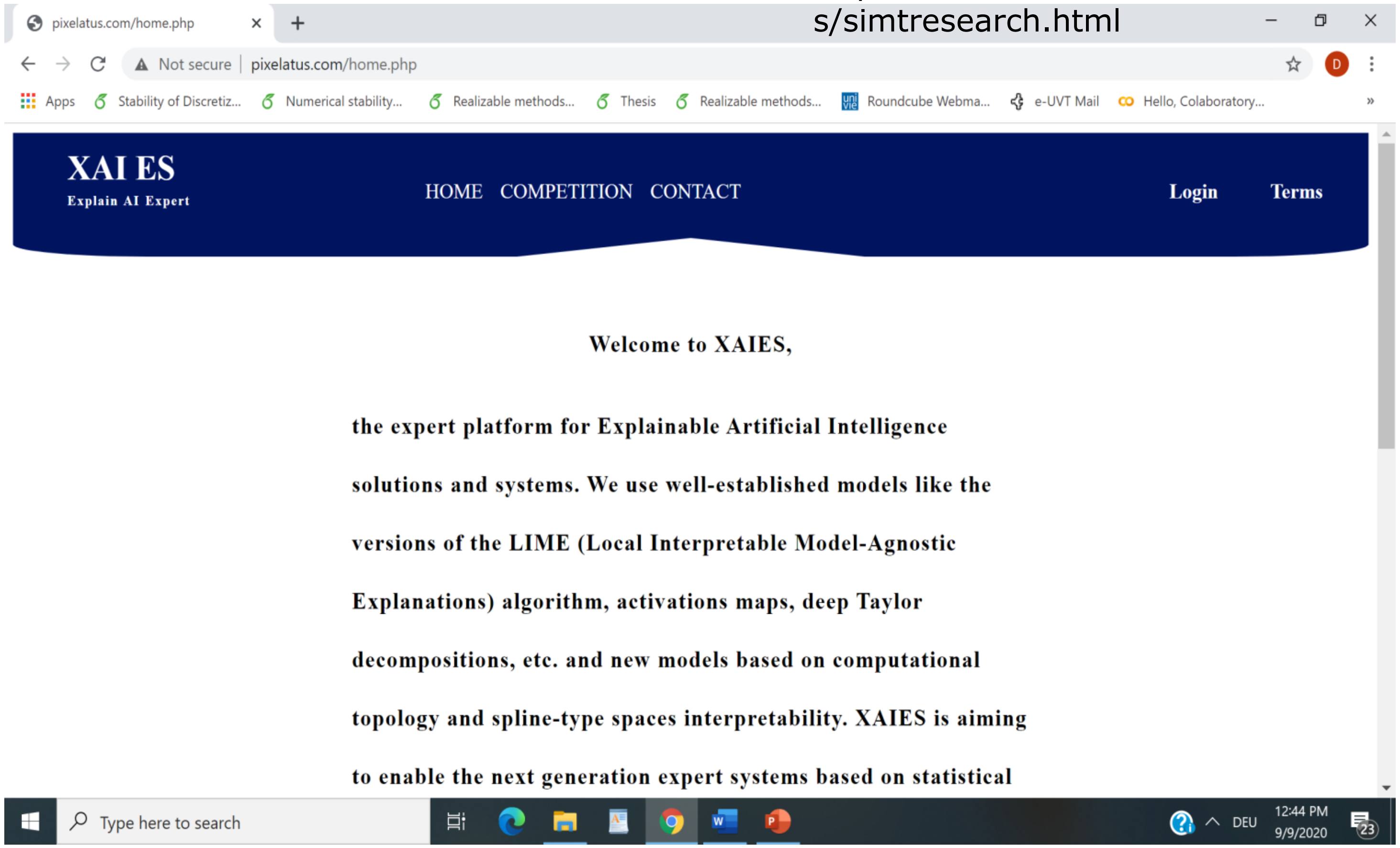
DARIAN.ONCHIS@E-UVT.RO

S·I·M·T = **Signal, Image and Machine Learning Team**

MYSELF, SIMT AND XAIES

https://staff.fmi.uvt.ro/~darian.onchis/simtresearch.html

DARIAN



XAI ES

Explain AI Expert

Welcome to XAIES,

the expert platform for Explainable Artificial Intelligence solutions and systems. We use well-established models like the versions of the LIME (Local Interpretable Model-Agnostic Explanations) algorithm, activations maps, deep Taylor decompositions, etc. and new models based on computational topology and spline-type spaces interpretability. XAIES is aiming to enable the next generation expert systems based on statistical

Evaluation

1. Topical Assignments: 50%
2. Small Team Project: 50%

THERE COULD BE A BONUS FOR IMPLICATION !

!

T

P

E

PRJ

PROJECT SELECTION

XAIES PROJECT

Intiate a new competition on XAIES and provide one solution for it: https://xaies.000webhostapp.com/

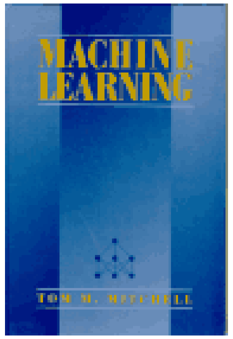
HOW TO: Check the existing competitions and propose one where the XAI part is needed.

WEB PROJECT

Check the KAGGLE web https://www.kaggle.com/

HOW TO:Further develop the XAIES web site to look and function like kaggle.com

PRJ



T = TASK

P = PERFORMANCE

E = EXPERIENCE

MACHINE LEARNING, TOM MITCHELL, MCGRAW HILL, 1997.

Other necessary references:

http://www.deeplearningbook.org/

http://neuralnetworksanddeeplearning.com/

T

P

E

PYTHON AND JUPYTER NOTEBOOKS E.G. Colab

https://colab.research.google.com/

MAIN LIBRARIES:

NUMPY, NUMERICS

PANDAS, DATA FRAMES

MATPLOTLIB, VIZUALIZATIONS

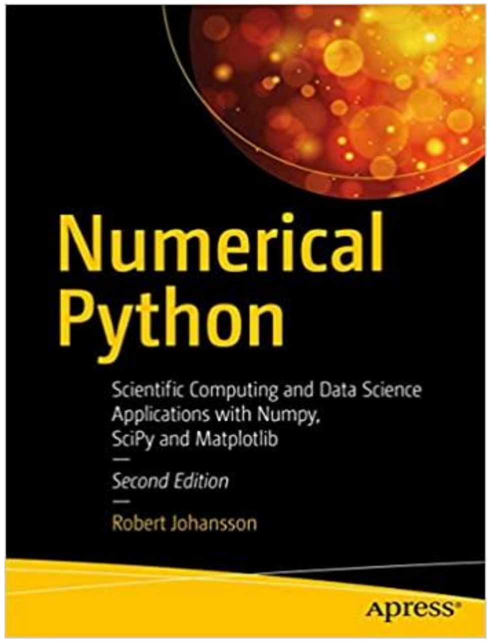
SCIKIT-LEARN, CLASSICAL ML

TENSORFLOW, DEEP LEARNING

KERAS, DEEP LEARNING EASY

GYM, REINFORCEMENT LEARNING

There will be more libraries on the way…


Numerical Python
Scientific Computing and Data Science Applications with Numpy, SciPy and Matplotlib
Second Edition
Robert Johansson
Apress

Neuro-Symbolic AI

- new AI methods that combine neural networks, which extract statistical structures from raw data files – context about image and sound files, for example – with symbolic representations of problems and logic. By fusing these two approaches, we're building a new class of AI that will be far more powerful than the sum of its parts.
- Logical Neural Networks: https://arxiv.org/pdf/2006.13155.pdf

Neurosymbolic AI: The 3rd Wave

- https://arxiv.org/abs/2012.05876

- Neural-symbolic computing has been an active area of research for many years seeking to bring together robust learning in neural networks with reasoning and explainability via symbolic representations for network models.

Symbolic AI vs Connectionism

- Symbolic AI theory presumes that the world can be understood in the terms of structured representations. It asserts that symbols that stand for things in the world are the core building blocks of cognition. Symbolic processing uses rules or operations on the set of symbols to encode understanding. This set of rules is called an expert system, which is a large base of if/then instructions.

Symbolic AI vs Connectionism

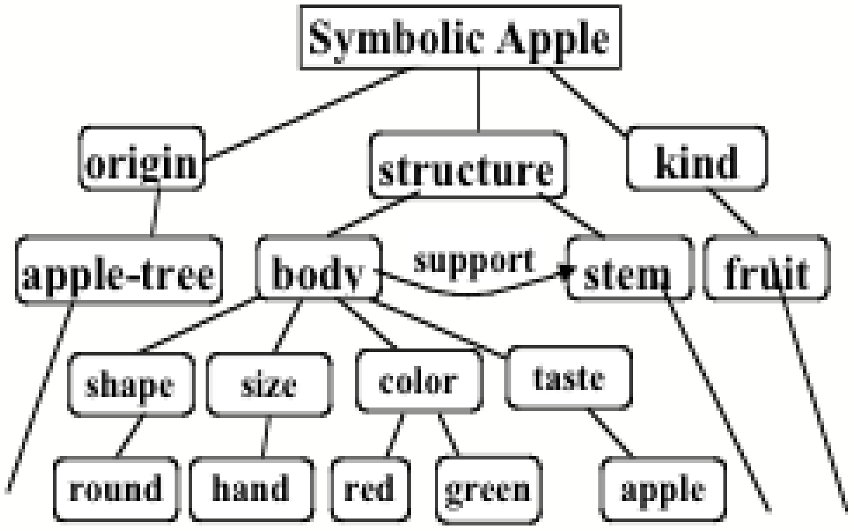
- Connectionism theory essentially states that intelligent decision-making can be done through an interconnected system of small processing nodes of unit size. Each of the neuron-like processing units is connected to other units, where the degree or magnitude of connection is determined by each neuron’s level of activation. As the interconnected system is introduced to more information (learns), each neuron processing unit also becomes either increasingly activated or deactivated. This system of transformations and convolutions, when trained with data, can learn in-depth models of the data generation distribution, and thus can perform intelligent decision-making, such as regression or classification.

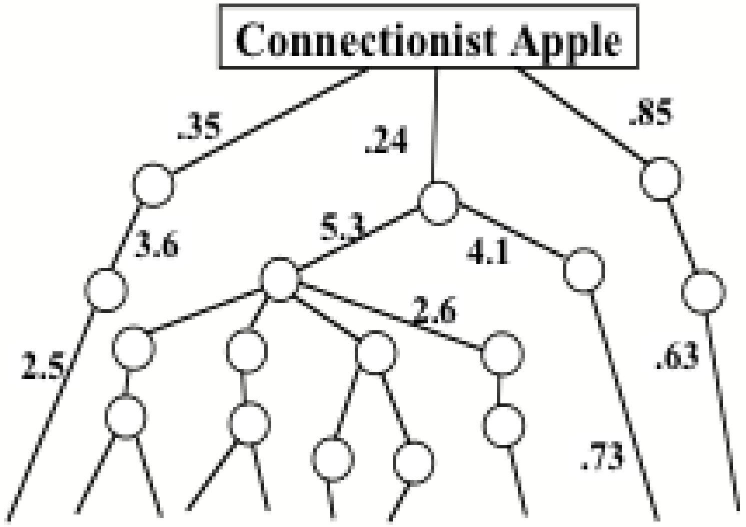
Advantages and Drawbacks

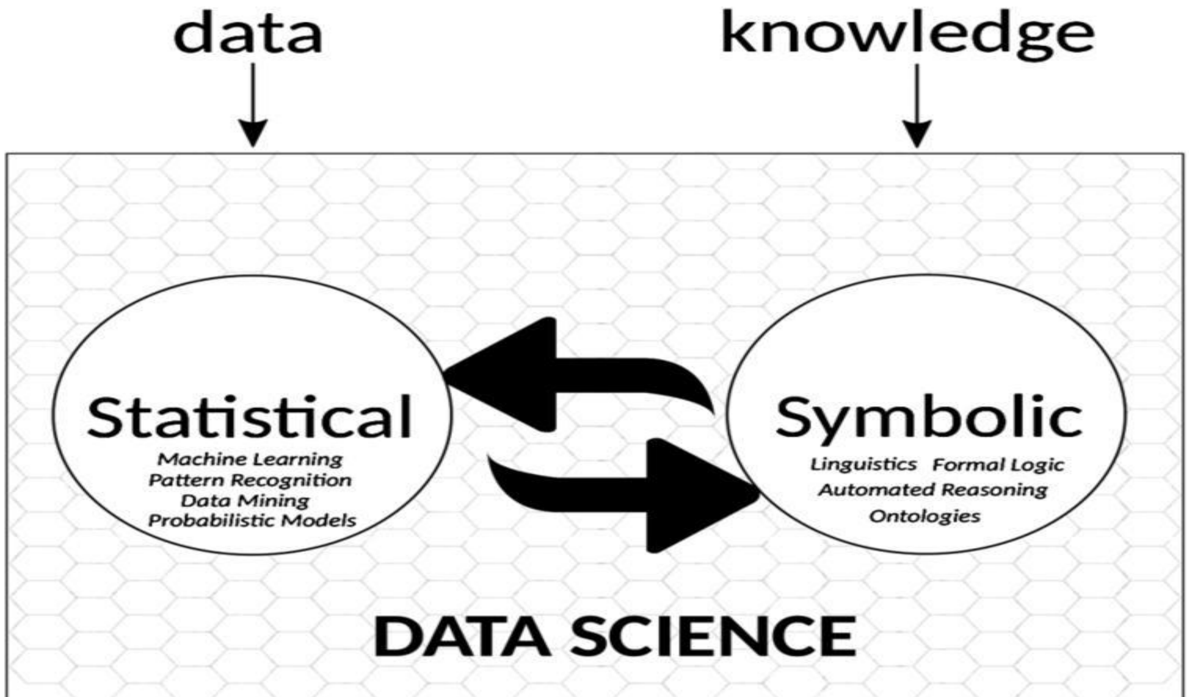
- The advantages of symbolic AI are that it performs well when restricted to the specific problem space that it is designed for. Symbolic AI is simple and solves toy problems well. However, the primary disadvantage of symbolic AI is that it does not generalize well.
- The main advantage of connectionism is that it is parallel, not serial. What this means is that connectionism is robust to changes. If one neuron or computation if removed, the system still performs decently due to all of the other neurons. This robustness is called graceful degradation. Additionally, the neuronal units can be abstract, and do not need to represent a particular symbolic entity, which means this network is more generalizable to different problems. Connectionism architectures have been shown to perform well on complex tasks like image recognition, computer vision, prediction, and supervised learning.
- One disadvantage is that connectionist networks take significantly higher computational power to train. Another critique is that connectionism models may be oversimplifying assumptions about the details of the underlying neural systems by making such general abstractions.

Symbolic AI vs Connectionism

- Symbolic AI goes by several other names, including rule-based AI, classic AI and good old-fashioned AI (GOFA).
- A system built with connectionist AI gets more intelligent through increased exposure to data and learning the patterns and relationships associated with it. In contrast, symbolic AI gets hand-coded by humans. One example of connectionist AI is an artificial neural network.


Symbolic Apple
origin
structure
kind
apple-tree
body
support
stem
fruit
shape
size
color
taste
round
hand
red
green
apple


Connectionist Apple
.35
.24
.85
3.6
5.3
4.1
2.6
2.5
.63
.73


data
knowledge
Statistical
Machine Learning
Pattern Recognition
Data Mining
Probabilistic Models
Symbolic
Linguistics Formal Logic
Automated Reasoning
Ontologies
DATA SCIENCE

What is an expert system?

"An expert system is a computer system that emulates, or acts in all respects, with the decision-making capabilities of a human expert."

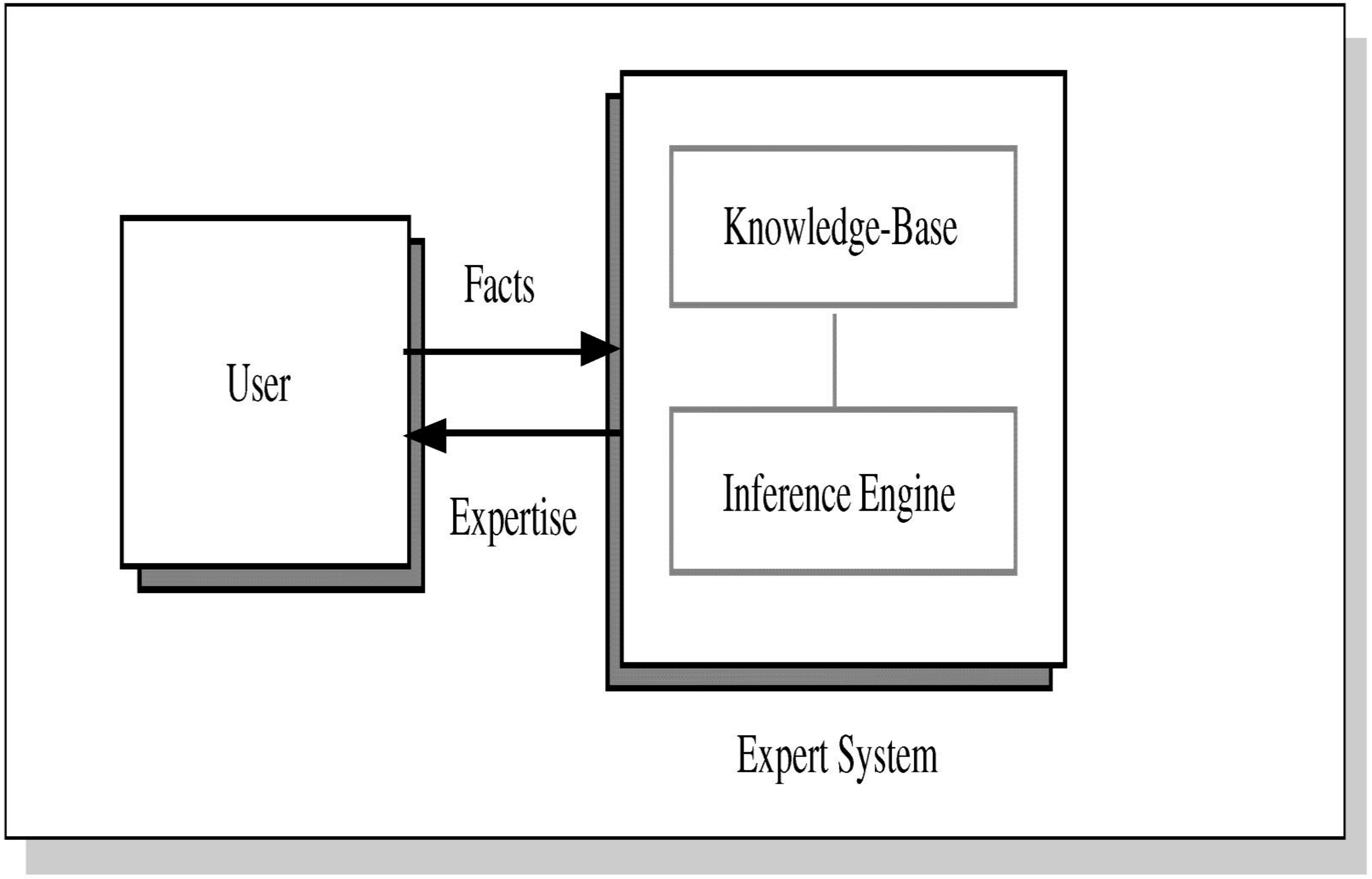
Professor Edward Feigenbaum

Stanford University, "father of experts systems"

Expert System Main Components

- Knowledge base – obtainable from books, magazines, knowledgeable persons, etc.
- Inference engine – draws conclusions from the knowledge base

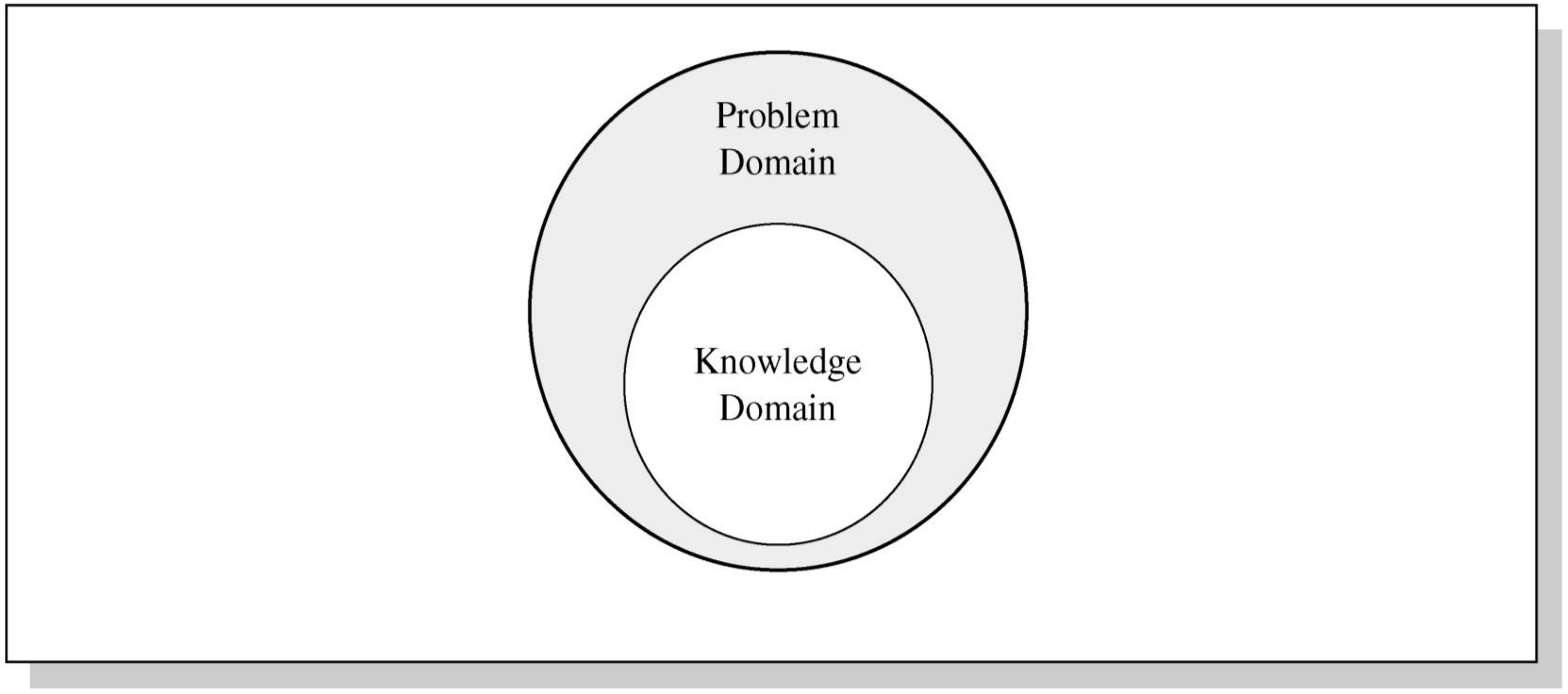
Figure 1.2 Basic Functions of Expert Systems



Problem Domain vs. Knowledge Domain

- An expert's knowledge is specific to one problem domain – medicine, finance, science, engineering, etc.
- The expert's knowledge about solving specific problems is called the knowledge domain.
- The problem domain is always a superset of the knowledge domain.

Problem and Knowledge Domain Relationship



Representing the Knowledge

The knowledge of an expert system can be represented in a number of ways, including IF-THEN rules:

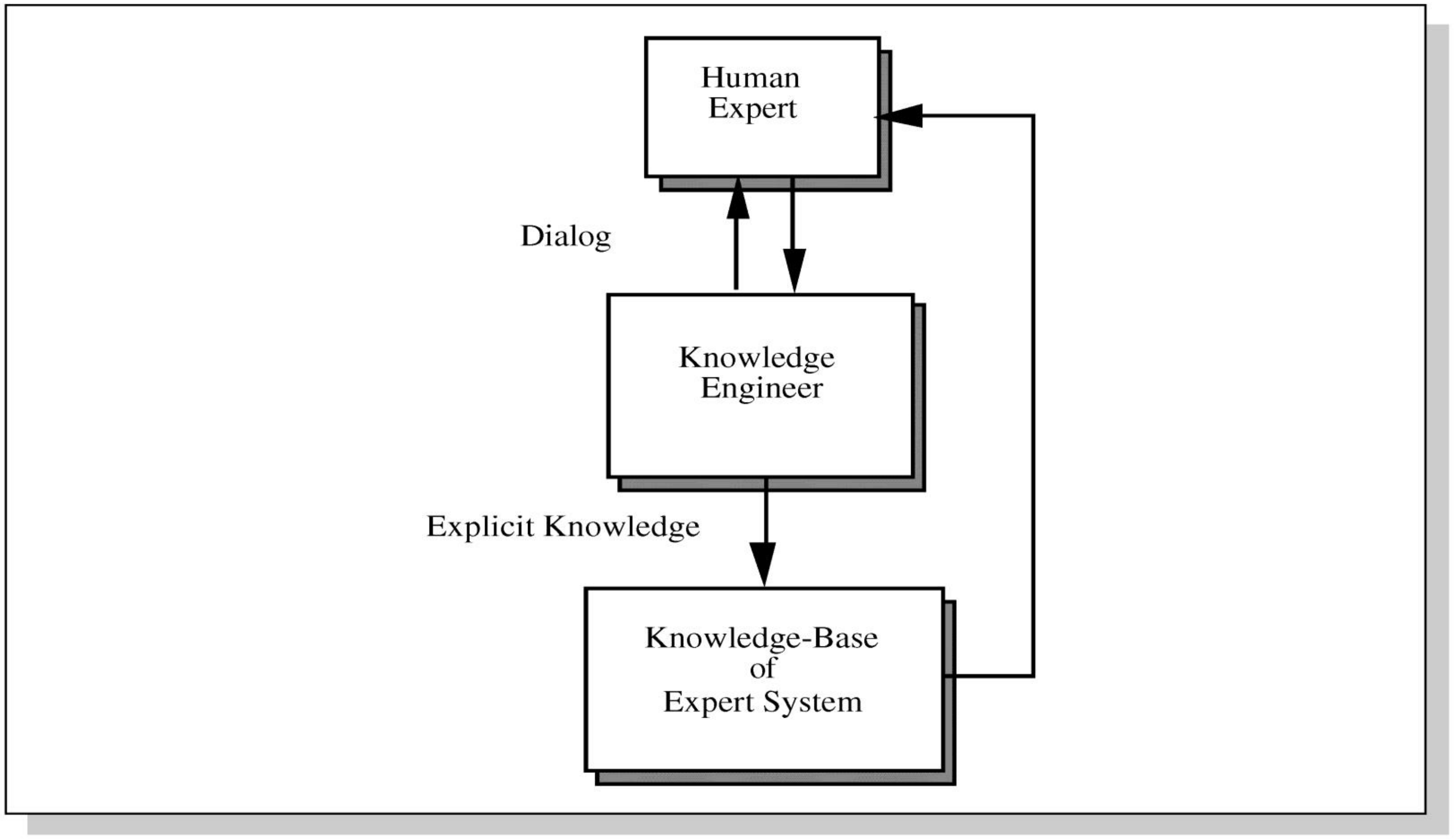
```
IF you are hungry THEN eat
```

Knowledge Engineering

The process of building an expert system:

1. The knowledge engineer establishes a dialog with the human expert to determine knowledge.
2. The knowledge engineer codes the knowledge explicitly in the knowledge base.
3. The expert evaluates the expert system and gives a critique to the knowledge engineer.

Development of an Expert System



The Role of AI

- An algorithm is an ideal solution guaranteed to yield a solution in a finite amount of time.
- When an algorithm is not available or is insufficient, we rely on artificial intelligence (AI).
- Expert system relies on inference – we accept a “reasonable solution.”

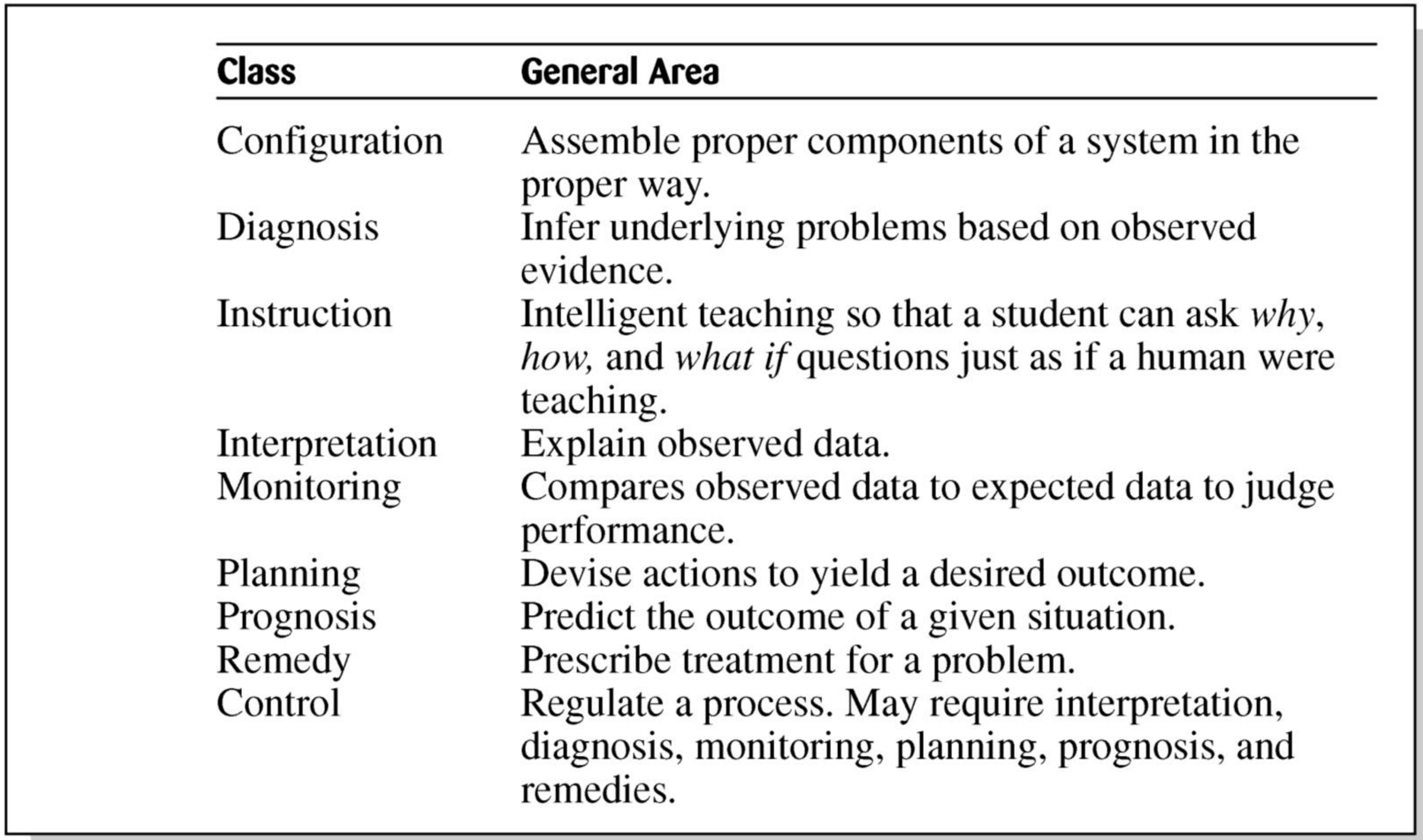
Shallow and Deep Knowledge

- It is easier to program expert systems with shallow knowledge than with deep knowledge.
- Shallow knowledge – based on empirical and heuristic knowledge.
- Deep knowledge – based on basic structure, function, and behavior of objects.

Early Expert Systems

- DENDRAL – used in chemical mass spectroscopy to identify chemical constituents
- MYCIN – medical diagnosis of illness
- DIPMETER – geological data analysis for oil
- PROSPECTOR – geological data analysis for minerals
- XCON/R1 – configuring computer systems

Table 1.3 Broad Classes of Expert Systems



Class	General Area
Configuration	Assemble proper components of a system in the proper way.
Diagnosis	Infer underlying problems based on observed evidence.
Instruction	Intelligent teaching so that a student can ask *why*, *how,* and *what if* questions just as if a human were teaching.
Interpretation	Explain observed data.
Monitoring	Compares observed data to expected data to judge performance.
Planning	Devise actions to yield a desired outcome.
Prognosis	Predict the outcome of a given situation.
Remedy	Prescribe treatment for a problem.
Control	Regulate a process. May require interpretation, diagnosis, monitoring, planning, prognosis, and remedies.

Problems with Algorithmic Solutions

- Conventional computer programs generally solve problems having algorithmic solutions.

- Algorithmic languages include C, Java, and C#.

- Classic AI languages include LISP and PROLOG.

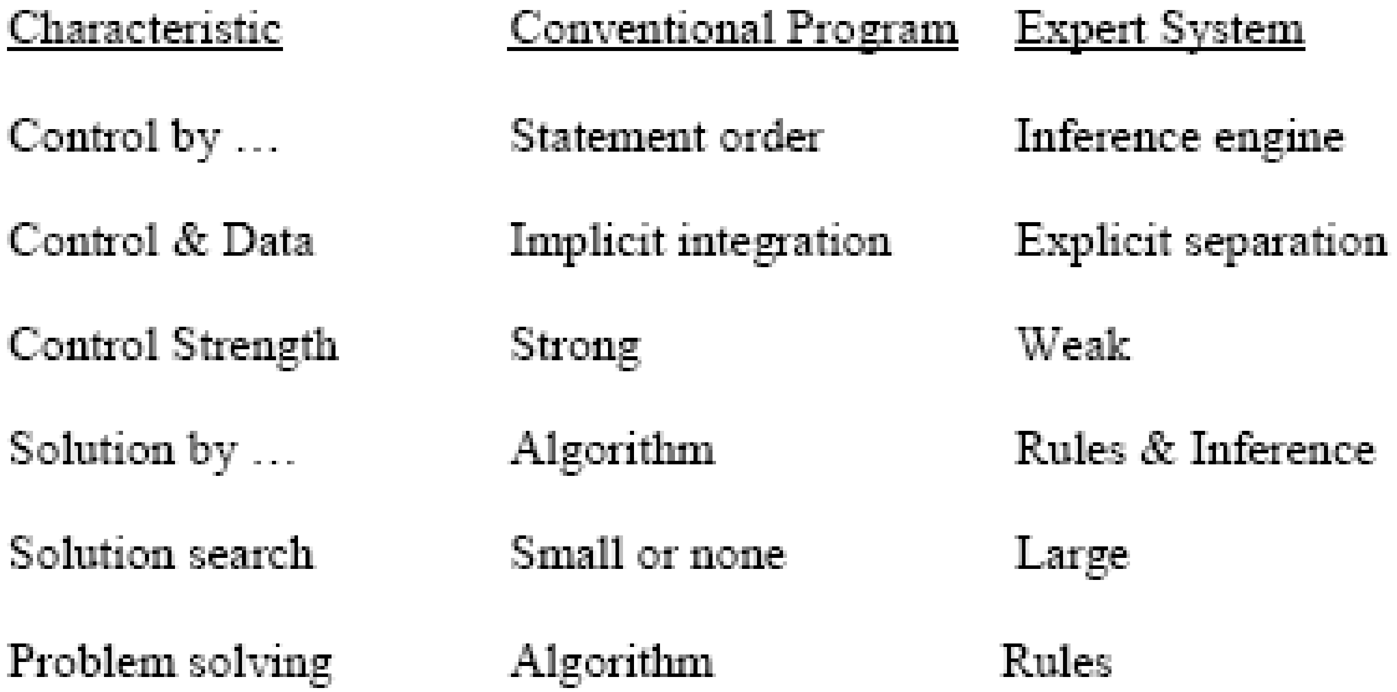
Considerations for Building Expert Systems

- Can the problem be solved effectively by conventional programming?
- Is there a need and a desire for an expert system?
- Is there at least one human expert who is willing to cooperate?
- Can the expert explain the knowledge to the knowledge engineer can understand it.
- Is the problem-solving knowledge mainly heuristic and uncertain?

Languages, Shells, and Tools

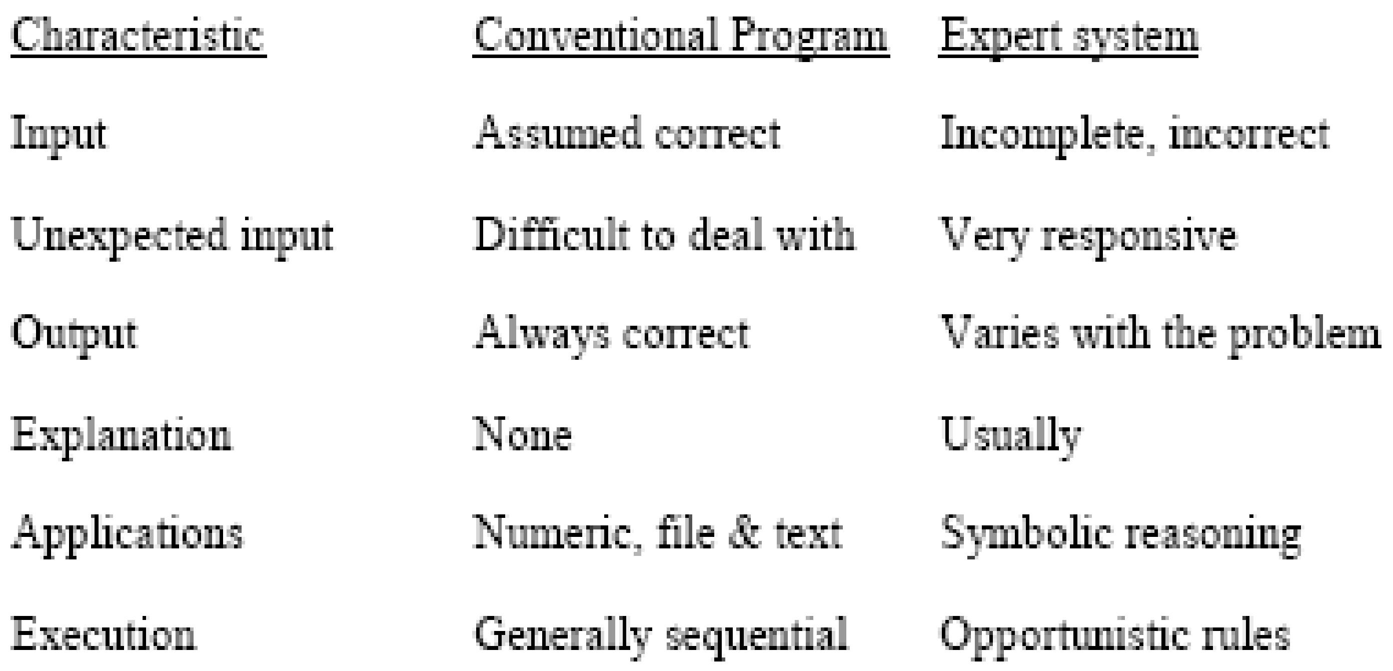
- Expert system languages are post-third generation.
- Procedural languages (e.g., C) focus on techniques to represent data.
- More modern languages (e.g., Java) focus on data abstraction.
- Expert system languages (e.g. CLIPS) focus on ways to represent knowledge.

Expert systems Vs conventional programs I



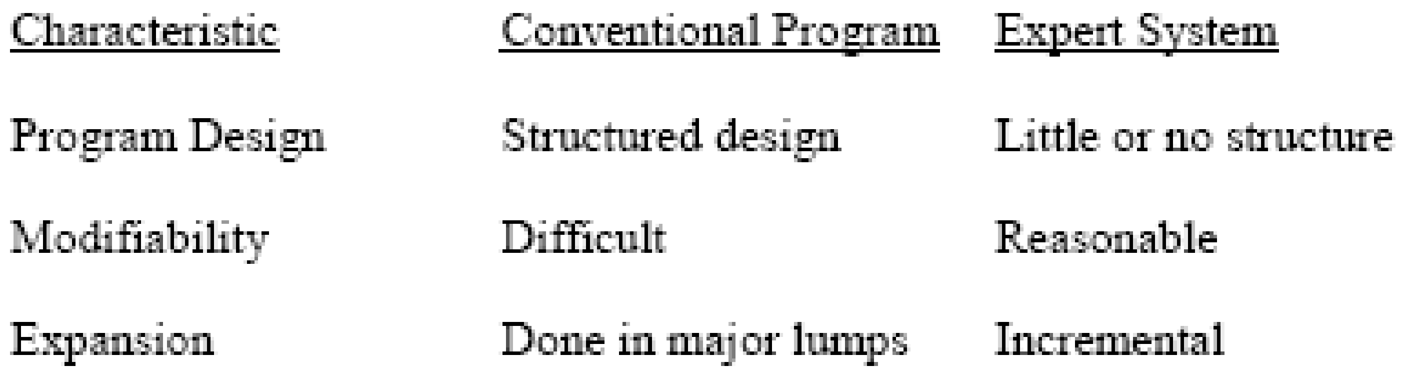
Characteristic	Conventional Program	Expert System
Control by …	Statement order	Inference engine
Control & Data	Implicit integration	Explicit separation
Control Strength	Strong	Weak
Solution by …	Algorithm	Rules & Inference
Solution search	Small or none	Large
Problem solving	Algorithm	Rules

Expert systems Vs conventional programs II



Characteristic	Conventional Program	Expert system
Input	Assumed correct	Incomplete, incorrect
Unexpected input	Difficult to deal with	Very responsive
Output	Always correct	Varies with the problem
Explanation	None	Usually
Applications	Numeric, file & text	Symbolic reasoning
Execution	Generally sequential	Opportunistic rules

Expert systems Vs conventional programs III



Characteristic	Conventional Program	Expert System
Program Design	Structured design	Little or no structure
Modifiability	Difficult	Reasonable
Expansion	Done in major lumps	Incremental

Elements of an Expert System

- User interface – mechanism by which user and system communicate.
- Exploration facility – explains reasoning of expert system to user.
- Working memory – global database of facts used by rules.
- Inference engine – makes inferences deciding which rules are satisfied and prioritizing.

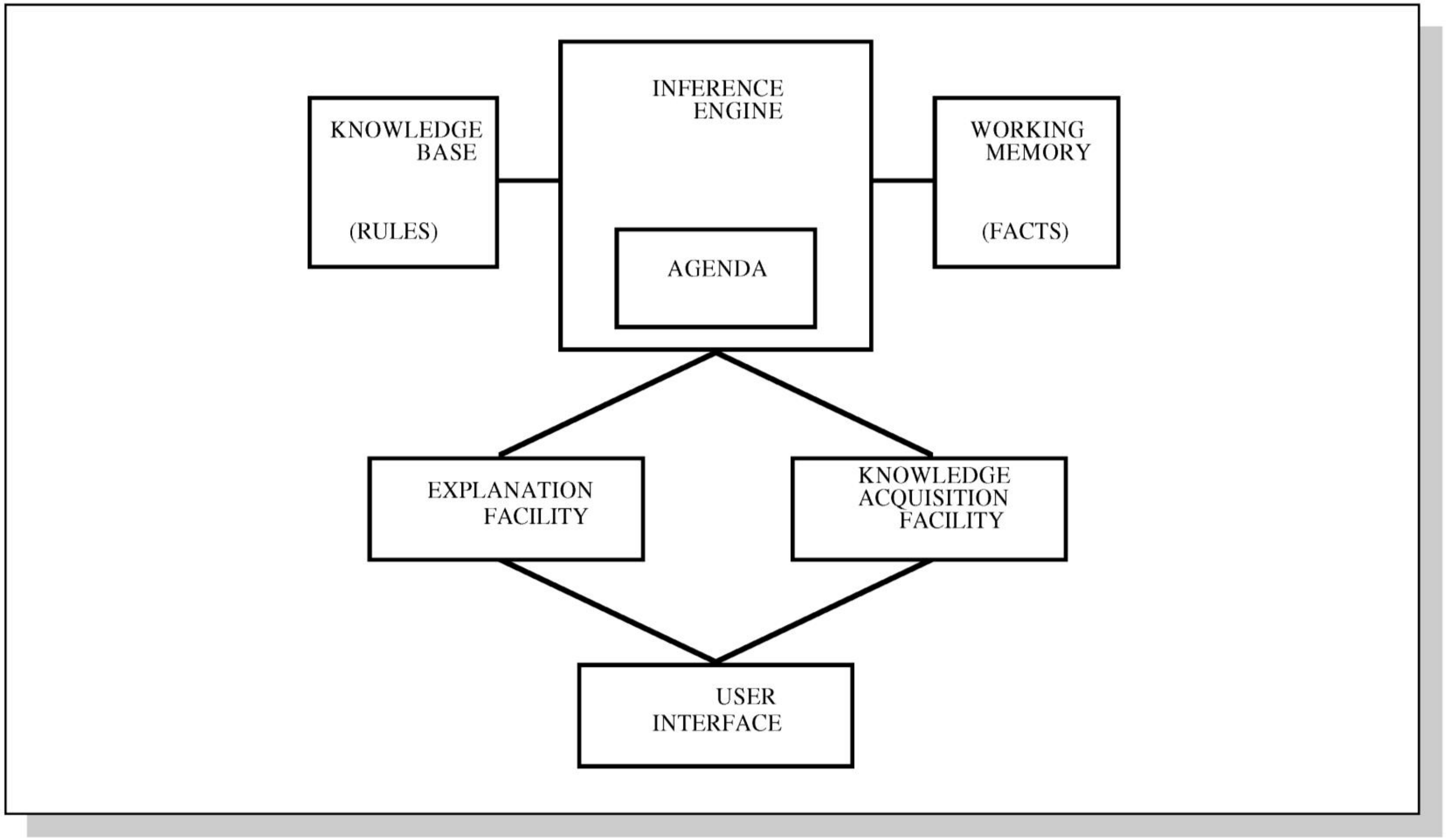
Elements Continued

- Agenda – a prioritized list of rules created by the inference engine, whose patterns are satisfied by facts or objects in working memory.
- Knowledge acquisition facility – automatic way for the user to enter knowledge in the system bypassing the explicit coding by knowledge engineer.
- Knowledge Base – includes the rules of the expert system

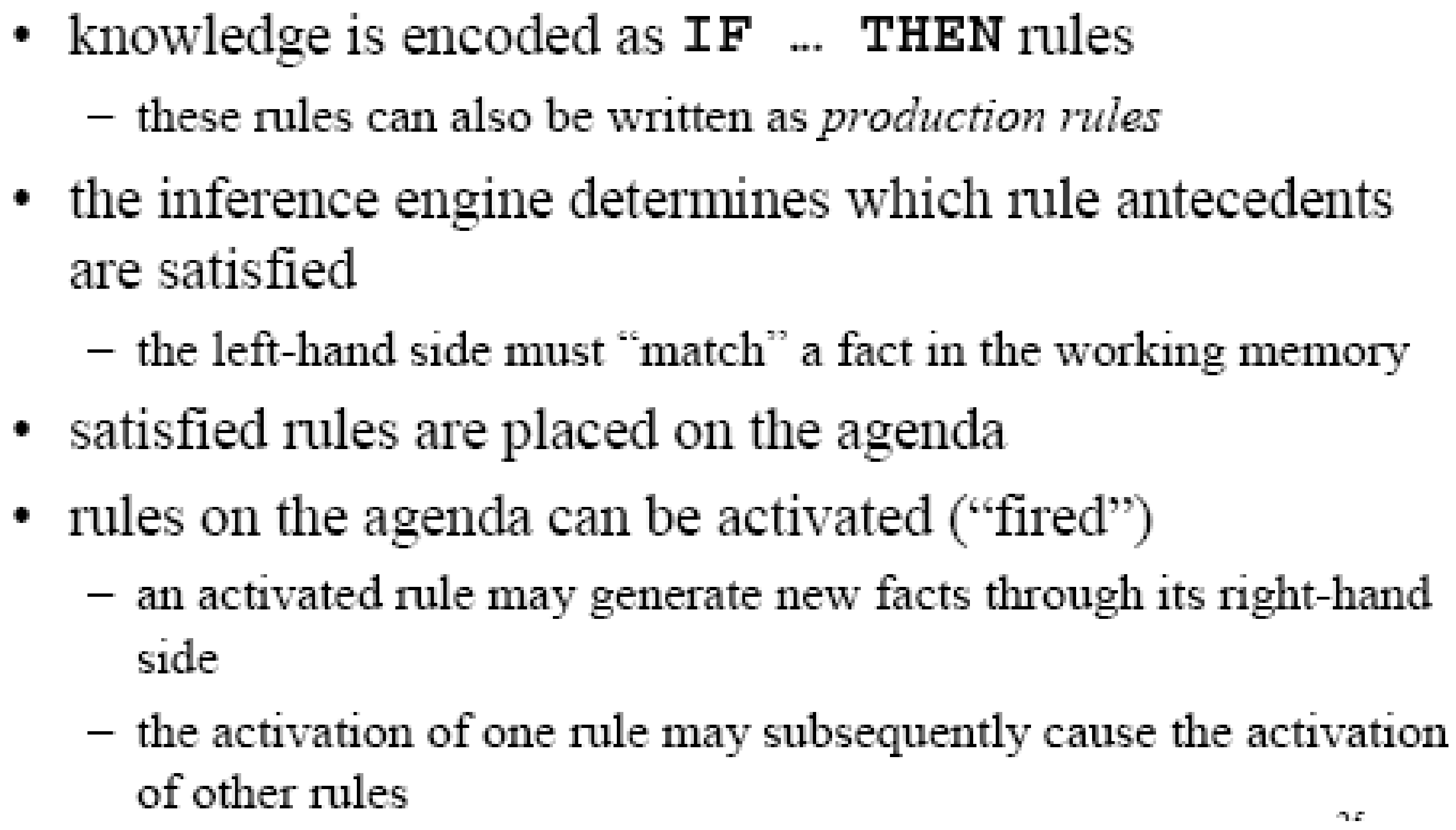
Production Rules

- Knowledge base is also called production memory.
- Production rules can be expressed in IF-THEN pseudocode format.
- In rule-based systems, the inference engine determines which rule antecedents are satisfied by the facts.

Structure of a Rule-Based Expert System

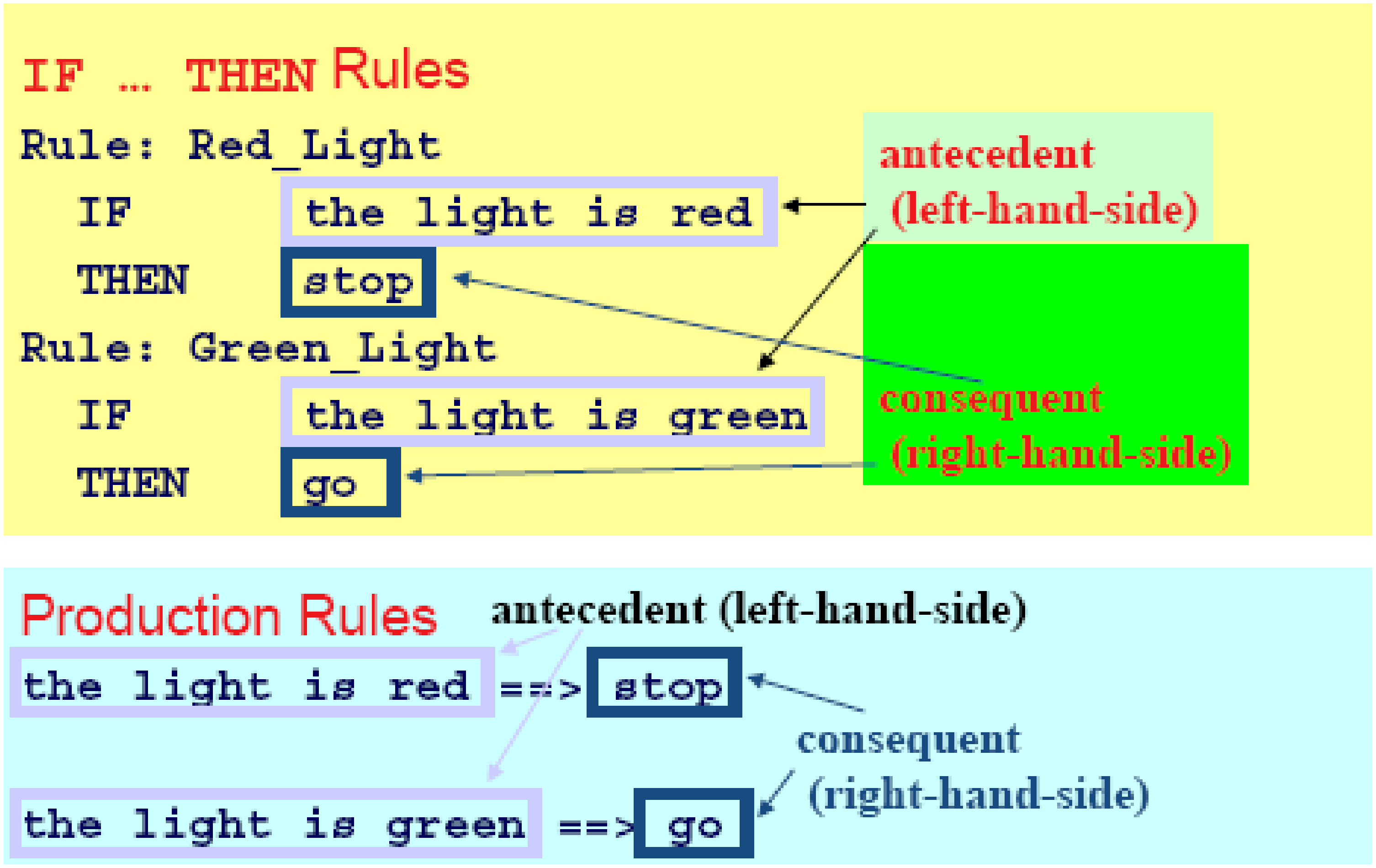


Rule-Based ES



- knowledge is encoded as **`IF ... THEN`** rules
  - these rules can also be written as *production rules*
- the inference engine determines which rule antecedents are satisfied
  - the left-hand side must "match" a fact in the working memory
- satisfied rules are placed on the agenda
- rules on the agenda can be activated ("fired")
  - an activated rule may generate new facts through its right-hand side
  - the activation of one rule may subsequently cause the activation of other rules

Example Rules



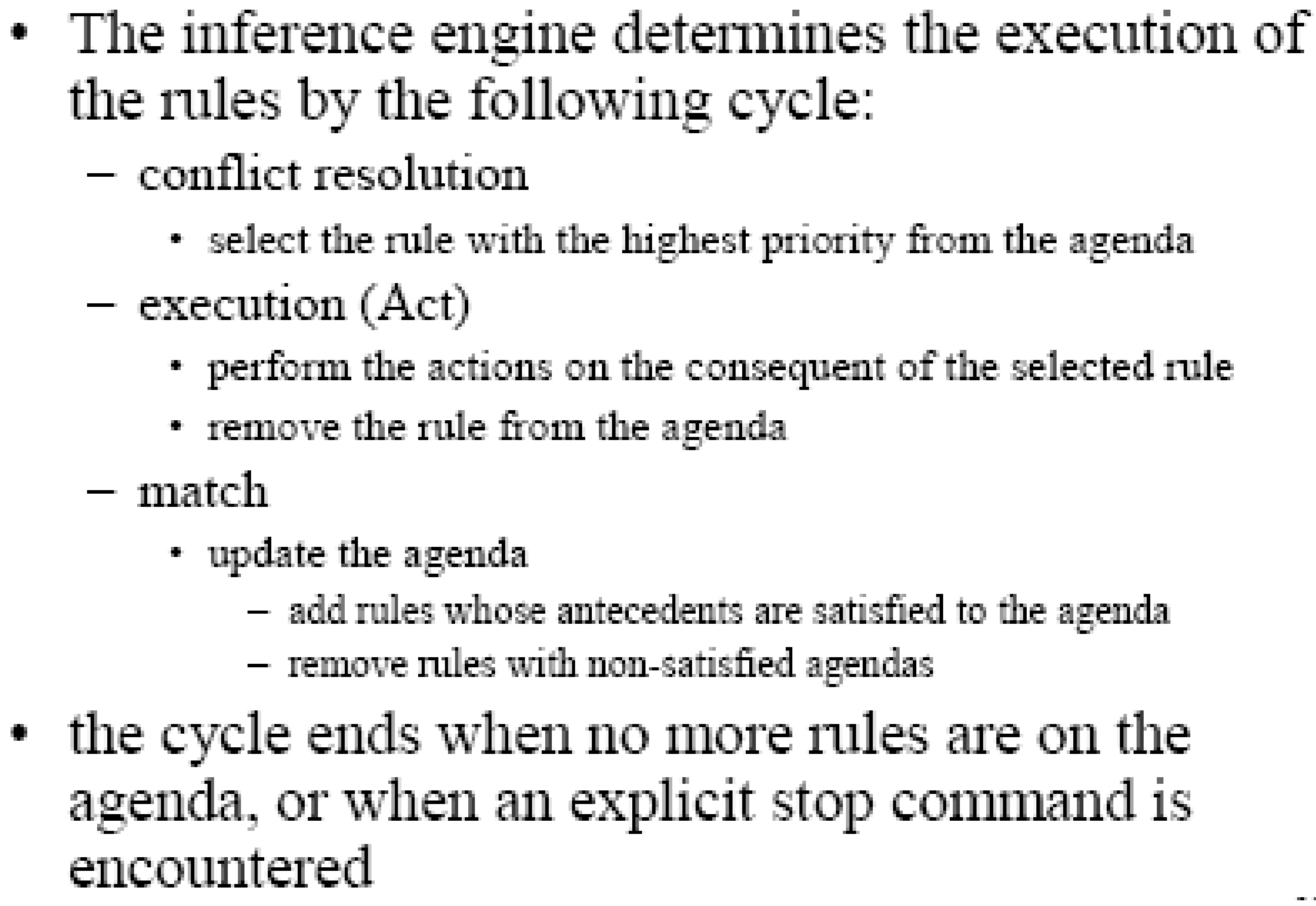
IF … THEN Rules

```
Rule: Red_Light
   IF       the light is red
   THEN     stop
Rule: Green_Light
   IF       the light is green
   THEN     go
```

Production Rules

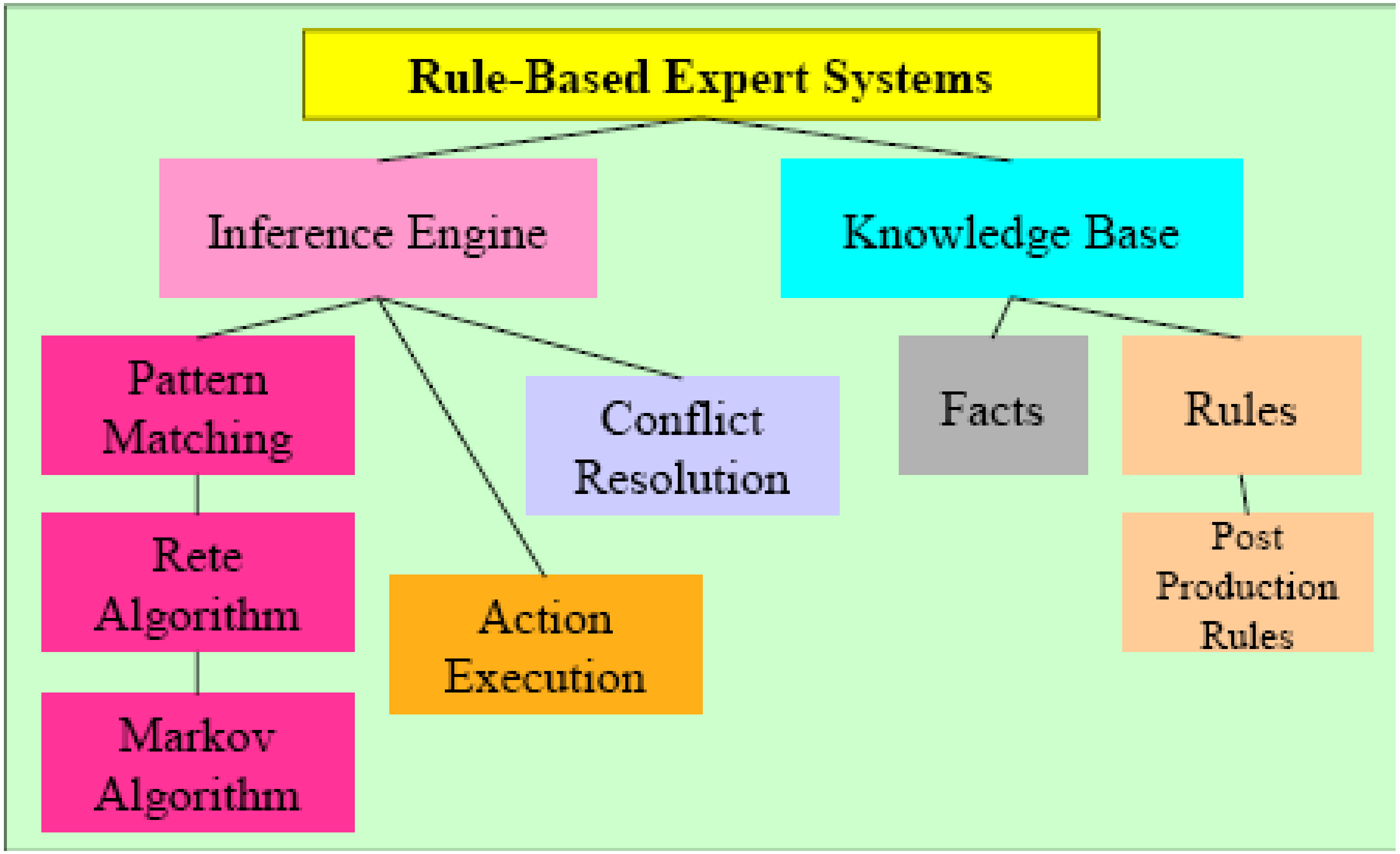
```
the light is red ==> stop

the light is green ==> go
```

Inference Engine Cycle



- The inference engine determines the execution of the rules by the following cycle:
  - conflict resolution
    - select the rule with the highest priority from the agenda
  - execution (Act)
    - perform the actions on the consequent of the selected rule
    - remove the rule from the agenda
  - match
    - update the agenda
      - add rules whose antecedents are satisfied to the agenda
      - remove rules with non-satisfied agendas
- the cycle ends when no more rules are on the agenda, or when an explicit stop command is encountered

Foundation of Expert Systems



General Methods of Inferencing

- Forward chaining (data-driven)– reasoning from facts to the conclusions resulting from those facts – best for prognosis, monitoring, and control.
  - Examples: CLIPS, OPS5
- Backward chaining (query driven)– reasoning in reverse from a hypothesis, a potential conclusion to be proved to the facts that support the hypothesis – best for diagnosis problems.
  - Examples: MYCIN

Production Systems

- Rule-based expert systems – most popular type today.
- Knowledge is represented as multiple rules that specify what should/not be concluded from different situations.
- Forward chaining – start w/facts and use rules do draw conclusions/take actions.
- Backward chaining – start w/hypothesis and look for rules that allow hypothesis to be proven true.

Forward/Backward Chaining

- Forward chaining – primarily data-driven.
- Backward chaining – primarily goal driven.

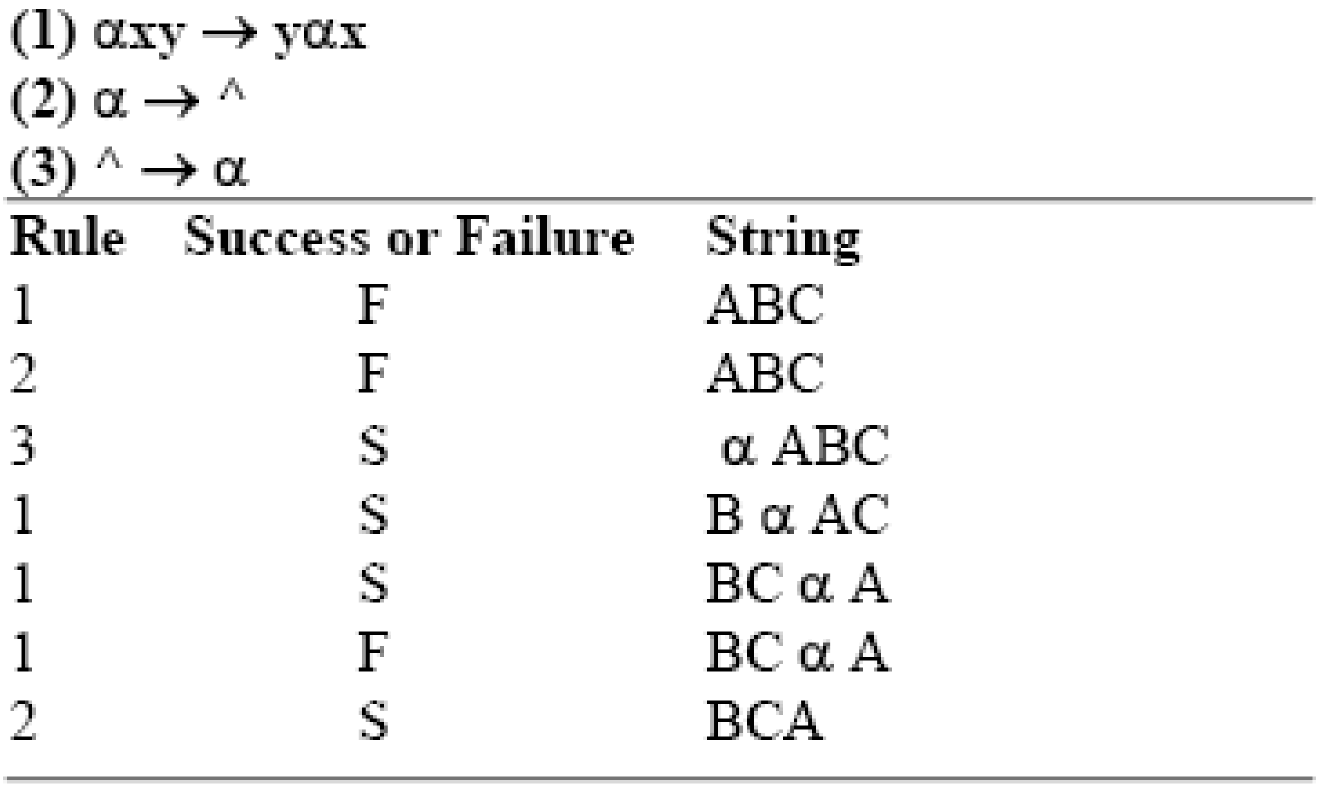
Post Production System

- Basic idea – any mathematical / logical system is simply a set of rules specifying how to change one string of symbols into another string of symbols.
    - these rules are also known as rewrite rules
    - simple syntactic string manipulation
    - no understanding or interpretation is required\also used to define grammars of languages
        - e.g BNF grammars of programming languages.
- Basic limitation – lack of control mechanism to guide the application of the rules.

Markov Algorithm

- An ordered group of productions applied in order or priority to an input string.
- If the highest priority rule is not applicable, we apply the next, and so on.
- An efficient algorithm for systems with many rules.
- Termination on (1) last production not applicable to a string, or (2) production ending with period applied
- Can be applied to substrings, beginning at left

Markov Algorithm



(1) $\alpha xy \rightarrow y\alpha x$

(2) $\alpha \rightarrow$ ^

(3) ^ $\rightarrow \alpha$

Rule	Success or Failure	String
1	F	ABC
2	F	ABC
3	S	α ABC
1	S	B α AC
1	S	BC α A
1	F	BC α A
2	S	BCA

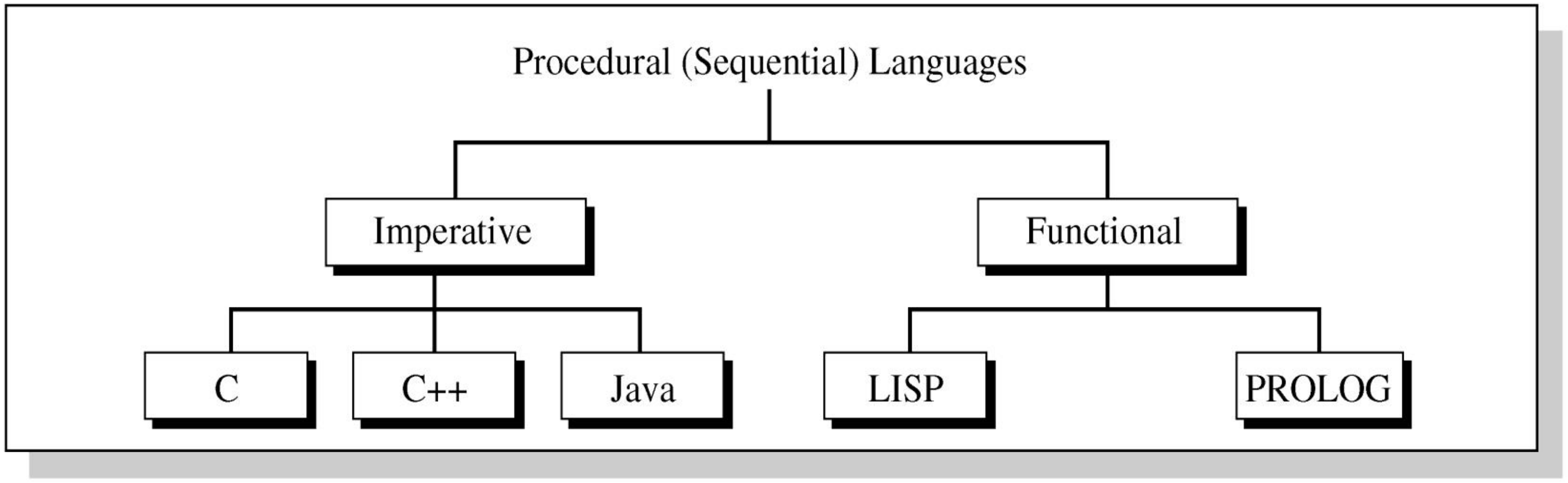
Table 1.11 Execution Trace of a Markov Algorithm

Procedural Paradigms

- Algorithm – method of solving a problem in a finite number of steps.
- Procedural programs are also called sequential programs.
- The programmer specifies exactly how a problem solution must be coded.

Procedural Languages

Figure 1.8 Procedural Languages



Imperative Programming

- Also known as statement-oriented
- During execution, program makes transition from the initial state to the final state by passing through series of intermediate states.
- Provide rigid control and top-down-design.
- Not efficient for directly implementing expert systems.

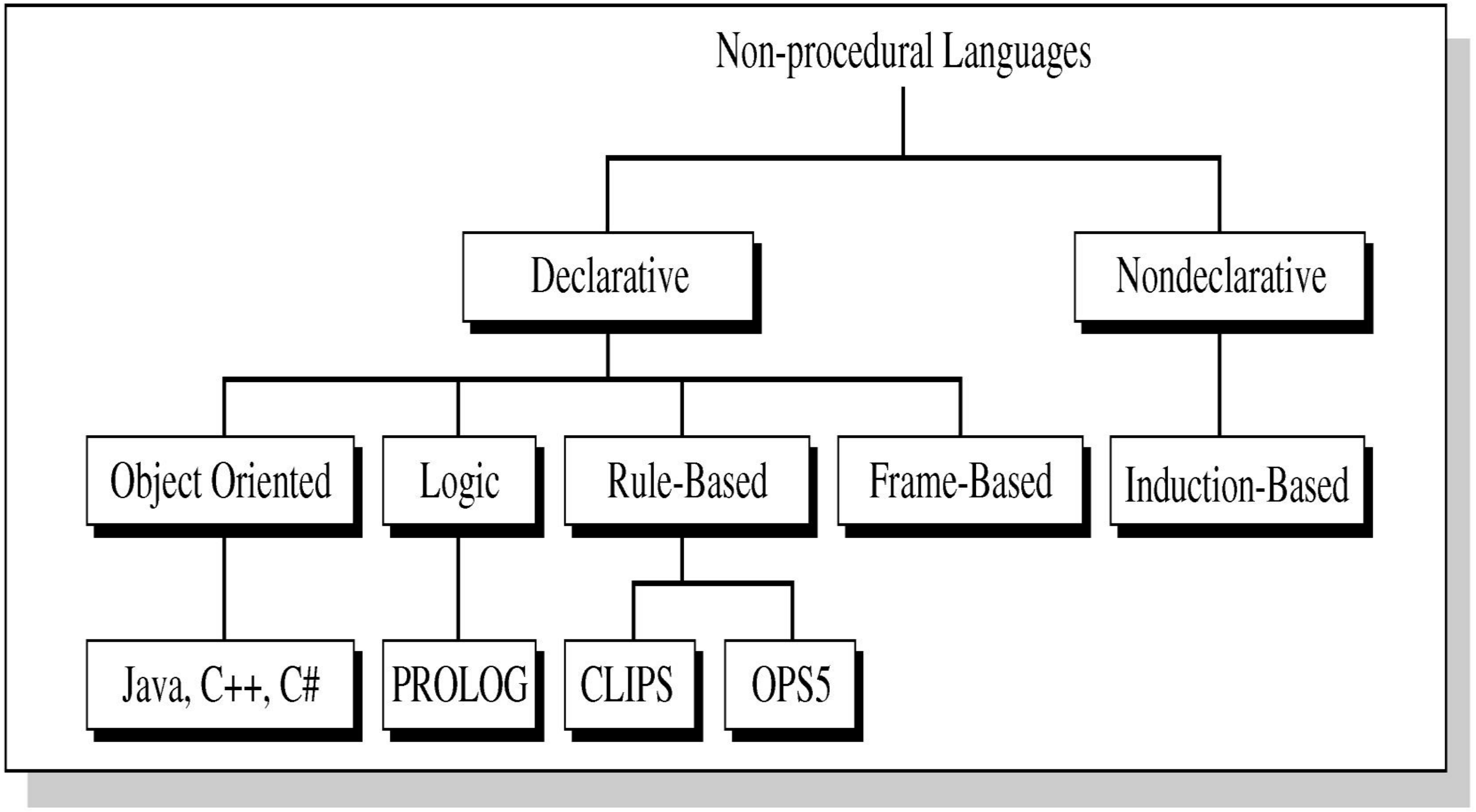
Functional Programming

- Function-based (association, domain, co-domain); f: S→ T
- Not much control
- Bottom-up→ combine simple functions to yield more powerful functions.
- Mathematically a function is an association or rule that maps members of one set, the domain, into another set, the codomain.
- e.g. LISP and Prolog

Nonprocedural Paradigms

- Do not depend on the programmer giving exact details how the program is to be solved.
- Declarative programming – goal is separated from the method to achieve it.
- Object-oriented programming – partly imperative and partly declarative – uses objects and methods that act on those objects.
- Inheritance – (OOP) subclasses derived from parent classes.

Nonprocedural Languages



What are Expert Systems?

Can be considered declarative languages:

- Programmer does not specify how to achieve a goal at the algorithm level.
- Induction-based programming – the program learns by generalizing from a sample.