Problem and Production System Characteristics

By

Dr Sanjay Kumar Singh

Problem Characteristics

1. Is problem decomposable into set of(nearly) independent smaller or easier sub problems?
2. Can solution steps be ignored or at least undone if they prove unwise?
3. Is the problem's universe predictable?
4. Is a good solution to the problem obvious without comparison to all other possible solutions?
5. Is a desire solution a state of the world or a path to a state?
6. Is a large amount of knowledge absolute required to solve the problem, or is knowledge important only to certain the search?
7. Can a computer that is simply given the problem return the solution, or will the solution of problem require interaction between the computer and a person?

1. Is the problem Decomposable?

By this method we can solve large problem easily.

Ex: Decomposable problem

Symbolic Integration

$$\int (x^2 + 3x + \sin^2 x.\cos^2 x)\,dx$$

Can be divided to

Integral of x^2

Integral of $3x$

Integral of $\sin^2 x.\cos^2 x$, which can be further divided to $(1 - \cos^2 x).\cos^2 x$ ….

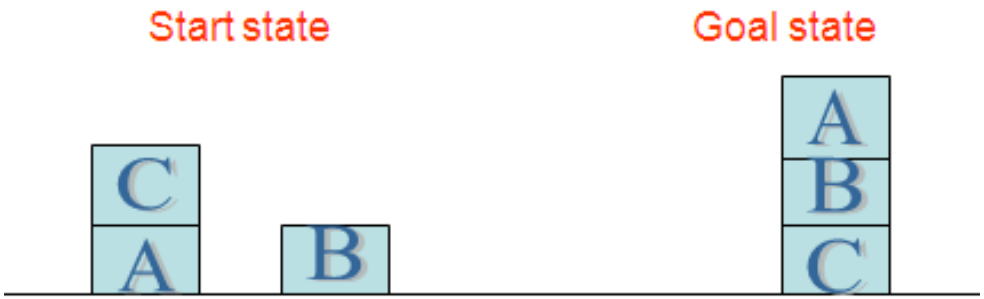
1. Is the problem Decomposable?

Ex: Non- decomposable problems

Block World Problem

Assume that only two operations are available:

1. CLEAR(x)[Block x has nothing on it]->ON(x,Table)[Pick up x and put on the table]
2. Clear(x) and Clear(y)->ON(x,y)[Put x on y]



ON(B,C) and ON(A,B)

2. Can Solution steps be ignored or undone?

- **Ignorable problem:** in which solution steps can be ignored.

Ex:- Theorem Proving

Suppose we are trying to prove a mathematical theorem. We proceed by first proving a lemma that we think will be useful. Eventually, we realize that the lemma is not help at all.

Every thing we need to know to prove theorem is still true and in memory, if it ever was. Any rule that could have been applied at the outset can still be applied. *All we have lost is the effort that was spent exploring the blind alley.*

2. Can Solution steps be ignored or undone?

- Recoverable problem: in which solution steps can be undone.

Ex:- The 8-Puzzle

The 8-puzzle is a square tray in which are placed, eight square tiles and remaining 9th square is uncovered. Each tile has number on it. A tile that is adjacent to blank space can be slide in to that space. A game consist of a starting position and a specific goal position.

We might make stupid move.

We can backtrack and undo the first move. Mistakes can still be recovered from but not quite as easy as in theorem proving.

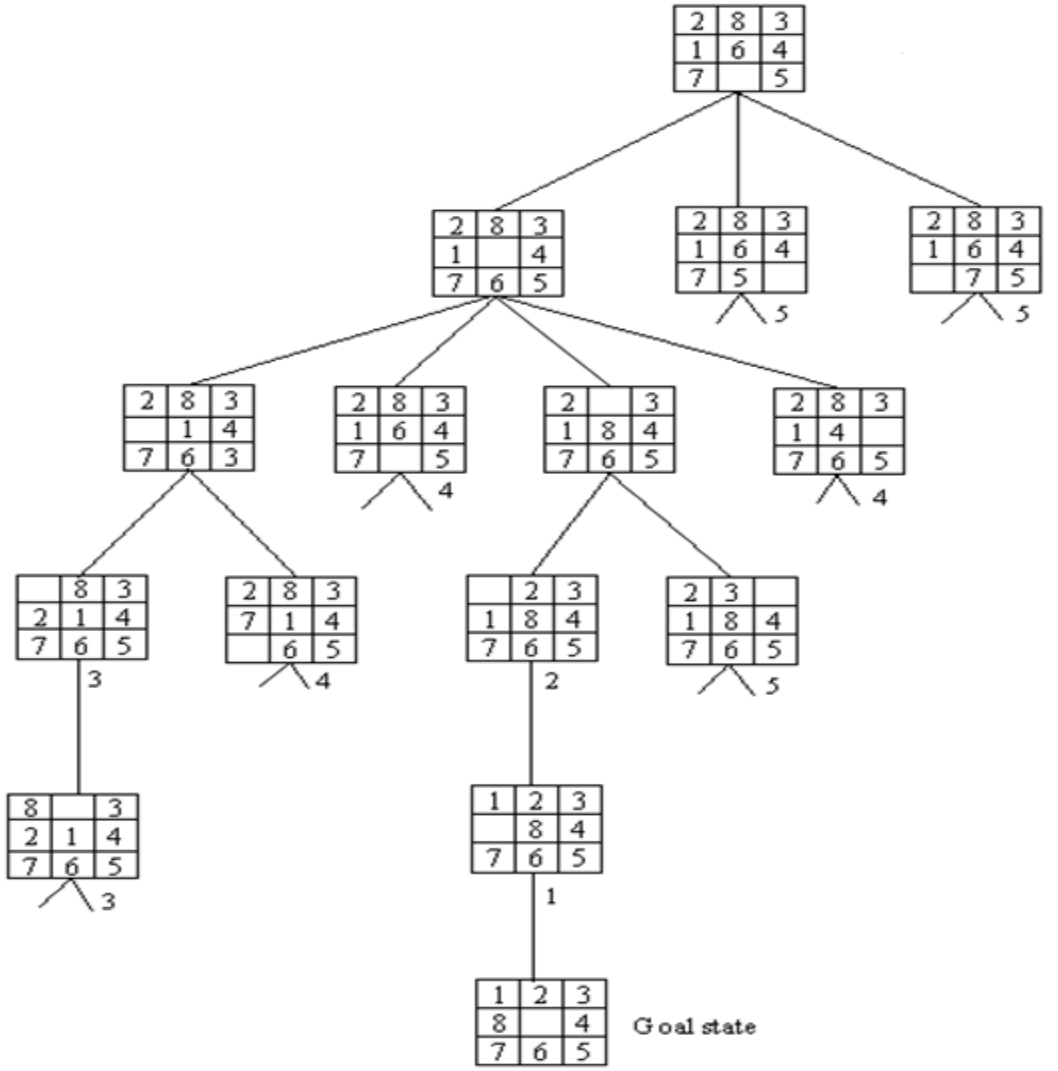
2	8	3
1	6	4
7		5

Initial State

1	2	3
8		4
7	6	5

Goal state

8-Puzzle



2. Can Solution steps be ignored or undone?

- Irrecoverable problem: in which solution steps cannot be undone.

Ex:- Chess

Suppose a chess playing program makes a stupid move and realize it a couple of move later. It cannot simply play as though it never made the stupid move. Nor can it simply backup and start the game over from that point. All it can do is to try to make best of the current situation and go on from there.

2. Can Solution steps be ignored or undone?

- **Ignorable problem** can be solved using a simple control structure that never backtracks. Such a control structure is easy to implement.
- **Recoverable problem** can be solved by slightly more complicated control strategy that does something mistakes and **backtracking** will be necessary to recover from such mistakes.
- **Irrecoverable** problems, solved by a system that expends a great deal of effort making each decision since each the decision must be final.
- **Some irrecoverable** problems can be solved by recoverable style methods used in a **planning process** , in which an entire sequence of steps is analyzed in advance to discover where it will lead before first step is actually taken.

3. Is universe predictable?

Certain-outcome problem

Ex: 8-Puzzle

Every time we make a move, we know exactly what will happen. This is possible to **plan** entire sequence of moves and be confident that we know what the resulting state will be.

Uncertain-outcome problem

Ex: play Bridge

One of the decisions we will have to make is which card to play on the first trick. What we would like to do is to plan entire hand before making the 1st hand. But now it is not possible to do such planning with certainty since we cannot know exactly where all the cards are or what the other players will do on their turn.

4. Is a good solution Absolute or Relative?

Any-path problem

Ex: Answer-question System

Consider the problem of answering the question based on following facts:

1. Marcus was a man.
2. Marcus was a Pompean.
3. Marcus was born in 40 AD.
4. All men are mortal.
5. All Pompeans died when volcano erupted in 79 AD.
6. No mortal lives longer than 150 years.
7. Now it is 1991 AD.

“ Is Marcus alive?”

4. Is a good solution Absolute or Relative?

1. Marcus was a man	- Axiom1
4. All men are mortal	-Axiom4
8. Marcus is Mortal	- 1&4
3. Marcus was born in 40 AD	-Axiom3
7. Now it is 1991 AD	-Axiom7
9. Marcus age is 1951 years	- 3&7
6. No mortal lives longer than 150 years	-Axiom6
10. **Marcus is dead**	-6,8,9

OR

7. It is now 1991AD	-axiom 7
5. All pompeians died in 79 AD	-axiom 5
11. All pompeians are died now	-7 & 5
2. Marcus was a pompeian	-axiom 2
12. **Marcus is dead**	-11,2

Since all we are interested in is the answer to question, it does not matter **which path we follow**.

If we do follow one path successfully to the answer, there is no reason to go back and see if some other path might also lead to a solution.

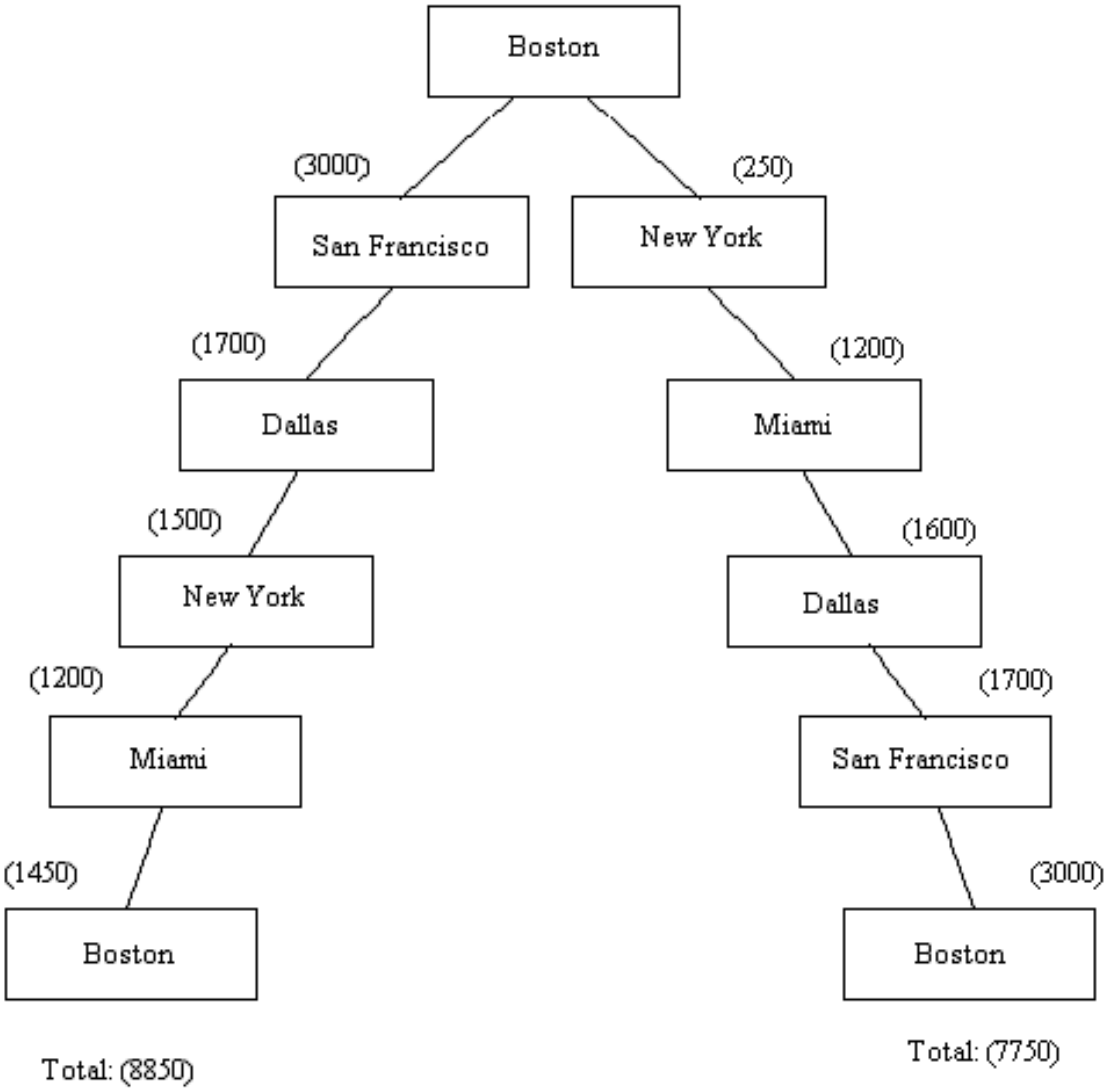
4. Is a good solution Absolute or Relative ?

Best-path problem

Ex: Traveling Salesman Problem

- Given a road map of n cities, find the **shortest** tour which visits every city on the map exactly once and then return to the original city (*Hamiltonian circuit*)

	Boston	New York	Miami	Dallas	S.F.
Boston		250	1450	1700	3000
New York	250		1200	1500	2900
Miami	1450	1200		1600	3300
Dallas	1700	1500	1600		1700
S.F.	3000	2900	3300	1700	


Boston
(3000)
San Francisco
(250)
New York
(1700)
Dallas
(1200)
Miami
(1500)
New York
(1600)
Dallas
(1200)
Miami
(1700)
San Francisco
(1450)
Boston
(3000)
Boston
Total: (8850)
Total: (7750)

4. Is a good solution Absolute or Relative ?

•Best-path problems are, in general, computationally harder than any-path problems.

•Any-path problems can often be solved in a reasonable amount of time by using heuristics that suggest good paths to explore. If the heuristics are not perfect, the search for a solution may not be as direct as possible, but that does not matter.

•For true best-path problems, however, no heuristic that could possibly miss the best solution can be used. So a much more exhaustive search will be performed.

5. Is the solution a State or Path ?

Solution is a path to state

Ex: Water jug problem

Here is not sufficient to report that we have solved the problem and the final state is (2,0).

Here we must report is not the final state but the path that we found to that state.

Thus a statement of solution to this problem must be a sequence of operations (some time called *apian*) that produce the final state.

Solution is a state of world

Ex: Natural language understanding

To solve the problem of finding the interpretation we need to produce interpretation itself. No record of processing by which the interpretation was found is necessary.

"The **bank** president ate a **dish** of pasta salad with the fork".

6. What is the role of knowledge?

Knowledge is important only to constrain the search for solution

Ex: playing chess

Suppose you have ultimate computing power available.

How much knowledge **would be required by a perfect program**?

just the rule for determining legal moves and some simple control mechanism that implement an appropriate search procedure.

Knowledge is required even to be able to recognize a solution

Ex: Scanning daily news paper to decide which are supporting the democrates and which are supporting the republicans in some upcoming elections.

you have ultimate computing power available.

How much knowledge **would be required by a perfect program?**

This time answer is great deal. It would have to know:

- The name of candidates in each party.
- For supporting republicans; you want to see done is have taxes lowered.
- For supporting democrats; you want to see done is improved education for minority students.
- And so on.....

7. Does the task require interaction with person?

Solitary:

in which the computer is given a problem description and produces an answer with no intermediate communication and with no demand for an explanation of the reasoning process.

Level of interaction b/w computer and user is **problem-in solution-out.**

EX: Theorem Proving

Conversational:

in which there is intermediate communication between a person and the computer, either to prove additional assistance to computer or to prove additional information to user, or both.

Ex: Medical diagnosis

1. Class of production Systems

- A ***monotonic production system*** is a system in which the application of rule never prevents the later application of another rule that could also have been applied at the time that the first rule was selected.

- A ***nonmonotonic production system*** is one in which this is not true.

- A ***partially commutative production system*** is a system in with the property that if the application of particular sequence of rules transforms state x into state y, then any permutation of those rules that is allowable also transform state x in to state y.

- A ***commutative production system*** is a production system that is both monotonic and partially commutative.

2. Relationship b/w problems and production systems

Ignorable problems; where creating new thins rather than changing old once

Change occur but can be reversed and in which order of operation is not critical

	Monotonic	**Nonmonotonic**
Partially Commutative	Theorem Proving	Robot Navigation, 8-puzzle
Not Partially Commutative	Chemical synthesis	Bridge, Chess

where creating new thins by changing old once

Reverse not possible and order matter.

It is particularly important to make correct decisions the first time, although Universe is predictable.

Thank You!!!

ARD203/ARI203::AI
Intelligent Agents

By Dr. Sanjay Kumar Singh

Outline

- Agents and environments
- Rationality
- PEAS (Performance measure, Environment, Actuators, Sensors)
- Environment types
- Agent types

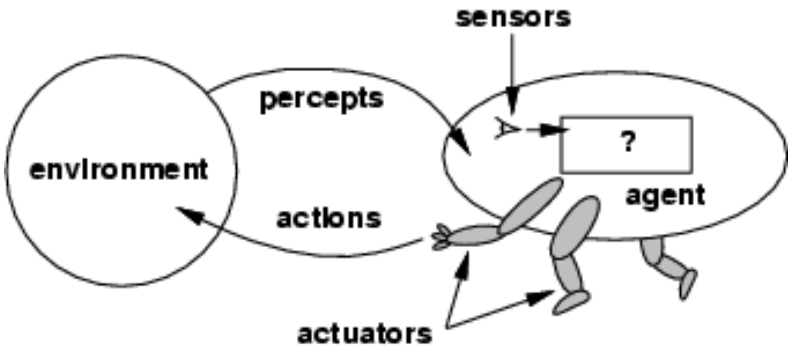
Agents

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators

- **Human agent:**
  - eyes, ears, and other organs for sensors;
  - hands, legs, mouth, and other body parts for actuators

- **Robotic agent:**
  - cameras and infrared range finders for sensors
  - various motors for actuators

Agents and environments

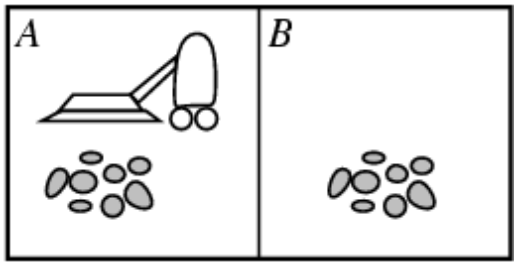


- The agent function maps from percept histories to actions:

$$[f: \mathcal{P}^{\star} \rightarrow \mathcal{A}]$$

- The agent program runs on the physical architecture to produce *f*
- agent = architecture + program

Vacuum-cleaner world



- Percepts: location and contents, e.g., [A,Dirty]
- Actions: *Left*, *Right*, *Suck*, *NoOp*
- *Agent's function → look-up table*
  - *For many agents this is a very large table*

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Dirty]$	$Suck$
⋮	⋮

Rational agents

- Rationality
  - Performance measuring success
  - Agents prior knowledge of environment
  - Actions that agent can perform
  - Agent's percept sequence to date

- Rational Agent: For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.
-

Rationality

- Rational is different from omniscience
  - Percepts may not supply all relevant information
  - E.g., in card game, don't know cards of others.

- Rational is different from being perfect
  - Rationality maximizes expected outcome while perfection maximizes actual outcome.

Characterizing a Task Environment

- Must first specify the setting for intelligent agent design.
- **PEAS: Performance measure, Environment, Actuators, Sensors**

- **Example:** the task of designing a self-driving car
-
  - **Performance measure** Safe, fast, legal, comfortable trip
  - **Environment** Roads, other traffic, pedestrians
  - **Actuators** Steering wheel, accelerator, brake, signal, horn
  - **Sensors** Cameras, LIDAR (light/radar), speedometer, GPS, odometer engine sensors, keyboard

Environment types

- Fully observable (vs. partially observable)
- Deterministic (vs. stochastic)
- Episodic (vs. sequential)
- Static (vs. dynamic)
- Discrete (vs. continuous)
- Single agent (vs. multiagent):

Fully observable (vs. partially observable)

- Is everything an agent requires to choose its actions available to it via its sensors? Perfect or Full information.
  - If so, the environment is fully accessible
- If not, parts of the environment are inaccessible
  - Agent must make informed guesses about world.
- In decision theory: perfect information vs. imperfect information.

Cross Word	**Poker**	**Backgammon**	**Taxi driver**	**Part picking robot**	**Image analysis**
Fully	Partially	Partially	Partially	Fully	Fully

Deterministic (vs. stochastic)

- Does the change in world state
  - Depend only on current state and agent's action?
- Non-deterministic environments
  - Have aspects beyond the control of the agent
  - Utility functions have to guess at changes in world

Cross Word	Poker	Backgammon	Taxi driver	Part picking robot	Image analysis
Deterministic	Stochastic	Stochastic	Stochastic	Stochastic	Deterministic

Episodic (vs. sequential):

- Is the choice of current action
  - Dependent on previous actions?
  - If not, then the environment is episodic
- In non-episodic environments:
  - Agent has to plan ahead:
    - Current choice will affect future actions

Cross Word	Poker	Backgammon	Taxi driver	Part picking robot	Image analysis
Sequential	Sequential	Sequential	Sequential	Episodic	Episodic

Static (vs. dynamic):

- Static environments don't change
  - While the agent is deliberating over what to do
- Dynamic environments do change
  - So agent should/could consult the world when choosing actions
  - Alternatively: anticipate the change during deliberation OR make decision very fast
- Semidynamic: If the environment itself does not change with the passage of time but the agent's performance score does.

Cross Word	Poker	Backgammon	Taxi driver	Part picking robot	Image analysis
Static	Static	Static	Dynamic	Dynamic	Semi

Another example: off-line route planning vs. on-board navigation system

Discrete (vs. continuous)

- A limited number of distinct, clearly defined percepts and actions vs. a range of values (continuous)

Cross Word	**Poker**	**Backgammon**	**Taxi driver**	**Part picking robot**	**Image analysis**
Discrete	Discrete	Discrete	Conti	Conti	Conti

Single agent (vs. multiagent):

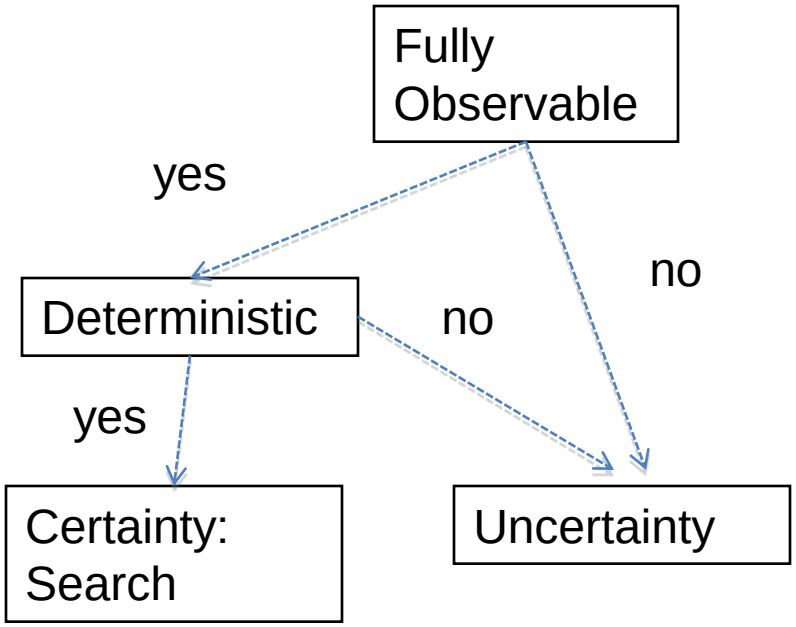
- An agent operating by itself in an environment or there are many agents working together

Cross Word	Poker	Backgammon	Taxi driver	Part picking robot	Image analysis
Single	Multi	Multi	Multi	Single	Single

Summary.

	Observable	Deterministic	Episodic	Static	Discrete	Agents
Cross Word	Fully	Deterministic	Sequential	Static	Discrete	Single
Poker	Fully	Stochastic	Sequential	Static	Discrete	Multi
Backgammon	Partially	Stochastic	Sequential	Static	Discrete	Multi
Taxi driver	Partially	Stochastic	Sequential	Dynamic	Conti	Multi
Part picking robot	Partially	Stochastic	Episodic	Dynamic	Conti	Single
Image analysis	Fully	Deterministic	Episodic	Semi	Conti	Single

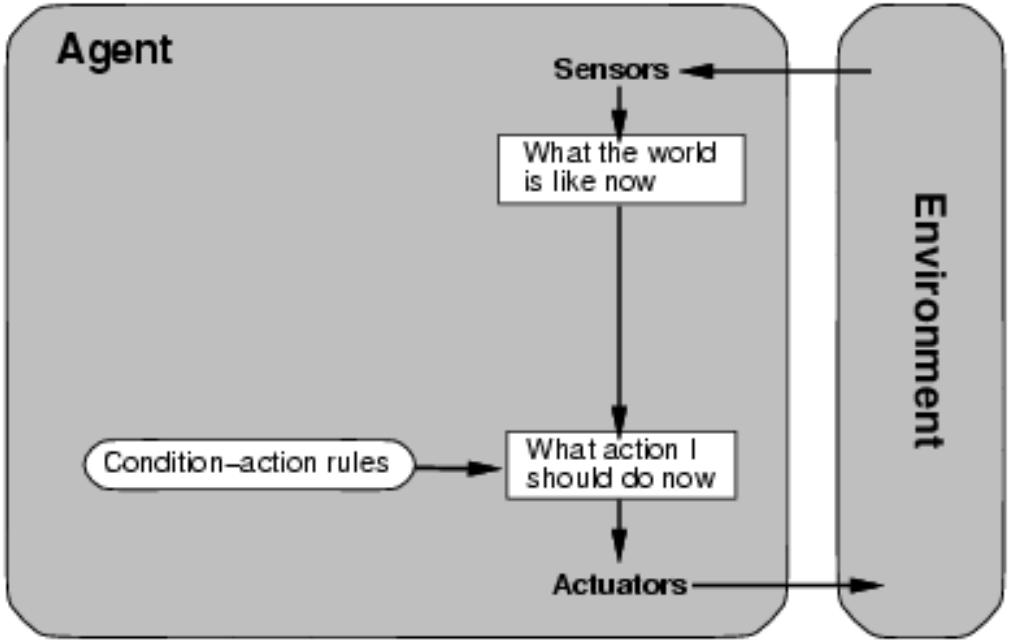
Choice under (Un)certainty



Agent types

- Four basic types in order of increasing generality:
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents
  - Learning agents

Simple reflex agents




```
function REFLEX-VACUUM-AGENT( [location,status]) returns an action
    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```

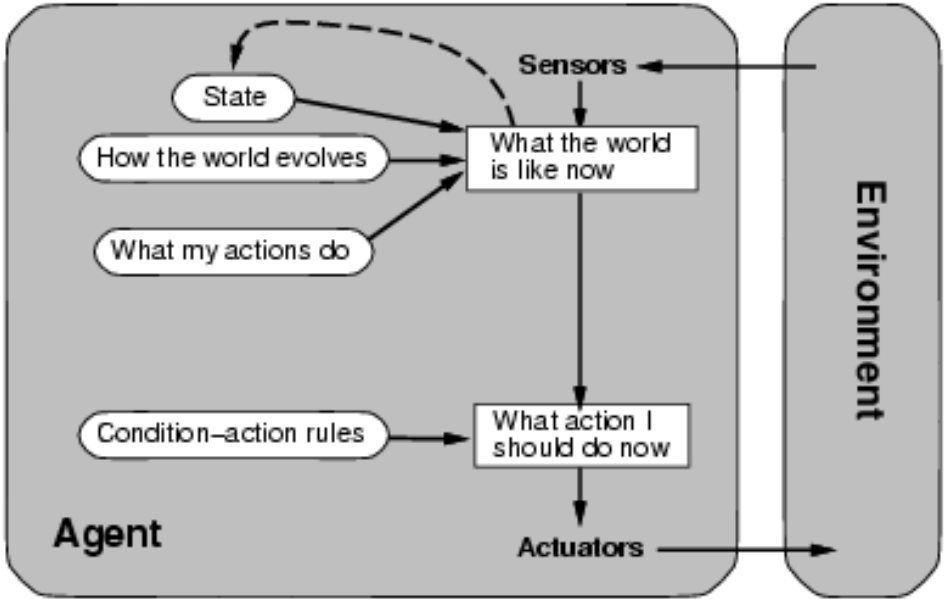
Simple reflex agents

- Simple but very limited intelligence.
- **Action does not depend on percept history, only on current percept.**
- Therefore no memory requirements.
- Infinite loops
  - Suppose vacuum cleaner does not observe location. What do you do given location = clean? Left of A or right on B -> infinite loop.
  - Fly buzzing around window or light.
  - Possible Solution: Randomize action.
- Chess – openings, endings
  - Lookup table (not a good idea in general)
    - 35^{100} entries required for the entire game

States and Memory: Game Theory

- If each state includes the information about the percepts and actions that led to it, the state space has **perfect recall**.
- **Perfect Information** = Perfect Recall + Full Observability + Deterministic Actions.

Model-based reflex agents



- **Know how world evolves**
  - Overtaking car gets closer from behind
- **How agents actions affect the world**
  - Wheel turned clockwise takes you right
- **Model base agents update their state**

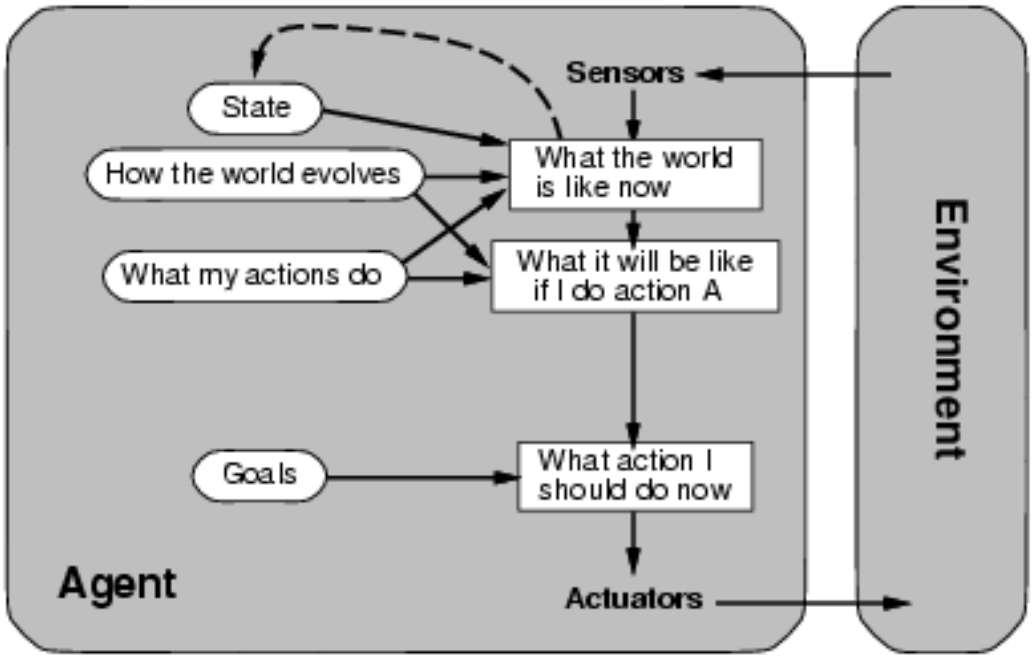
```
function REFLEX-AGENT-WITH-STATE(percept) returns action
  static: state, a description of the current world state
          rules, a set of condition-action rules

  state ← UPDATE-STATE(state, percept)
  rule ← RULE-MATCH(state, rules)
  action ← RULE-ACTION[rule]
  state ← UPDATE-STATE(state, action)
  return action
```

Goal-based agents

- knowing state and environment? Enough?
  - Taxi can go left, right, straight
- Have a goal
  - A destination to get to
- Uses knowledge about a goal to guide its actions
  - E.g., Search, planning

Goal-based agents

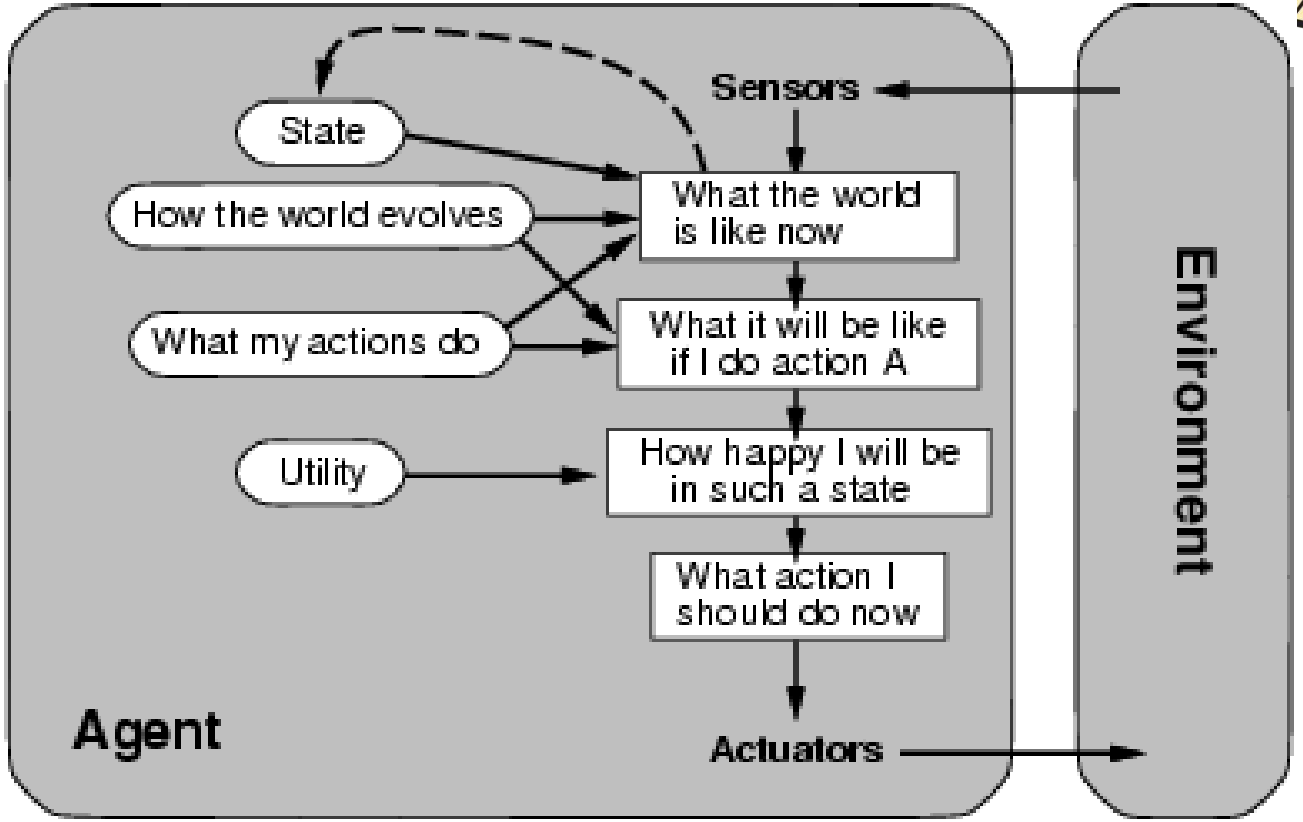


- Reflex agent breaks when it sees brake lights. Goal based agent reasons
  - Brake light -> car in front is stopping -> I should stop -> I should use brake

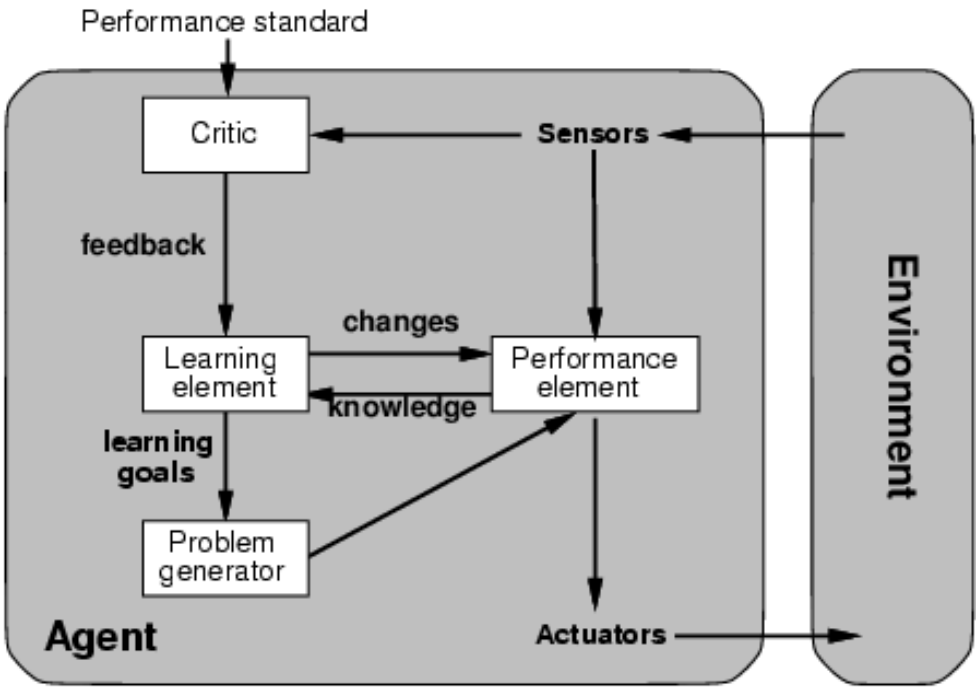
Utility-based agents

- Goals are not always enough
  - Many action sequences get taxi to destination
  - Consider other things. How fast, how safe…..
- A utility function maps a state onto a real number which describes the associated degree of “happiness”, “goodness”, “success”.
- Where does the utility measure come from?
  - Economics: money.
  - Biology: number of offspring.
  - Your life?

Utility-based agents

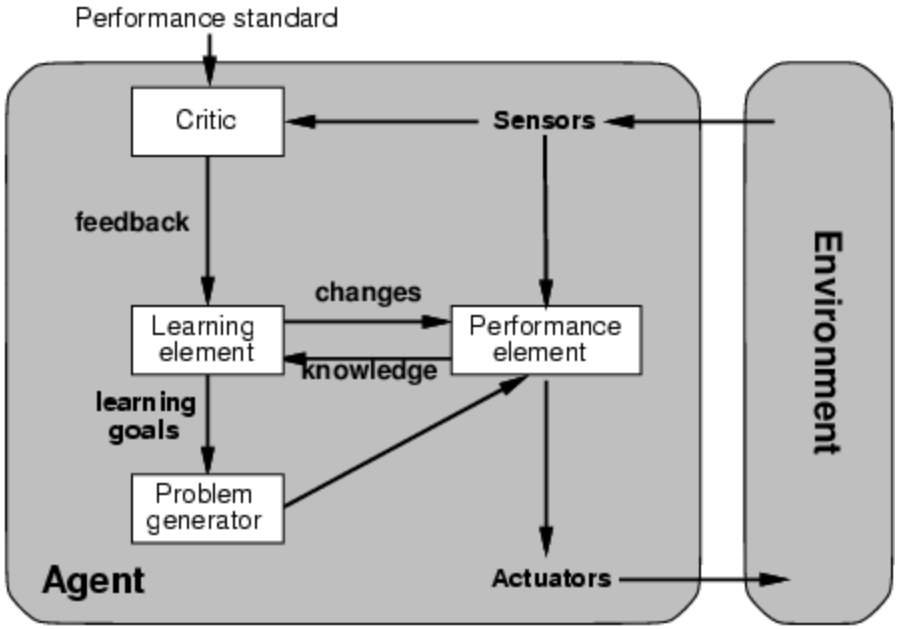


Learning agents



- Performance element is what was previously the whole agent
  - Input sensor
  - Output action
- Learning element
  - Modifies performance element.

Learning agents

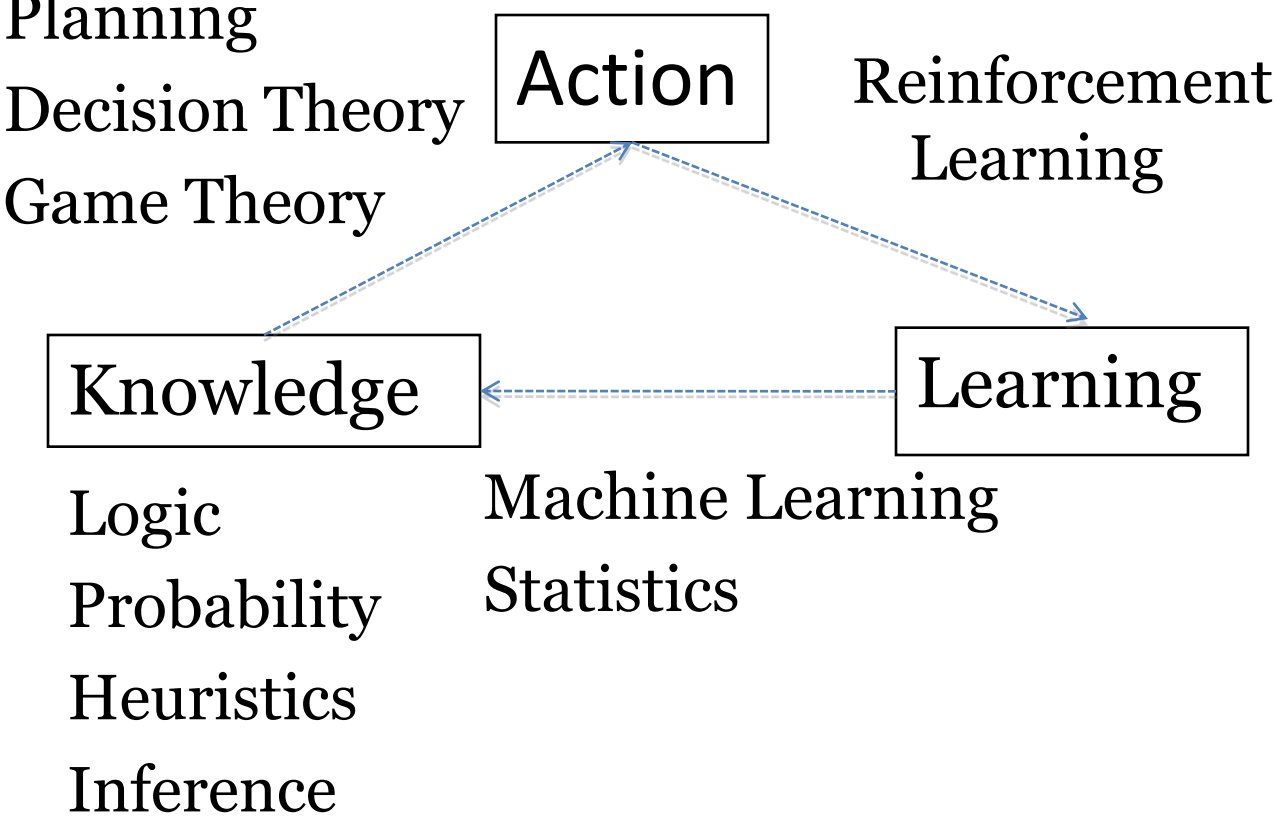


- Critic: how the agent is doing
  - Input: checkmate?
  - Fixed
- Problem generator
  - Tries to solve the problem differently instead of optimizing.
  - Suggests **exploring** new actions -> new problems.

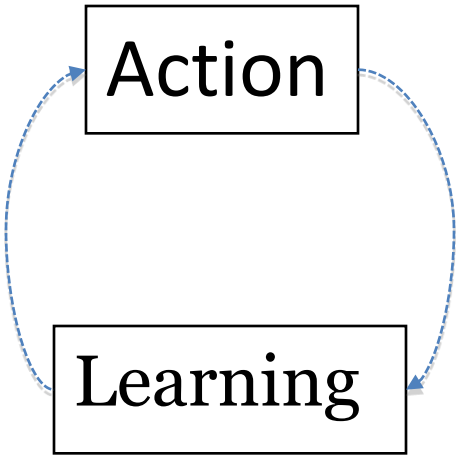
Learning agents(Taxi driver)

- Performance element
  - How it currently drives
- Taxi driver Makes quick left turn across 3 lanes
  - Critics observe shocking language by passenger and other drivers and informs bad action
  - Learning element tries to modify performance elements for future
  - Problem generator suggests experiment out something called Brakes on different Road conditions
- Exploration vs. Exploitation
  - Learning experience can be costly in the short run
  - shocking language from other drivers
  - Less tip
  - Fewer passengers

The Big Picture: AI for Model-Based Agents



The Picture for Reflex-Based Agents



Reinforcement Learning

- Studied in AI, Cybernetics, Control Theory, Biology, Psychology.

Thank You !!!

Artificial Intelligence

ARD203/ARI203

Definitions of AI

There are various definitions given by experts in describing what is Artificial Intelligence

There are three kinds of intelligence:
one kind understand things for itself,
the other appreciate what other can understand,
the third understands neither for itself nor through others.
The first kind is excellent, the second good and the third kind useless.

-Niccolo Machiavelli
(1469-1527, Italian Diplomat)

What is artificial intelligence?

There are no clear consensus on the definition of AI

INTELLIGENCE

Intelligence is the computational part of the ability to achieve goals in the world. Varying kinds and degrees of intelligence occur in people, many animals and some machines.

ARTIFICIAL INTELLIGENCE

It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable.

Other possible AI definitions

- AI is a collection of hard problems which can be solved by humans and other living things, but for which we don't have good algorithms for solving.
  - e. g., understanding spoken natural language, medical diagnosis, circuit design, learning, self-adaptation, reasoning, chess playing, proving math theories, etc.

- A program that
  - Acts like human (Turing test)
  - Thinks like human (human-like patterns of thinking steps)
  - Acts or thinks rationally (logically, correctly)

Easy Problems in AI

- It's been easier to mechanize many of the high level cogniti
  tasks we usually associate with "intelligence" in people
  - e. g., symbolic integration, proving theorems, playing chess, some aspect of medical diagnosis, etc.

Hard Problems in AI

- It's been very hard to mechanize tasks that animals can do easily
  - catching prey and avoiding predators
  - interpreting complex sensory information (visual, aural, …)
  - modeling the internal states of other animals from their behavior
  - working as a team (ants, bees)

Examples of AI problems (Application Areas)

1.Expert Consulting Systems

A key problem in the development of Expert Consulting System is how to represent and use the knowledge that human experts in these subjects obviously posses and use.

This problem is more difficult by the fact that the expert knowledge in any important field is imprecise and uncertain.

Examples of AI problems (contd.) (Application Areas)

2. Theorem Proving

Finding proof of a mathematical theorem requires following intelligence.

- Requires the ability to make deductions from hypothesis.
- It demands intuitive scales such as guessing which path should be proved first in order to help proving the theorem.
- It also requires judgments to guess accurately about which previously proven theorems in a subject area will be useful in the present proof.
- Also sometimes it is needed to break the main problem into sub-problems to work on independently.

Examples of AI problems(contd.)
(Application Areas)

3. Robotics

It deals with the problems of controlling the physical actions of a mobile Robot.

4. Automatic Programming

In automatic programming, a system takes in a high level description of what program is to accomplish and produce a program.

Examples of AI problems (contd.) (Application Areas)

5. Perceptional Problems

- Computers are made to see their surroundings by fitting input devices.
- Also they are made to hear speaking voices by providing with microphone inputs.
- But it requires processing of large base knowledge about the things being perceived.

6. Natural Language Processor

- This field concerned with the efforts of making computers to understand spoken and written languages.
- In order to understand sentences about a topic, it is necessary not only a lot about the vocabulary and grammar, but also a good deal about the topic so that unstated assumptions can be recognized.

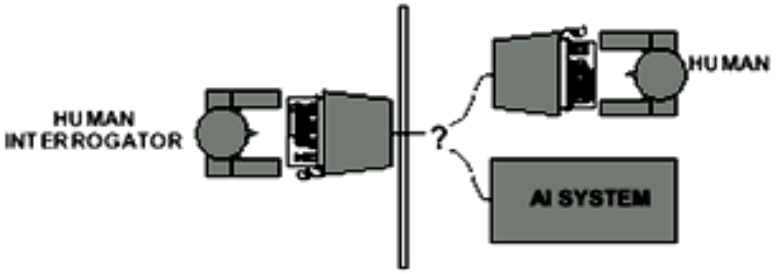
A Brief History of Artificial Intelligence

The birth of AI (1943 – 1956)

- Pitts and McCulloch (1943): simplified mathematical model of neurons (resting/firing states) can realize all propositional logic primitives (can compute all Turing computable functions)
- Allen Turing: Turing machine and Turing test (1950)
- **John McCarthy**, an American computer scientist pioneer and inventor, was called the "*Father of Artificial Intelligence*."

Turing Test

- Alan Turing's 1950 article *Computing Machinery and Intelligence* discussed conditions for considering a machine to be intelligent
  - "Can machines think?" ←→ "Can machines behave intelligently?"
  - The Turing test (The Imitation Game): Operational definition of intelligence.



- Three rooms contain a person, a computer, and an interrogator.
- The interrogator can communicate with the other two by teleprinter.
- The interrogator tries to determine which is the person and which is the machine.
- The machine tries to fool the interrogator into believing that it is the person.
- If the machine succeeds, then we conclude that the machine can think.

What would a computer need to pass the Turing test?

- Natural language processing: to communicate with examiner.
- Knowledge representation: to store and retrieve information provided before or during interrogation.
- Automated reasoning: to use the stored information to answer questions and to draw new conclusions.
- Machine learning: to adapt to new circumstances and to detect and extrapolate patterns.
- Vision (for Total Turing test): to recognize the examiner's actions and various objects presented by the examiner.
- Motor control (total test): to act upon objects as requested.
- Other senses (total test): such as audition, smell, touch, etc.

Simple Definitions

- **Data**

  Data are the facts and figures about a particular activity.

- **Data Processing**

  The process of collecting all the required data together to produce meaningful information.

- **Information**

  Information is obtained by processing the data into meaningful form

- **Knowledge**

  It is structured representation of all the facts of an AI problem

- **Knowledge base**

  The facts or assertions about the problem domain.

- **Data base**

  The storage medium for the state variables.

Programming languages for AI

- The relational languages like PROLOG [PROgramming in LOgic] AND LISP [LISt Processing] in AI.
- LISP is well suited for handling lists, where as PROLOG is designed for logic Programming

How is AI research done?

- AI research has both theoretical and experimental sides. The experimental side has both basic and applied aspects.

- There are two main lines of research:
  - One is biological, based on the idea that since humans are intelligent, AI should study humans and imitate their psychology or physiology.
  - The other is phenomenal, based on studying and formalizing common sense facts about the world and the problems that the world presents to the achievement of goals.

- The two approaches interact to some extent, and both should eventually succeed. It is a race, but both racers seem to be walking. [**John McCarthy]**
- -**John McCarthy**: Computer scientist known as the father of AI,
- -In 1958, McCarthy invented the computer programming language LISP, the second oldest programming language after FORTRAN

Some of the Task Domains of Artificial Intelligence

Mundane Tasks

- Perception
  - Vision
  - Speech
- Natural language
  - Understanding
  - Generation
  - Translation
- Commonsense reasoning
- Robot control

Some of the Task Domains of Artificial Intelligence

Formal Tasks

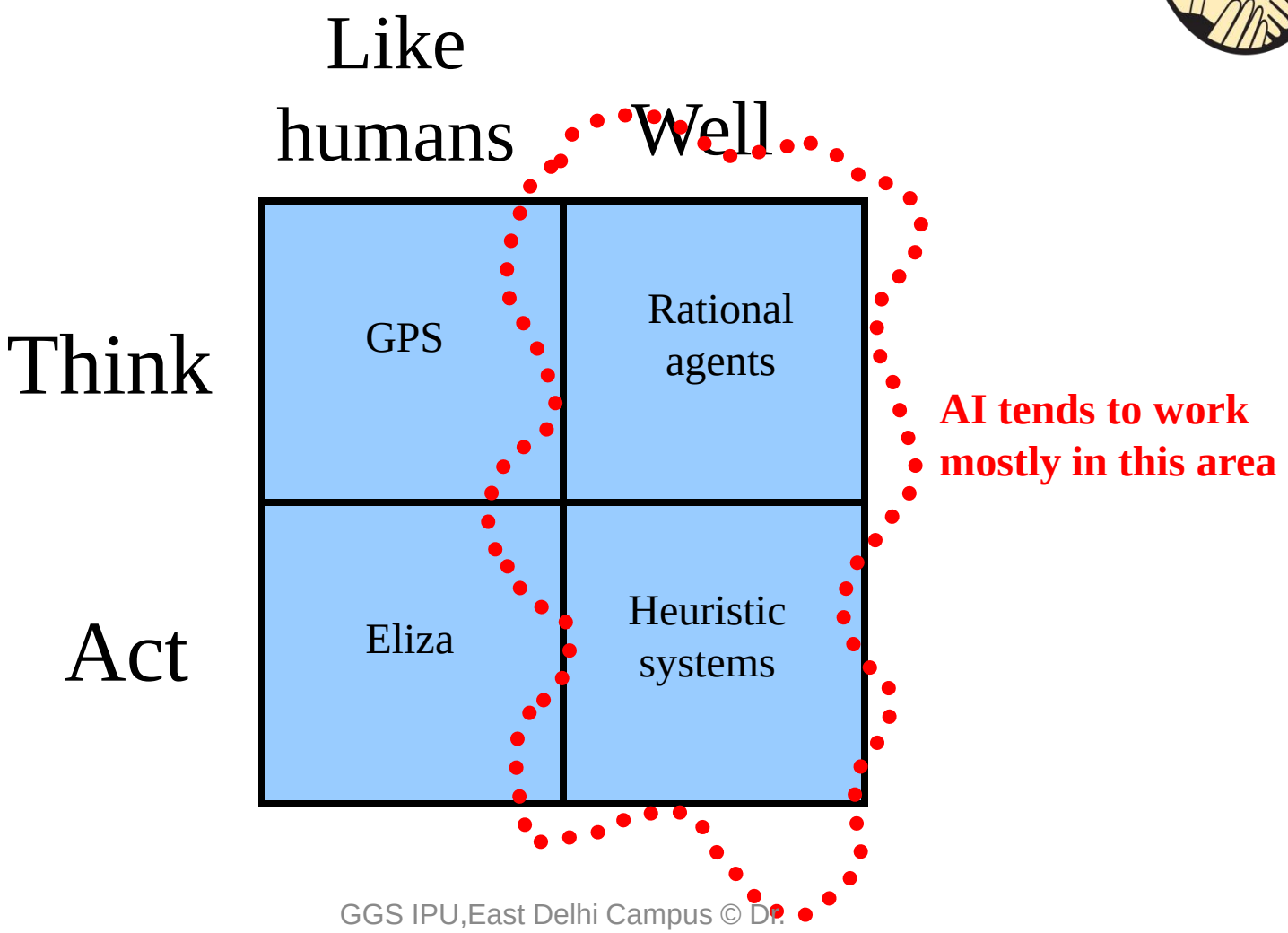
- Games
  - Chess
  - Backgammon
  - Checkers -Go
- Mathematics
  - Geometry
  - Logic
  - Integral calculus
  - Proving properties of programs

Some of the Task Domains of Artificial Intelligence

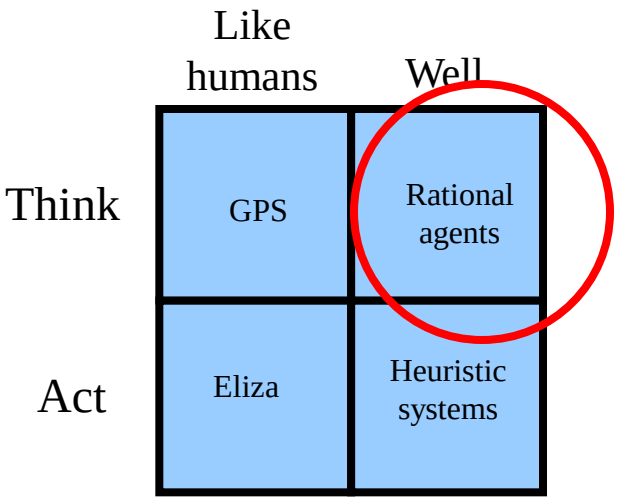
Expert Tasks

- Engineering
  - Design
  - Fault finding
  - Manufacturing planning
- Scientific analysis
- Medical diagnosis
- Financial analysis

Possible Approaches



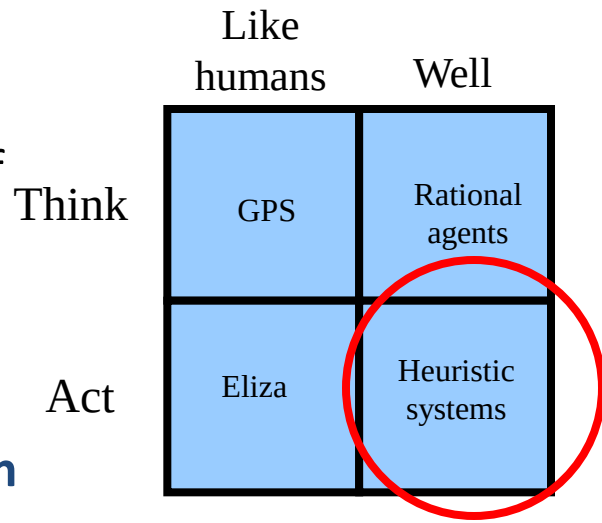
Think well



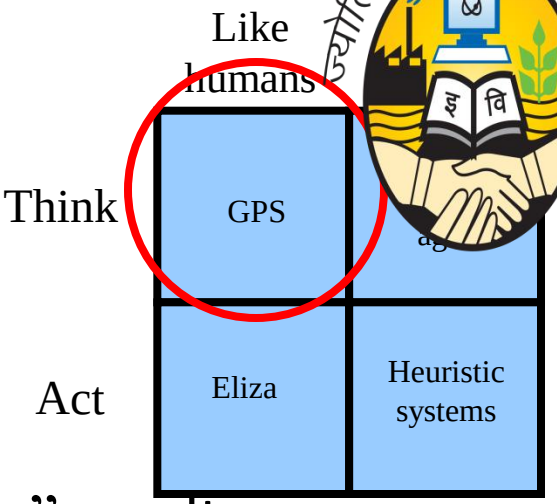
- Develop formal models of knowledge representation, reasoning, learning memory, problem solving, that can be rendered in algorithms.
- There is often an emphasis on a systems that are provably correct, and guarantee finding an optimal solution.

Act well

- For a given set of inputs, generate an appropriate output that is not necessarily correct but gets the job done.
- A **heuristic** (**heuristic rule**, **heuristic method**) is a rule of thumb, strategy, trick, simplification, or any other kind of device which drastically limits search for solutions in large problem spaces.
- Heuristics do not guarantee optimal solutions; in fact, they do not guarantee any solution at all: **all that can be said for a useful heuristic is that it offers solutions which are good enough most of the time.**
  – Feigenbaum and Feldman, 1963, p. 6

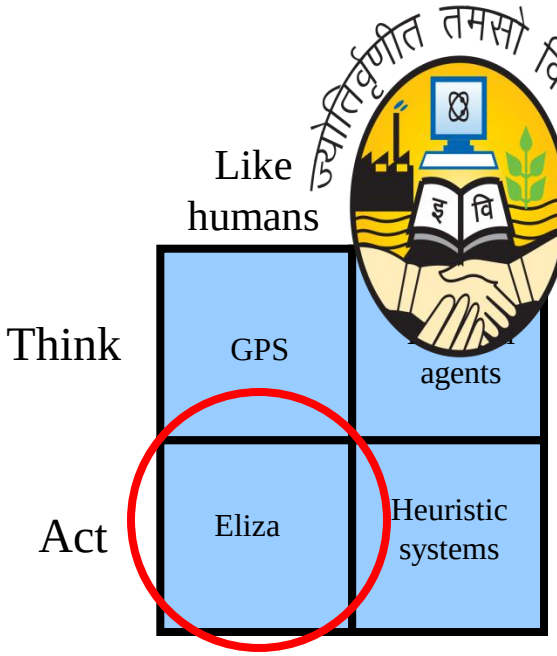


Think like humans



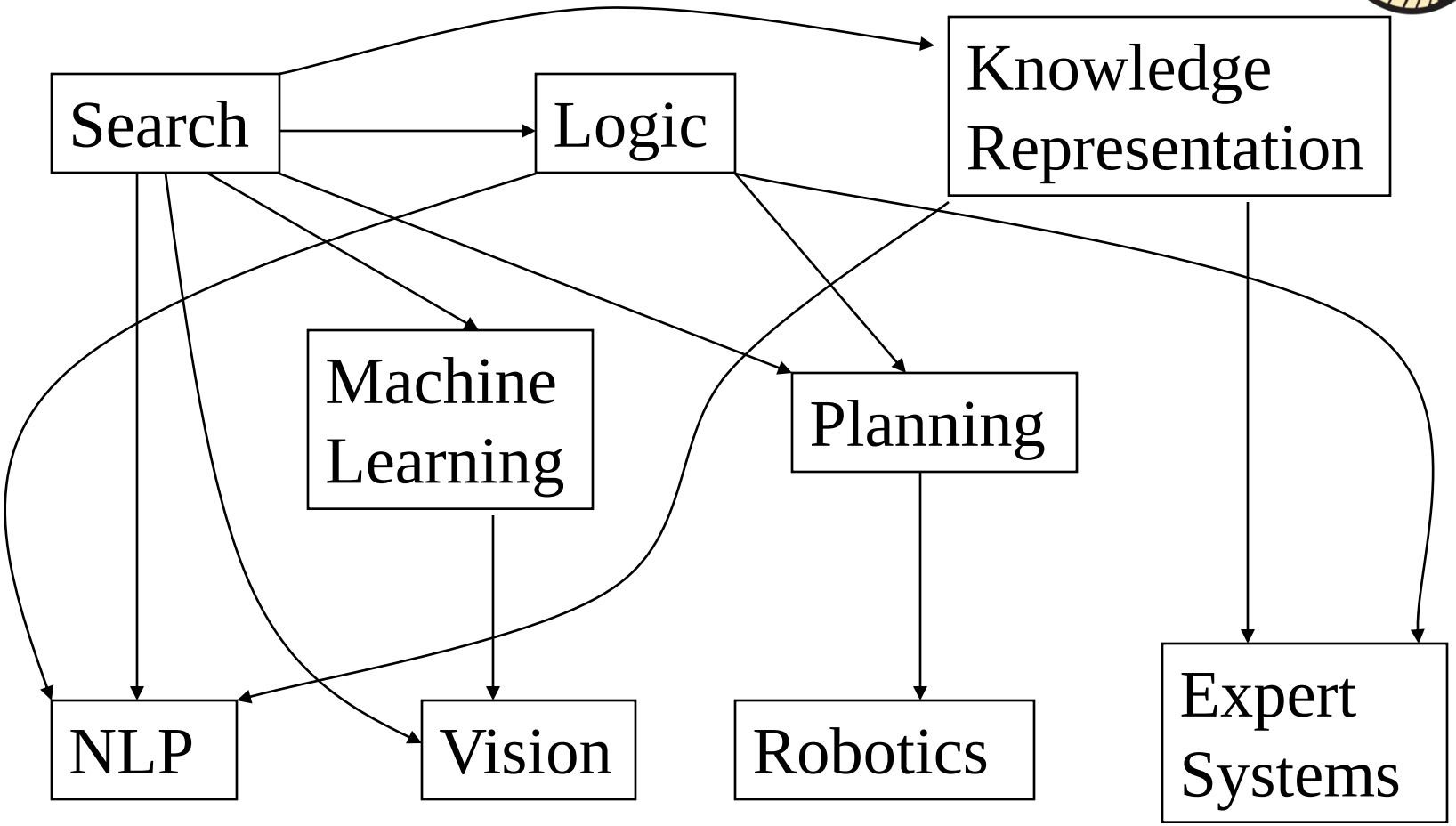
- Cognitive science approach
- Focus not just on behavior and I/O also look at reasoning process.
- Computational model should reflect "how" results were obtained.
- Provide a new language for expressing cognitive theories and new mechanisms for evaluating them
- **GPS (General Problem Solver):** Goal not just to produce humanlike behavior (like ELIZA), but to produce a sequence of steps of the reasoning process that was similar to the steps followed by a person in solving the same task.

Act like humans



- Behaviorist approach.
- Not interested in how you get results, just the similarity to what human results are.
- Exemplified by the Turing Test (Alan Turing, 1950).
- ELIZA: A program that simulated a psychotherapist interacting with a patient and successfully passed the Turing Test.
- Coded at MIT during 1964-1966 by Joel Weizenbaum.

Areas of AI and their inter-dependenc



Branches of AI

- **Logical AI**
- **Search**
- **Natural language processing**
- **pattern recognition**
- **Knowledge representation**
- **Inference** From some facts, others can be inferred.
- **Automated reasoning**
- **Learning from experience**
- **Planning** To generate a strategy for achieving some goal
- **Epistemology** This is a study of the kinds of knowledge that are required for solving problems in the world.
- **Genetic programming**
- **Emotions???**
- …

Thank You !!!

Introduction to Expert Systems

Dr. Sanjay Kumar Singh

Objectives

- Learn the meaning of an expert system
- Understand the problem domain and knowledge domain
- Learn the advantages of an expert system
- Understand the stages in the development of an expert system
- Examine the general characteristics of an expert system

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

- Expert Systems = knowledge-based systems = knowledge-based expert systems

What is an expert system?

- Emulation (mimics cause/process) is stronger than simulation (mimics outward appearance) which is required to act like the real thing in only some aspects.
- The basic idea is that if a human expert can specify the steps of reasoning by which a problem may be solved, so too can an expert system.
- Restricted domain expert systems (extensive use of specialized knowledge at the level of human expert) function well which is not the case of general-purpose problem solver.
- Knowledge based system are designed to act as an intelligent systems.

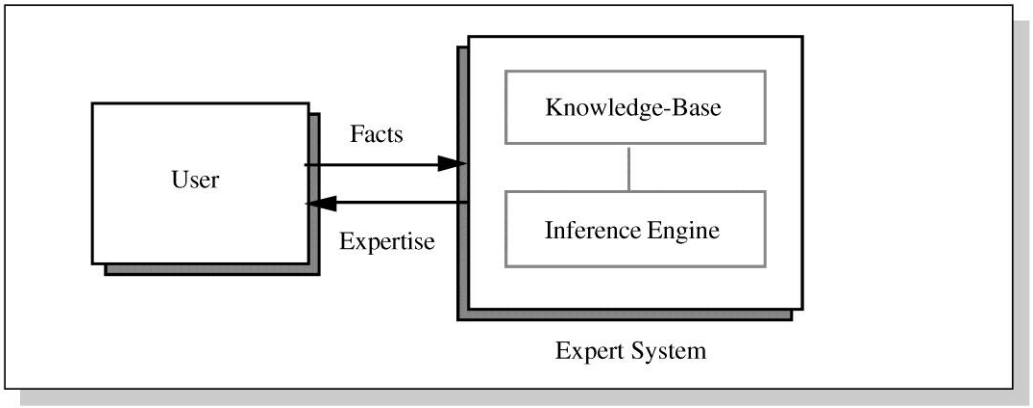
Expert System Main Components

- Knowledge base – obtainable from books, magazines, knowledgeable persons, etc; or expertise knowledge.

- Inference engine – draws conclusions from the knowledge base.

Basic Functions of Expert Systems

Figure 1.2 Basic Function of an Expert System



Advantages of Expert Systems

- Increased availability: on suitable computer hardware
- Reduced cost
- Reduced danger: can be used in hazardous environment.
- Permanence: last for ever, unlike human who may die, retire, quit.
- Multiple expertise: several experts' knowledge leads to
- Increased reliability

Advantages Continued

- Explanation: explain in detail how arrived at conclusions.
- Fast response: (e.g. emergency situations).
- Steady, unemotional, and complete responses at all times: unlike human who may be inefficient because of stress or fatigue.
- Intelligent tutor: provides direct instructions (student may run sample programs and explaining the system's reasoning).
- Intelligent database: access a database intelligently (e.g. data mining).

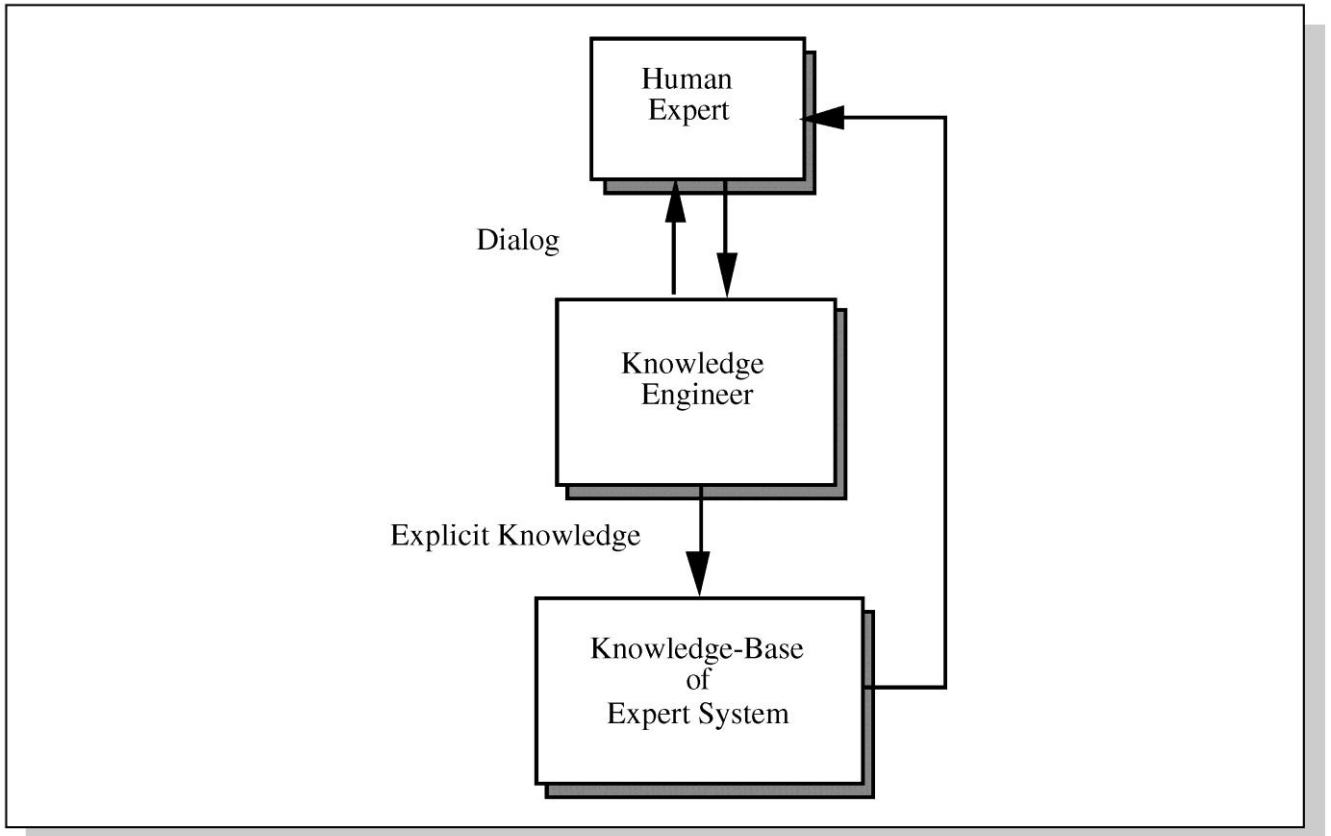
Representing the Knowledge

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

```
IF the light is red THEN stop
```

Development of an Expert System

Figure 1.4 Development of an Expert System



Limitations of Expert Systems

- Uncertainty = having limited knowledge (more than possible outcomes)
- Both human experts and expert systems must be able to deal with uncertainty.
- **Limitation 1**: most expert systems deals with shallow knowledge than with deep knowledge.
- Shallow knowledge – based on empirical and heuristic knowledge. (aspirin for headache.)
- Deep knowledge – based on basic structure, function, and behavior of objects.

Limitations of Expert Systems

- **Limitation 2**: typical expert systems cannot generalize through analogy to reason about new situations in the way people can.
- Solution 1 for limitation 2: repeating the cycle of interviewing the expert.
- Limitation raised form Solution 1: A knowledge acquisition bottleneck results from the time-consuming and labor intensive task of building an expert system.

Early Expert Systems

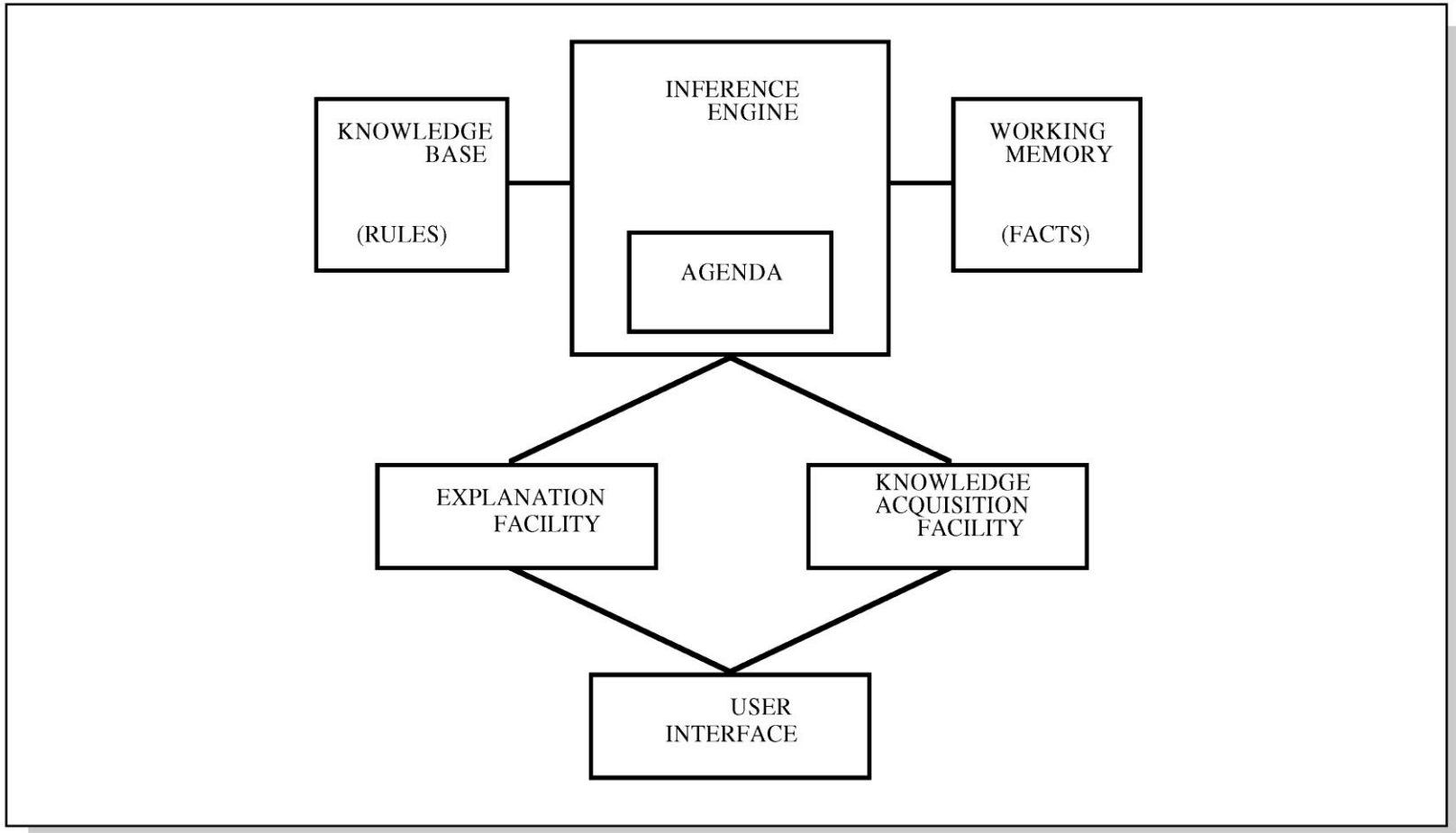
- DENDRAL – used in chemical mass spectroscopy to identify chemical constituents
- MYCIN – medical diagnosis of illness
- DIPMETER – geological data analysis for oil
- PROSPECTOR – geological data analysis for minerals
- XCON/R1 – configuring computer systems

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.

Structure of a Rule-Based Expert System

Figure 1.6 Structure of a Rule-Based Expert System



Elements of an Expert System

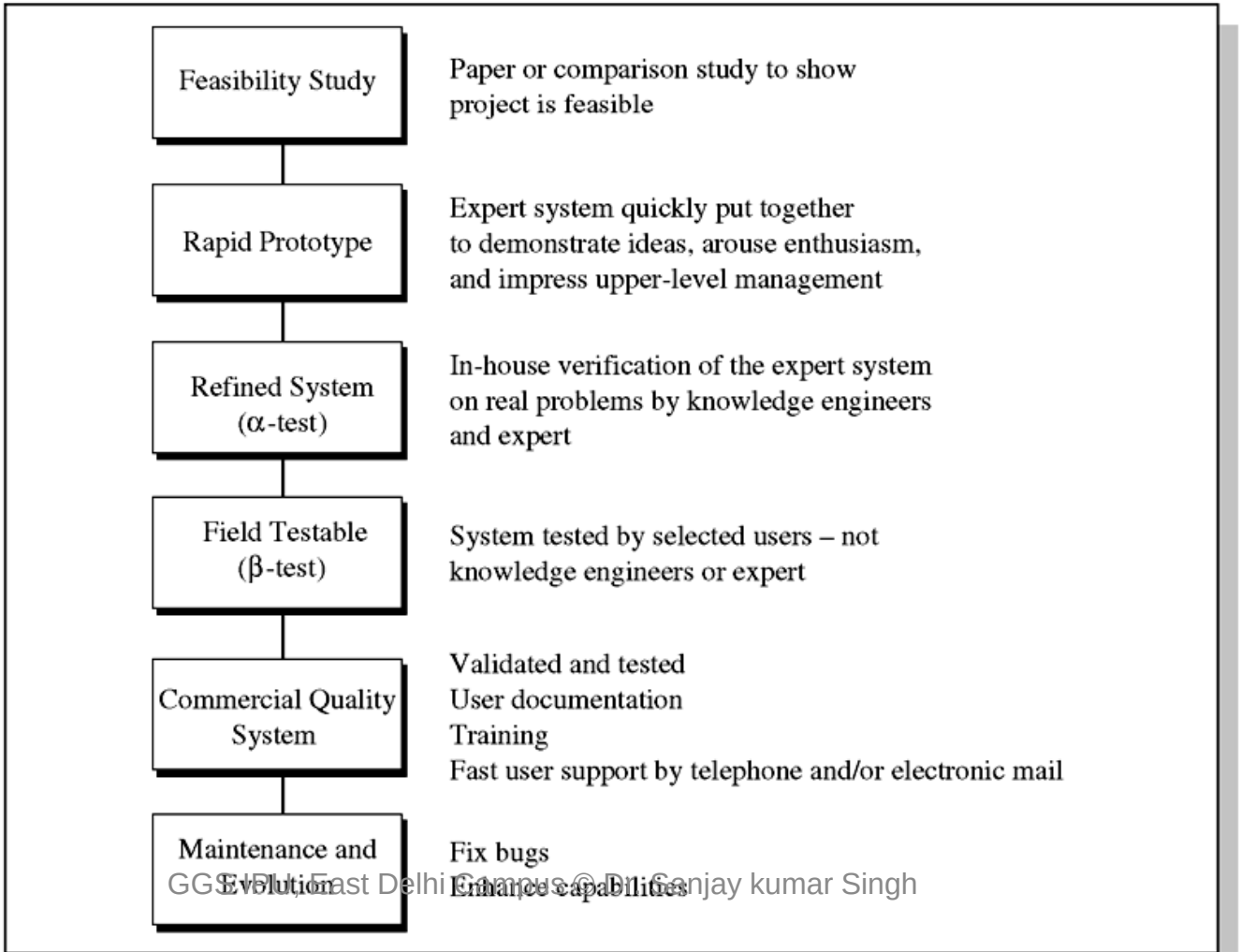
- User interface – mechanism by which user and system communicate.
- Exploration facility – explains reasoning of expert system to user.
- Working memory – global database of facts used by rules.
- Inference engine – makes inferences deciding which rules are satisfied and prioritizing.

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.

General Stages in the Development of an Expert System

Figure 6.2 General Stages in the Development of an Expert System



Expert System Life Cycle

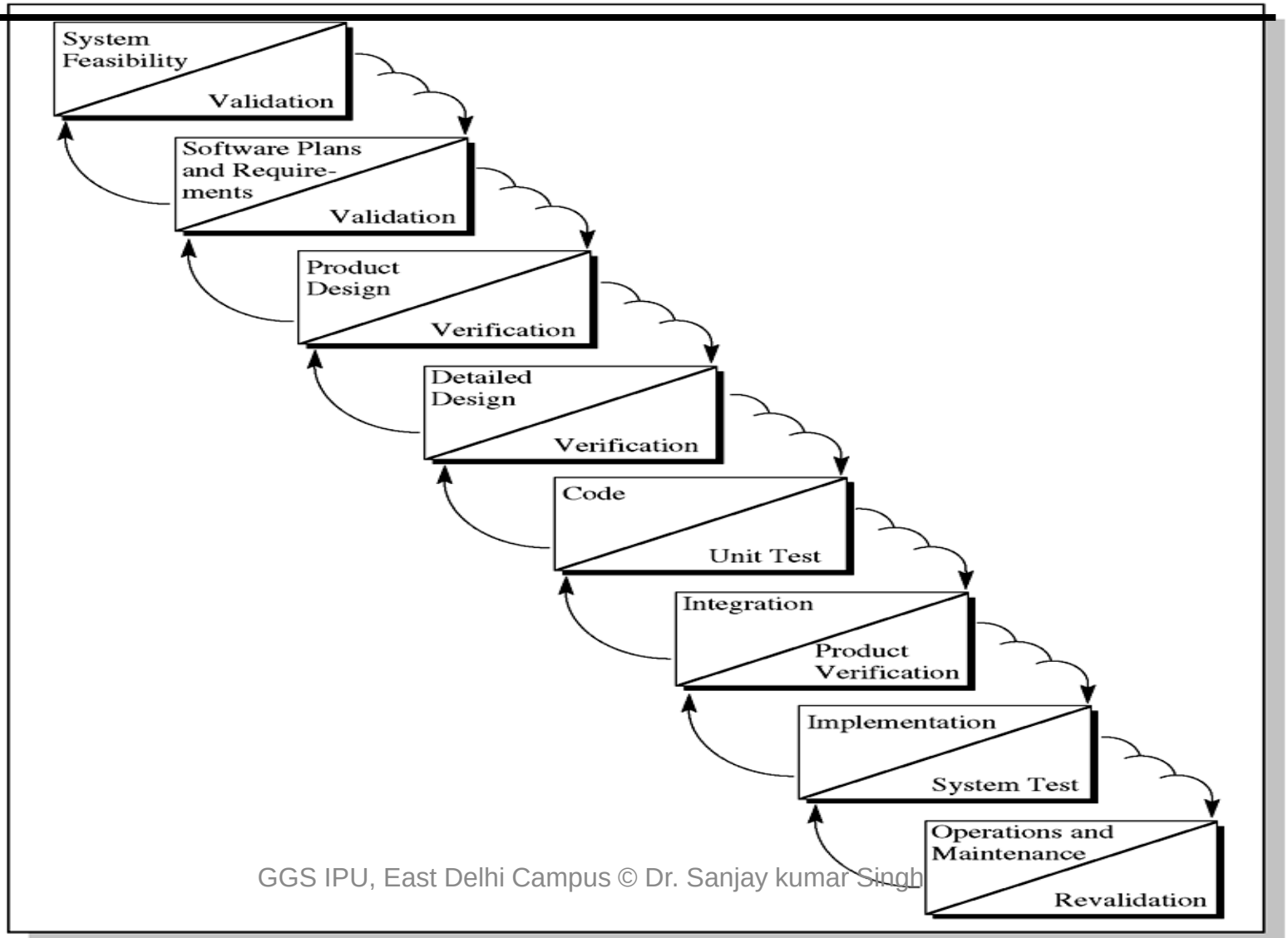
- Begins with the initial concept of the software and ends with its retirement from use.
- Expert systems require more maintenance because they are based on knowledge that is:
  - Heuristic
  - Experiential
- A number of life cycle models have been developed.

Waterfall Model

- Each stage ends with a verification and validation activity to minimize any problems in that stage.
- Arrows go back and forth only one stage at a time.
- It is assumed that all information necessary for a stage is known.

Waterfall Model of the Software Life Cycle

Figure 6.5 The Waterfall Model of the Software Life Cycle



Code-and-Fix Model

- Some code is written and then fixed when it does not work correctly.
- Usually the method of choice for new programming students in conventional and expert systems
- This eventually led to the do-it-twice concept where a prototype then a final system was built.

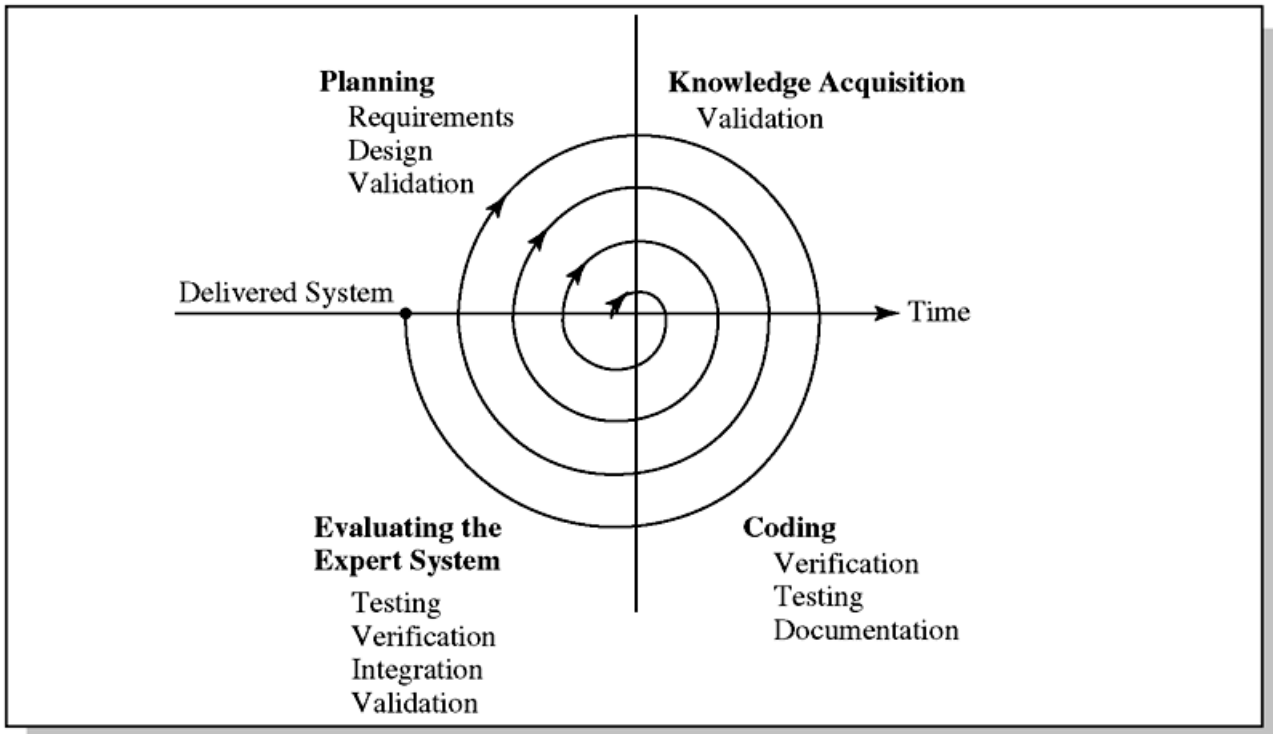
Incremental Model

- This is a refinement of the waterfall and top-down-approach.
- The idea is to develop software in increments of functional capability.
  - Major increment – assistant → colleague → expert
  - Minor increment – expertise within each level
  - Microincrement – add/refining individual rules

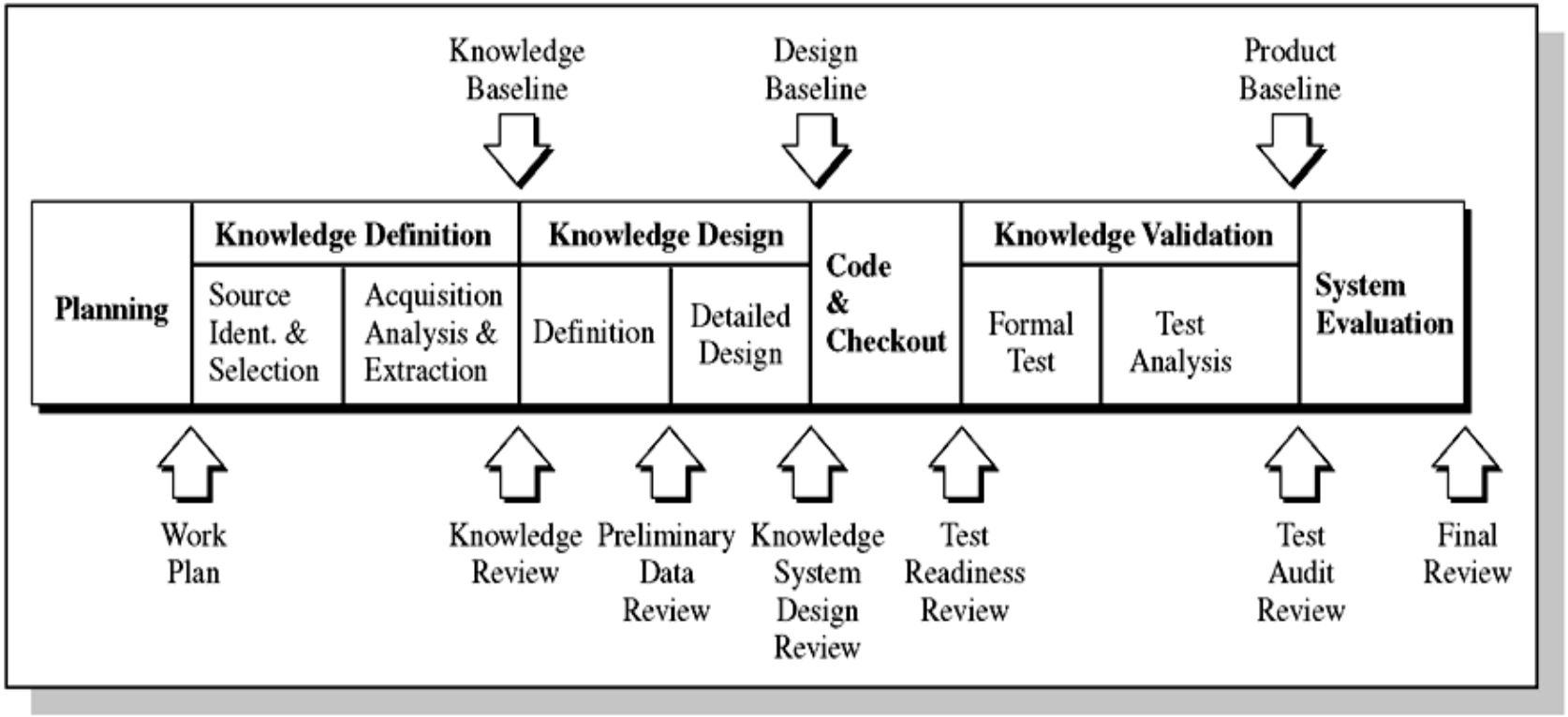
Spiral Model

Each circuit of the spiral adds some functional capability to the system.

Figure 6.6 A Spiral Model of Expert System Development



Linear Model of Expert System Development Life Cycle



Detailed Life Cycle Model
Linear Model

1. Planning Stage

The purpose of this stage is to produce a formal work plan for the expert system development – documents to guide and evaluate the development.

Table 6.2 Planning Stage Tasks

Task	Objective
Feasibility assessment	Determine if it is worthwhile to build the system and if so, whether expert systems technology should be used.
Resource management	Assess resources of people, time, money, software, and hardware required. Acquire and manage the required resources.
Task phasing	Specify the tasks and their order in the stages.
Schedules	Specify the starting and delivery dates of tasks in the stages.
Preliminary functional layout	Define what the system should accomplish by specifying the high-level functions of the system. This task specifies the purpose of the system.
High-level requirements	Describe in high-level terms how the functions of the system will be accomplished.

Linear Model

2. Knowledge Definition

The objective of this stage is to define the knowledge requirements of the expert system, which consists of two main tasks:

- Knowledge source identification and selection
- Knowledge acquisition, analysis, and extraction

Knowledge Source / Identification

Task	Objective
Source identification	Who and what are the knowledge sources, without regard to availability.
Source importance	Prioritized list of knowledge sources in order of importance to development.
Source availability	List of knowledge sources ranked in order of availability. The Web, books and other documents are generally much more available than human experts.
Source selection	Select the knowledge sources based on importance and availability.

Knowledge Acquisition, Analysis, and Extraction Tasks

Table 6.4 Knowledge Acquisition, Analysis, and Extraction Tasks

Task	Objective
Acquisition strategy	Specify how knowledge will be acquired by methods for interviewing experts, reading documents, rule induction, repertory grids, and so forth.
Knowledge element identification	Pick out the specific knowledge from sources that will be useful in this iteration of the life cycle.
Knowledge classification system	Classify and organize the knowledge to aid in knowledge verification and understanding by developers. Use hierarchical groups whenever possible.
Detailed functional layout	Specify the functional capabilities of the system in detail. This level is at a more technical level while the preliminary functional layout was at a managerial level.
Preliminary control flow	Describe general phases that the expert system will execute. Phases correspond to logical groups of rules that are activated/deactivated in groups to control execution flow.
Preliminary user's manual	Describes system from user's viewpoint. An often ignored, but essential part of the system. It is absolutely important to involve users as soon as possible for feedback. If they don't use the system, it's worthless.
Requirements specifications	Define exactly what the system is supposed to do. The expert system will be validated using these requirements.
Knowledge baseline	Baseline knowledge for the system. Any changes must now be done by a formal change request. The high-level knowledge is now adequate for the next stage of knowledge design.

Linear Model

3. Knowledge Design

The objective is to produce the detailed design for an expert system and involves:

- Knowledge definition
- Detailed design

Knowledge Definition Tasks

Task	Objective
Knowledge representation	Specify how knowledge will be represented, such as rules, frames, or logic. Dependent upon what the expert systems tool will support.
Detailed control structure	Specify three general control structures: (1) if the system is embedded in procedural code, how it will be called; (2) control of related groups of rules within an executing system; (3) metalevel control structures for rules.
Internal fact structure	Specify the internal structure of facts in a consistent manner to aid in understanding and good style.
Preliminary user interface	Specify a preliminary user interface. Get feedback from users about the interface.
Initial test plan	Specify how code will be tested. Define test data, test drivers, and how test results will be analyzed.

Detailed Design of Knowledge Tasks

Task	Objective
Design structure	Specify how knowledge is logically organized in the knowledge base and what is in the knowledge base.
Implementation strategy	Specify how the system is to be implemented.
Detailed user interface	Specify the detailed user interface after receiving user feedback from the preliminary user interface design.
Design specifications and report	Document the design.
Detailed test plan	Specify exactly how the code will be tested and verified.

Linear Model

4. Code and Checkout

This begins the actual code implementation

Task	Objective
Coding	Implement coding.
Tests	Test code using test data, test drivers, and test analysis procedures.
Source listings	Produce commented, documented source code.
User manual	Produce working user's manual so experts and users can provide feedback on system.
Installation/operations guide	Document installation/operation of system for users.
System description document	Document overall expert system functionality, limitations, and problems.

Linear Model

5. Knowledge Verification

The objective here is to determine the correctness, completeness, and consistency of the system.

- Formal tests
- Test Analysis

Table 6.8 Formal Test Tasks of Knowledge Verification Stage

Task	Objective
Test procedures	Implement formal test procedures.
Test reports	Document test results.

Test Analysis Tasks

Task	Objective
Results evaluations	Analyze test results.
Recommendations	Document recommendations and conclusions of tests.

Linear Model

6. System Evaluation

This stage is for summarizing what has been learned with recommendations for improvements and corrections.

Task	Objective
Results evaluation	Summarize the results of testing and verification.
Recommendations	Recommend any changes to the system.
Validation	Validate that the system is correct with respect to user needs and requirements.
Interim or final report	If the system is complete, then issue final report. If not, issue an interim report. Ask for more money.

Thank You !!!