ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS (SMEX1041)

UNIT I INTRODUCTION

Introduction to AI

Artificial Intelligence is the branch of computer science concerned with making computers behave like humans.

Systems that think like humans —"The exciting new effort to make computers think … machines with minds,in the full and literal sense."(Haugeland,1985)	Systems that think rationally —"The study of mental faculties through the use of computer models." (Charniak and McDermont,1985)
Systems that act like humans "The art of creating machines that perform functions that require intelligence when performed by people."(Kurzweil,1990)	Systems that act rationally "Computational intelligence is the study of the design of intelligent agents."(Poole et al.,1998)

The four approaches in more detail are as follows:

(a) Acting humanly: Turning test approach

The art of creating machines that perform functions requiring intelligence when performed by people, i.e., the study of, how to make computers do things which at the moment people do better.

Eg: 3 rooms contain a person, a computer, and an interrogator. The interrogator can communicate with the other 2 by teletype. The interrogator tried to determine which is the person and machine. The machine tries to fool the interrogator to believe that it is the human but the person also tries to convince the interrogator that it is the human. If the machines succeeds in fooling the interrogator then conclude the machine is intelligent.

Goal is to develop system that are human like.

(b)Thinking humanly: The cognitive modeling approach

An exciting new effort to make computers think, i.e., the machines with minds in the full and literal sense. Goal is not just to produce human like behavior but to produce a sequence of steps of the reasoning process similar to human in solving the same task.

(c) Thinking rationally: The "laws of thought approach"

The study of mental faculties through the use of computational models, i.e. the study of the computations that make it possible to perceive and act. Goal is to formalize the reasoning process as a system of logical rules and procedures for inference.

(d) Acting rationally: The rational agent approach

Ties to explain and emulate intelligent behavior in terms of computational processes i.e. concerned with the automation of intelligence.

OMNISCIENCE, LEARNING, AND AUTONOMY

An **omniscient agent** knows the *actual* outcome of its actions and can act accordingly; but omniscience is impossible in reality. Doing actions in order to modify future percepts-sometimes called **information gathering**-is an important part of rationality. Our definition requires a rational agent not only to gather information, but also to **learn** asmuch as possible from what it perceives.
To the extent that an agent relies on the prior knowledge of its designer rather than on its own percepts, we say that the agent lacks autonomy. A rational agent should be **autonomous**it should learn what it can to compensate for partial or incorrect prior knowledge
DIFFERENT TYPES OF AGENTS:

1. A human agent has eyes, ears, and other organs for sensors and hands, legs, mouth, and other body parts for actuators.
2. A robotic agent might have cameras and infrared range finders for sensors and various motors for actuators.
3. A software agent receives keystrokes, file contents and network packets as sensory inputs and acts as on the environment by displaying on the same screen, writing files and sending networks pockets.
4. Generic agent: a general structure of an agent who interface with the environment.

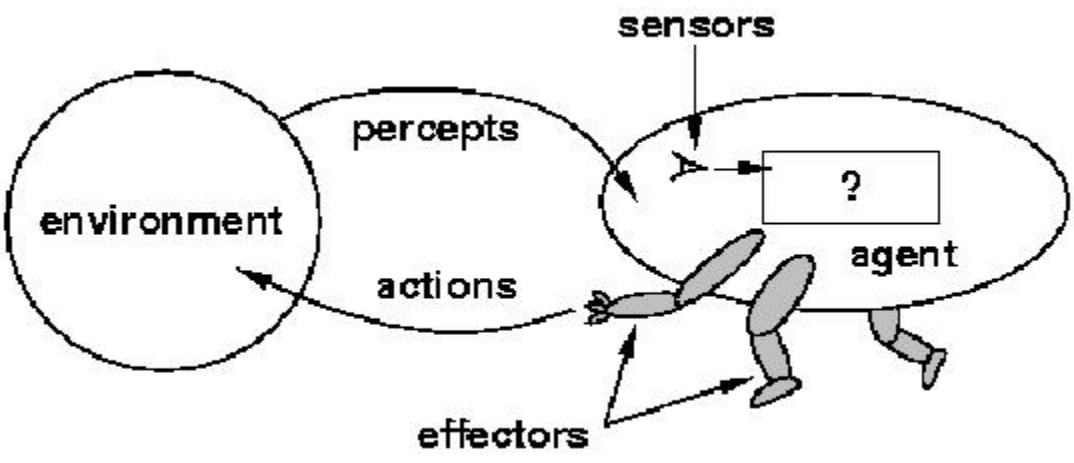


Fig 1: Agent interacting with environment through sensors and actuators.

PERCEPT SEQUENCE: Agents percept sequence is the complete history of everything the agent has ever perceived. An agent's behavior is discussed by the agent function that maps any given percept sequence to an action.

AGENT FUNCTION

1.The *agent function* is a mathematical function that maps a sequence of perceptions into action.

2. The function is implemented as the *agent program.*

3. The part of the agent taking an action is called an *actuator.*

4. Environment sensors agent function actuators environment

AGENT PROGRAM: The agent function for an artificial agent will be implemented by an agent program.

Eg:

1. Vacuum cleaner world function has just two locations; squares A and B
2. The vacuum agent perceives which square it is in and whether there is dirt in the square.
3. It can choose to move left, move right, suck up and do nothing,.
4. One very simple agent function is the following: if the current is dirty, then suck, otherwise move to the other square.

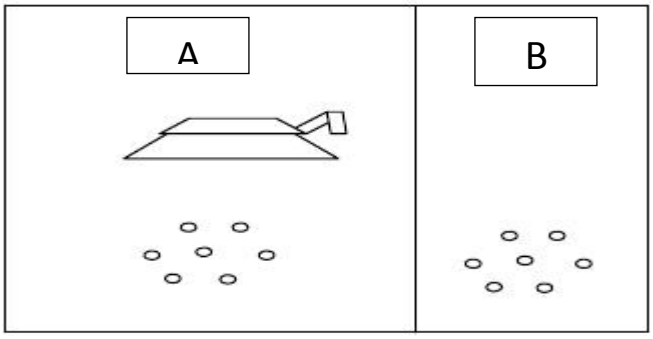


Fig 2: A vacuum cleaner world with just two locations

Partial tabulation of a simple agent function for the vacuum cleaner world

Percepts: location and status e.g., [A, Dirty]

Actions: Left, Right, Suck and No OP

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

Partial tabulation of a simple agent function for the Vacuum cleaner shown in fig 2

Function VACUUM – AGENT (location, status)

```
If status = Dirty then return suck
Else if location = A then return Right
Else if location = B then return Left
```

Concept of Rationality

1. A rational agent is one that does the right thing.
2. The right thing action is the one that will cause the agent to be most successful.

Performance measures

1. A performance measure embodies the criterion for success of an agent's behavior
2. When an agent is plunked down in an environment, it generates a sequence of actions according to the percepts it receives.

Rationality

Rationality at any given time depends on four things:

1. The performance measure that defines the criterion of success.
2. The agent's prior knowledge of the environment.
3. The actions that the agent can perform.
4. The agent's percept sequence to date.

DEFINITION OF A 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."

A rational agent should be autonomous

Definition of an omniscient agent:

An omniscient agent knows the actual outcome of its actions and can act accordingly; but omniscience is impossible in reality.

Autonomy

A rational agent should be autonomous-it should learn what it can to compensate for partial or incorrect prior knowledge.

Information Gathering

Doing actions in order to modify future percepts is called as information gathering.

Specifying the task environment

1. The rationality or any agent, we had to specify the performance measure, the environment, and the agent's actuators and sensors.
2. We call this as PEAS (Performance, Environment, Actuators, Sensors) or PAGE (Percept, Action, Goal, Environment) description.
3. In designing an agent, the first step must always be to specify the task environment.
4. Task environment is the "problems" to which rational agents are the "solutions".

Example: PEAS description of the task environment for agents

Agent Type	Performance measure	Environment	Actuators	Sensors
Automated Taxi Driver	Safe, Fast, Legal, Comfortable trip, maximize profits	Roads, traffic, pedestrian customers	Steering accelerator, brake, signal, horn	Cameras, speedometer, GPS, odometers
Medical Diagnosis system	Healthy Patent, minimize costs, lawsuits	Patient, hospital, staff	Screen display	keyboard
Part picking robot	Percentage of parts in correct bin	Conveyor belt with parts, bins	Jointed arm and hand	Camera, joint angle sensors
Interactive English tutor	Maximize student's score on test	Set of students	Screen display	keyboard
Robot soccer player	Amount of goals scored	Soccer match field	legs	Cameras, sonar or infrared
Satellite image analysis	Correct image categorization	Downlink from satellite	Display categorization screen	Color pixel arrays
Refinery controller	Maximum purity, safety, yield	Refinery operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Vacuum agent	Minimize energy, consumption	Two squares	Left, Right, Suck NoOP	Sensors to identify dirt

Properties of task environments

a) Fully Vs partially observable:

If an agent accesses the environment in all aspects using sensors to perform action then the environment is accessible by the agent.

If the sensors are not active or the environment changes a very second then the environment is inaccessible.

b) Deterministic Vs non- deterministic or stochastic

If the next state of the environment is completely determined by the current state and actions selected by the agents, then the environment is deterministic.

Accessible - deterministic

Inaccessible - non- deterministic

c) Static vs. dynamic

If the environment can change while an agent is deliberating, then we say the environment is dynamic for that agent; otherwise, it is static. Taxi driving is clearly dynamic. Crossword puzzles are static.

d) Episodic vs. Sequential

If an episodic environment, the agent's experience is divided into episodes.

Each episode consists of its own projects and actions and it does not depend on the previous episode.

e) Discrete vs. Continuous

If the number of distinct percepts and actions is limited, the environment is discrete, otherwise it is continuous.

Taxi driving is a continuous state and continuous-time problem. Chess game has a finite number of distinct states.

f) Single agent vs. Multi agent

The distinction between single-agent and multi agent environment may seem simple enough.

An agent solving a crossword puzzle by itself is clearly in a single-agent environment,

An agent playing chess is in a two-agent environment.

Examples of task environment and their characteristics

Task Environment	**Observable**	**Deterministic**	**Episodic**	**Static**	**Discrete**	**Agent**
Crossword puzzle	Fully	Deterministic	Sequential	Static	Discrete	Single
Chess with a clock	Fully	Stochastic	Sequential	Semi	Discrete	Multi
Poker	Partially	Stochastic	Sequential	Semi	Discrete	Multi
Backgammon	Fully	Stochastic	Sequential	Static	Discrete	Multi
Taxi driving	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi
Medical diagnosis	Partially	Stochastic	Sequential	Dynamic	Continuous	Single
Image-analysis	Fully	Deterministic	Episodic	Semi	Continuous	Single

Part-picking robot	Partially	Stochastic	Episodic	Dynamic	Continuous	Single
Refinery controller	Partially	Stochastic	Sequential	Dynamic	Continuous	Single
Interactive English tutor	Partially	Stochastic	Sequential	Dynamic	Discrete	Multi

The simplest environment is – Fully observable, deterministic, episodic, static, discrete and single-agent.

Most real situations are – Partially observable, stochastic, sequential, dynamic, continuous and multi-agent.

FOUR TYPES OF AGENTS:

1. Simple reflex agent
2. Model based reflex agent
3. goal-based agent
4. utility- base agent

SIMPLE REFLEX AGENT

Definition:

SRA works only if the correct decision can be made on the basis of only the current percept that is only if the environment is fully observable.

Characteristics

- no plan, no goal
- do not know what they want to achieve
- do not know what they are doing

Condition-action rule

- If condition then action

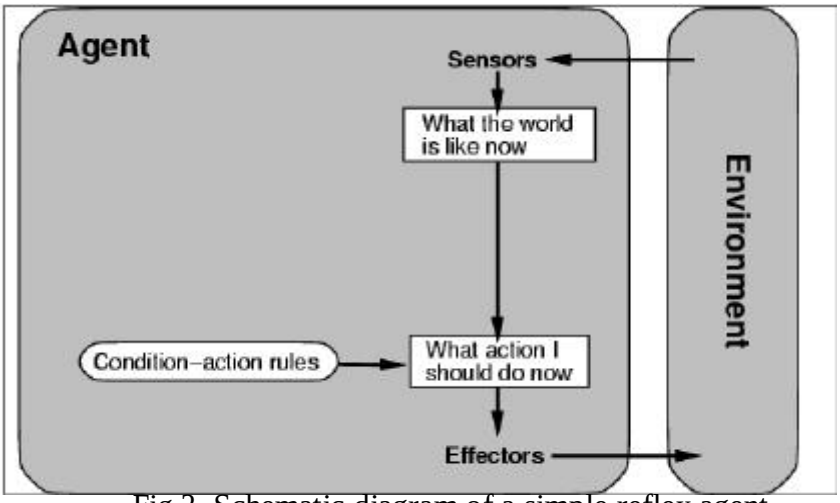


Fig 3 Schematic diagram of a simple reflex agent.

function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action
static: *rules*, a set of condition-action rules
state INTERPRET-INPUT(*percept*)
rule RULE-MATCH(*state*, *rule*)
action RULE-ACTION[*rule*]
return *action*

A simple reflex agent. It acts according to a rule whose condition matchesthe current state, as def ined by the percept.

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

The agent program for a simple reflex agent in the twostate vacuum environment. This program implements the agent function tabulated in the figure 2.

Model-based reflex agents

Definition:

An agent which combines the current percept with the old internal state to generate updated description of the current state.

If the world is not fully observable, the agent must remember observations about the parts of the environment it cannot currently observe.

This usually requires an internal representation of the world (or internal state).

Since this representation is a model of the world, we call this model-based agent.

Ex: Braking problem

characteristics

1.Reflex agent with internal state

2.Sensor does not provide the complete state of the world.

3. must keep its internal state

Updating the internal world requires two kinds of knowledge

1. How world evolves

2. How agent's action affect the world

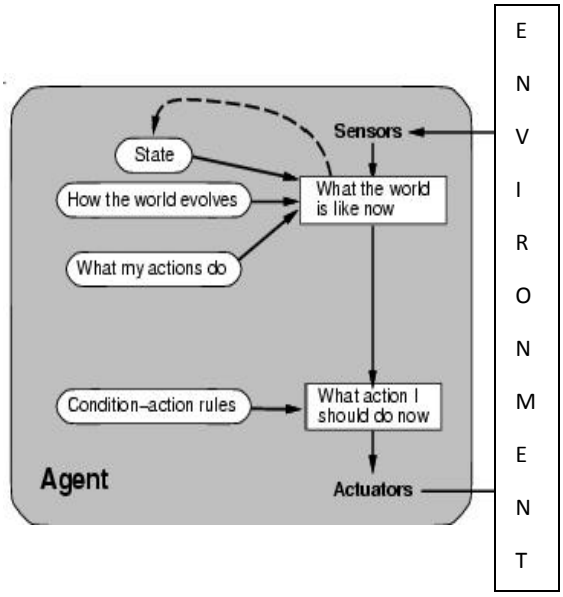


Fig 4 A model based reflex agent

function REFLEX-AGENT-WITH-STATE(*percept*) **returns** an action
static: *rules,* a set of condition-action rules
state, a description of the current world state
action, the most recent action.
state UPDATE-STATE(*state, action, percept*)
rule RULE-MATCH(*state, rule*)
action RULE-ACTION[*rule*]
return *action*

Model based reflex agent. It keeps track of the current state of the world using an internal model. It then chooses an action in the same way as the reflex agent.

Algorithm Explanation

***UPDATE-INPUT*:** This is responsible for creating the new internal stated description

GOAL-BASED AGENTS:

The agent has a *purpose* and the action to be taken depends on the current state

and on what it tries to accomplish (the goal).
In some cases the goal is easy to achieve. In others it involves *planning*, sifting through a*search* sp ace for possible solutions, developing a *strategy*.

Characteriscs

– Action depends on the goal. (consideration of future)
e.g. path finding

– Fundamentally different from the condition-action rule

  – Search and Planning
  – Solving "car-braking" problem?
  – Yes, possible … but not likely natural.

• Appears less efficient.

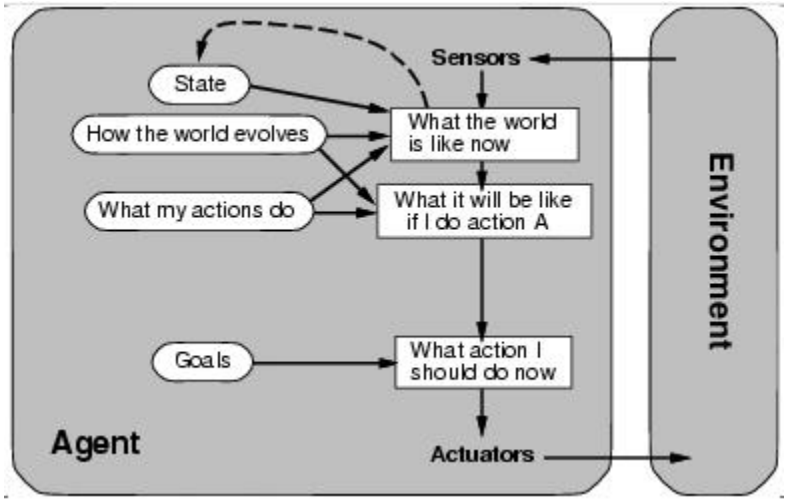


Fig 5 A goal based agent

UTILITY-BASED AGENTS

If one state is preferred over the other, then it has higher utility for the agent
Utility-Function (state) = real number (degree of happiness)
The agent is aware of a utility function that estimates how close the current state is to the agent's goal.

•**Characteristics**

- to generate high-quality behavior
- Map the internal states to real numbers. (e.g., game playing)
- Looking for higher utility value utility function

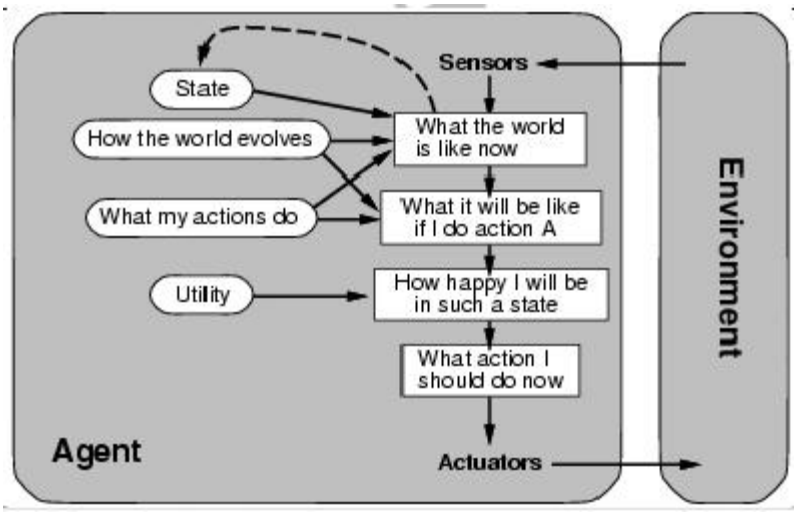


Fig 6 A model-based, utility-based agent

Early work in AI

 "ArtificialIntelligence (AI) is the part of computer science concerned with designing intelligent computer systems, that is, systems that exhibit characteristics we associate with intelligence in human behaviour – understanding language, learning, reasoning, solving problems, and so on."

 Scientific Goal To determine which ideas about knowledge representation, learning, rule systems, search, and so on, explain various sorts of real intelligence.

 Engineering Goal To solve real world problems using AI techniques such as knowledge representation, learning, rule systems, search, and so on.

 Traditionally, computer scientists and engineers have been more interested inthe engineering goal, while psychologists, philosophers and cognitive scientists have been more interested in the scientific goal.

 The Roots - Artificial Intelligence has identifiable roots in a number of older disciplines, particularly:

- Philosophy
- Logic/Mathematics
- Computation
- Psychology/Cognitive Science

- Biology/Neuroscience
- Evolution

 There is inevitably much overlap, e.g. between philosophy and logic,or between mathematics and computation. By looking at each of these in turn, we can gain a better understanding of their role in AI, and how these underlying disciplines have developed to play that role.

 Philosophy

 ~400 BC Socrates asks for an algorithm to distinguish piety from non-piety.

 ~350 BC Aristotle formulated different styles of deductive reasoning, which could mechanically generate conclusions from initial premises, e.g. Modus Ponens

IfA?BandA thenB

IfA implies B andA is true then B is true when it's raining you get wet and it's raining then you get wet

 1596 – 1650Rene Descartes idea of mind-body dualism – part of the mind is exempt from physical laws.

 1646 – 1716 Wilhelm Leibnitz was one of the first to take the materialist position which holds that the mind operates by ordinary physical processes – this has the implication that mental processes can potentially be carried out by machines.

 Logic/Mathematics

 EarlStanhope's Logic Demonstrator was a machine that was able to solve syllogisms, numerical problems in a logical form, and elementary questions of probability.

 1815 – 1864George Boole introduced his formal language for making logical inference in 1847 – Boolean algebra.

 1848 – 1925 Gottlob Frege produced a logic that is essentially the first-order logic that today forms the most basic knowledge representation system.

 1906 – 1978Kurt Gödel showed in 1931 that there are limits to what logic can do. His Incompleteness Theorem showed that in any formal logic powerful enough to describe the properties of natural numbers, there are true statements whose truth cannot be established by any algorithm

 1995 Roger Penrose tries to prove the human mind has non-computable capabilities.

 Computation

 1869William Jevon's Logic Machine could handle Boolean Algebra and Venn Diagrams, and was able to solve logical problems faster than human beings.

 1912 – 1954Alan Turing tried to characterise exactly which functions are capable of being computed. Unfortunately it is difficult to give the notion of computation a formal definition. However, the Church-Turing thesis, which states that a Turing machine is capable of computing any computable function, is generally accepted as providing a sufficient definition. Turing also showed that there were some functions which no Turing machine can compute (e.g. HaltingProblem).

 1903 – 1957John von Neumann proposed the von Neuman architecture which allows a description of computation that is independent of the particular realisation of the computer.

 1960sTwo important concepts emerged: Intractability (when solution timegrows atleast ex ponentially) and Reduction (to 'easier' problems).

Psychology / Cognitive Science

 Modern Psychology / Cognitive Psychology / Cognitive Science is the science which studies how the mind operates, how we behave, and how our brains process information.

 Language is an important part of human intelligence. Much of the early work on knowledge representation was tied to language and informed by research into linguistics.

 It is natural for us to try to use our understanding of how human (and other animal) brains lead to intelligent behavior in our quest to build artificial intelligent systems. Conversely, it makes sense to explore the properties of artificial systems(computer mo dels/simulations) to test our hypotheses concerning human systems.

 Many sub-fields of AI are simultaneously building models of how the human system operates, and artificial systems for solving real world problems, and are allowing useful ideas to transfer between them.

Biology / Neuroscience

 Our brains (which give rise to our intelligence) are made up of tens of billions of neurons, each connected to hundreds or thousands of other neurons.

 Each neuron is a simple processing device (e.g. just firing or not firing depending on the total amount of activity feeding into it).However, large networks of neurons are extremely powerful computational devices that can learn how best to operate.

 The field of Connectionism or Neural Networks attempts to build artificial systems based on simplified networks of simplified artificial neurons.

 The aim is to build powerful AI systems, as well as models of various human abilities.

 Neural networks work at a sub-symbolic level, whereas much of conscious human reasoning appears to operate at a symbolic level.

 Artificial neural networks perform well at many simple tasks, and provide good models of many human abilities. However, there are many tasks that they are not so good at, and other approaches seem more promising in those areas.

 Evolution

 One advantage humans have over current machines/computers is that they have a long evolutionary history.

 Charles Darwin (1809 – 1882) is famous for his work on evolution by natural selection. The idea is that fitter individuals will naturally tend to live longer and produce more children, and hence after many generations a population will automatically emerge with good innate properties.

 This has resulted in brains that have much structure, or even knowledge, built in at birth.

 This gives them at the advantage over simple artificial neural network systems that have to learn everything.

 Computers are finally becoming powerful enough that we can simulate evolution and evolve good AI systems.

 We can now even evolve systems (e.g. neural networks) so that they are good at learning.

 A related field called genetic programming has had some success in evolving

programs, rather than programming them by hand.

ASPECTS OF INTELLIGENCE

For example, you are hiring a developer. If you look at the previous abilities, here is what they mean for that role:

1. **Ability to grasp information from data**: Understand specifications and other written documents.
2. **Ability to remember information**: Remember programming standards and libraries, and also important information conveyed through conversations.
3. **Ability to juggle multiple things:** Work in multiple modules at the same time.
4. **Ability to concentrate on one thing**: Spend vast amounts of uninterrupted time building programs, thus increasing quality.
5. **Ability to apply solutions to problems**: Know when to use which best practices and design patterns.
6. **Ability to devise new solutions**: Create new programming modules that solve common problems faced by other developers.
7. **Ability to imagine**: Design better. Debug better.

1. **Ability to grasp information from data**: Two people may receive the same information, but one of them is able to comprehend it better and derive meaning from it. This is not necessarily a function of past knowledge, but an ability to recognize patterns in the data and derive conclusions. A person with this ability is best suited to work in analytical situations, being able to process huge amounts of information and make sense from them.
2. **Ability to remember information**: This is perhaps a misunderstood aspect of intelligence, and usually negatively associated with exam cramming. However, you can notice differences there too. Some people are able to cram more information than others. Some are able to remember relevant information from long ago. Memory is an essential part of the brain function. A person who is able to operate with large volumes of relevant information easily accessible while working can be highly efficient. Such a person is well suited for technical work. For example, a software developer who is very familiar with the language API's.
3. **Ability to juggle multiple things:** A good soccer player can run down the field with the ball, and also simultaneously remember exactly when and where to pass the ball, because while running, he can visualize where the other players are. Or think of how an aircraft controller works. Many people break down when confronted with multiple things at the same time. The person with the ability to multi-task revels in such situations.
4. **Ability to concentrate on one thing**: This seems the opposite of the previous point. But in some cases, the ability to juggle also requires the ability to tune out certain things. The soccer player in the previous example tunes out the thousands of cheering fans, his personal life and whatever happened 5 minutes ago as he runs. The aircraft controller shuts out all the other

distractions in the room. By only focusing on the essential, the entire brainpower is devoted to the main task at hand.

5. **Ability to apply solutions to problems**: Most people associate intelligence with problem-solving, but it is not the solution per-se that displays intelligence, but it is the process. Intelligence is the ability to match and apply strategies to solving problems. All strategies have to be learnt, but some people do better than others at understanding when to use them. For example, most accountants are familiar with the various strategies to minimize taxes, but some are just better at actually doing it.

6. **Ability to devise new solutions**: When faced with a unique problem, a person with this ability can come up with new ways of solving problems. For example, Leibniz and Newton (independently) invented calculus to solve their mathematical (and physics) challenges. The key difference with the previous point is that a person without this skill will get stuck when faced with new challenges, even though they can use the strategies they know to devise a new solution.

7. **Ability to imagine**: This goes beyond logical ability to devise solutions. People with this skill think unconventionally. Their ideas do not come out of some combination of putting existing ideas together or improving them. People working in creative fields are good examples of this. Another example is the discovery of the ring structure of benzene. Without such people, innovation would always be incremental and human progress would not be where it is. In many cases, companies measure only a few of these aspects. Some of these are difficult to measure in a typical interview. For example, asking people to solve puzzles in an interview may only be measuring their ability to remember information, if they already know the answer. But I still think it is worthwhile to think about ways to understand where a person stands with regard to these traits.

THE HISTORY OF ARTIFICIAL INTELLIGENCE

The gestation of artificial intelligence (1943-1955)

There were a number of early examples of work that can be characterized as AI, but itwas Alan Turing who first articulated a complete vision of A1 in his 1950 article "Computing Machinery a nd Intelligence." Therein, he introduced the Turing test, machine learning,genetic algorithms, an d reinforcement learning.

The birth of artificial intelligence (1956)

McCarthy convinced Minsky, Claude Shannon, and Nathaniel Rochester to help himbring togeth er U.S. researchers interested in automata theory, neural nets, and the study of,ntelligence. They organized a two,month workshop at Dartmouth in the summer of 1956.Perhaps the longestlasting thing to come out of the workshop was an agreement to adopt McCarthy's new name for the field: **artificial intelligence.**

Early enthusiasm, great expectations (1952-1969)

The early years of A1 were full of successes-in a limited way.

General Problem Solver (**GPS**) was a computer program created in 1957 by Herbert Simon an,A llen Newell to build a universal problem solver machine. The order in which the program consid eredsubgoals and possible actions was similar to that in which humans approached the same prob lems. Thus,GPS was probably the first program to embody the "thinking humanly" approach.

At IBM, Nathaniel Rochester and his colleagues produced some of the first A1 programs. Herber t Gelernter (1959) constructed the Geometry Theorem Prover, which wasable to prove theorems t hat many students of mathematics would find quite tricky.Lisp was invented by John McCarthy i n 1958 while he was at the Massachusetts Institute of
Technology (MIT). In 1963, McCarthy started the AI lab at Stanford.

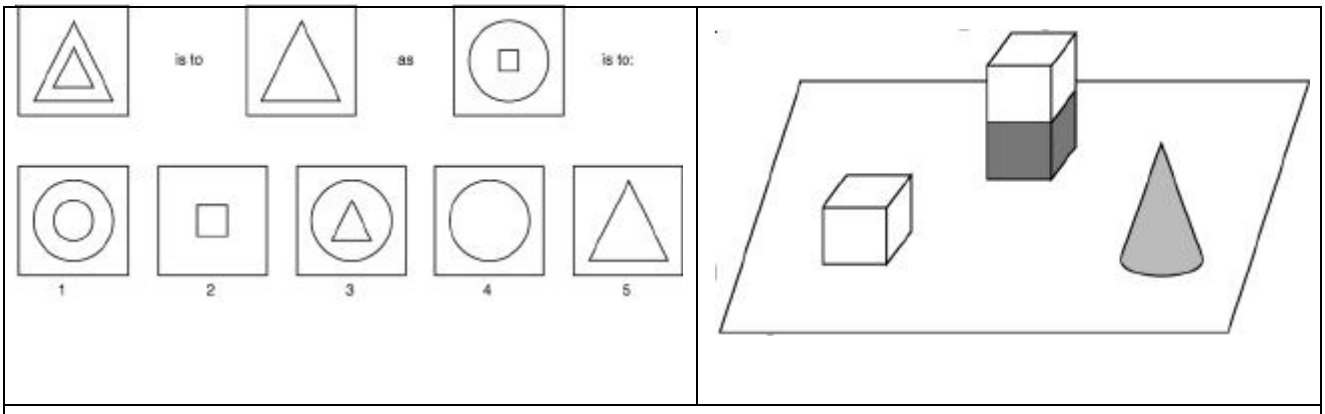


Fig 7 The Tom Evan's ANALOGY program could solve geometric analogy problems as shown.

Tom Evans's ANALOGY program (1968) solved geometric analogy problems that appear in IQ t ests.

A dose of reality (1966-
1973) From the beginning, AI researchers were not shy about making predictions of their comings uccesses. The following statement by Herbert Simon in 1957 is often quoted:

–It is not my aim to surprise or shock you-
but the simplest way I can summarize is to saythat there are now in the world machinesthat think , that learn and that create. Moreover,their ability to do these things is going toincrease rapidly u ntil-in a visible future-
therange of problems they can handle will be coextensive with the range to which the humanmin d has been applied.

Knowledge-based systems: The key to power? (1969-1979)

Dendral was an influential pioneer project in artificial intelligence (AI) of the 1960s, and thecom puter software **expert system** that it produced. Its primary aim was to help organic chemists inde ntifying unknown organic molecules, by analyzing their mass spectra and using knowledge ofche mistry. It was done at Stanford University by Edward Feigenbaum, Bruce Buchanan, JoshuaLede rberg, and Carl Djerassi.

A1 becomes an industry (1980-present)

In 1981, the Japanese announced the "Fifth Generation" project, a 10-year plan to build intelligent computers running Prolog. Overall, the A1 industry boomed from a few million dollar s in 1980 to billions of dollars in 1988.

The return of neural networks (1986-present)

Psychologists including David Rumelhart and Geoff Hinton continued the study of neural-net models of memory.

A1 becomes a science (1987-present)

In recent years, approaches based on **hidden Markov models** (HMMs) have come to dominate the area.Speech technology and the related field of handwritten character recognition are already

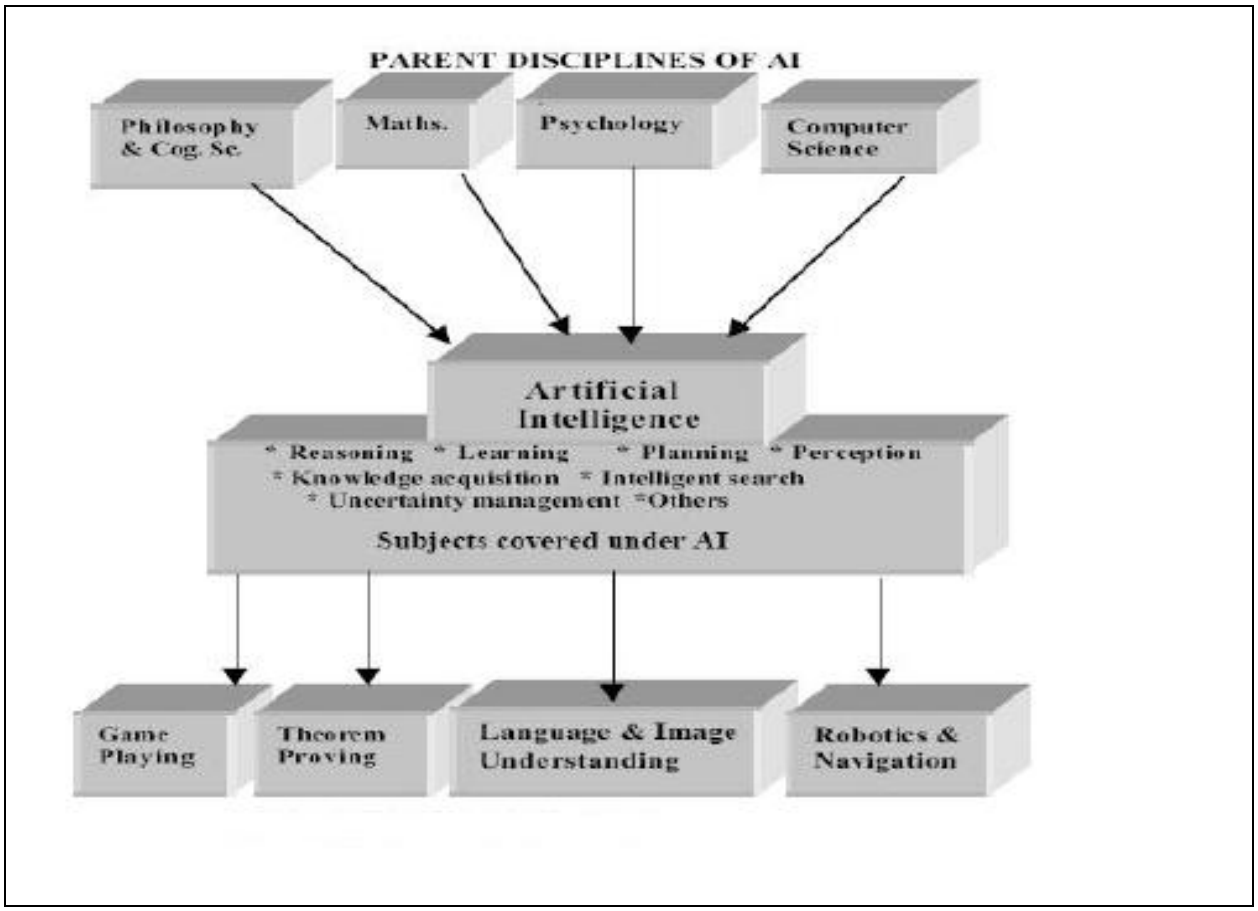
making thetransition to widespread industrial and consumer applications.The **Bayesian network** formalism was invented to allow efficient representation of, and rigorous reasoning with, uncertain knowledge.

The emergence of intelligent agents (1995-present)

One of the most important environments for intelligent agents is the Internet.

APPLICATIONS OF AI

1. Finance
2. Medical
3. Industries
4. Telephone maintenance
5. Telecom
6. Transport
7. Entertainment
8. Pattern Recognition
9. Robotics
10. Data Mining



AI LANGUAGES:

There are primarily two computer languages used in artificial intelligent work, LISP and PROLOG. LISP, which is short for List Processing, was created by John McCarthy of Stanford University. It looks Klutzy but it is based upon the lamba calculus and works quite well for computation associated with artificial intelligence. PROLOG has an elegent formulation but it does not have the range of application that LISP has. The Japanese when they formulated the fifth Generation project chose PROLOG over LISP as the programing language. This was perhaps one of the factors that contributes to the failure of the Fifth Generation project. Neverthless PROLOG is worth knowing for its power in solving questions in relationships.

PROLOG:

Prolog is a declarative language where programs are expressed in terms of relations, and execution occurs by running *queries* over these relations. Prolog is particularly useful for symbolic reasoning, database and language parsing applications. Prolog is widely used in AI today.

A PROLOG program consists of:

- Declaration of the facts of the relations involved.

- Declaration of rules concerning relations.
- Formulation of questions to be answered.

APPLICATION OF PROLOG:

Some applications of Prolog are:

- intelligent data base retrieval
- natural language understanding
- expert systems
- specification language
- machine learning
- robot planning
- automated reasoning
- problem solving

Lisp (list processing)

LISP, an acronym for list processing, is a programming language that was designed for easy manipulation of data strings. Developed in 1959 by John McCarthy, it is a commonly used language for artificial intelligence (AI) programming. It is one of the oldest programming languages still in relatively wide use.

- Lisp is a practical mathematical notation for computer programs based on lambda calculus. Linked lists are one of Lisp languages' major data structures, and Lisp sources code is itself made up of lists. As a result, Lisp programs can manipulate source code as a data structure, giving rise to the macro systems that allow programmers to create new syntax or even new domain-specific programming languages embedded in Lisp. There are many dialects of Lisp in use today, among them are Common Lisp, Scheme, and Clojure.
- In LISP, all computation is expressed as a function of at least one object. Objects can be other functions, data items (such as constants or variables), or data structures. LISP's ability to compute with symbolic expressions rather numbers makes it convenient for AI applications.

HISTORY OF LISP

LISP was invented by John McCarthy in 1958 while he was at the MIT.

McCarthy published its design in a paper in communications of the ACM in 1960.

LISP was first implemented by steve Russell on an IBM 704 computer.

CONNECTION TO AI:

LISP was closely connected to AI research communities, especially on PDP-10 systems.

LISP was used as the implementations of the programing languages Micro Planer which was used in the famous AI system SHRUDLU.

Over its Fifty-year history, lisp has spawned many variations on the core theme of an S-expression language.

INTRODUCTION TO LISP:

Lisp is second oldest high-level programing languages with a long history and a distinctive, fully parenthesized syntax.

Lisp is a Tool to solve some of the most difficult problems in the world of computing.

It is an example of elegant, minimalist language.

Lisp is one of the most popular programing languages that is used for Artificial Intelligence.

CHARACTERISTICS OF LISP:

1. **Language for artificial intelligence programming**
    a. Originally designed for symbolic computing
2. **Imperative language**
    a. Describe *how* to perform an algorithm
    b. Contrasts with *declarative* languages such as PROLOG
3. **Functional programming**
    a. Syntax and semantics are derived from the mathematical theory of recursive functions.
    b. Combined with a rich set of high-level tools for building symbolic data structures such as predicates, frames, networks, and objects
4. **Popularity in the AI community**
    a. Widely used as a language for implementing AI tools and models
    b. High-level functionality and rich development environment make it an ideal language for building and testing prototype systems.

LISP FEATURES:

Built in support for Lists.

Atomic storage management.

Dynamic Typing

Uniform syntax

Interactive environment.

Extensibility

Standard macros.

Special forms (loop, do, dotimes...)

Simple syntax. So, it's very easy to parse. Programing in Lisp is distinguished from other programing languages due to its unique syntax and development mode. The interchangeability of code and data also gives Lisp instantly recognizable syntax. All lisp program code is written as S-expressions or parenthesized lists.

APPLICATIONS OF LISP PROGRAMMING:

I. Common Lisp is used to develop research applications (often in Artificial Intelligence).
II. For rapid development of proto types
III. Lisp language is often used in interactive command line, which may be combines with an IDE.
IV. Common Lisp is used in many commercial applications, including the Yahoo! store web-commerce site.
V. Other visible applications people have developed using Lisp are:

- Emacs
- G2
- AutoCAD
- Igor Engraver

PROBLEM SOLVING AGENTS

- Problem solving agent is one kind of goal based agent, where the agent decides what to do by finding sequence of actions that lead to desirable states.
- The complexity arises here is the knowledge about the formulation process, (from current state to outcome action) of the agent.
- If the agent understood the definition of problem, it is relatively straight forward to construct a search process for finding solutions.
- It implies that problem solving agent should be an intelligent agent to maximize the performance measure.

The sequence of steps done by the intelligent agent to maximize the performance measure:

1. Goal formulation – based on current situation is the first step in problem solving. Actions that result to a failure case can be rejected without further consideration.
2. Problem formulation – is the process of deciding what actions and states to consider and follows goal formulation.

Function SIMPLE-PROBLEM-SOLVING-AGENT (p) returns an action

Input: p, a percept

Static: s, an action sequence, initially empty

state, some description of the current world state

g a goal initially null

problem, a problem formulation state <- UPDATE-STATE(state, p)

if s is empty then

g <-FORMULATE-GOAL (state)

problem<-FORMULATE-PROBLEM(state,g)

s <- SEARCH(problem)

action<-RECOMMENDATION(s, state)

s <- REMAINDER(s, state)

return action

3. Search – is the process of finding different possible sequence of actions that lead to state of known value, and choosing the best one from the states.
4. Solution – a search algorithm takes a problem as input and returns a solution in the form of action sequence.
5. Execution phase – if the solution exists, the action it recommends can be carried out.

A simple problem solving agent

Note:
RECOMMENDATION – first action in the sequence
REMAINDER – returns the rest
SEARCH – choosing the best one from the sequence of actions
FORMULATE-PROBLEM – sequence of actions and states that lead to goal state.
UPDATE-STATE – initial state is forced to next state to reach the goal state.

Well-defined problems and solutions

A problem is really a collection of information that the agent will use to decide what to do.

A problem can be defined formally by four compnents:

1. Initial state
2. Successor function
3. Goal test
4. Path cost

The initial state – the agent knows itself to be start in.

Successor function (S) – Given a particular state x, S(x) returns a set of states reachable from x by any single action.

I. The goal test – Single state description to check whether the goal state is reached or not.
II. If more than one goal state exists, then we can check whether any one of the goal state is reached or not.

A path cost function that assigns a numeric cost to each path.

I. Then problem- solving agent chooses a cost function that reflects its own performance measure.
II. The sum of the cost of the individual action along the path.

A solution to a problem is a path from the initial state to a goal state

Operator – The set of possible actions available to the agent.

State space (or) state set space – The set of all possible states reachable from the initial state by any sequence of actions.

Path (state space) – The sequence of action leading from one state to another.

The effectiveness of a search can be measured using three factors. They are:

1. Solution is identified or not?
2. Is it a good solution? If yes, then path cost to be minimum.
3. Search cost of the problem that is associated with time and memory required to find a solution.

Toy problems

i) Vacuum world problem

States: The agent is in one of two locations, each of which might or might not contain dirt. Thus there are $2 * 2^2 = 8$ possible world state.

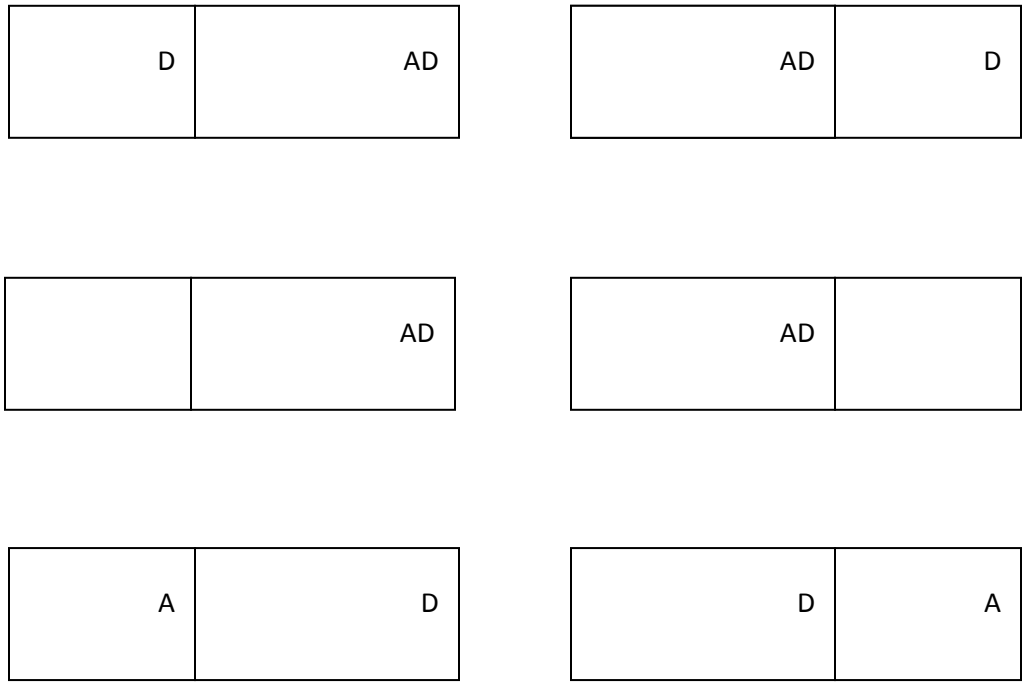
Initial state: Any state can be designated as the initial state.

Successor function: three actions (Left, Right, and Suck).

Goal test: This checks whether all the squares are clean.

Path cost: Each step costs 1, so the path cost is the number of steps in the path.

The 8 possible state of the simplified vacuum world problem.



ii) 8- puzzle problem:

1. The 8-puzzle problem consists of a 3 x 3 board with eight numbered titles and a blank-space.
2. A tile adjacent to the blank space can slide into the space.
3. The object is to reach a specified goal state

States: A state description specifies the location of each of the eight tiles and the blank in one of the nine squares.

Initial state: Any state can be designated as the initial state.

Successor function: This generates the legal states that result from trying the four actions (blank moves Left, Right, Up or Down).

Goal test : This checks whether the state matches the goal configuration (Other goal configurations are possible.)

Post cost: Each step costs 1, so the path cost is the number of steps in the path.

Initial State

2	8	3
1	6	4
7		5

Goal State

	1	2
3	4	5
6	7	8

Water Jug Problem

1. Given two jugs, a 4-gallon one and a 3-gallon one.
2. Neither have any ameasuring markers on it.
3. There is a pump that can be used to fill the jugs with water.
4. How can you get exactly 2 gallons of water into the 4-gallon jug?

States: The state space for this problem can be described as the set of ordered pairs of integers(x,y) such that x=0,1,2,3, or 4 and y=0,1,2, or 3:x represents the number of gallons of water in the 4-gallon jug, and y represents the quality of the water in the 3-gallon jug.

Initial state: The starting state is (0,0) where 3 – gallon jug and 4-gallon jug are empty

Successor function: This generates the legal productions that result from trying the 12 actions (empty the jug, Pour some water from jug, Fill the jug).

1. (x,y) if x<4	->(4,y)

2. (x,y) if y<3	->(x,3)
3. (x,y) if x>0	->(x-d, y)
4. (x,y) if y>0	->(x, y-d)
5. (x,y) if x>0	->(0, y)
6. (x,y) if y>0	->(x, 0)
7. (x,y) if x+y>=4 and y>0	->(4, y-(4-x)
8. (x,y) if x+y>=3 and x>0	->(x-(3-y), 3)
9. (x,y) if x+y<=4 and y>0	->(x+y, 0)
10. (x,y) if x+y<=3 and x>0	->(0, x+y)
11. (0,2)	->(2, 0)
12. (2,y)	->(0, y)

Goal test: The goal state is (2,n) for any value of n (the problem does not specify how many gallons need to be in the 3-gallon jug)
Path cost: Each step costs 1, so the path cost is the number of steps in the path.

Gallons in the 4-Gallon Jug	Gallons in the 3-Gallon Jug	Rule Applied
0	0	2
0	3	9
3	0	2
3	3	7
4	2	5 or 12
0	2	9 or 11
2	0	