Introduction to Artificial Intelligence

Text Book and reference

Artificial Intelligence: A Modern Approach

Stuart J. Russell and Peter Norvig

Introduction

- **Intelligence:**

  The ability to acquire and apply knowledge and skills.

- **Knowledge:**

  The facts, information, and skills acquired through experience or education

  Awareness or familiarity gained by experience of a fact or situation.

Introduction

- **Wisdom:**

  The quality of having experience, knowledge, and good judgment; the quality of being wise.

  The body of knowledge and experience that develops within a specified society or period.

  The fact of being based on sensible or wise thinking.

What is AI?

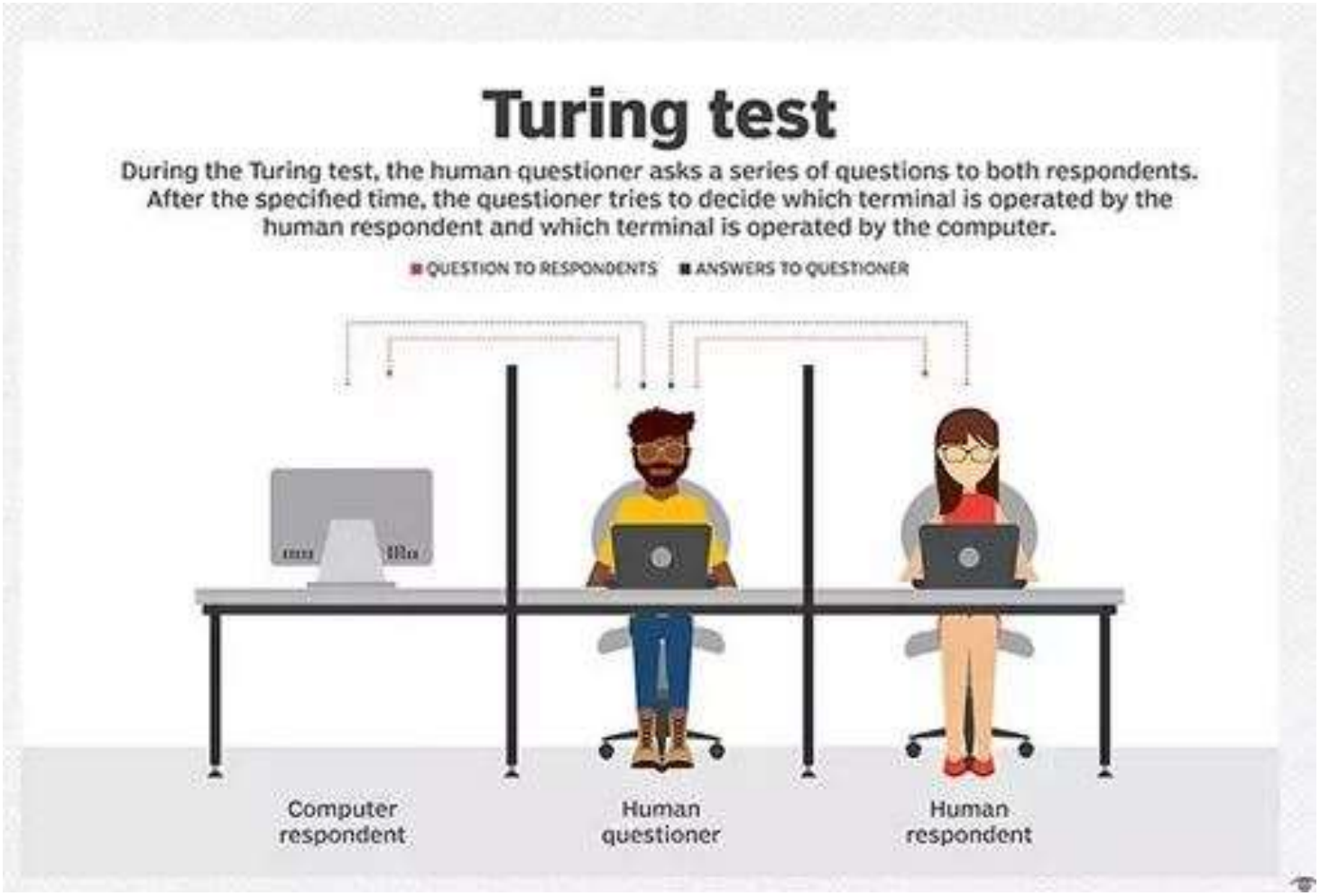
- Build intelligent artifacts vs. understanding human behavior.
- Does it matter how I built it as long as it does the job well?
- Should the system behave like a human or behave *intelligently*?

Definitions of AI

if our system can be more rational than humans in some cases, why not?

Systems that think like humans	Systems that think rationally
Decision making system, Problem solving , learning (Bellman,1978)	Study of the computations that make it possible to perceive, reason and act (Winston, 1992)
Systems that act like humans	**Systems that act rationally**
Study of how to make computers do things at the moment people are better (Rich and Knight,1991)	AI is concerned with intelligent behaviour in artifacts. (Nilson,1998)

Acting humanly: Turing Test



Acting humanly: Turing Test

- Natural language processing
- Knowledge Representation
- Automated reasoning.
- Machine Learning
- Computer vision, speech recognition, finding data on the web, robotics, and much more.

Thinking humanly: cognitive modeling

- 1960s "cognitive revolution": information-processing psychology
- Requires scientific theories of internal activities of the brain
- -- How to validate? Requires

  1) Predicting and testing behavior of human subjects (top-down)

  or 2) Direct identification from neurological data (bottom-up)

  Both approaches (roughly, Cognitive Science and Cognitive Neuroscience) are now distinct from AI

Thinking rationally: "laws of thought"

- Aristotle: what are correct arguments/thought processes?
- Several Greek schools developed various forms of *logic*: *notation* and *rules of derivation* for thoughts; may or may not have proceeded to the idea of mechanization
- Direct line through mathematics and philosophy to modern AI
- Problems:
  1. Not all intelligent behavior is mediated by logical deliberation
  2. What is the purpose of thinking? What thoughts should I have?

Acting rationally: rational agent

- Rational behavior: doing the right thing.
- An agent is an entity that perceives and acts.
- Abstractly, an agent is a function from percept histories to actions:
-

$$[f: P^* \rightarrow A]$$

- For any given class of environments and tasks, we seek the agent (or class of agents) with the best performance.
  → design best program for given machine resources
- The right thing: that which is expected to maximize goal achievement, given the available information
- Doesn't necessarily involve thinking – e.g., blinking reflex – but thinking should be in the service of rational action

Acting rationally: rational agent

- Rational behavior: Doing that was is expected to maximize

  one's "utility function" in this world.

- An agent is an entity that perceives and acts.
- A rational agent acts rationally.

FOUNDATIONS of AI

FOUNDATIONS of AI

.

- **Philosophy:** Logic, methods of reasoning, mind as physical system, foundations of learning, language, rationality.
- **Mathematics:** Formal representation and proof, algorithms, computation,(un)decidability, (in)tractability, probability.
- **Economics:** utility, decision theory, rational economic agents.
- **Neuroscience:** neurons as information processing units. physical substrate for mental activity

FOUNDATIONS of AI

.

- **Psychology:** how do people behave, perceive, process information, represent knowledge. phenomena of perception and motor control, experimental techniques.
- **Computer engineering** :building fast computers
- **Control theory Cybernetics :** design systems that maximize an function over time
- **Linguistics:** knowledge representation, grammar

Philosophy (428 B.C.-present)

- Can formal rules be used to draw valid conclusions?
- How does the mind arise from a physical brain?
- Where does knowledge come from?
- How does knowledge lead to action?

Can formal rules be used to draw valid conclusions?

- **Aristotle**(384-322 B.C.)**:** informal system of syllogisms for proper reasoning.

  Try to formulate laws of rational part of the mind. Believed in another part, intuitive reason.
- **Ramon Lull (d. 1315)** had the idea that useful reasoning could actually be carried out by a mechanical artifact
- **Thomas Hobbes (1588-1679)** proposed that reasoning was like numerical computation, that "we add and subtract in our silent thoughts." The automation of computation itself was already well under *way*

- **Leonardo (La Vinci (l452-1519)** designed but did not build a mechanical calculator

  The first known calculating machine was constructed around 1623 by the German scientist Wilhelm Schickard (1592-1635).
- **Blaise Pascal (1623-1662),** whereas the Pascaline could only add and sub-tract. Some speculated that machines might not just do calculations but actually be able to think and act on their own is more famous.

- Pascal wrote that "the arithmetical machine produces effects which appear nearer to thought than all the actions of animals."
- **Gottfried Wilhelm Leibniz** (1646-1716) built a mechanical device intended to carry out operations on concepts rather than numbers, but its scope was rather limited.

How does knowledge lead to action?

- Rene Descartes (1596-1650) gave the first clear discussion of the distinction between mind and matter and of the problems that arise.
- Descartes was a strong advocate of the power of reasoning in understanding the world, a philosophy now called rationalism.

Philosophy: Dualism vs. materialism

- Rene Descartes (1596-1650): *dualism* (part of mind that is outside of nature)
- An alternative to dualism is materialism, which holds that the brain's operation according to the laws of physics constitutes the mind.
- Animals did not possess the dual quality.
- *Materialism*. Wilhelm Leibniz (1646-1716) built a mechanical device to carry out mental operations; could not produce interesting results.

Philosophy: Source of knowledge

- *Empiricism* (Francis Bacon 1561-1626)
  - John Locke (1632-1704): “Nothing is in the understanding which was not first in the senses”
  - David Hume (1711-1776): Principle of induction: General rules from repeated associations between their elements
    - Bertrand Russell (1872-1970): *Logical positivism*: All knowledge can be characterized by logical theories connected, ultimately, to observation sentences that correspond to sensory inputs.

- the famous Vienna Circle, led by Rudolf Carnap (1891-1970), developed the doctrine of logical positivism. This doctrine holds that all knowledge can be characterized by logical theories connected to observation sentences that correspond to sensory inputs.
- Camap's book The Logical Structure of the World (1928) defined an explicit computational procedure for extracting knowledge from elementary experiences. It was probably the first theory of mind as a computational process.

Mathematics

- What are the formal rules to draw valid conclusions?
- What can be computed?
- How do we reason with uncertain information?

Mathematics

- Logic
  - George Boole (1815-1864): formal language for making logical inference
  - Gottlob Frege (1848-1925): First-order logic (FOL)
  - Computability
    - David Hilbert (1862-1943): He presented a list of 23 problems that he correctly predicted.
    - Kurt Godel (1906-1978): There exists an effective procedure to prove any true statement in the FOL but that FOL could not capture the principle of mathematical induction needed to characterize the natural number.
    - Alan Turing (1912-1954): which functions are computable?
      - Church-Turing thesis: any computable function is computable via a Turing machine
      - No machine can tell in general whether a given program will return an answer on a given input, or run forever

ALGORITHM: The first nontrivial algorithm is thought to be Euclid's algorithm for computing greatest common divisors.

Boole and others discussed algorithms for logical deduction and efforts were under way to formalize general mathematical reasoning as logical deduction.

- In 1931, Godel showed that limits on deduc tion, INCOMPLETENESS do exist.
- Alan Turing (1912-1954) try to characterize exactly which functions *are* COMPUTABLE
- COMPUTABLE—**capable of being computed**

- Although decidability and computability are important to an understanding of computation.
- How can one recognize an intractable problem? The theory of NP-completeness, pioneered by Steven Cook (1971) and Richard Karp (1972).

- Cook and Karp showed the existence of large classes of canonical combinatorial search and reasoning problems that are NP-complete
- Besides logic and computation, the third great contribution of mathematics to AI is the theory of probability

Mathematics…

- Probability
  - Gerolamo Cardano (1501-1576): probability in gambling
  - Pierre Fermat (1601-1665), Blaise Pascal (1623-1662), James Bernoulli (1654-1705), Pierre Laplace (1749-1827): new methods
  - Bernoulli: subjective beliefs->updating
  - Thomas Bayes (1702-1761): updating rule

Economics

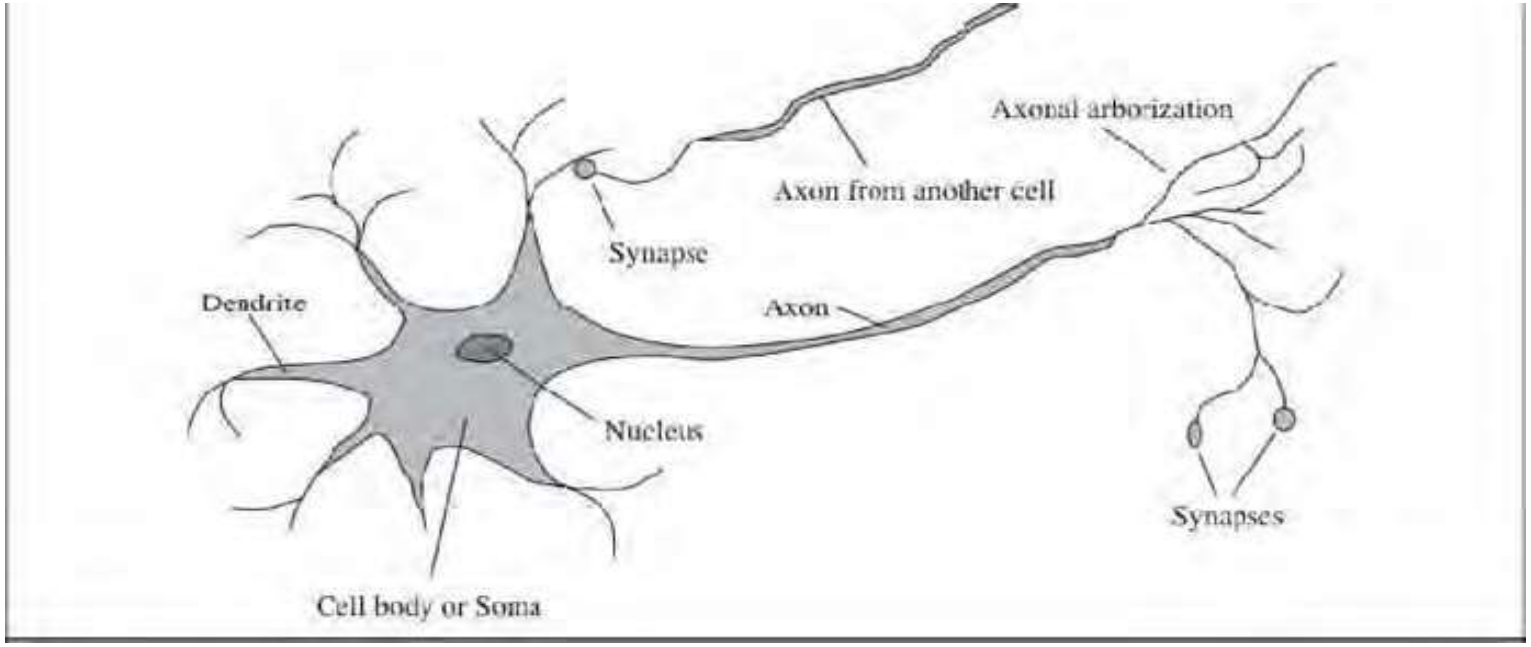
- How should we make decisions so as to maximize payoff?
- How should we do this when others may not go along?
- How should we do this when the payoff may be far in the future?

- Economics can be thought of as consisting of individual agents maximizing their own economic well-being.
- Most people think of economics as being about money, but economist will say that they are really studying how people make choices that lead to preferred outcomes.
- The mathematical treatment of "preferred outcomes" or **utility** was first formalized by Leon Walras (pronounced "Valrasse") (1834-1910) and was improved by Frank Ramsey (1931).
- Decision theory = probability theory + utility theory

- Game theory includes a rational agent that act in a random fashion.
- Economist make rational decisions when payoffs from actions are not immediate but instead result from several actions taken in sequence. This topic was pursued in the field of operation research.
- The work of Richard Gellman (l957) formalized a class of sequential decision problems called Marko decision processes.

Neuroscience

How do brains process information?


Axonal arborization
Axon from another cell
Synapse
Dendrite
Axon
Nucleus
Synapses
Cell body or Soma

Can we build hardware as complex as the brain?

- How complicated is our brain?
  - a neuron, or nerve cell, is the basic information processing unit
  - estimated to be on the order of 10^{11} neurons in a human brain
  - many more synapses (10^{14}) connecting these neurons
  - cycle time: 10^{-3} seconds (1 millisecond)

- How complex can we make computers?
  - 10^6 or more transistors per CPU
  - supercomputer: hundreds of CPUs, 10^9 bits of RAM
  - cycle times: order of 10^{-8} seconds

- Conclusion
  - **YES**: in the near future we can have computers with as many basic processing elements as our brain, but with
    - far fewer interconnections (wires or synapses) than the brain
    - much faster updates than the brain
  - **but** building hardware is very different from making a computer behave like a brain!

Psychology

- How do humans and animals think and act?

- In 1879, Wundt opened the first laboratory of experimental psychology at the university of Leipzig.
- Wundt insisted on carefully controlled experiments in which his workers would perform a perceptual task while introspecting on their thought processes.

- The behaviorism movement, led by John Watson(1878-1958), rejected any theory involving mental processes on the grounds that introspection could not provide reliable evidence.

Psychology

- Cognitive psychology
  - Brain posesses and processes information
  - Kenneth Craik 1943: knowledge-based agent:
    - Stimulus -> representation
    - Representation is manipulated to derive new representations
    - These are translated back into actions
  - Widely accepted now
  - Anderson 1980: “A cognitive theory should be like a computer program”

Computer engineering

- **How can we build an efficient computer**

Computer engineering

- *Abacus* (7000 years old)
- *Pascaline*: mechanical adder & substractor (Pascal; mid 1600's)
  - Leibniz added multiplication, 1694
- *Analytic Engine*: universal computation; never completed (ideas: addressable memory, stored programs, conditional jumps)
  - Charles Babbage (1792-1871), Ada Lovelace

Computer engineering…

[See Wired magazine late Fall 1999]

- *Heath Robinson*: digital electronic computer for cracking codes
  - Alan Turing 1940, England
- *Z-3*: first programmable computer
  - Konrad Zuse 1941, Germany
- *ABC*: first electronic computer
  - John Atanasoff 1940-42, US
- *ENIAC*: first general-purpose, electronic, digital computer
  - John Mauchy & John Eckert

ENIAC-Electronic Numerical Integrator And **Computer**

Control theory and cybernetics

How can artifacts operate under their own control?

- Ktesibios of Alexandria (c. 250 B.C.) built the first self-controlling machine. Before this only living things could modify their behavior in response to changes in the environment.
- In the late 1940s, Wiener, along with Warren McCulloch, Walter Pitts, and John von Neumann, explored the new mathematical and computational models of cognition.
- Wiener's book Cybernetics (1948) be-came a bestseller and awoke the public to the possibility of artificially intelligent machines.

- Modern control theory, especially the branch known as stochastic optimal control, has as its goal the design of systems that maximize an objective function over time.

- Why, they are AI and control theory two different fields, despite the close connections among their founders?

- Calculus and matrix algebra, the tools of control theory, lend themselves to systems that are describable by fixed sets of continuous variables, whereas AI was founded in part as a way to escape from the these perceived limitations.

- The tools of logical inference and computation allowed Al researchers to consider problems such as language, vision, and planning that fell completely outside the control theorist's purview.

Linguistics

How does language relate to thought?

How does language relate to thought

- Noam Chomsky pointed out that the behaviorist theory did not address the notion of creativity in language—it did not explain how a child could understand and make up sentences that he or she had never heard before. Chomsky's theory—based on syntactic models

- Modem linguistics and AI then, were "born" at about the same time, and grew up together, intersecting in a hybrid field called computational linguistics or natural language processing.

- Understanding language requires an understanding of the subject matter and context, not just an understanding of the structure of sentences.

History of AI

History of AI

- The gestation of artificial intelligence (1943-1955)
- The birth of artificial intelligence (1956)
- Early enthusiasm, great expectations (1952-1969)
- A dose of reality (1966-1973)

- Knowledge-based systems: The key to power? (1969-1979)
- AI becomes an industry (1980—present)
- The return of neural networks (1986—present)
- AI becomes a science (1987–present)
- The emergence of intelligent agents (1995—present)

The gestation of artificial intelligence (1943-1955)

- The first AI work was done by Warren McCulloch and Walter Pitts (1943)
- They drew on three socurces:
    1. knowledge of the basic physiology and function of neurons in the brain
    2. A formal analysis of propositional logic
    3. Turing theory of computation
- They proposed a model of artificial neurons in which each neuron is characterized as being “on” or “off”

- They showed that any computable function could be computed by network of neurons and all the logical connectives could be implemented by simple net structures.

- Donald Hebb (1949) demostrated a simple updating rule for modifying the connection strengths between neurons. It is know as Hebbian learning.

- Alan Turing first articulated a complete vision of AI in his 1950 article “Computing Machinery and Intelligence”

The birth of artificial intelligence (1956)

- Dartmouth workshop (1955) we can see why it was necessary for AI to become a separate field.

- First answer is that AI from the start embraced the idea of duplicating human faculties like creativity, self-improvement and language use.

- Second answer is methodology. AI is the only one these fields that is clearly a branch of computer science and AI is the only field to attempt to build machines that will function autonomously in complex, changing environment.

Early enthusiasm (1952-69)

- claim: computers can do X

- *General Problem Solver*, Newell & Simon
  - Intentionally solved puzzles in a similar way as humans do (order of subgoals, etc)

- *Geometry Theorem Prover*, Herbert Gelernter, 1959

- Arthur Samuel's learning checkers program 1952.

  Along the way, he disproved the idea that computers can do only what they are told to: his program quickly learned to play a better game than its creator.

- LISP, time sharing, Advice taker: John McCarthy 1958
- Blocks world: vision, learning, NLP, planning
- Adalines [Widrow & Hoff 1960]
- Perceptron convergence theorem [Rosenblatt 1962]

A dose of reality (1966-74)

- Herbert Simon(1957): It is not may aim to surprise, that there are machines that think, learn and create.
- Simon over confident on early AI systems.
- Early systems turned out to off, when tried out on more difficult problems.
  - Early programs contained little or no knowledge of their subject matter.
  - Intractability of the problems
  - Fundamental limitations on the basic structures being used to generate intelligent behavior.

Knowledge-based systems (1969-79)

- *DENDRAL(Buchanam et al., 1969)*: molecule structure identification
  - It was the first successful Knowledge Intensive system
- *MYCIN* (Feigenbaum, Buchanan, Dr. Edward) :medical diagnosis
  - 450 rules; knowledge from experts
  - Better than junior doctors
  - Certainty factors
- *PROSPECTOR*: For mineral exploration
- Domain knowledge in NLP
- Knowledge representation: logic, frames...

AI becomes an industry (1980-present)

- *R1(McDermott,1982)*

  first successful commercial expert system

  configured order for new computer systems at DEC;

  saved 40M$/year
- 1988: DEC had 40 expert systems
- 1981: Japanese announced 5th generation project, a 10 year plan to build intelligent computer running Prolog.

Return of ANNs (1986-present)

- John Hopfield (1982) used techniques from statistical mechanics to analyze the storage and optimization properties of networks.

- David Rumelhart and Geoff Hinton continued the study of neural net model of memory.

- Back-propagation learning algorithm first found in1969 by Bryson and Ho. It is applied to many learning problems caused great excitement.

AI becomes a science (1987–present)

- AI was founded the limitations of existing fields like control theory and statistics, but now it is embracing those fields.

- David McAllester(1988), stated as in the early period of AI it seemed that new forms of symbolic computation e.g. frames and semantic networks made much of classical theory obsolete. This led to a form of isolation in which AI became largely seperated from the rest o f computer science.

- In terms of methodology, AI has use scientific method. To be accepted, hypotheses must be subjected to empirical experiments and results must be analyzed statistically.
- E.g. speech recognition
- Hidden Markov Models: Two aspects are relevant
  - Mathematical theory
  - Training on large real speech data
- Neural Network: Improved methodology and theoretical framework.
- As a result data mining technology has spawned.

- Bayesian network: It allow efficient representation and reasoning uncertain knowledge.
- It dominates AI research on uncertain reasoning and expert systems.
- It allows learning from experience and combines the best classical and neural nets.
- Similar revolutions have occurred in robotics, computer vision and knowledge representation.

The emergence of intelligent agents (1995-present)

- The work of Allen Newell, John Laird. and Paul Rosenbloom on SOAR (Newell, 1990; Laird et al., 1987) is the best-known example of a complete agent architecture.
- One of the most important environments for intelligent agents is the Internet.
- Al systems have become so common in Web-based applications that the "-hot" suffix has entered everyday language.
- AI technologies underlie many internet tools, such as search engines, recommender systems and web site construction system.

Applications of AI

- Autonomous planning and scheduling

  Autonomous planning program to control the scheduling of operations for a spacecraft.

- Game playing

  IBM's Deep Blue become the first program to defeat the world champion Garry Kasparov in a chess match.

- Autonomous control

  The ALVINN computer vision system was trained to steer a car to keep it following a lane.

Applications of AI

- Diagnosis

  Medical diagnosis programs based on probabilistic analysis have been able to perform at the level of an expert physician in the area of medicine.

- Logistics planning

  In 1991, U.S. forces deployed a Dynamic Analysis and Replacement Tool, DART to do automated logistics planning and scheduling for transportation.

Applications of AI

- Robotics

  Many surgeons use robot assistant in microsurgery.

  HipNav is a system that uses computer vision techniques to create a three dimensional model of a patient's internal anatomy and then uses robotics control to guide the insertion of a hip replacement prosthesis.

- Language understanding and problem solving

  PROVERB is a computer program that solves crossword puzzles better than most human.

PEAS

- Example: Agent = taxi driver
  - Performance measure:
  - Environment:
  - Actuators:
  - Sensors:

PEAS

- Example: Agent = taxi driver
  - Performance measure: Safe, fast, legal, comfortable trip, maximize profits
  - Environment: Roads, other traffic, pedestrians, customers
  - Actuators: Steering wheel, accelerator, brake, signal, horn
  - Sensors: Cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard

PEAS

- Example: Agent = Medical diagnosis system

Performance measure:

Environment:

Actuators:

Sensors:

PEAS

- Example: Agent = Medical diagnosis system

Performance measure: Healthy patient, minimize costs, lawsuits

Environment: Patient, hospital, staff

Actuators: Screen display (questions, tests, diagnoses, treatments, referrals)

Sensors: Keyboard (entry of symptoms, findings, patient's answers)

PEAS

- Example: Agent = Part-picking robot
- Performance measure:
- Environment:
- Actuators:
- Sensors:

PEAS

- Example: Agent = Part-picking robot
- Performance measure: Percentage of parts in correct bins
- Environment: Conveyor belt with parts, bins
- Actuators: Jointed arm and hand
- Sensors: Camera, joint angle sensors

PEAS

- Agent: Interactive English tutor
- Performance measure:

- Environment:

- Actuators:

- Sensors:

PEAS

- Agent: Interactive English tutor
- Performance measure: Maximize student's score on test
- Environment: Set of students
- Actuators: Screen display (exercises, suggestions, corrections)
- Sensors: Keyboard

Examples of agents in different types of applications

Agent type	Percepts	Actions	Goals	Environment
Medical diagnosis system	Symptoms, findings, patient's answers	Questions, tests, treatments	Healthy patients, minimize costs	Patient, hospital
Satellite image analysis system	Pixels of varying intensity, color	Print a categorization of scene	Correct categorization	Images from orbiting satellite
Part-picking robot	Pixels of varying intensity	Pick up parts and sort into bins	Place parts in correct bins	Conveyor belts with parts
Refinery controller	Temperature, pressure readings	Open, close valves; adjust temperature	Maximize purity, yield, safety	Refinery
Interactive English tutor	Typed words	Print exercises, suggestions, corrections	Maximize student's score on test	Set of students

task environm.	observable	determ./ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle						
chess with clock						
poker						
back gammon						
taxi driving						
medical diagnosis						
image analysis						
partpicking robot						
refinery controller						
interact. Eng. tutor						

task environm.	observable	determ./ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle	fully	determ.	sequential	static	discrete	single
chess with clock						
poker						
back gammon						
taxi driving						
medical diagnosis						
image analysis						
partpicking robot						
refinery controller						
interact. Eng. tutor						

task environm.	observable	determ./ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle	fully	determ.	sequential	static	discrete	single
chess with clock	fully	strategic	sequential	semi	discrete	multi
poker						
back gammon						
taxi driving						
medical diagnosis						
image analysis						
partpicking robot						
refinery controller						
interact. Eng. tutor						

task environm.	observable	determ./ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle	fully	determ.	sequential	static	discrete	single
chess with clock	fully	strategic	sequential	semi	discrete	multi
poker	partial	stochastic	sequential	static	discrete	multi
back gammon						
taxi driving						
medical diagnosis						
image analysis						
partpicking robot						
refinery controller						
interact. Eng. tutor						

task environm.	observable	determ./ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle	fully	determ.	sequential	static	discrete	single
chess with clock	fully	strategic	sequential	semi	discrete	multi
poker	partial	stochastic	sequential	static	discrete	multi
back gammon	fully	stochastic	sequential	static	discrete	multi
taxi driving						
medical diagnosis						
image analysis						
partpicking robot						
refinery controller						
interact. Eng. tutor						

task environm.	observable	determ./ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle	fully	determ.	sequential	static	discrete	single
chess with clock	fully	strategic	sequential	semi	discrete	multi
poker	partial	stochastic	sequential	static	discrete	multi
back gammon	fully	stochastic	sequential	static	discrete	multi
taxi driving	partial	stochastic	sequential	dynamic	continuous	multi
medical diagnosis						
image analysis						
partpicking robot						
refinery controller						
interact. Eng. tutor						

task environm.	observable	determ./ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle	fully	determ.	sequential	static	discrete	single
chess with clock	fully	strategic	sequential	semi	discrete	multi
poker	partial	stochastic	sequential	static	discrete	multi
back gammon	fully	stochastic	sequential	static	discrete	multi
taxi driving	partial	stochastic	sequential	dynamic	continuous	multi
medical diagnosis	partial	stochastic	sequential	dynamic	continuous	single
image analysis						
partpicking robot						
refinery controller						
interact. Eng. tutor						

task environm.	observable	determ./ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle	fully	determ.	sequential	static	discrete	single
chess with clock	fully	strategic	sequential	semi	discrete	multi
poker	partial	stochastic	sequential	static	discrete	multi
back gammon	fully	stochastic	sequential	static	discrete	multi
taxi driving	partial	stochastic	sequential	dynamic	continuous	multi
medical diagnosis	partial	stochastic	sequential	dynamic	continuous	single
image analysis	fully	determ.	episodic	semi	continuous	single
partpicking robot						
refinery controller						
interact. Eng. tutor						

task environm.	observable	determ./ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle	fully	determ.	sequential	static	discrete	single
chess with clock	fully	strategic	sequential	semi	discrete	multi
poker	partial	stochastic	sequential	static	discrete	multi
back gammon	fully	stochastic	sequential	static	discrete	multi
taxi driving	partial	stochastic	sequential	dynamic	continuous	multi
medical diagnosis	partial	stochastic	sequential	dynamic	continuous	single
image analysis	fully	determ.	episodic	semi	continuous	single
partpicking robot	partial	stochastic	episodic	dynamic	continuous	single
refinery controller						
interact. Eng. tutor						

task environm.	observable	determ./ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle	fully	determ.	sequential	static	discrete	single
chess with clock	fully	strategic	sequential	semi	discrete	multi
poker	partial	stochastic	sequential	static	discrete	multi
back gammon	fully	stochastic	sequential	static	discrete	multi
taxi driving	partial	stochastic	sequential	dynamic	continuous	multi
medical diagnosis	partial	stochastic	sequential	dynamic	continuous	single
image analysis	fully	determ.	episodic	semi	continuous	single
partpicking robot	partial	stochastic	episodic	dynamic	continuous	single
refinery controller	partial	stochastic	sequential	dynamic	continuous	single
interact. Eng. tutor						

task environm.	observable	determ./ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle	fully	determ.	sequential	static	discrete	single
chess with clock	fully	strategic	sequential	semi	discrete	multi
poker	partial	stochastic	sequential	static	discrete	multi
back gammon	fully	stochastic	sequential	static	discrete	multi
taxi driving	partial	stochastic	sequential	dynamic	continuous	multi
medical diagnosis	partial	stochastic	sequential	dynamic	continuous	single
image analysis	fully	determ.	episodic	semi	continuous	single
partpicking robot	partial	stochastic	episodic	dynamic	continuous	single
refinery controller	partial	stochastic	sequential	dynamic	continuous	single
interact. Eng. tutor	partial	stochastic	sequential	dynamic	discrete	multi

Properties of task environments

	Solitaire	Backgammon	Internet shopping	Taxi
Observable (hidden)				
Deterministic (stochastic)				
Episodic (sequential)				
Static (Dynamic)				
Discrete (Continuous)				
Single-agent (multi-agent)				

Properties of task environments (cont'd)

	Solitaire	Backgammon	Internet shopping	Taxi
Observable (hidden)	Yes	Yes	No	No
Deterministic (stochastic)				
Episodic (sequential)				
Static (Dynamic)				
Discrete (Continuous)				
Single-agent (multi-agent)				

Properties of task environments (cont'd)

	Solitaire	Backgammon	Internet shopping	Taxi
Observable (hidden)	Yes	Yes	No	No
Deterministic (stochastic)	Yes	No	Partly	No
Episodic (sequential)				
Static (Dynamic)				
Discrete (Continuous)				
Single-agent (multi-agent)				

Properties of task environments (cont'd)

	Solitaire	Backgammon	Internet shopping	Taxi
Observable (hidden)	Yes	Yes	No	No
Deterministic (stochastic)	Yes	No	Partly	No
Episodic (sequential)	No	No	No	No
Static (Dynamic)				
Discrete (Continuous)				
Single-agent (multi-agent)				

Properties of task environments (cont'd)

	Solitaire	Backgammon	Internet shopping	Taxi
Observable (hidden)	Yes	Yes	No	No
Deterministic (stochastic)	Yes	No	Partly	No
Episodic (sequential)	No	No	No	No
Static (Dynamic)	Yes	Semi	Semi	No
Discrete (Continuous)				
Single-agent (multi-agent)				

Properties of task environments (cont'd)

	Solitaire	Backgammon	Internet shopping	Taxi
Observable (hidden)	Yes	Yes	No	No
Deterministic (stochastic)	Yes	No	Partly	No
Episodic (sequential)	No	No	No	No
Static (Dynamic)	Yes	Semi	Semi	No
Discrete (Continuous)	Yes	Yes	Yes	No
Single-agent (multi-agent)				

Properties of task environments (cont'd)

	Solitaire	Backgammon	Internet shopping	Taxi
Observable (hidden)	Yes	Yes	No	No
Deterministic (stochastic)	Yes	No	Partly	No
Episodic (sequential)	No	No	No	No
Static (Dynamic)	Yes	Semi	Semi	No
Discrete (Continuous)	Yes	Yes	Yes	No
Single-agent (multi-agent)	Yes	No	No	No

Intelligent Agents

- What is an agent ?
  - An **agent** is anything that **perceiving** its environment through **sensors** and **acting** upon that environment through **actuators**
  - Example:
    - Human is an agent
    - A robot is also an agent with cameras and motors
    - A thermostat detecting room temperature.

Intelligent Agents

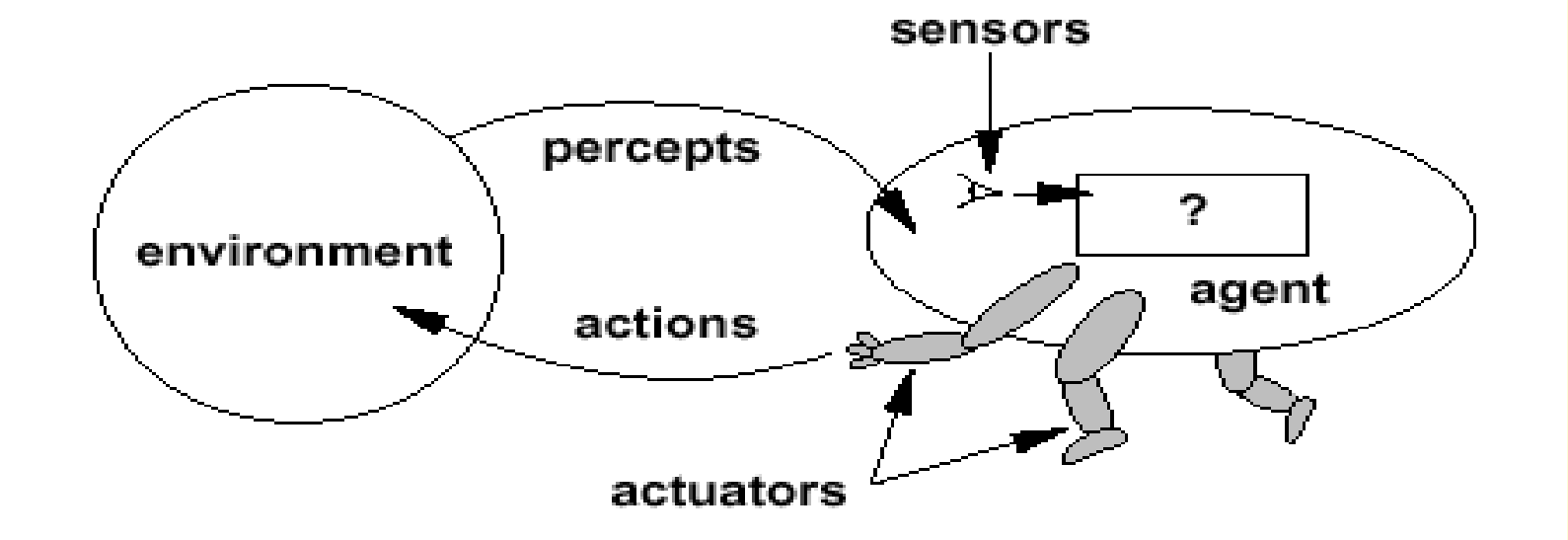
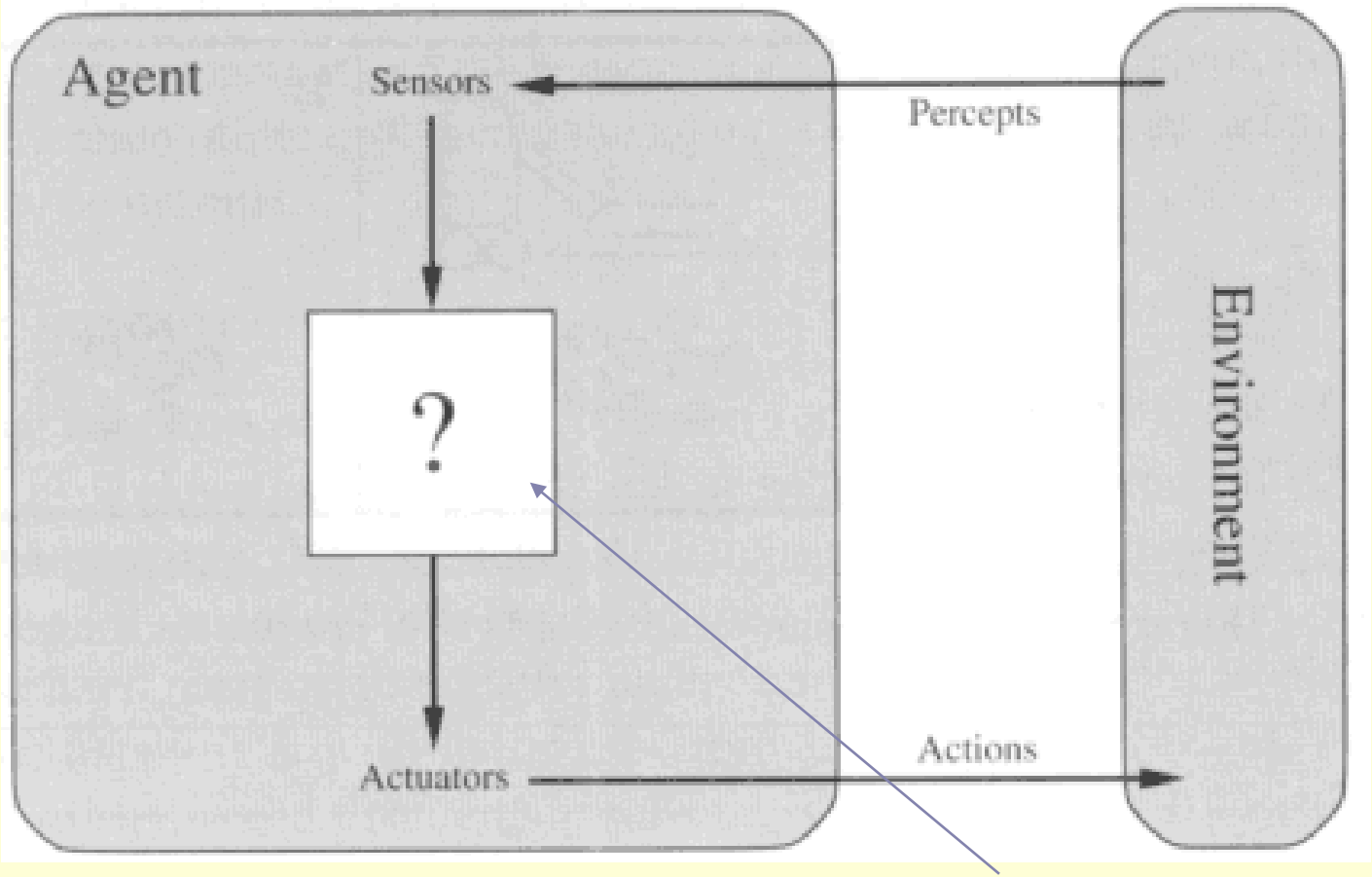


Diagram of an agent



What AI should fill

Simple Terms

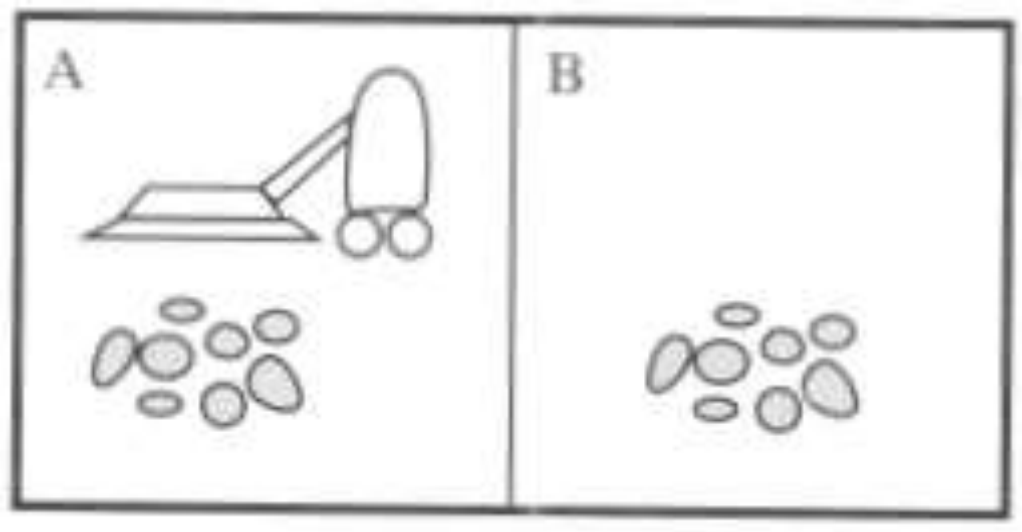
- Percept
  - Agent's perceptual inputs at any given instant
- Percept sequence
  - Complete history of everything that the agent has ever perceived.

Agent function & program

- Agent's behavior is *mathematically* described by
  - **Agent function**
  - A function mapping any given percept sequence to an action
- *Practically* it is described by
  - An **agent program**
  - The real implementation

Vacuum-cleaner world

- Perception: Clean or Dirty? where it is in?
- Actions: Move left, Move right, suck, do nothing



Vacuum-cleaner world

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*
⋮	⋮
$[A, Clean], [A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Clean], [A, Dirty]$	*Suck*
⋮	⋮

Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

Program implements the agent function tabulated in Fig. 2.3

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*

Concept of Rationality

- Rational agent
  - One that does the right thing
  - = every entry in the table for the agent function is correct (rational).
- What is correct?
  - The actions that cause the agent to be most successful
  - So we need ways to measure success.

Performance measure

- Performance measure
  - An objective function that determines
    - How the agent does successfully
    - E.g., 90% or 30% ?
- An agent, based on its percepts
  - → action sequence :

  if desirable, it is said to be performing well.
  - No universal performance measure for all agents

Performance measure

- A general rule:
  - Design performance measures according to
    - What one actually wants in the environment
    - Rather than how one thinks the agent should behave
- E.g., in vacuum-cleaner world
  - We want the floor clean, no matter how the agent behave
  - We don’t restrict how the agent behaves

Rationality

- What is rational at any given time depends on four things:
  - The performance measure defining the criterion of success
  - The agent's prior knowledge of the environment
  - The actions that the agent can perform
  - The agents's percept sequence to date

Rational agent

- For each possible percept sequence,
  - an rational agent should select
    - an action expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has
- E.g., an exam
  - Maximize marks, based on

the questions on the paper & your knowledge

Rational agent

- Performance measure
  - Awards one point for each clean square
    - at each time step, over 10000 time steps
- Prior knowledge about the environment
  - The geography of the environment
  - Only two squares
  - The *effect* of the actions

Example of a rational agent

- Actions that can perform
  - Left, Right, Suck and NoOp
- Percept sequences
  - Where is the agent?
  - Whether the location contains dirt?
- Under this circumstance, the agent is rational.

Omniscience

- An omniscient agent
  - Knows the *actual* outcome of its actions in advance
  - No other possible outcomes
  - However, impossible in real world
- An example
  - crossing a street but died of the fallen cargo door from 33,000ft → irrational?

Omniscience

- Based on the circumstance, it is rational.
- As rationality maximizes
  - *Expected* performance
- Perfection maximizes
  - *Actual* performance
- Hence rational agents are not *omniscient*.

Learning

- Does a rational agent depend on only current percept?
  - No, the past percept sequence should also be used
  - This is called **learning**
  - After experiencing an episode, the agent
    - should adjust its behaviors to perform better for the same job next time.

Autonomy

- If an agent just relies on the prior knowledge of its designer rather than its own percepts then the agent lacks ***autonomy***

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

Software Agents

- Sometimes, the environment may not be the real world
  - E.g., flight simulator, video games, Internet
  - They are all artificial but very complex environments
  - Those agents working in these environments are called
    - Software agent (softbots)
    - Because all parts of the agent are software

Task environments

- Task environments are the problems
  - While the rational agents are the solutions
- Specifying the task environment
  - PEAS description as fully as possible
    - Performance
    - Environment
    - Actuators
    - Sensors

    In designing an agent, the first step must always be to specify the task environment as fully as possible.
- Use automated taxi driver as an example

Task environments

- Performance measure
  - How can we judge the automated driver?
  - Which factors are considered?
    - getting to the correct destination
    - minimizing fuel consumption
    - minimizing the trip time and/or cost
    - minimizing the violations of traffic laws
    - maximizing the safety and comfort, etc.

Task environments

- Environment
  - A taxi must deal with a variety of roads
  - Traffic lights, other vehicles, pedestrians, stray animals, road works, police cars, etc.
  - Interact with the customer

Task environments

- Actuators (for outputs)
  - Control over the accelerator, steering, gear shifting and braking
  - A display to communicate with the customers
- Sensors (for inputs)
  - Detect other vehicles, road situations
  - GPS (Global Positioning System) to know where the taxi is
  - Many more devices are necessary

Task environments

- A sketch of automated taxi driver

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

Figure 2.4 PEAS description of the task environment for an automated taxi.

Properties of task environments

- Fully observable vs. Partially observable
  - If an agent's sensors give it access to the complete state of the environment at each point in time then the environment is effectively and fully observable
    - if the sensors detect all aspects
    - That are relevant to the choice of action

    Example: Chess game

- Partially observable

An environment might be Partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data.

Example:

- A local dirt sensor of the cleaner cannot tell
- Whether other squares are clean or not
- Automated Taxi driver

Properties of task environments

- Deterministic vs. stochastic
  - next state of the environment Completely determined by the current state and the actions executed by the agent, then the environment is deterministic, otherwise, it is Stochastic.
  - Strategic environment: deterministic except for actions of other agents

  Example: Chess game

  -vaccume Cleaner and taxi driver are:
    - Stochastic because of some unobservable aspects → noise or unknown

Properties of task environments

- Episodic vs. sequential
  - An episode = agent's single pair of perception & action
  - The quality of the agent's action does not depend on other episodes
    - Every episode is independent of each other
  - Episodic environment is simpler
    - The agent does not need to think ahead

    Example: Classification
- Sequential
  - Current action may affect all future decisions

  -Ex. Taxi driving and chess.

Properties of task environments

- Static vs. dynamic
  - A dynamic environment is always changing over time
    - E.g., Automated Taxi driver
  - While static environment
    - E.g., crossword puzzle
- Semidynamic
  - environment is not changed over time

but the agent's performance score does.

Properties of task environments

- Discrete vs. continuous
  - If there are a limited number of distinct states, clearly defined percepts and actions, the environment is discrete
  - E.g., Chess game
  - Continuous: Taxi driving because the speed and location of the taxi and the other vehicles sweep through a range of continuous values.

Properties of task environments

- Single agent VS. multiagent
  - Playing a crossword puzzle – single agent
  - Chess playing – two agents
  - Competitive multiagent environment
    - Chess playing
  - Cooperative multiagent environment
    - Automated taxi driver
    - Avoiding collision

Properties of task environments

- Known vs. unknown

This distinction refers not to the environment itslef but to the agent's (or designer's) state of knowledge about the environment.

-In known environment, the outcomes for all actions are given. (example: solitaire card games).

- If the environment is unknown, the agent will have to learn how it works in order to make good decisions.(example: new video game).

Examples of task environments

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Crossword puzzle	Fully	Deterministic	Sequential	Static	Discrete	Single
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Poker	Partially	Strategic	Sequential	Static	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

Figure 2.6 Examples of task environments and their characteristics.

Structure of agents

- Agent = architecture + program
  - Architecture = some sort of computing device (sensors + actuators)
  - (Agent) Program = some function that implements the agent mapping = “?”
  - Agent Program = Job of AI

Agent programs

- Input for Agent Program
  - Only the current percept
- Input for Agent Function
  - The entire percept sequence
  - The agent must remember all of them
- Implement the agent program as
  - A look up table (agent function)

Agent programs

- Skeleton design of an agent program

```
function TABLE-DRIVEN-AGENT(percept) returns action
  static: percepts, a sequence, initially empty
          table, a table, indexed by percept sequences, initially fully specified

  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action
```

Agent Programs

- P = the set of possible percepts
- T= lifetime of the agent
  - The total number of percepts it receives
- Size of the look up table $\sum_{t=1}^{T} |P|^t$
- Consider playing chess
  - P =10, T=150
  - Will require a table of at least 10^{150} entries

Agent programs

- Despite of huge size, look up table does what we want.
- The key challenge of AI
  - Find out how to write programs that, to the extent possible, produce rational behavior
    - From a small amount of code
    - Rather than a large amount of table entries
  - E.g., a five-line program of Newton's Method
  - V.s. huge tables of square roots, sine, cosine, …

Types of agent programs

- Four types
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents

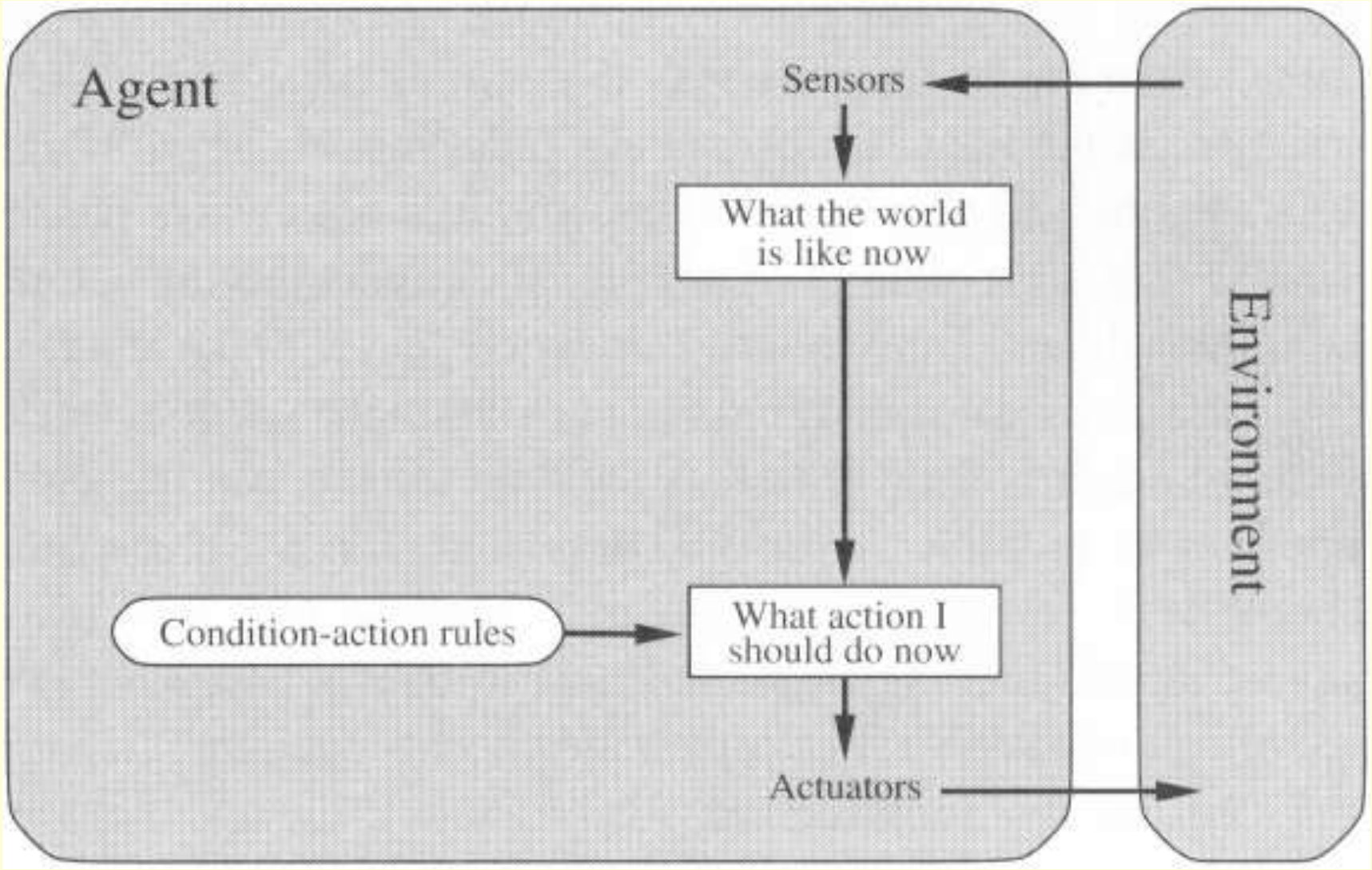
Simple reflex agents

- It uses just ***condition-action rules***
  - The rules are like the form “if … then …”
  - efficient but have narrow range of applicability
  - Because knowledge sometimes cannot be stated explicitly
  - Work only
    - if the environment is fully observable

Simple reflex agents

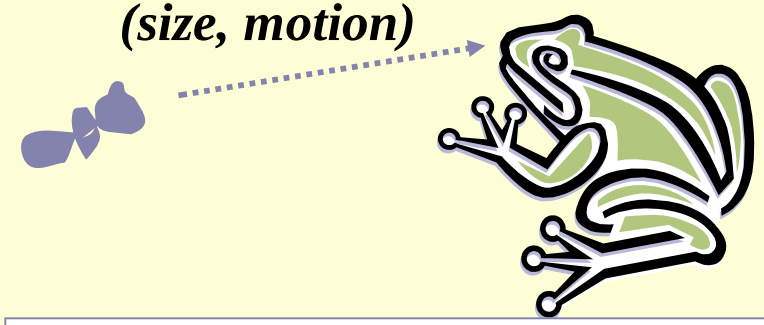
```
function SIMPLE-REFLEX-AGENT(percept) returns action
    static: rules, a set of condition-action rules

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

Simple reflex agents (2)



A Simple Reflex Agent in Nature

percepts
(size, motion)



RULES:

(1) If small moving object,
then activate SNAP

(2) If large moving object,
then activate AVOID and inhibit SNAP

ELSE (not moving) then NOOP

needed for completeness

Action: SNAP or AVOID or NOOP

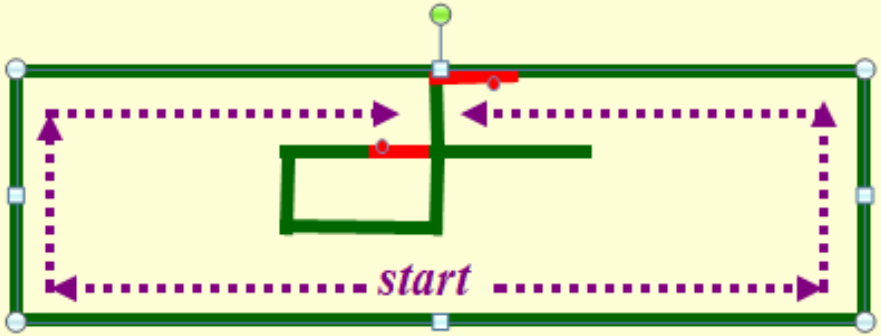
Model-based Reflex Agents

- For the world that is partially observable
  - the agent has to keep track of an internal state
    - That depends on the percept history
    - Reflecting some of the unobserved aspects
    - E.g., driving a car and changing lane
- Requiring two types of knowledge
  - How the world evolves independently of the agent
  - How the agent’s actions affect the world

Example Table Agent With Internal State

IF	THEN
Saw an object ahead, and turned right, and it's now clear ahead	Go straight
Saw an object Ahead, turned right, and object ahead again	Halt
See no objects ahead	Go straight
See an object ahead	Turn randomly

Example Reflex Agent With Internal State: **Wall-Following**



Actions: left, right, straight, open-door

Rules:

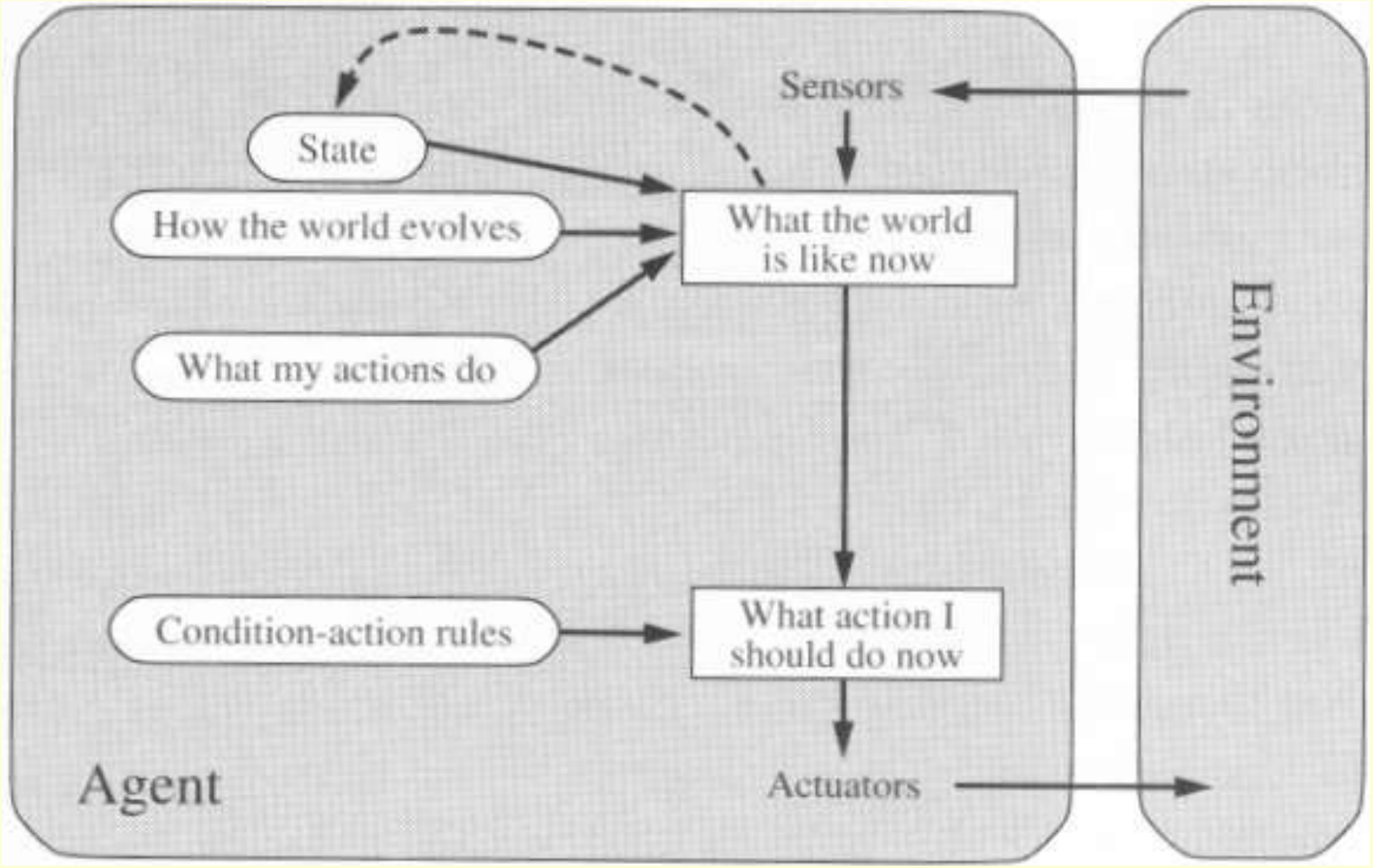
1. If open(left) & open(right) and open(straight) then choose randomly between right and left
2. If wall(left) and open(right) and open(straight) then straight
3. If wall(right) and open(left) and open(straight) then straight
4. If wall(right) and open(left) and wall(straight) then left
5. If wall(left) and open(right) and wall(straight) then right
6. If wall(left) and door(right) and wall(straight) then open-door
7. If wall(right) and wall(left) and open(straight) then straight.
8. (Default) Move randomly

Model-based Reflex Agents

```
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
```

The agent is with memory

Model-based Reflex Agents



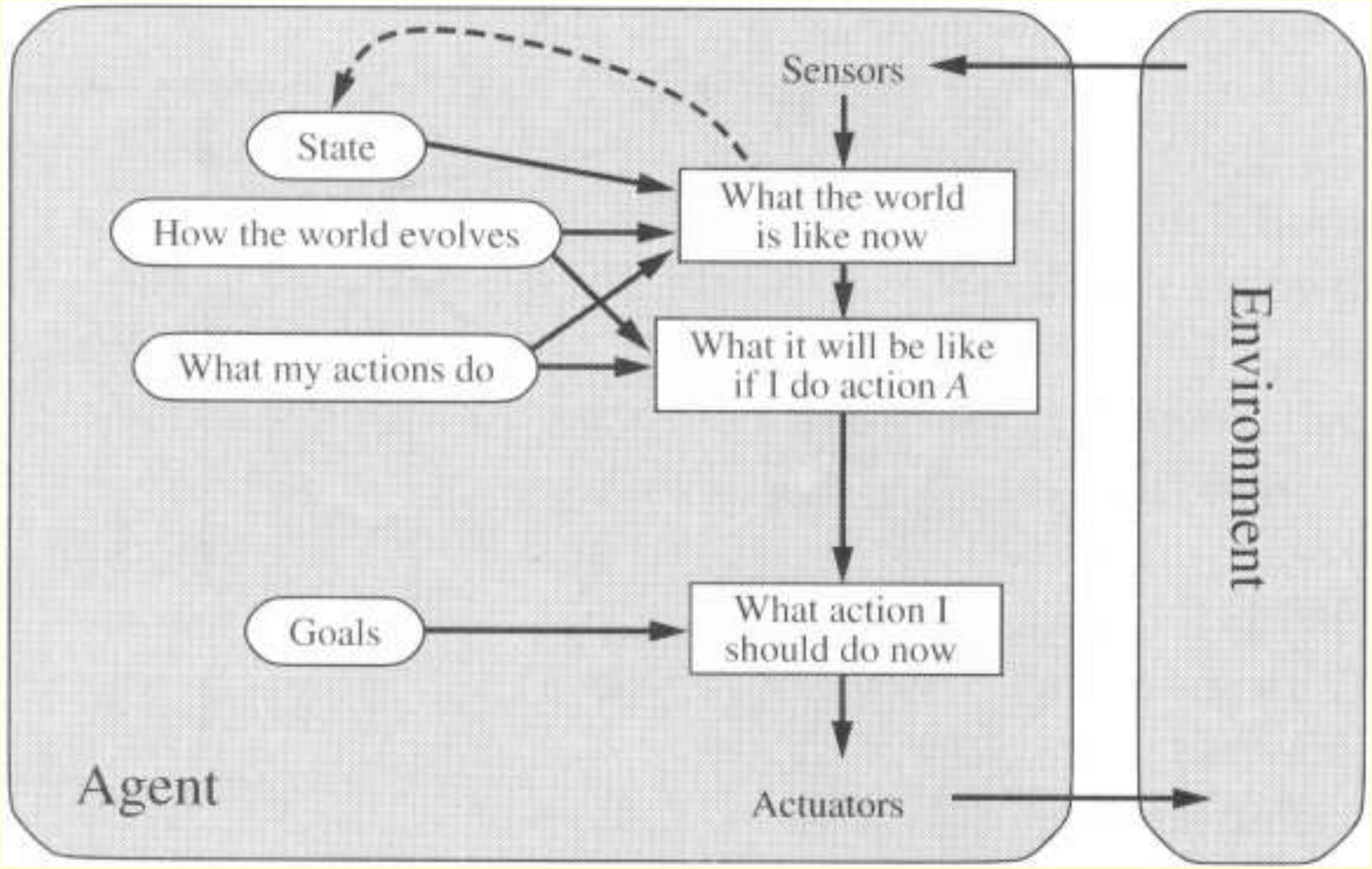
Goal-based agents

- Current state of the environment is always not enough
- The goal is another issue to achieve
  - Judgment of rationality / correctness
- Actions chosen → goals, based on
  - the current state
  - the current percept

Goal-based agents

- Conclusion
  - Goal-based agents are less efficient
  - but more flexible
    - Agent ← Different goals ← different tasks
  - Search and planning
    - two other sub-fields in AI
    - to find out the action sequences to achieve its goal

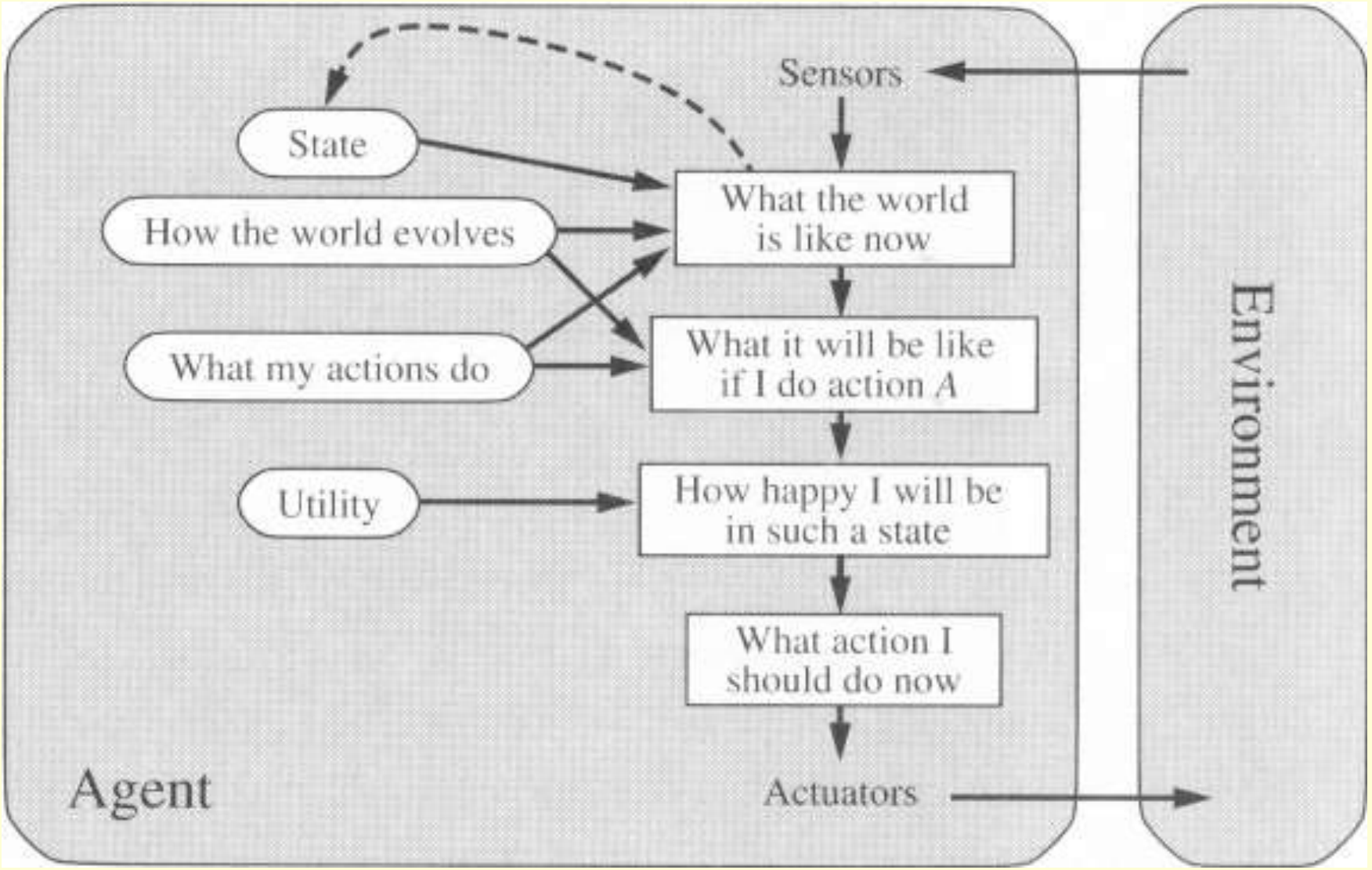
Goal-based agents



Utility-based agents

- Goals alone are not enough
  - to generate **high-quality** behavior
  - E.g. meals in Canteen, good or not ?
- Many action sequences → the goals
  - some are better and some worse
  - If goal means success,
  - then **utility** means the degree of success (how successful it is)

Utility-based agents (4)



Utility-based agents

- it is said state A has higher utility
  - If state A is more preferred than others
- Utility is therefore a function
  - that maps a state onto a real number
  - the degree of success

Utility-based agents (3)

- Utility has several advantages:
  - When there are conflicting goals,
    - Only some of the goals but not all can be achieved
    - utility describes the appropriate trade-off
  - When there are several goals
    - None of them are achieved **certainly**
    - utility provides a way for the decision-making

Learning Agents

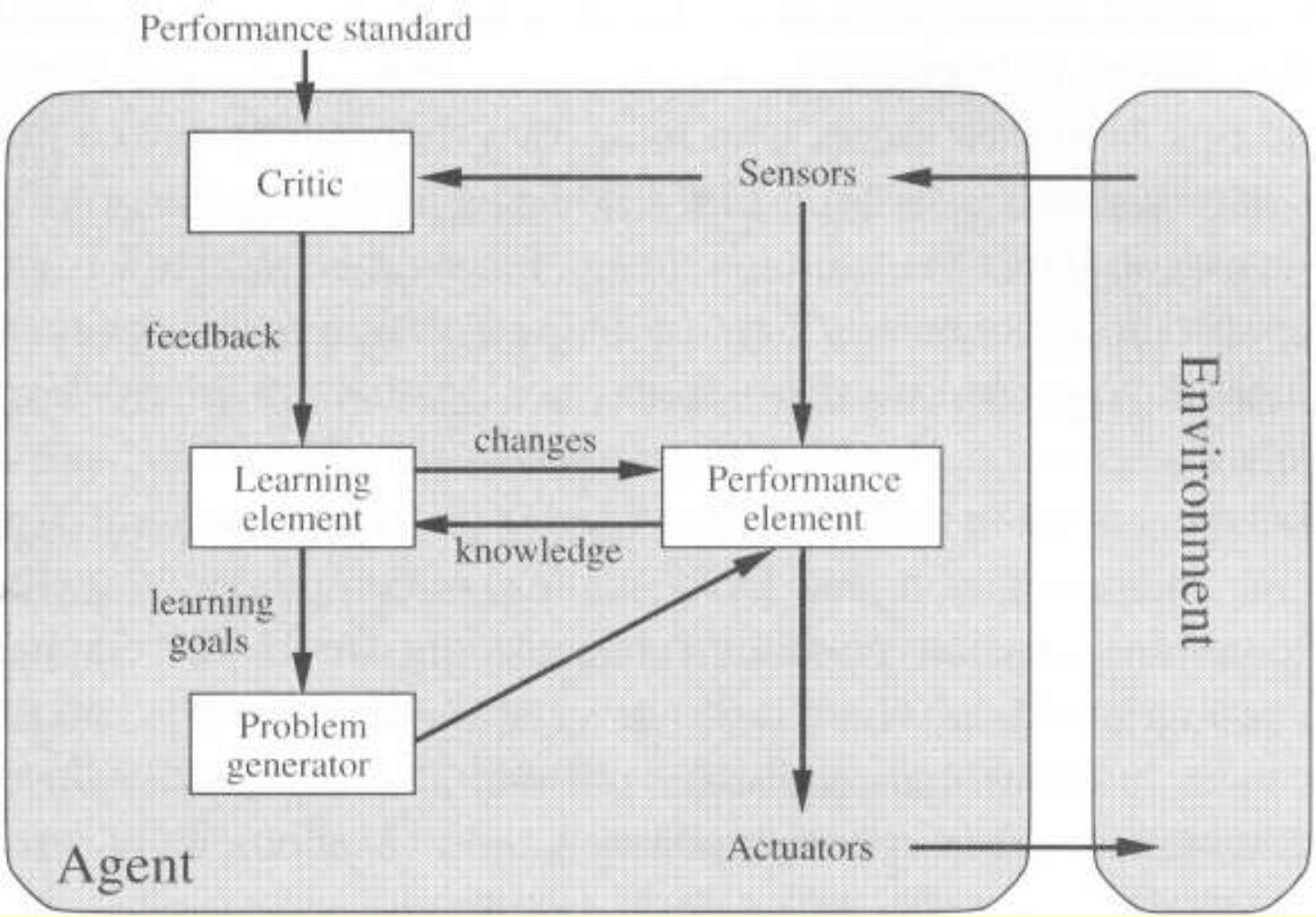
- After an agent is programmed, can it work immediately?
  - No, it still need teaching
- In AI,
  - Once an agent is done
  - We teach it by giving it a set of examples
  - Test it by using another set of examples
- We then say the agent learns
  - A learning agent

Learning Agents

- Four conceptual components
  - Learning element
    - Making improvement
  - Performance element
    - Selecting external actions
  - Critic
    - Tells the Learning element how well the agent is doing with respect to fixed performance standard.

    (Feedback from user or examples, good or not?)
  - Problem generator
    - Suggest actions that will lead to new and informative experiences.

Learning Agents



Solving Problems by Searching

- Reflex agent is simple
  - base their actions on
  - a direct mapping from states to actions
  - but cannot work well in environments
    - which this mapping would be too large to store
    - and would take too long to learn
- Hence, goal-based agent is used

Problem-solving agent

- Problem-solving agent
  - A kind of goal-based agent
  - It solves problem by
    - finding sequences of actions that lead to desirable states (goals)
  - To solve a problem,
    - the first step is the ***goal formulation***, based on the current situation

Goal formulation

- The goal is formulated
  - as a set of states, in which the goal is satisfied
- Reaching from initial state → goal state
  - Actions are required
- *Goal formulation, based on the current situation and the agent's performance measure, is the first step in problem solving.*
- Actions are the operators
  - causing transitions between states
  - **Actions** should be abstract enough at a certain degree, instead of very detailed
  - E.g., turn left VS turn left 30 degree, etc.

Problem formulation

- The process of deciding
  - what actions and states to consider, given a goal
- E.g., driving Amman → Zarqa
  - in-between states and actions defined
  - States: Some places in Amman & Zarqa
  - Actions: Turn left, Turn right, go straight, accelerate & brake, etc.

Search

- Because there are many ways to achieve the same goal
  - Those ways are together expressed as a tree
  - Multiple options of unknown value at a point,
    - the agent can examine different possible sequences of actions, and choose the best
  - This process of looking for such a sequence is called ***search***
  - A search algorithm takes a problem as input and returns a ***solution*** *in the form of an action sequence.*

Search algorithm

- Defined as
  - taking a *problem*
  - and returns a *solution*
- Once a solution is found
  - the agent follows the solution
  - and carries out the list of actions – execution phase
- Design of an agent
  - “Formulate, search, execute”

A simple problem-solving agent

```
function SIMPLE-PROBLEM-SOLVING-AGENT(p) returns an action
    inputs: 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
```

Well-defined problems and solutions

- A problem is defined by 4 components:
  - The ***initial state***
    - that the agent starts in
  - The set of possible **actions**

Transition model: *description of what each action does.*

(***successor functions***): refer to any state reachable from given state by a single action

Initial state, actions and Transition model define the ***state space***

- the set of all states reachable from the initial state by any sequence of actions.

A ***path*** in the state space:

- any sequence of states connected by a sequence of actions.

Well-defined problems and solutions

- The ***goal test***
  - Applied to the current state to test
    - if the agent is in its goal

-Sometimes there is an explicit set of possible goal states. (example: in Amman).

-Sometimes the goal is described by the properties

- instead of stating explicitly the set of states

Example: Chess

- the agent wins if it can capture the KING of the opponent on next move (checkmate).
- no matter what the opponent does

Well-defined problems and solutions

- A ***path cost*** function,
  - assigns a numeric cost to each path
  - = performance measure
  - denoted by ***g***
  - to distinguish the best path from others
- Usually the path cost is
  - the sum of the **step costs** of the individual actions (in the action list)

Well-defined problems and solutions

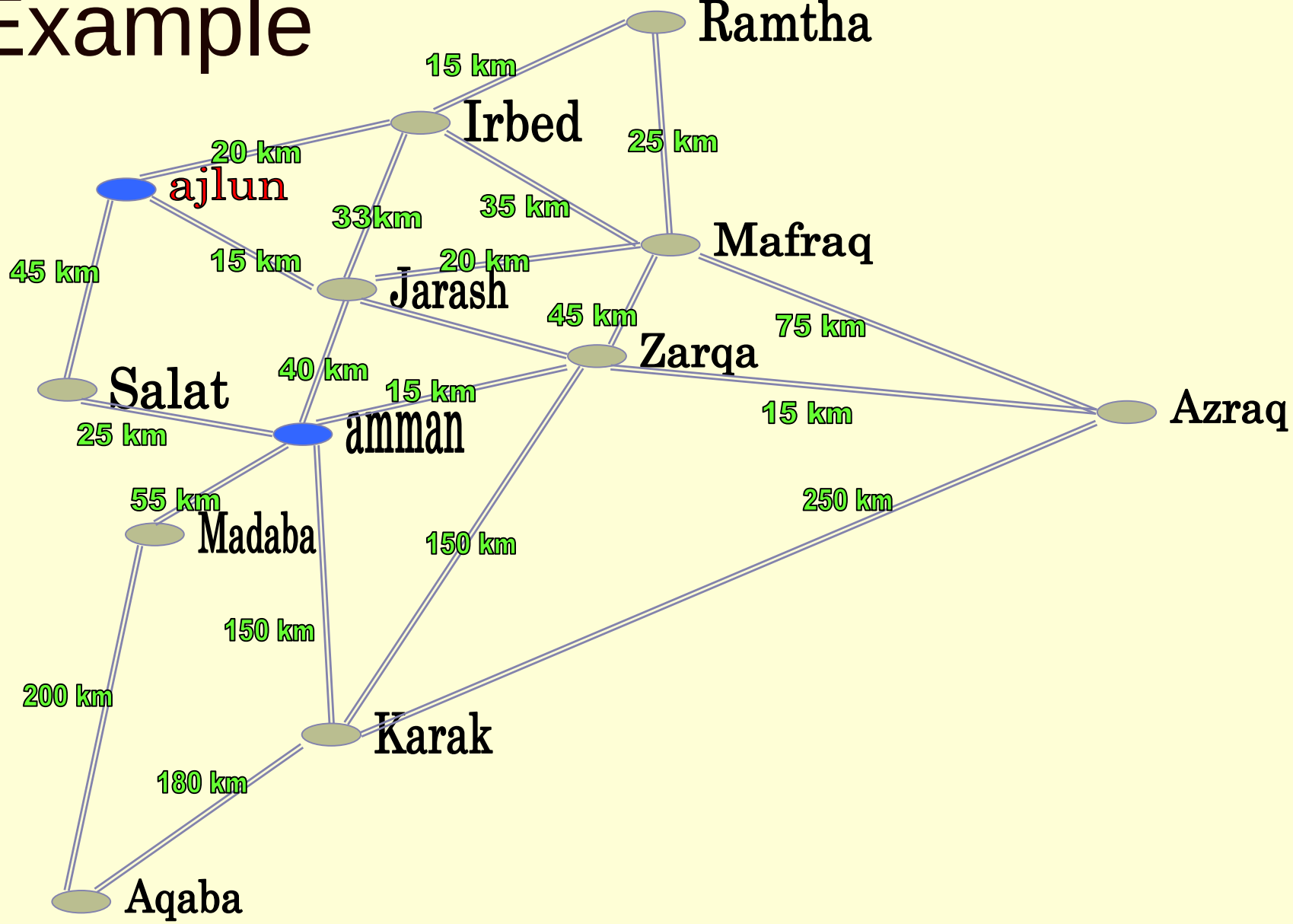
- Together a problem is defined by
  - Initial state
  - Actions
  - Successor function
  - Goal test
  - Path cost function
- The ***solution*** of a problem is then
  - *a path from the initial state to a state satisfying the goal test*
- ***Optimal*** solution
  - the solution with lowest path cost among all solutions

Formulating problems

- Besides the four components for problem formulation
  - anything else?
- ***Abstraction***
  - the process to take out the irrelevant information
  - leave the most essential parts to the description of the states

  (Remove detail from representation)
  - **Conclusion**: Only the most important parts *that are contributing to searching* are used

Example


Ramtha
15 km
Irbed
20 km
25 km
ajlun
33km
35 km
Mafraq
45 km
15 km
20 km
Jarash
45 km
75 km
Zarqa
Salat
40 km
15 km
15 km
Azraq
25 km
amman
55 km
250 km
Madaba
150 km
150 km
200 km
Karak
180 km
Aqaba

From our Example

1. Formulate Goal

- Be In Amman

2. Formulate Problem

- States : Cities
- actions : Drive Between Cities

3. Find Solution

- Sequence of Cities : ajlun – Jarash - Amman

Our Example

1. **Problem : To Go from Ajlun to Amman**

2. **Initial State : Ajlun**

3. **Operator : Go from One City To another .**

4. **State Space : {Jarash , Salat , irbed,……..}**

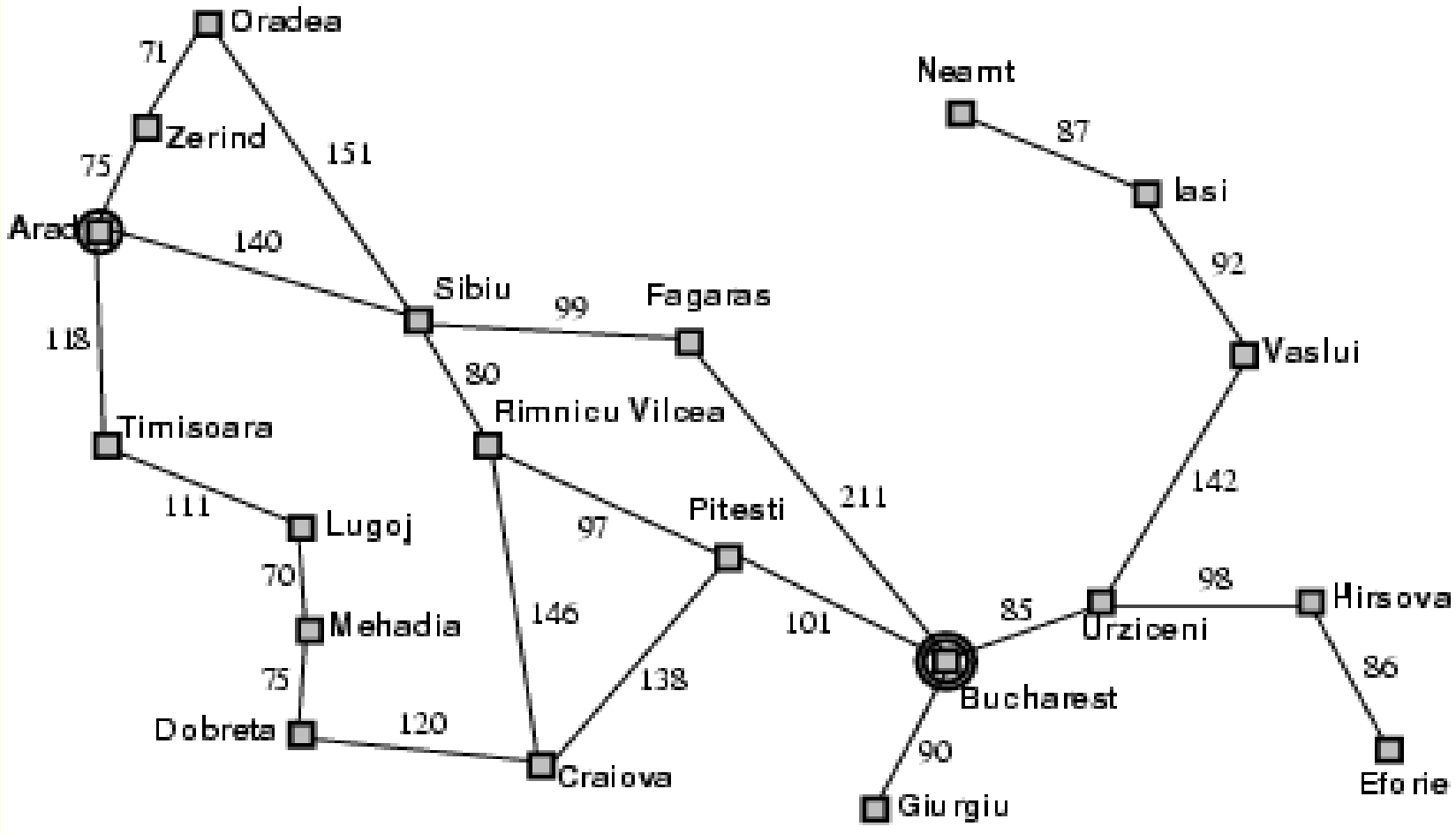
5. **Goal Test : are the agent in Amman.**

6. **Path Cost Function : Get The Cost From The Map.**

7. **Solution :** { {Aj → Ja → Ir → Ma → Za → Am} , {Aj →Ir → Ma → Za → Am} …. {Aj → Ja → Am} }

8. **State Set Space :** {Ajlun → Jarash → Amman}

Example: Romania

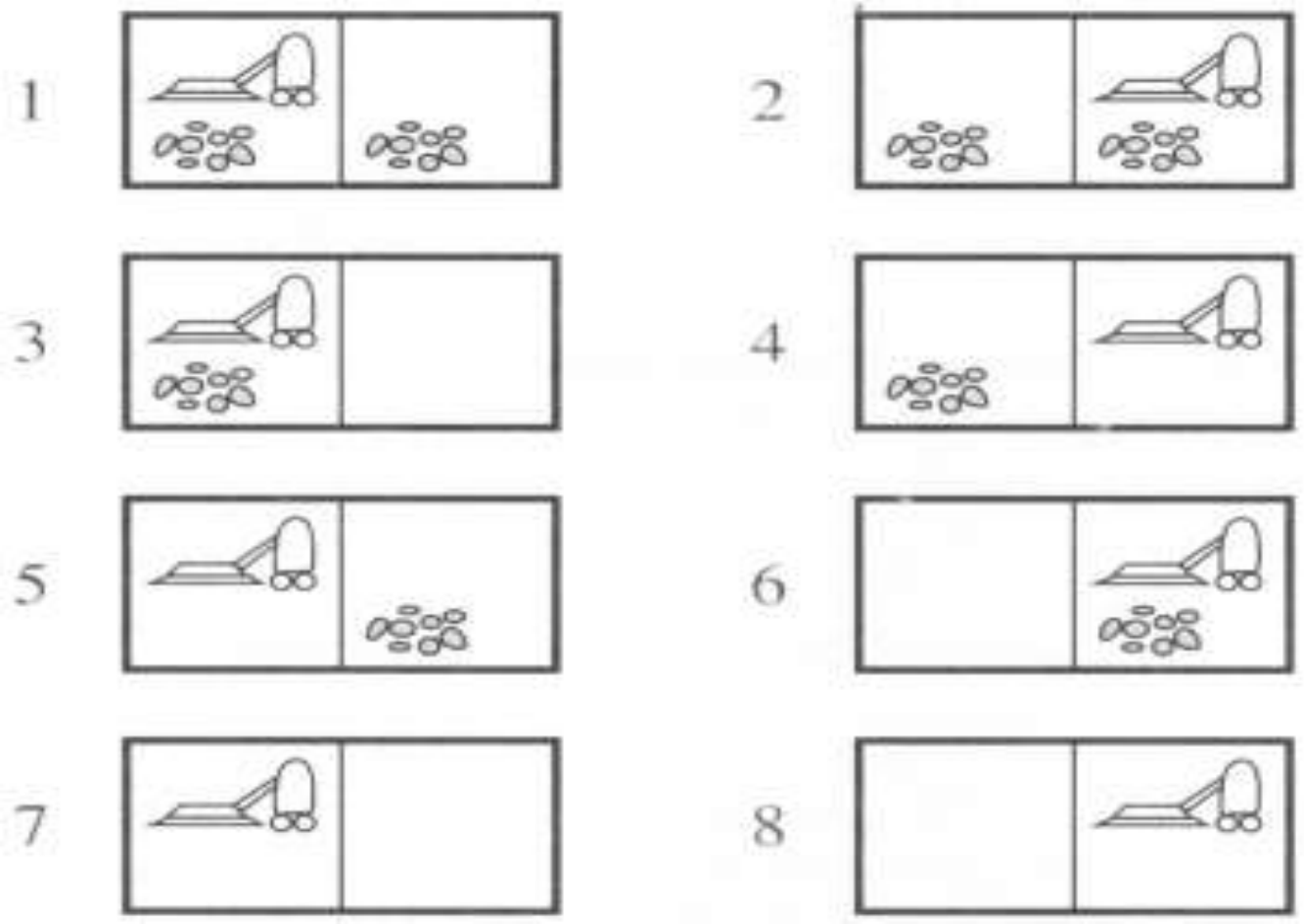


Example problems

- *Toy problems*
  - those intended to illustrate or exercise various problem-solving methods
  - E.g., puzzle, chess, etc.
- *Real-world problems*
  - tend to be more difficult and whose solutions people actually care about
  - E.g., Design, planning, etc.

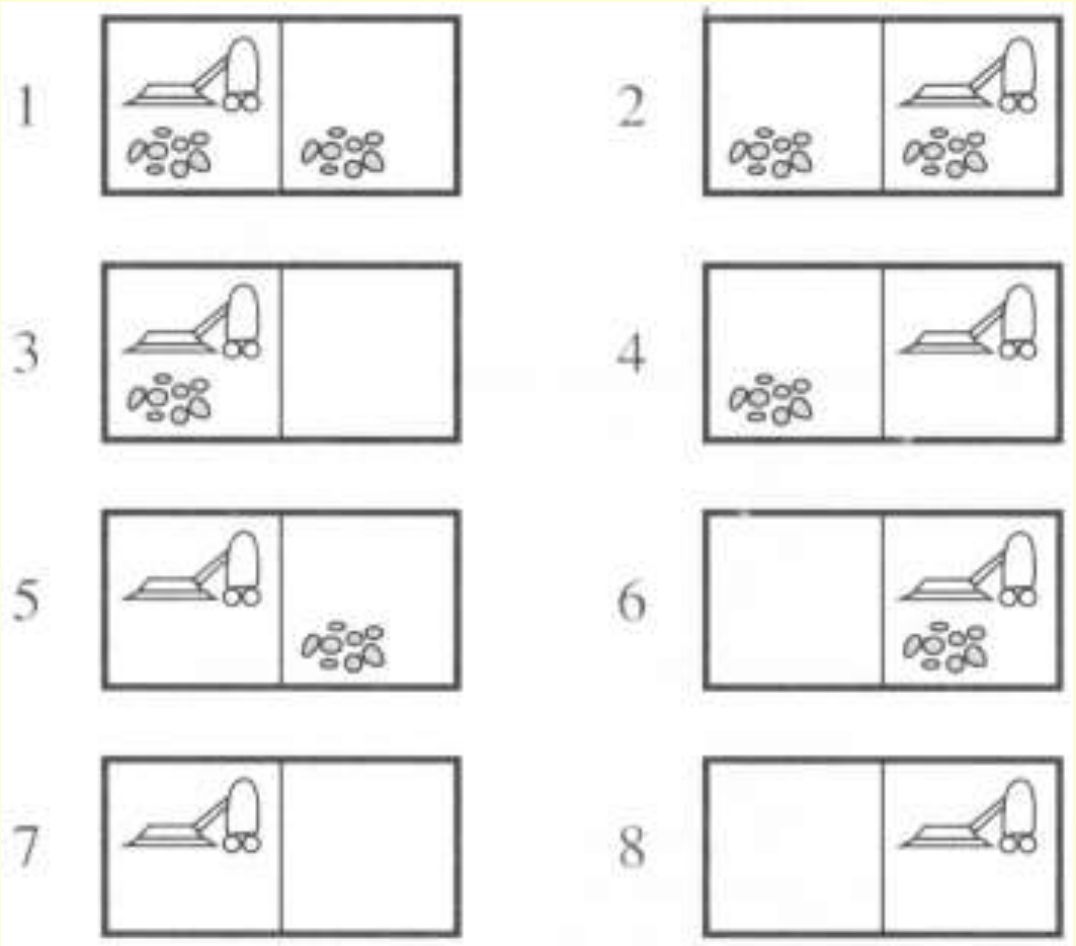
Toy problems

- Example: vacuum world

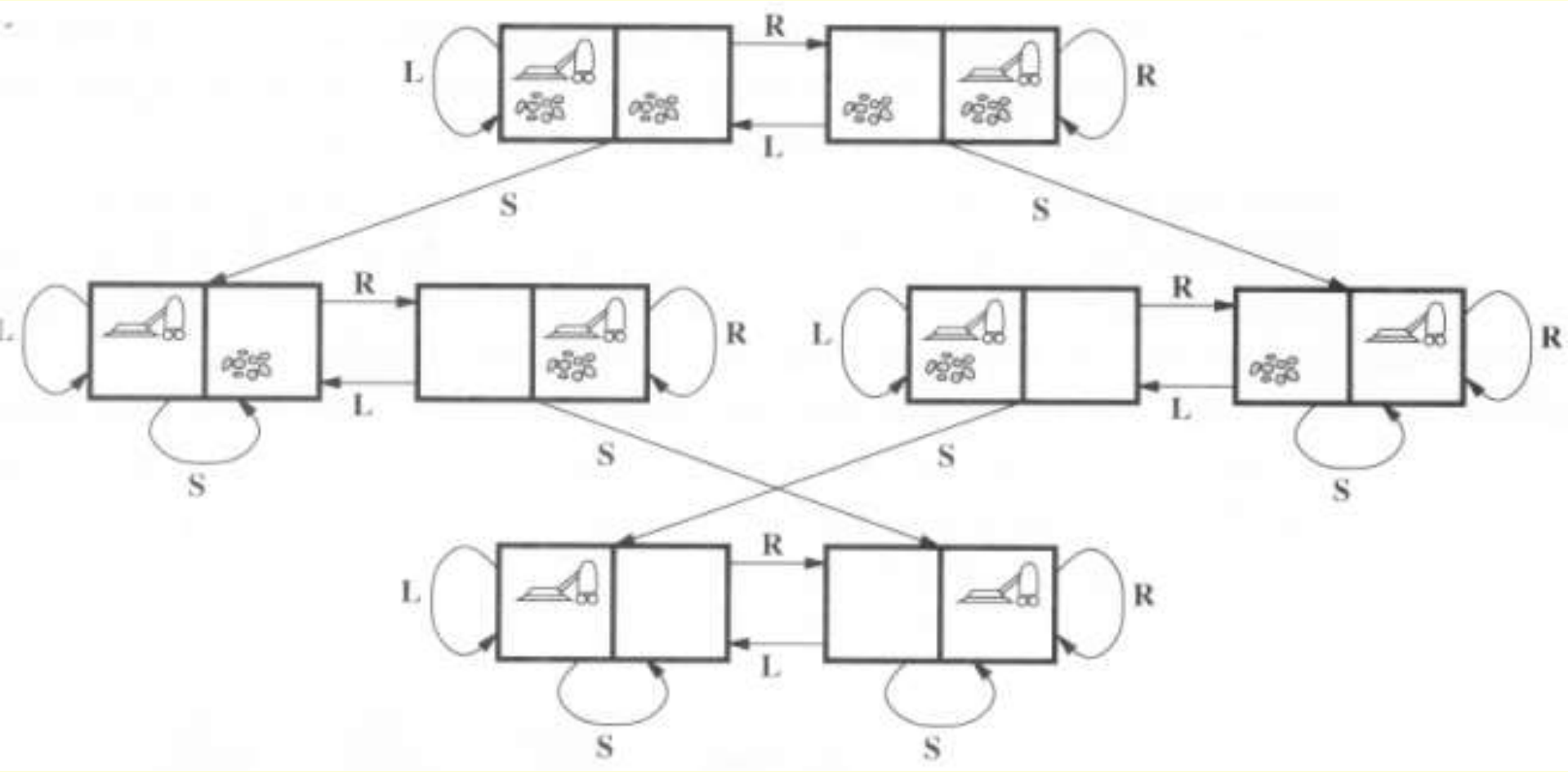


Toy problems

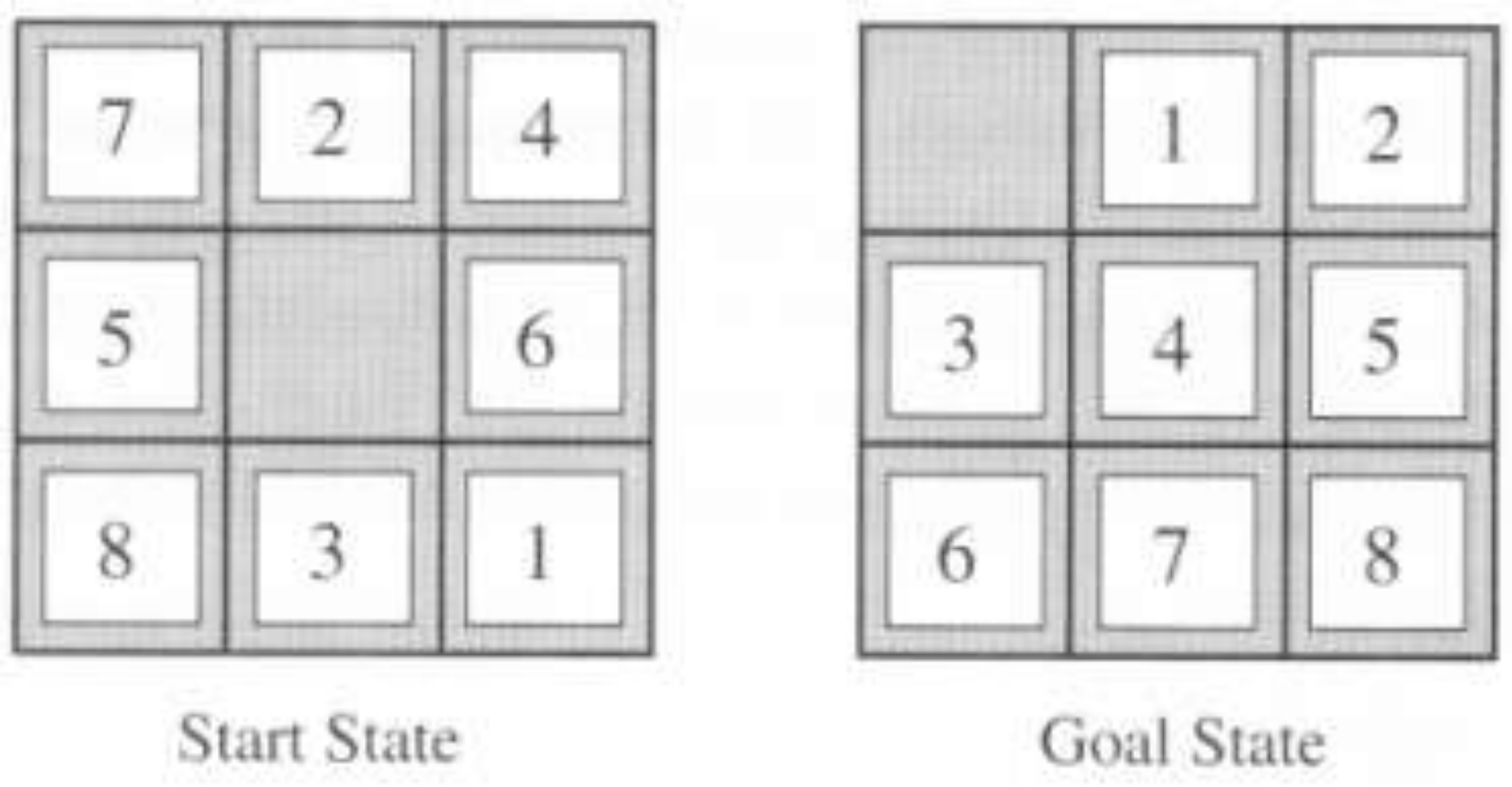
- Example: vacuum world



- Number of states: 8
- Initial state: Any
- Number of actions: 4
  - *left, right, suck, noOp*
- Goal: clean up all dirt
  - Goal states: {7, 8}
- Path Cost:
  - Each step costs 1


L
R
L
R
S
S
L
R
R
L
S
S
L
R
R
L
S
S
L
R
R
L
S
S

The 8-puzzle



The 8-puzzle

- **States**:
  - a state description specifies the location of each of the eight tiles and blank in one of the nine squares
- **Initial State**:
  - Any state in state space
- **Successor function**:
  - the blank moves *Left*, *Right*, *Up*, or *Down*
- **Goal test**:
  - current state matches the goal configuration
- **Path cost**:
  - each step costs 1, so the path cost is just the length of the path

The 8-queens

- There are two ways to formulate the problem
- All of them have the common followings:
  - Goal test: 8 queens on board, not attacking to each other

The 8-queens

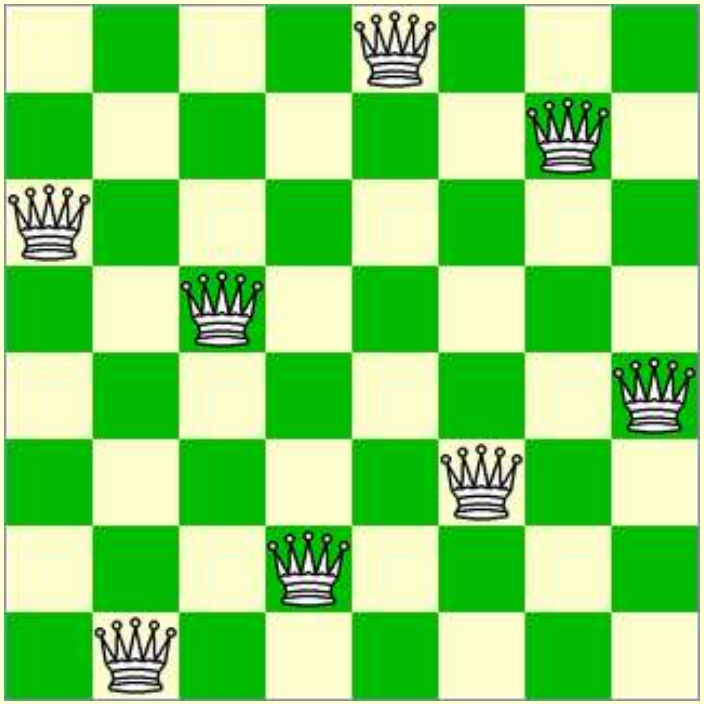
- (1) Incremental formulation
  - involves operators that augment the state description starting from an empty state
  - Each action adds a queen to the state
  - States:
    - any arrangement of 0 to 8 queens on board
  - Successor function:
    - add a queen to any empty square

The 8-queens

- (2) Complete-state formulation
  - starts with all 8 queens on the board
  - move the queens individually around
  - States:
    - any arrangement of 8 queens, one per column in the leftmost columns
  - Operators: move an attacked queen to a row, not attacked by any other

The 8-queens

- Conclusion:
  - the right formulation makes a big difference to the size of the search space



Examples

- Route finding problem
- Touring problem
- VLSI layout
- Robot navigation
- Automatic assembly sequence
- Protein design
- Internet searching

3.3 Searching for solutions

3.3 Searching for solutions

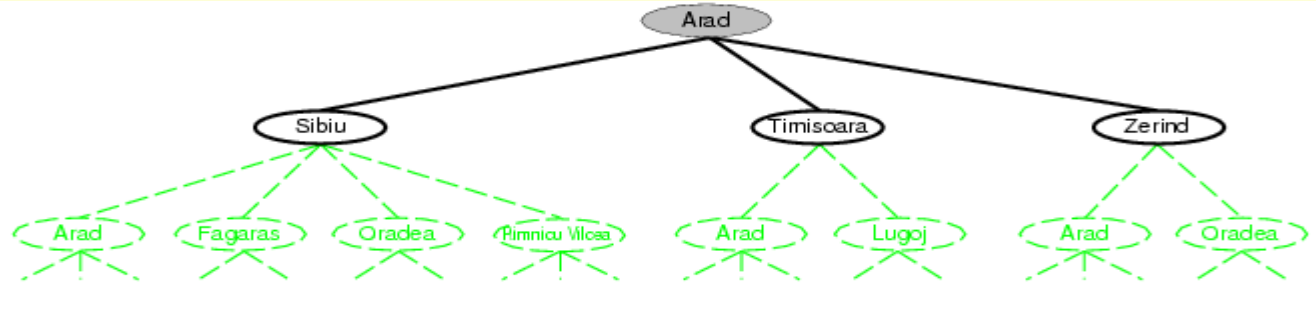
- Finding out a solution is done by
  - searching through the state space
- All problems are transformed
  - as a search tree
  - generated by the initial state and successor function

Search tree

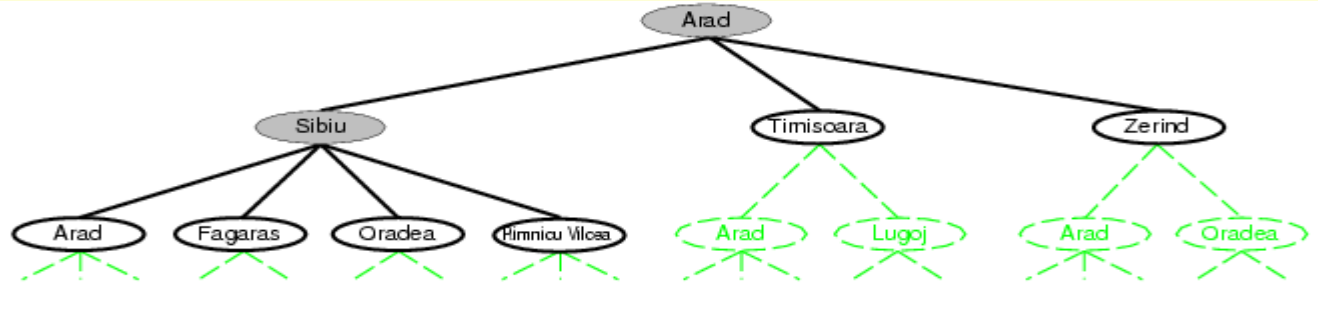
- **Initial state**
  - The root of the search tree is a **search node**
- **Expanding**
  - applying successor function to the current state
  - thereby generating a new set of states
- **leaf nodes**
  - the states having no successors

  Fringe : Set of search nodes that have not been expanded yet.
- Refer to next figure

Tree search example



Tree search example



Search tree

- The **essence** of searching
  - in case the first choice is not correct
  - choosing one option and keep others for later inspection
- Hence we have the **search strategy**
  - which determines the choice of which state to expand
  - good choice → fewer work → faster
- Important:
  - state space ≠ search tree

Search tree

- A node is having five components:
  - STATE: which state it is in the state space
  - PARENT-NODE: from which node it is generated
  - ACTION: which action applied to its parent-node to generate it
  - PATH-COST: the cost, g(n), from initial state to the node n itself
  - DEPTH: number of steps along the path from the initial state

Search tree

```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
```

Informal Description of Genearl search Algorithm

Search tree

```
function TREE-SEARCH( problem, fringe) returns a solution, or failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST[problem](STATE[node]) then return SOLUTION(node)
        fringe ← INSERTALL(EXPAND(node, problem), fringe)

function EXPAND( node, problem) returns a set of nodes
    successors ← the empty set
    for each action, result in SUCCESSOR-FN[problem](STATE[node]) do
        s ← a new NODE
        PARENT-NODE[s] ← node;  ACTION[s] ← action;  STATE[s] ← result
        PATH-COST[s] ← PATH-COST[node] + STEP-COST(node, action, s)
        DEPTH[s] ← DEPTH[node] + 1
        add s to successors
    return successors
```

Measuring problem-solving performance

- The evaluation of a search strategy
  - **Completeness:**
    - is the strategy guaranteed to find a solution when there is one?
  - **Optimality**:
    - does the strategy find the highest-quality solution when there are several different solutions?
  - **Time complexity**:
    - how long does it take to find a solution?
  - **Space complexity**:
    - how much memory is needed to perform the search?

Measuring problem-solving performance

- In AI, complexity is expressed in
  - **b**, branching factor, maximum number of successors of any node
  - **d**, the depth of the shallowest goal node.

  (depth of the least-cost solution)
  - **m**, the maximum length of any path in the state space
- Time and Space is measured in
  - number of nodes generated during the search
  - maximum number of nodes stored in memory

Measuring problem-solving performance

- For effectiveness of a search algorithm
  - we can just consider the total cost
  - **The total cost** = path cost (g) of the solution found **+** search cost
    - search cost = time necessary to find the solution
- Tradeoff:
  - (long time, optimal solution with least g)
  - vs. (shorter time, solution with slightly larger path cost g)

3.4 Uninformed search strategies

3.4 Uninformed search strategies

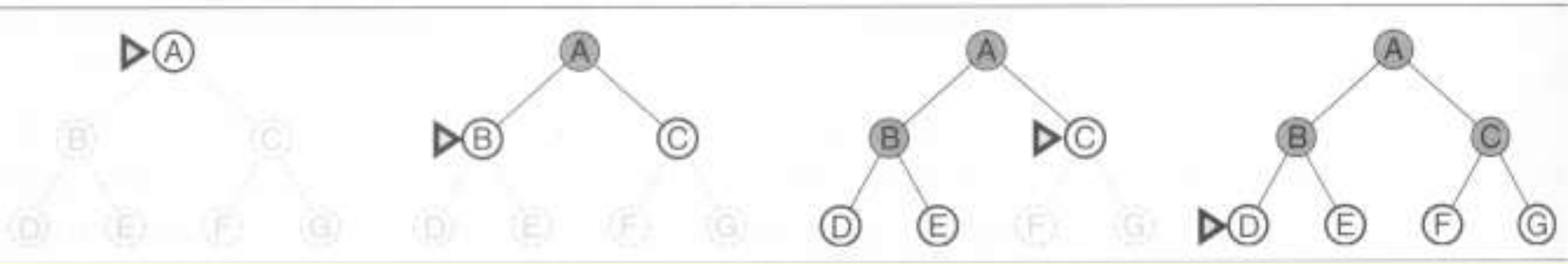
- **Uninformed search**
  - no information about the number of steps
  - or the path cost from the current state to the goal
  - search the state space **blindly**
- **Informed search, or heuristic search**
  - a cleverer strategy that searches toward the goal,
  - based on the information from the current state so far

Uninformed search strategies

- Breadth-first search
  - Uniform cost search
- Depth-first search
  - Depth-limited search
  - Iterative deepening search
- Bidirectional search

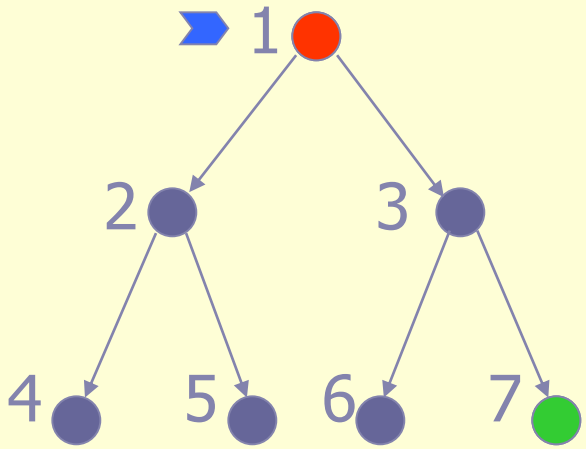
Breadth-first search

- The root node is expanded first (FIFO)
- All the nodes generated by the root node are then expanded
- And then their successors and so on



Breadth-First Strategy

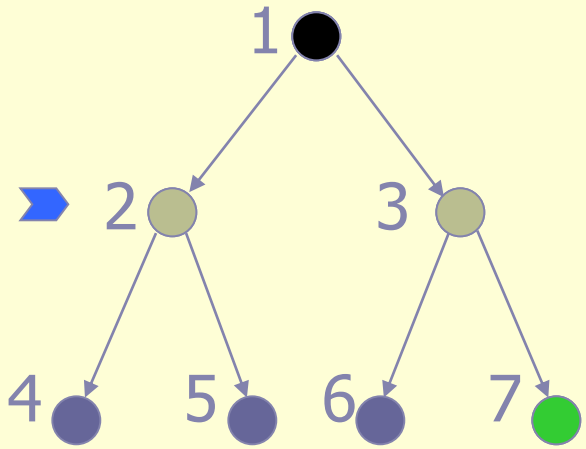
New nodes are inserted at the end of FRINGE



FRINGE = (1)

Breadth-First Strategy

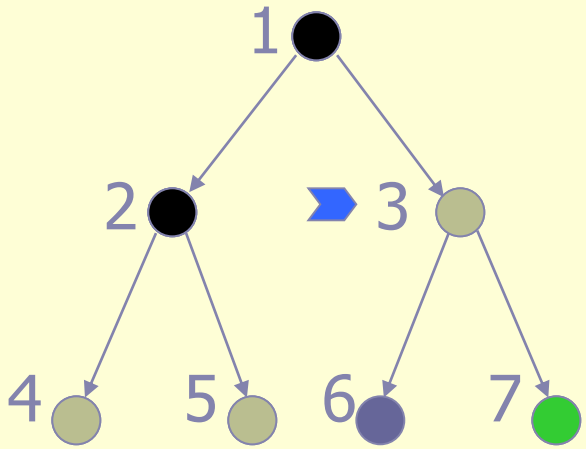
New nodes are inserted at the end of FRINGE



FRINGE = (2, 3)

Breadth-First Strategy

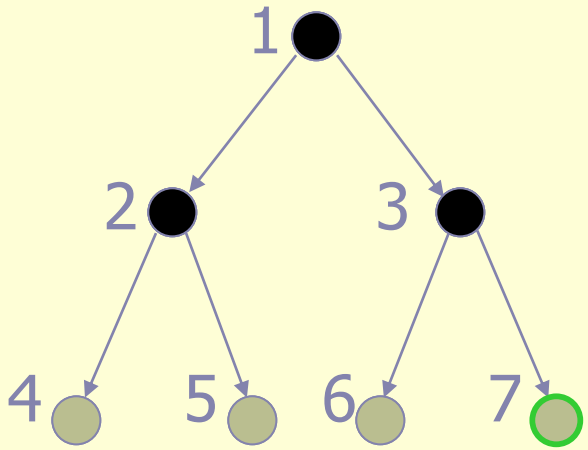
New nodes are inserted at the end of FRINGE



FRINGE = (3, 4, 5)

Breadth-First Strategy

New nodes are inserted at the end of FRINGE



FRINGE = (4, 5, 6, 7)

Breadth-first search (Analysis)

- Breadth-first search
  - Complete – find the solution eventually
  - Optimal, if step cost is 1

  The disadvantage
  - if the branching factor of a node is large,
  - for even small instances (e.g., chess)
    - the ***space complexity*** and the ***time complexity*** are enormous

Properties of breadth-first search

- Complete? Yes (if *b* is finite)
- Time? $1+b+b^2+b^3+\ldots+b^d = b(b^d-1) = O(b^{d+1})$
- Space? $O(b^{d+1})$ (keeps every node in memory)
- Optimal? Yes (if cost = 1 per step)
- Space is the bigger problem (more than time)

Breadth-first search (Analysis)

- assuming 10000 nodes can be processed per second, each with 1000 bytes of storage

Depth	Nodes	Time	Memory
2	1100	.11 seconds	1 megabyte
4	111,100	11 seconds	106 megabytes
6	10^7	19 minutes	10 gigabytes
8	10^9	31 hours	1 terabytes
10	10^{11}	129 days	101 terabytes
12	10^{13}	35 years	10 petabytes
14	10^{15}	3,523 years	1 exabyte

Figure 3.11 Time and memory requirements for breadth-first search. The numbers shown assume branching factor $b = 10$; 10,000 nodes/second; 1000 bytes/node.

Uniform cost search

- **Breadth-first** finds the shallowest goal state
  - but not necessarily be the least-cost solution
  - work only if all step costs are equal
- **Uniform cost search**
  - modifies breadth-first strategy
    - by always expanding the lowest-cost node
  - The lowest-cost node is measured by the path cost *g(n)*

Uniform-cost search

- Expand least-cost unexpanded node
- Implementation:
  - *fringe* = queue ordered by path cost
- Equivalent to breadth-first if step costs all equal
- Complete? Yes, if step cost ≥ ε
- Time? numbr of nodes with $g \leq$ cost of optimal solution, $O(b^{ceiling(C^*/\varepsilon)})$ where C^* is the cost of the optimal solution
- Space? Number of nodes with $g \leq$ cost of optimal solution, $O(b^{ceiling(C^*/\varepsilon)})$
- Optimal? Yes – nodes expanded in increasing order of $g(n)$

let

C be the cost of optimal solution.*

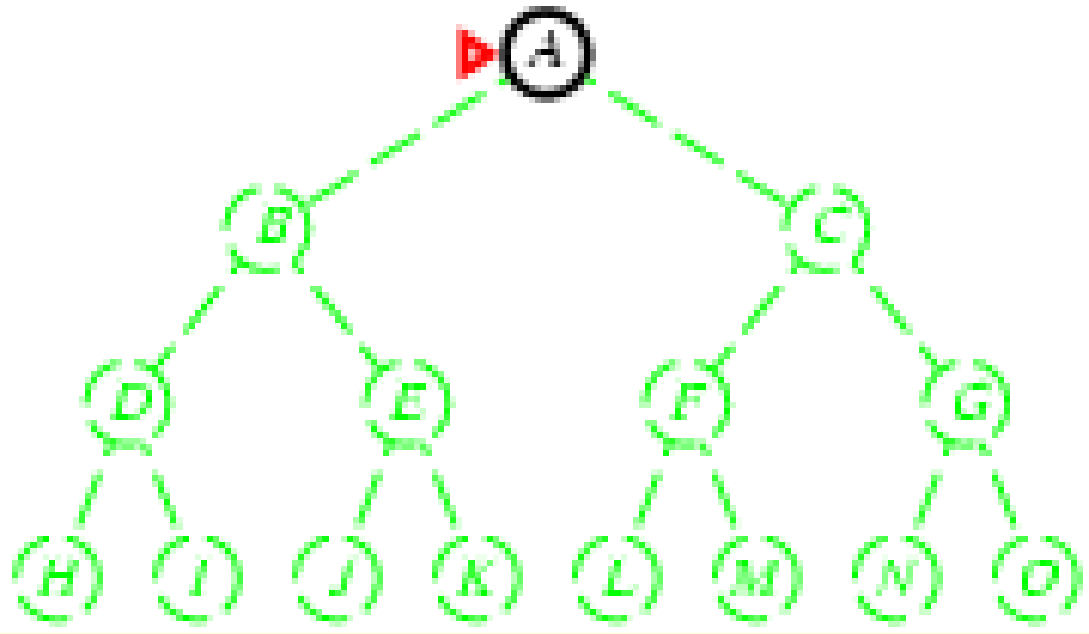
ε is possitive constant (every action cost)

Depth-first search

- Always expands one of the nodes at the ***deepest*** level of the tree
- Only when the search hits a dead end
  - goes back and expands nodes at shallower levels
  - Dead end → leaf nodes but not the goal
- Backtracking search
  - only one successor is generated on expansion
  - rather than all successors
  - fewer memory

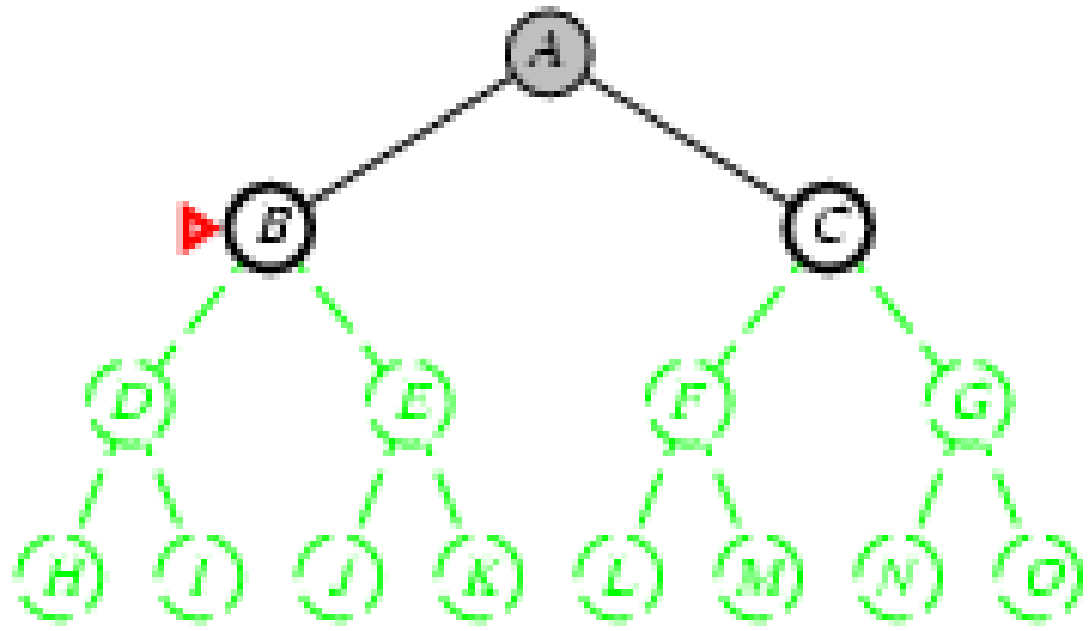
Depth-first search

- Expand deepest unexpanded node
- Implementation:
  - *fringe* = LIFO queue, i.e., put successors at front



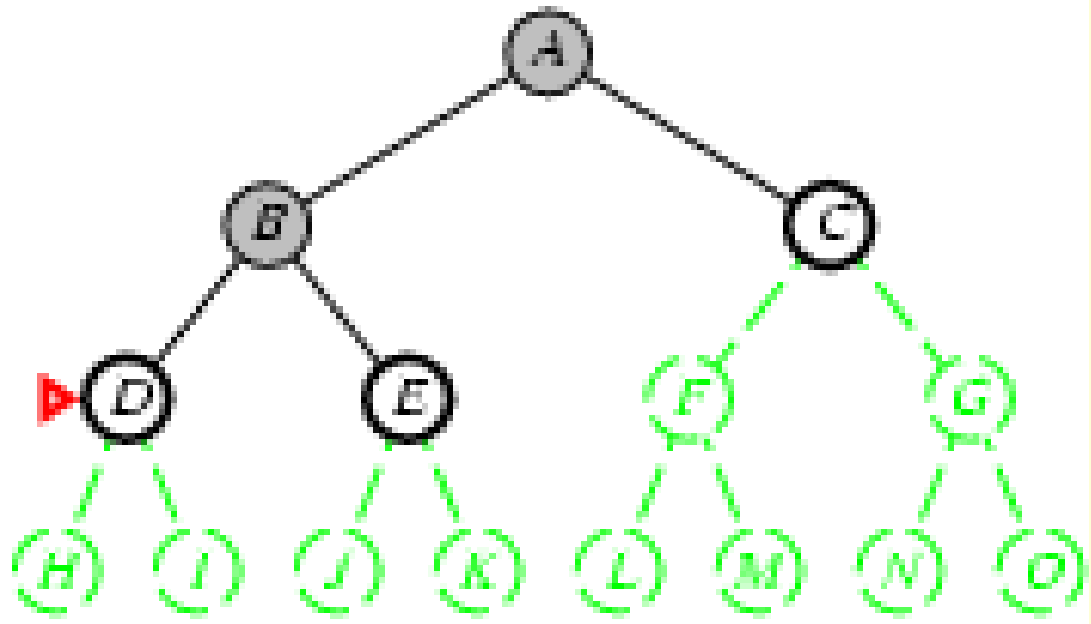
Depth-first search

- Expand deepest unexpanded node
- Implementation:
  - *fringe* = LIFO queue, i.e., put successors at front



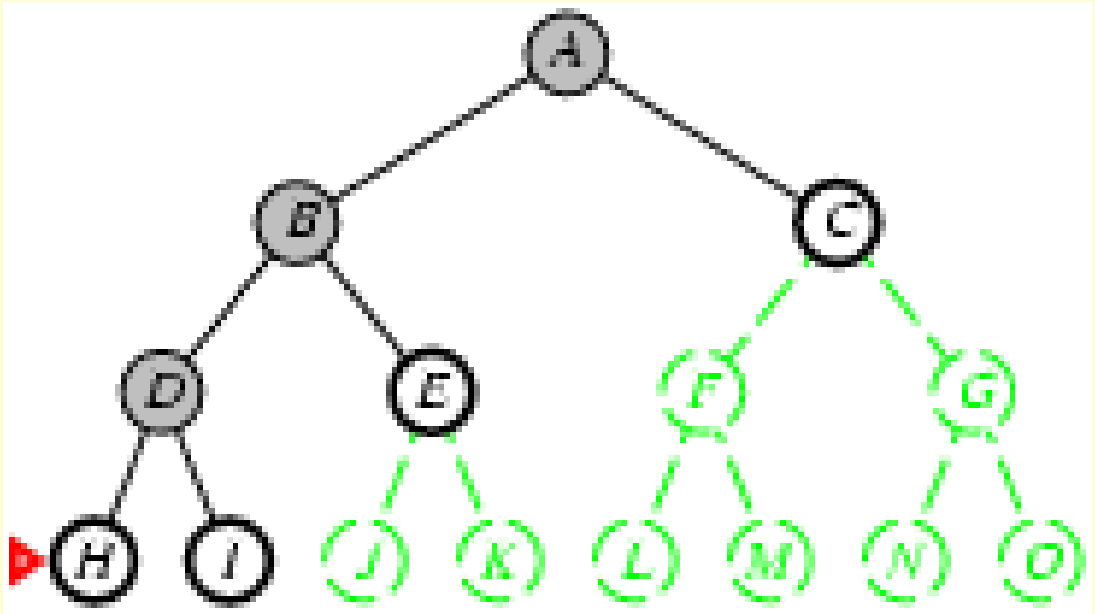
Depth-first search

- Expand deepest unexpanded node
- Implementation:
  - *fringe* = LIFO queue, i.e., put successors at front



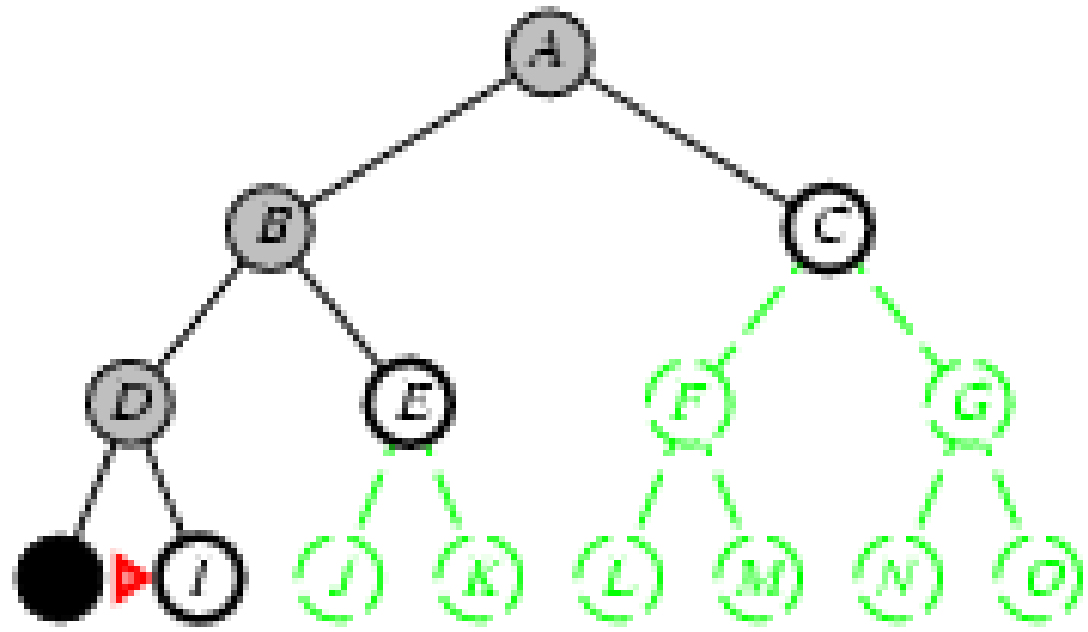
Depth-first search

- Expand deepest unexpanded node
- Implementation:
  - *fringe* = LIFO queue, i.e., put successors at front
  -



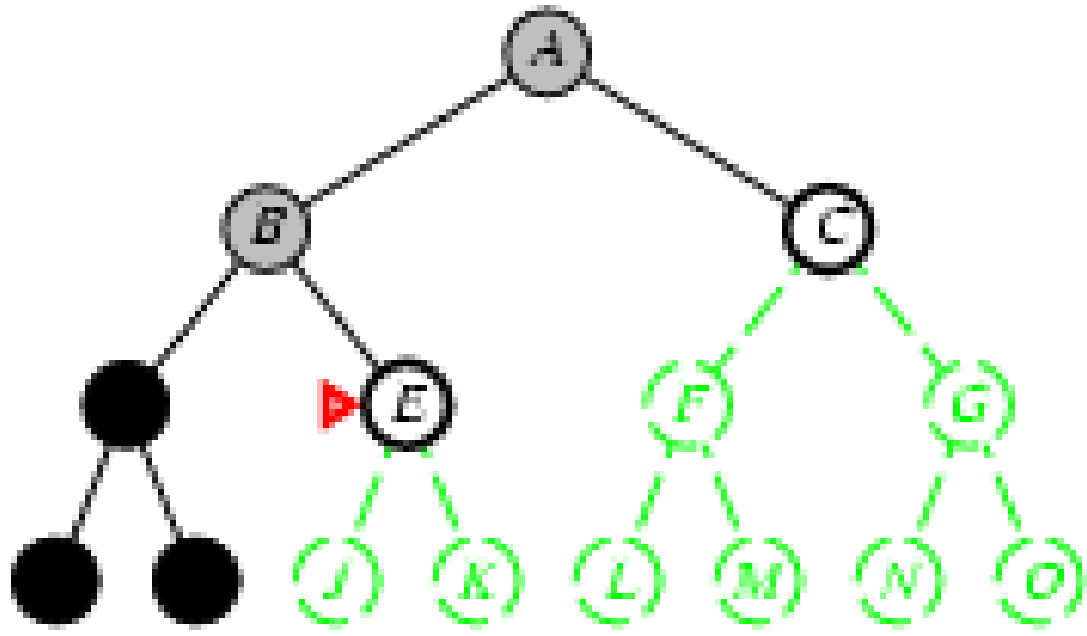
Depth-first search

- Expand deepest unexpanded node
- Implementation:
  - *fringe* = LIFO queue, i.e., put successors at front



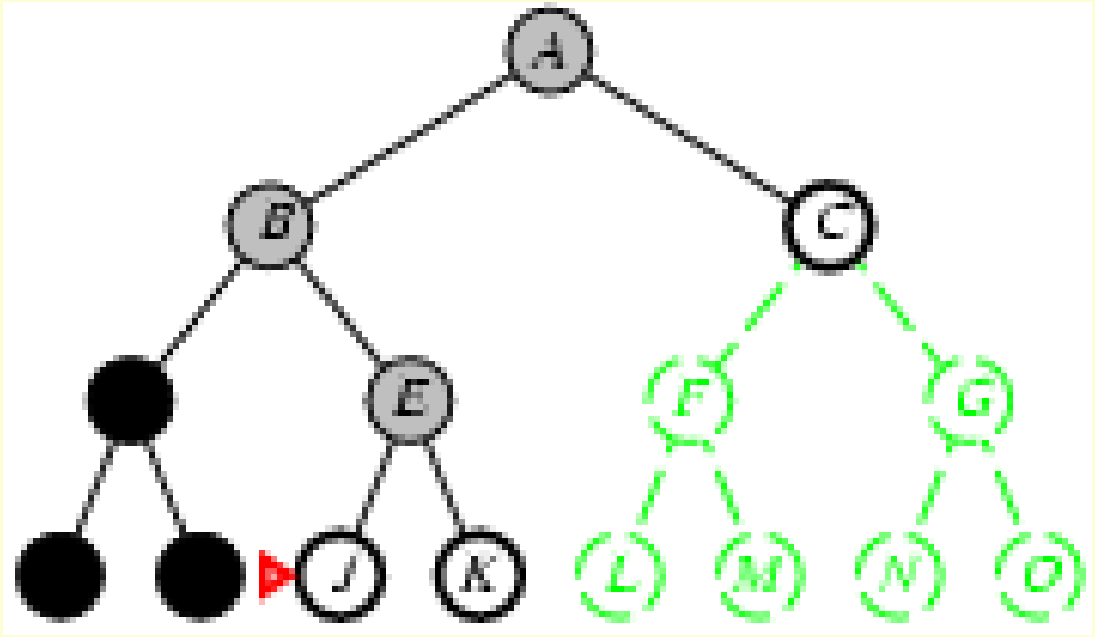
Depth-first search

- Expand deepest unexpanded node
- Implementation:
  - *fringe* = LIFO queue, i.e., put successors at front



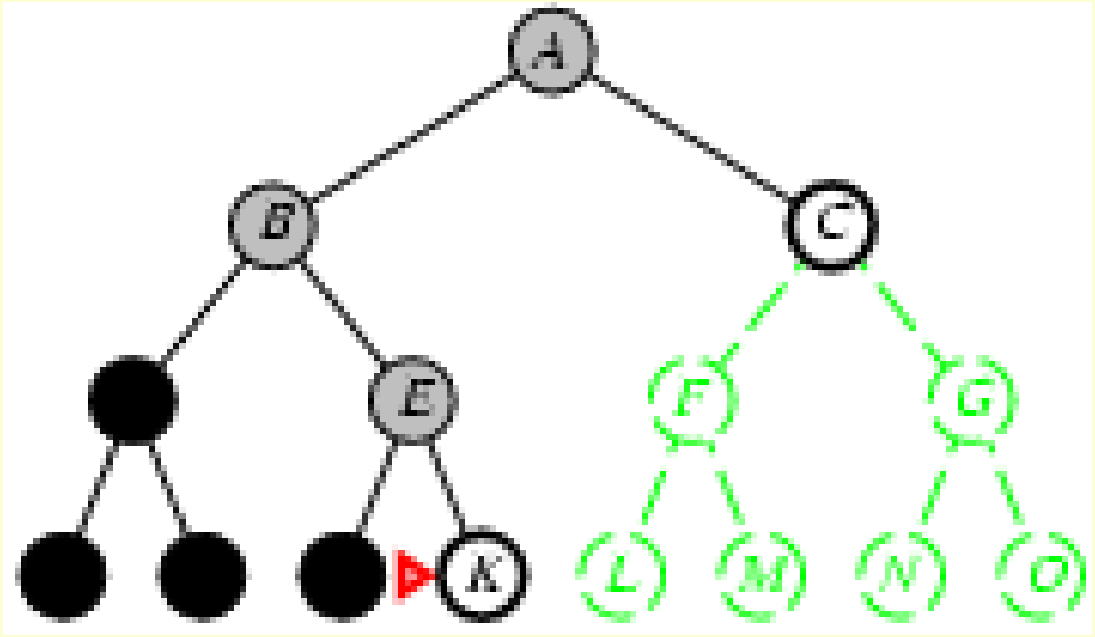
Depth-first search

- Expand deepest unexpanded node
- Implementation:
  - *fringe* = LIFO queue, i.e., put successors at front
  -



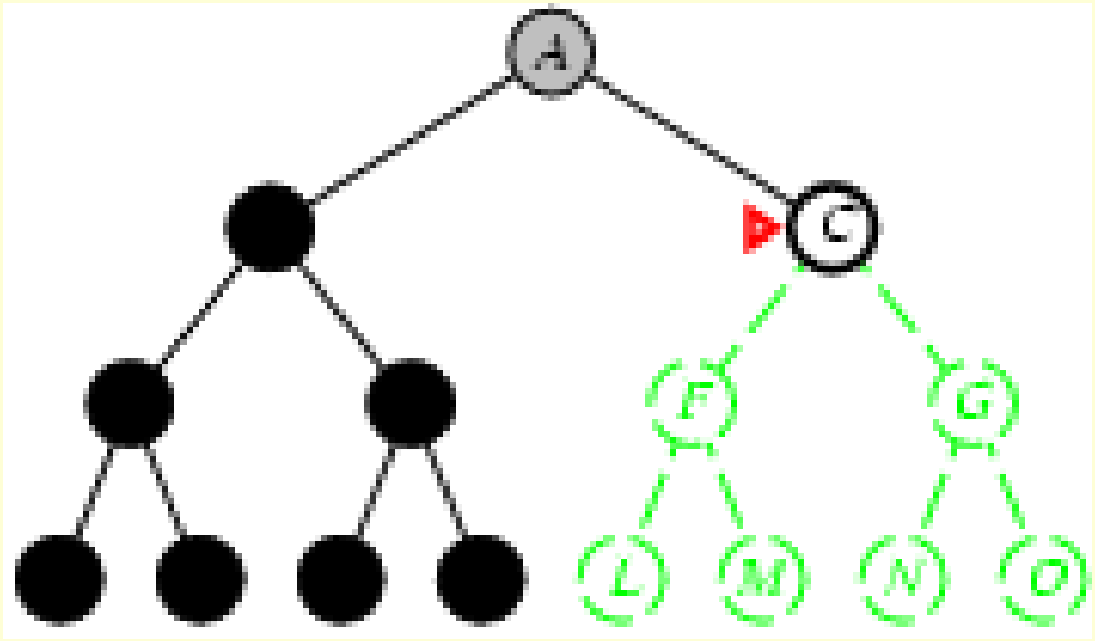
Depth-first search

- Expand deepest unexpanded node
- Implementation:
  - *fringe* = LIFO queue, i.e., put successors at front
  -



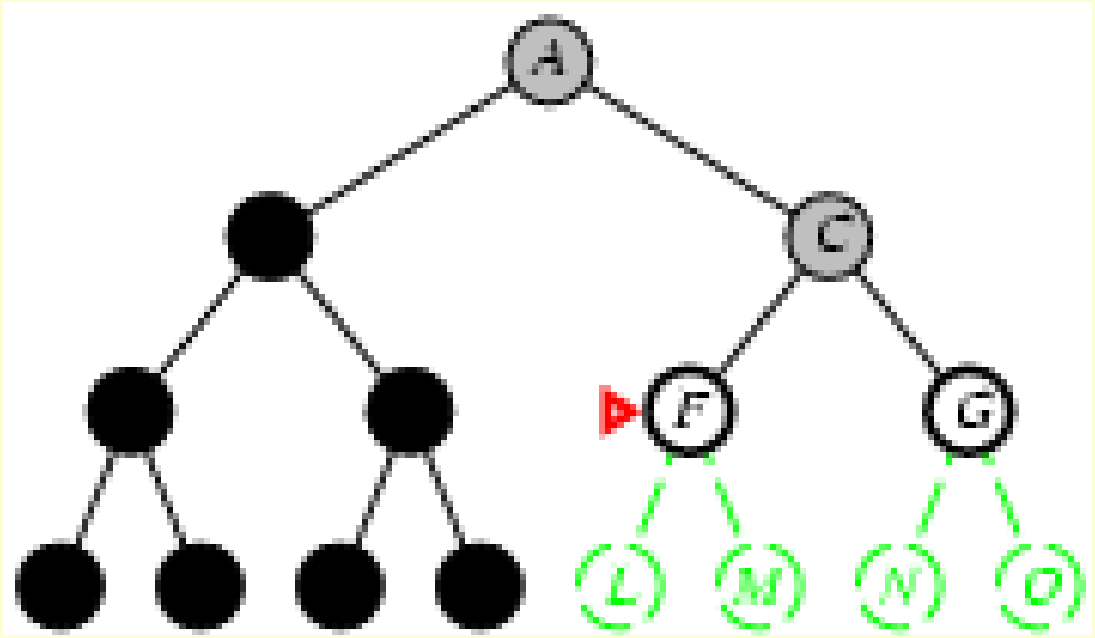
Depth-first search

- Expand deepest unexpanded node
- Implementation:
  - *fringe* = LIFO queue, i.e., put successors at front
  -



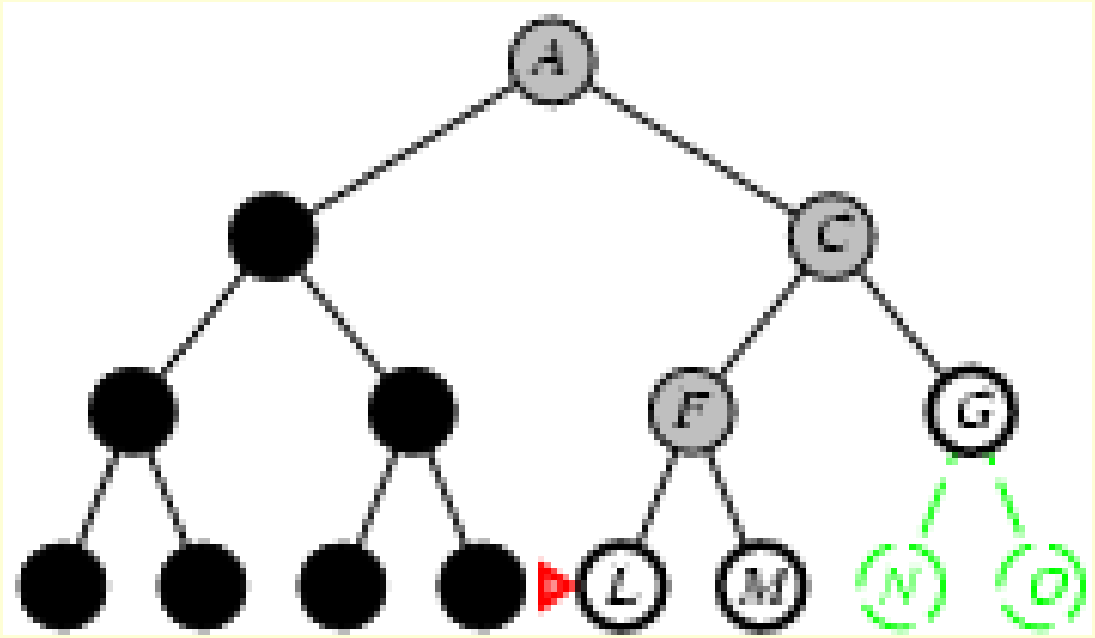
Depth-first search

- Expand deepest unexpanded node
- Implementation:
  - *fringe* = LIFO queue, i.e., put successors at front
  -



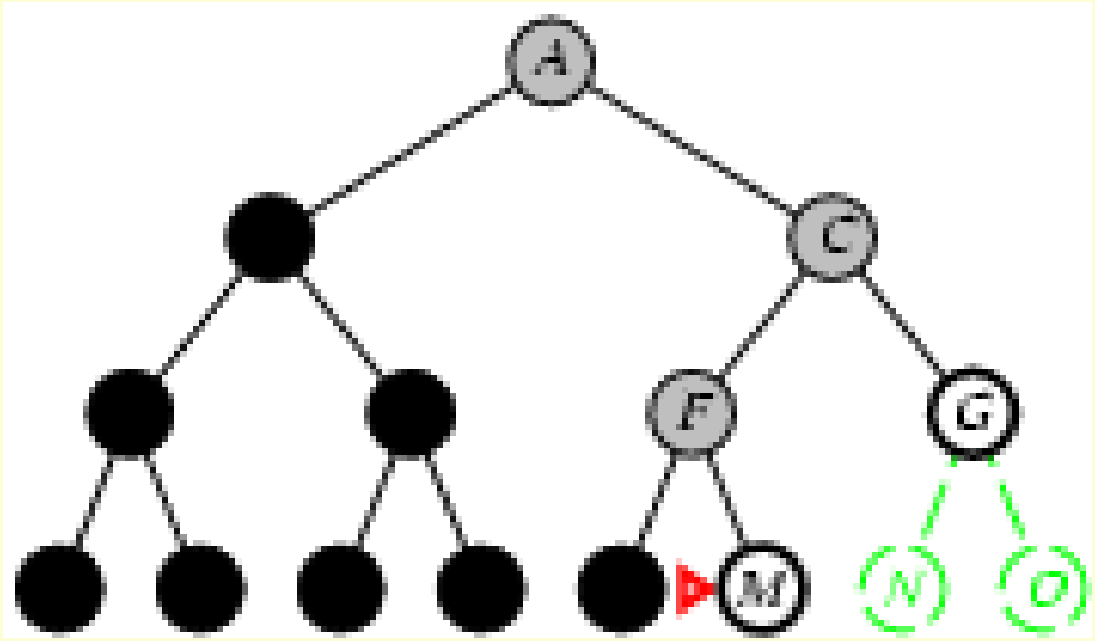
Depth-first search

- Expand deepest unexpanded node
- Implementation:
  - *fringe* = LIFO queue, i.e., put successors at front
  -

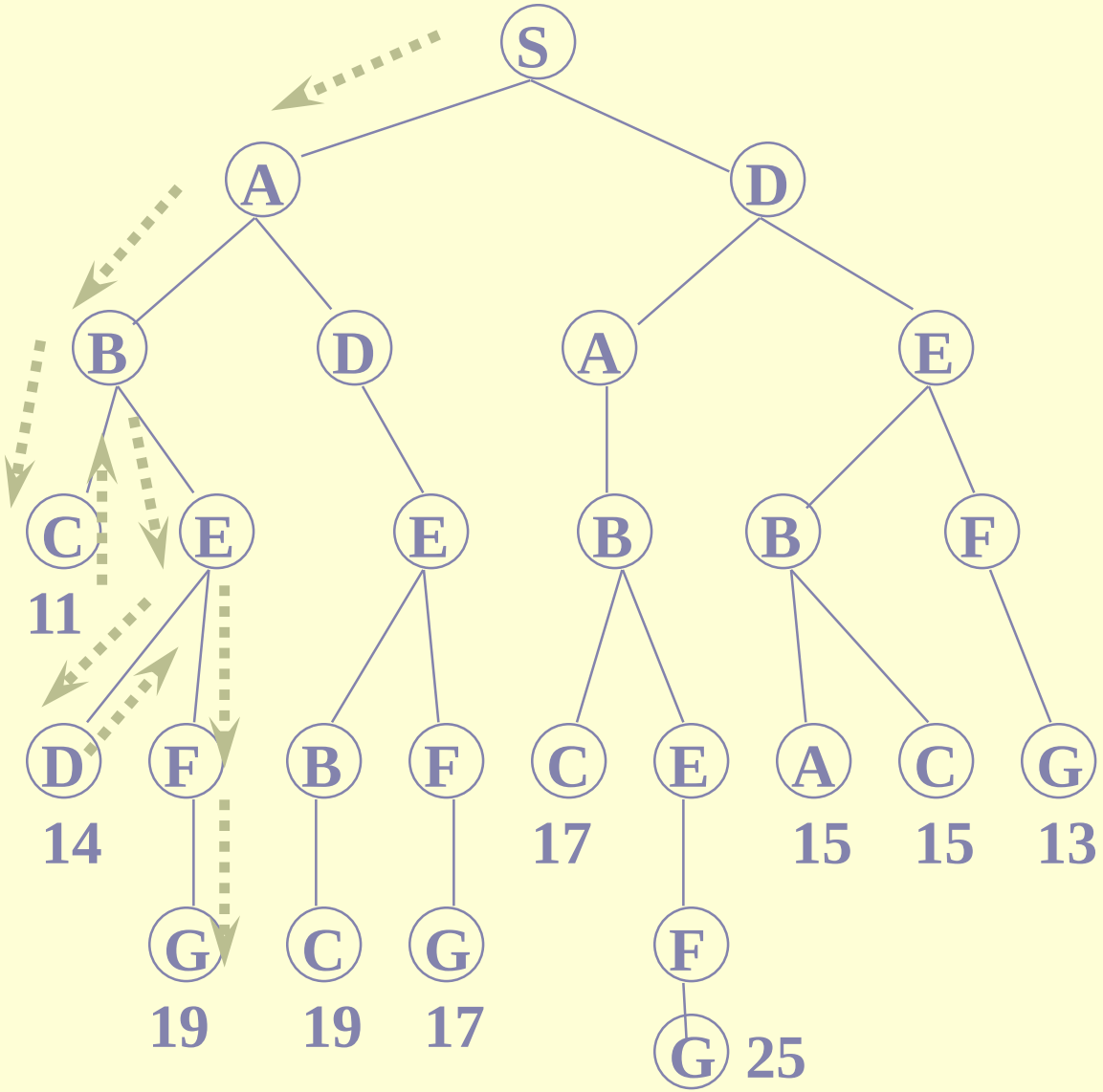


Depth-first search

- Expand deepest unexpanded node
- Implementation:
  - *fringe* = LIFO queue, i.e., put successors at front
  -



Depth-first search



Depth-first search (Analysis)

- Not complete
  - because a path may be infinite or looping
  - then the path will never fail and go back try another option
- Not optimal
  - it doesn't guarantee the best solution
- It overcomes
  - the time and space complexities

Properties of depth-first search

- Complete? No: fails in infinite-depth spaces, spaces with loops
  - Modify to avoid repeated states along path
  - → complete in finite spaces
- Time? $O(b^m)$: terrible if m is much larger than d
  - but if solutions are dense, may be much faster than breadth-first
- Space? $O(bm)$, i.e., linear space!
- Optimal? No

Depth-Limited Strategy

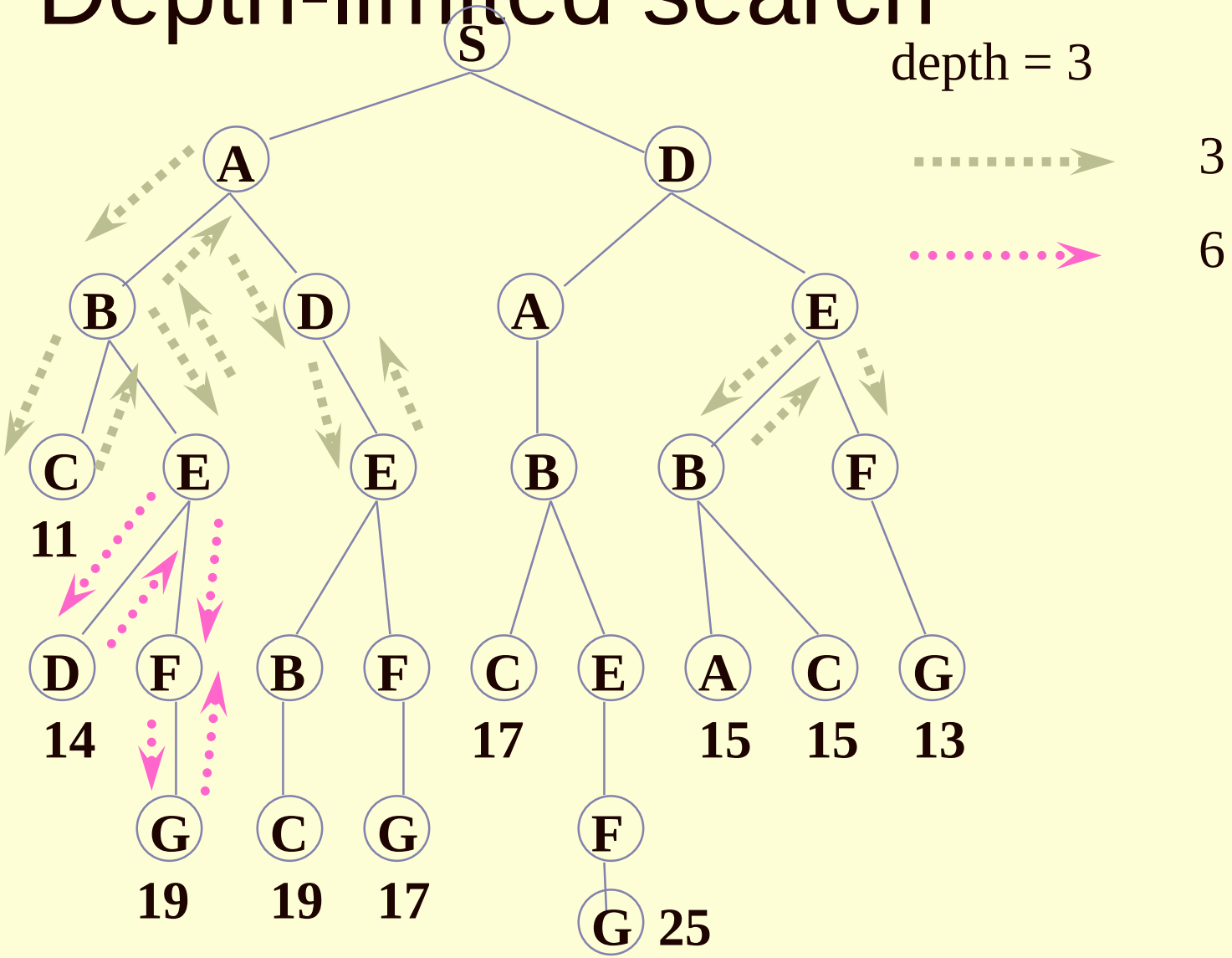
- Depth-first with depth cutoff k (maximal depth below which nodes are not expanded)

- Three possible outcomes:
  - Solution
  - Failure (no solution)
  - Cutoff (no solution within cutoff)

Depth-limited search

- It is depth-first search
  - with a **predefined** maximum depth
  - However, it is usually not easy to define the suitable maximum depth
  - too small → no solution can be found
  - too large → the same problems are suffered from
- Anyway the search is
  - complete
  - but still not optimal

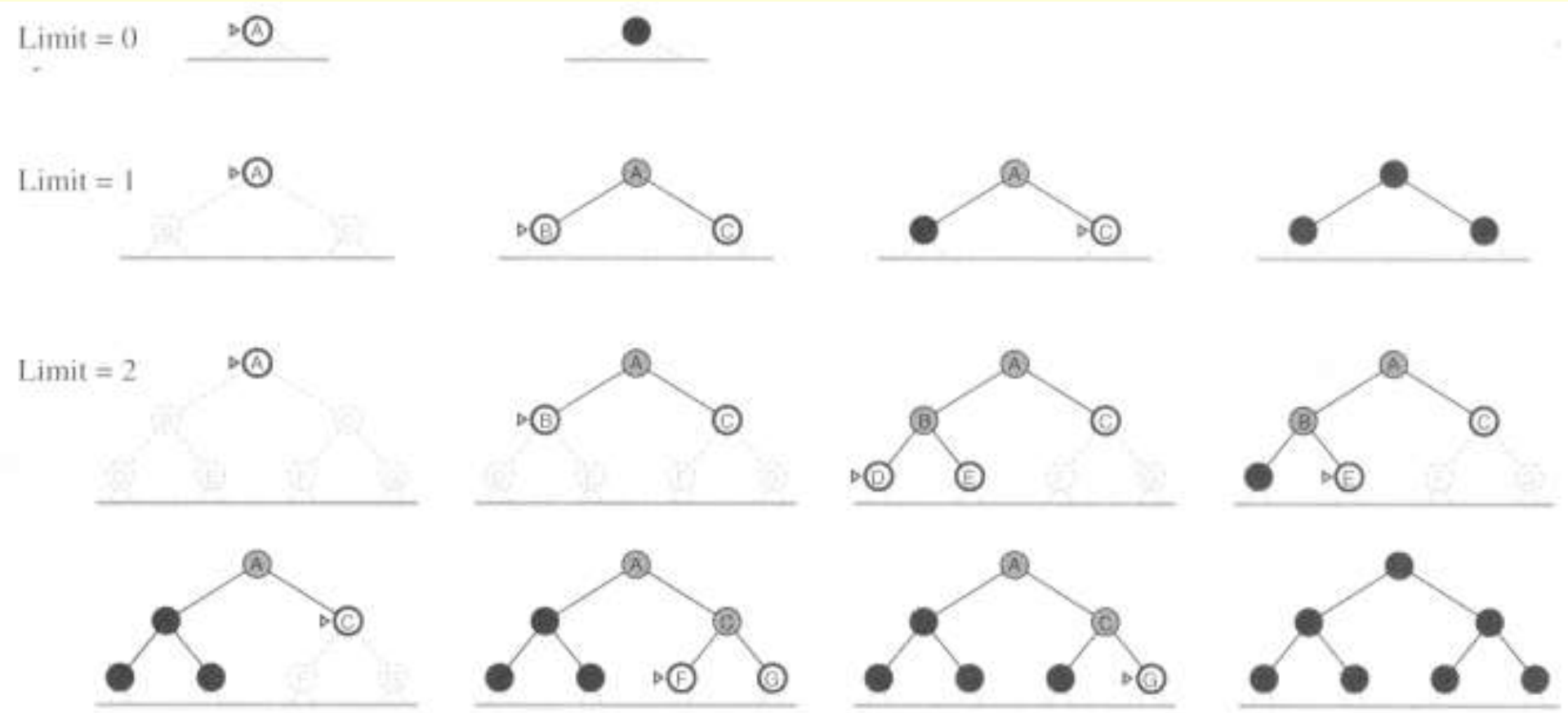
Depth-limited search



Iterative deepening search

- No choosing of the best depth limit
- It tries all possible depth limits:
  - first 0, then 1, 2, and so on
  - combines the benefits of depth-first and breadth-first search

Iterative deepening search



Iterative deepening search (Analysis)

- optimal
- complete
- Time and space complexities
  - reasonable
- suitable for the problem
  - having a large search space
  - and the depth of the solution is not known

Properties of iterative deepening search

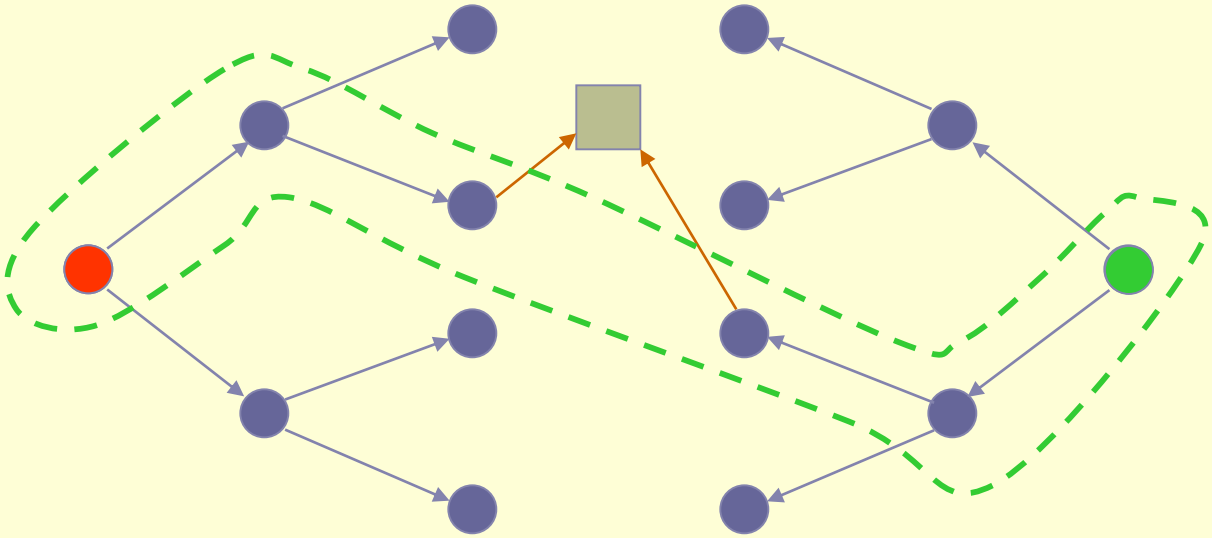
- Complete? Yes
- Time? $(d+1)b^0 + d\,b^1 + (d-1)b^2 + \ldots + b^d = O(b^d)$
- Space? $O(bd)$
- Optimal? Yes, if step cost = 1

Bidirectional search

- Run two simultaneous searches
  - one forward from the initial state another backward from the goal
  - stop when the two searches meet
- However, computing backward is difficult
  - A huge amount of goal states
  - at the goal state, which actions are used to compute it?
  - can the actions be reversible to computer its predecessors?

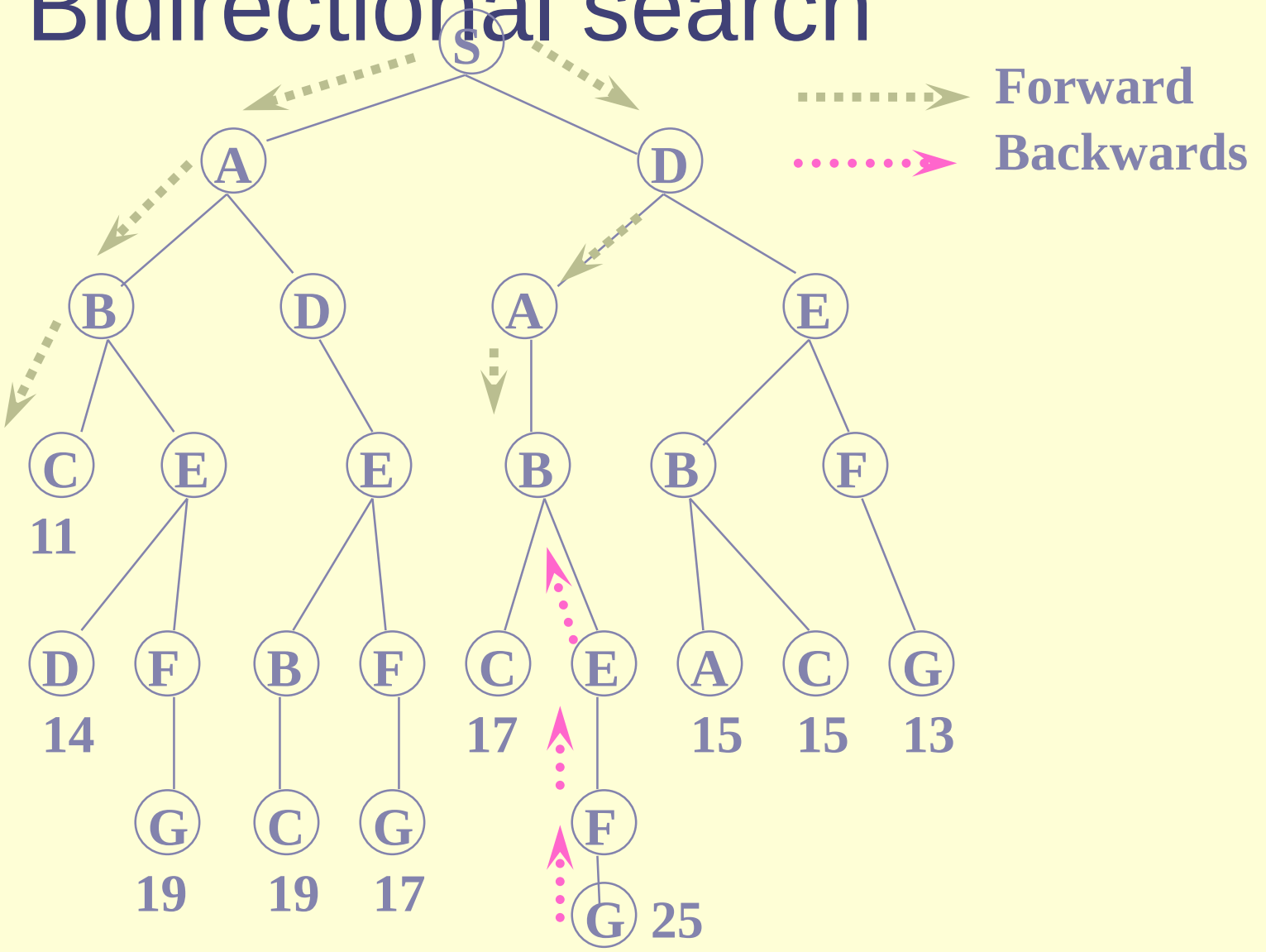
Bidirectional Strategy

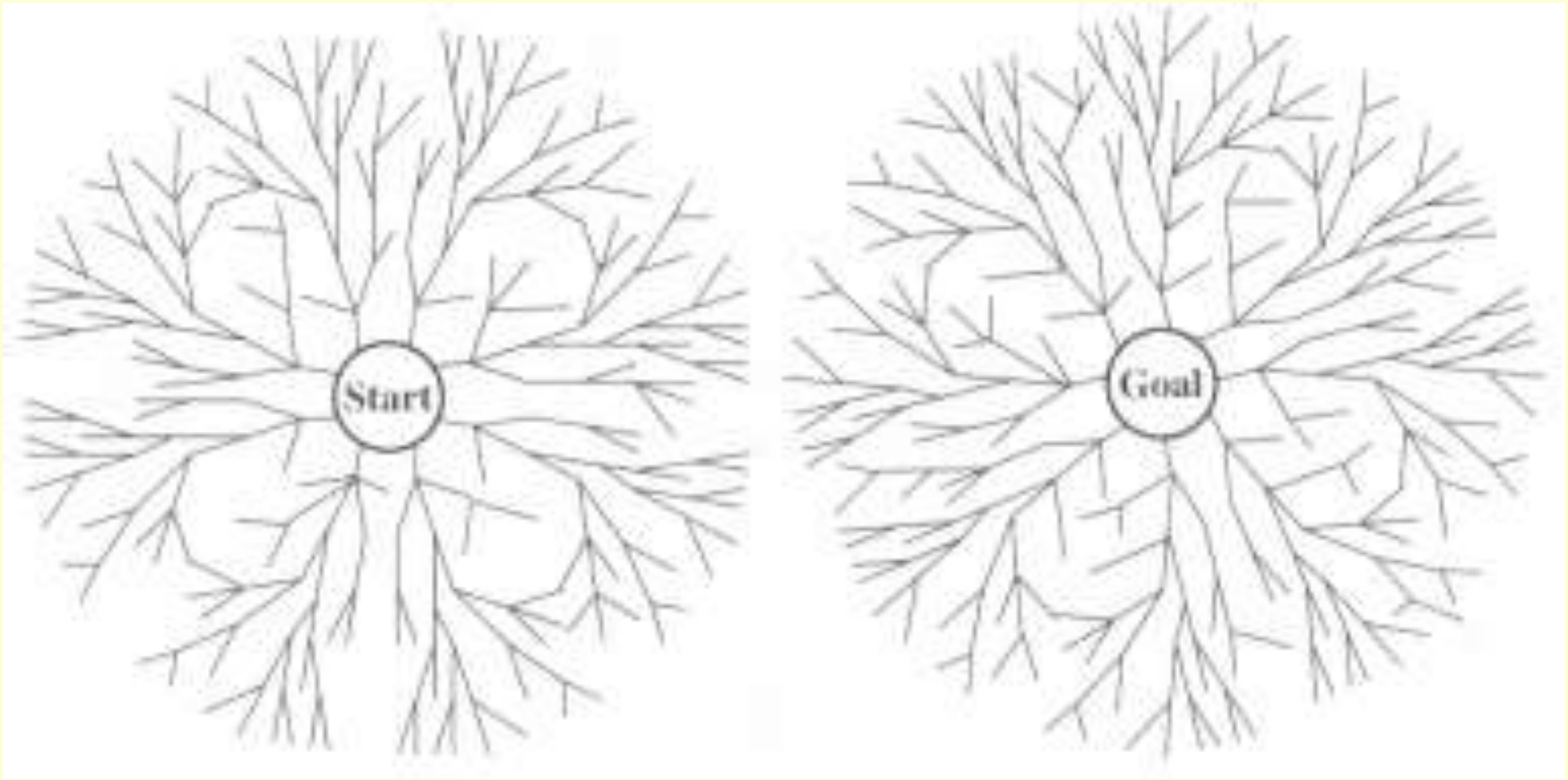
2 fringe queues: FRINGE1 and FRINGE2



Time and space complexity = $O(b^{d/2}) << O(b^d)$

Bidirectional search




Start
Goal

Comparing search strategies

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Figure 3.17 Evaluation of search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: [a] complete if b is finite; [b] complete if step costs $\geq \epsilon$ for positive ϵ; [c] optimal if step costs are all identical; [d] if both directions use breadth-first search.

Uncertainty in the World Model

- *The agent can never be completely certain about the state of the external world since there is ambiguity and uncertainty.*
- Why?
  - sensors have limited precision
    e.g. camera has only so many pixels to capture an image
  - sensors have limited accuracy
    e.g. tachometer's estimate of velocity is approximate
  - there are hidden variables that sensors can't "see"
    e.g. vehicle behind large truck or storm clouds approaching
  - the future is unknown, uncertain, i.e. cannot foresee all possible future events which may happen

Rules and Uncertainty

- Say we have a rule:

  if toothache then problem is cavity

- But not all patients have toothaches due to cavities

  so we could set up rules like:

  if toothache and not(gum disease) and not(filling) and …
  then problem = cavity

- **This gets complicated, a better method would be:**

  if toothache then problem is cavity with 0.8 probability

  or $P(cavity \mid toothache) = 0.8$

  the probability of cavity is 0.8 given toothache is all that is known

Uncertainty in the World Model

- True uncertainty: **rules *are* probabilistic in nature**
  - rolling dice, flipping a coin?
- Laziness: **too hard to determine exceptionless rules**
  - takes too much work to determine all of the relevant factors
  - too hard to use the enormous rules that result
- Theoretical ignorance: **don't know all the rules**
  - problem domain has no complete theory (medical diagnosis)
- Practical ignorance: **do know all the rules** BUT
  - haven't collected all relevant information for a particular case

Logics

- *Logics are characterized by what they commit to as "primitives".*

Logic	What Exists in World	Knowledge States
Propositional	**facts**	**true/false/unknown**
First-Order	**facts, objects, relations**	**true/false/unknown**
Temporal	**facts, objects, relations, times**	**true/false/unknown**
Probability Theory	**facts**	**degree of belief 0..1**
Fuzzy	**degree of truth**	**degree of belief 0..1**

Probability Theory

- *Probability theory serves as a formal means*
  - *for representing and reasoning with uncertain knowledge*
  - *of manipulating degrees of belief in a proposition (event, conclusion, diagnosis, etc.)*

Utility Theory

- Every state has a degree of usefulness or utility and the agent will prefer states with higher utility.

Decision Theory

- *An agent is rational if and only if it chooses the action that yields the highest expected utility, averaged over all the possible outcomes of the action.*

Decision theory = probability theory + utility theory

Baye's Theorem

Let X and Y be a pair of random variables. Their joint probability, $P(X = x, Y = y)$, refers to the probability that variable X will take on the value x and variable Y will take on the value y. A conditional probability is the probability that a random variable will take on a particular value given that the outcome for another random variable is known. For example, the conditional probability $P(Y = y|X = x)$ refers to the probability that the variable Y will take on the value y, given that the variable X is observed to have the value x. The joint and conditional probabilities for X and Y are related in the following way:

$$P(X, Y) = P(Y|X) \times P(X) = P(X|Y) \times P(Y).$$

Rearranging the last two expressions in Equation 5.9 leads to the following formula, known as the Bayes theorem:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}.$$

$$P(A \wedge B) = P(A|B)P(B)$$
$$P(B \wedge A) = P(B|A)P(A)$$

$$P(B|A)P(A) = P(A|B)P(B)$$

$$\mathbf{P(B|A) = \frac{P(A|B)P(B)}{P(A)}}$$

Bayes' law (also Bayes' law or Bayes' rule) is fundamental to probabilistic reasoning in AI!!

Bayes' in Action

- Bayes' rule requires three terms - a conditional probability and two unconditional probabilities - just to compute one conditional probability.
- Bayes' rule is useful in practice because there are many cases where we do have good probability estimates for these three numbers and need to compute the fourth.
- In a task such as medical diagnosis, we often have conditional probabilities on causal relationships and want to derive a diagnosis.

Bayes' in Action

Example:

A doctor knows that the disease meningitis causes the patient to have a stiff neck, say, 50% of the time. The doctor also knows some unconditional facts: the prior probability that a patient has meningitis is 1/50,000, and the prior probability that any patient has a stiff neck is 1/20. Let *s be the proposition that the patient has a stiff neck and m be the proposition that the patient has meningitis.*

Solution:

- This question can be answered by using the well-known Bayes' theorem.

Solution:

$$P(s|m) = 0.5$$

$$P(m) = 1/50000$$

$$P(s) = 1/20$$

$$P(m|s) = \frac{P(s|m)P(m)}{P(s)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002 .$$

Example

Consider a football game between two rival teams: Team 0 and Team 1. Suppose Team 0 wins 65% of the time and Team 1 wins the remaining matches. Among the games won by Team 0, only 30% of them come from playing on Team 1 's football field. On the other hand, 75% of the victories for Team 1 are obtained while playing at home. If Team 1 is to host the next match between the two teams, which team will most likely emerge as the winner?

Solution:

This question can be answered by using the well-known Bayes' theorem.

let X be the random variable that represents the team hosting the match and Y be the random variable that represents the winner of the match. Both X and Y can take on values from the set $\{0, 1\}$. We can summarize the information given in the problem as follows:

Probability Team 0 wins is $P(Y = 0) = 0.65$.
Probability Team 1 wins is $P(Y = 1) = 1 - P(Y = 0) = 0.35$.
Probability Team 1 hosted the match it won is $P(X = 1|Y = 1) = 0.75$.
Probability Team 1 hosted the match won by Team 0 is $P(X = 1|Y = 0) = 0.3$.

Our objective is to compute $P(Y = 1|X = 1)$, which is the conditional probability that Team 1 wins the next match it will be hosting, and compares it against $P(Y = 0|X = 1)$. Using the Bayes theorem, we obtain

$$
\begin{aligned}
P(Y=1|X=1) &= \frac{P(X=1|Y=1)\times P(Y=1)}{P(X=1)} \\
&= \frac{P(X=1|Y=1)\times P(Y=1)}{P(X=1,Y=1)+P(X=1,Y=0)} \\
&= \frac{P(X=1|Y=1)\times P(Y=1)}{P(X=1|Y=1)P(Y=1)+P(X=1|Y=0)P(Y=0)} \\
&= \frac{0.75\times 0.35}{0.75\times 0.35+0.3\times 0.65} \\
&= 0.5738,
\end{aligned}
$$

where the law of total probability (see Equation C.5 on page 722) was applied in the second line. Furthermore, $P(Y=0|X=1)=1-P(Y=1|X=1)=0.4262$. Since $P(Y=1|X=1)>P(Y=0|X=1)$, Team 1 has a better chance than Team 0 of winning the next match.

Example 19

In a factory which manufactures bolts, machines A, B and C manufacture respectively 25%, 35% and 40% of the bolts. Of their outputs, 5, 4 and 2 percent are respectively defective bolts. A bolt is drawn at random from the product and is found to be defective. What is the probability that it is manufactured by the machine B?

Let

A : bolt manufactured from machine A

B : bolt manufactured from machine B

C : bolt manufactured from machine C

D : bolt is defective

We need to find the Probability that the bolt is manufactured by machine B, if it is defective

i.e. P(B | D)

So, $$\mathbf{P(B|D)} = \frac{P(B).P(D|B)}{P(A).\ P(D|A) + P(B).\ P(D|B) + P(C).\ P(D|C)}$$

P(A) = Probability that the bolt is made by machine A	P(B) = Probability that the bolt is made by machine B	P(C) = Probability tha the bolt is made by machine C
$= 25\% = \frac{25}{100} = \mathbf{0.25}$	$= 35\% = \frac{35}{100} = 0.35$	$= 40\% = \frac{40}{100} = 0.40$
$\boldsymbol{P(D|A)}$ = Probability of a defective bolt from machine A	$\boldsymbol{P(D|B)}$ = Probability of a defective bolt from machine B	$\boldsymbol{P(D|C)}$ = Probabilit of a defective bolt from machine C
$= 5\% = \frac{5}{100} = \mathbf{0.05}$	$= 4\% = \frac{4}{100} = \mathbf{0.04}$	$= 2\% = \frac{2}{100} = \mathbf{0.02}$

Putting Values in formula,

$$P(B|D) = \frac{0.35 \times 0.04}{0.25 \times 0.05 + 0.35 \times 0.04 + 0.04 \times 0.02}$$

$$= \frac{0.014}{0.0125 + 0.014 + 0.008}$$

$$= \frac{0.014}{0.0345}$$

$$= \frac{140}{345}$$

$$= \frac{28}{69}$$

Therefore, required probability is $\frac{28}{69}$

Using Bayes' Theorem More Realistically

- Bayes'ian updating

$$P(Cavity \mid Toothache \wedge Catch) = P(Toothache \wedge Catch \mid Cavity)\ P(Cavity)$$