Introduction to Agents

John Lloyd

School of Computer Science
College of Engineering and Computer Science
Australian National University

Topics

- Agents and agent architectures
- Historical issues
- Philosophical issues

Reference:
Artificial Intelligence – A Modern Approach, S. Russell and P. Norvig, Prentice Hall, 2nd Edition, 2003.
Chapters 1, 2, 26, 27

Overview

- These lectures introduce the field of artificial intelligence as being that of the *construction of rational agents*

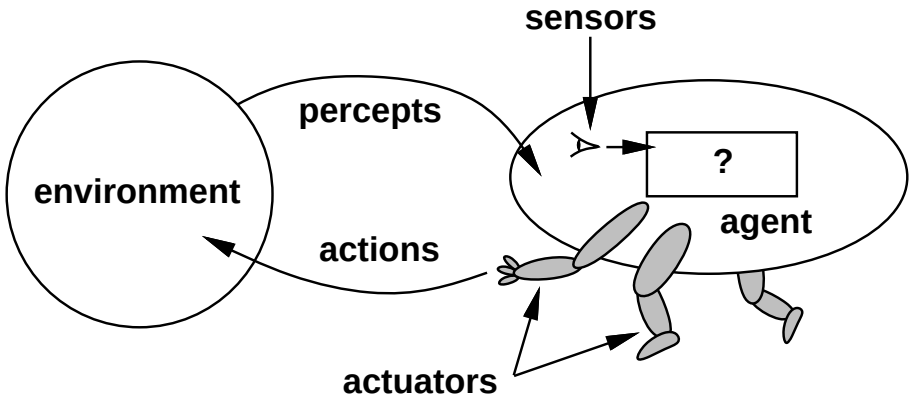
- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators

- A rational agent is one that maximizes its performance according to some performance measure.
  A rational agent *does the right thing*

- Agent applications are extremely diverse, from robots to software agents whose environment is the Internet

- There is now developing an agent-based approach to software engineering (that generalises object-oriented software engineering)

Agents and Environments



Agents interact with environments through sensors and actuators

Agent Function

A *percept* refers to the agent's perceptual inputs at any given instant

A *percept sequence* is the complete history of everything the agent has ever perceived

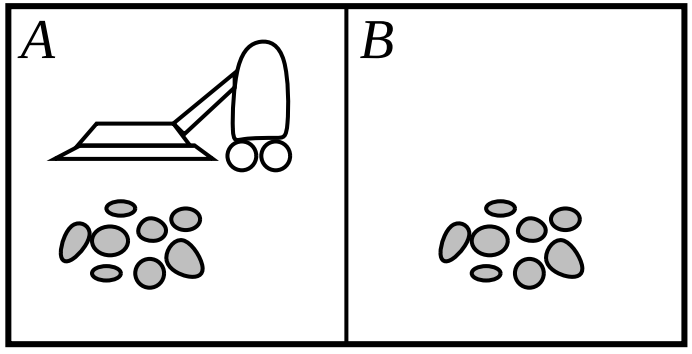
In general, an agent's choice of action at any given instant can depend on the entire percept sequence observed to date

An agent's behaviour is described by the *agent function* that maps any given percept sequence to an action

The agent function is implemented by an *agent program*

The agent function is an abstract mathematical description; the agent program is a concrete implementation of the agent function running on the agent architecture

Vacuum-cleaner world



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

Actions: $Left$, $Right$, $Suck$, $NoOp$

A vacuum-cleaner agent

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

function Reflex-Vacuum-Agent([*location*,*status*]) **returns** an action

if *status* = *Dirty* **then return** *Suck*
else if *location* = *A* **then return** *Right*
else if *location* = *B* **then return** *Left*

Rationality

A rational agent does the right thing – to define the 'right thing' we need a performance measure

A *performance measure* embodies the criterion for success of an agent's behaviour

Typically, a performance measure is objectively imposed by the agent's designer

As a general rule, it is better to design performance measures according to what one actually wants in the environment, rather than according to how one thinks the agent should behave

Utility is a way of accounting for how desirable a particular state of the environment is and can therefore be used as a performance measure

One important rationality principle is *Maximum Expected Utility*, that is, select an action that maximises the agent's expected utility

Rationality 2

What is rational at any given time depends on four things:

- The performance measure that defines the criterion of success
- The agent's prior knowledge of the environment
- The actions that the agent can perform
- 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 maximise its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

Omniscience, Learning and Autonomy

Rationality is not the same as omniscience – an omniscient agent knows the actual outcome of its actions and can act accordingly (impossible in practice)

Rationality is not the same as perfection – rationality maximises expected performance; whereas perfection maximises actual performance

Rationality requires the agent to learn as much as possible from its percept sequence – adaptive behaviour is extremely important in many agent applications

A rational agent should be autonomous, that is, it should not solely rely on the prior knowledge provided by the agent designer – it should learn what it can from the environment to compensate for partial or incorrect knowledge, and/or changing circumstances

Properties of Task Environments

- Fully observable vs. partially observable
  If the agent's sensors give it access to the complete state of the environment at each point in time, then we say the task environment is fully observable

- Deterministic vs. stochastic
  If the next state of the environment is completely determined by the current state and the action executed by the agent, then we say the environment is deterministic; otherwise, it is stochastic

- Episodic vs. sequential
  If the the agent's experience is divided into atomic episodes, then we say the task environment is episodic; otherwise, it is sequential

Properties of Task Environments

- Static vs. dynamic
  If the environment can change while the agent is deliberating, then we say the task environment is dynamic; otherwise, it is static
  If the environment itself does not change with the passage of time but the agent's performance score does, then we say the task environment is semi-dynamic

- Discrete vs. continuous
  The discrete/continuous distinction can be applied to the state of the environment, to the way time is handled, and to the percepts and actions of the agent

- Single agent vs. multi-agent
  If other agents can be identified in the environment or if the agent itself consists of several (sub)agents, then it is a multi-agent task environment

Multi-agent Systems

Some applications can be handled by a single agent, but it is much more common to require a multi-agent system:

Several agents may need to co-operate to achieve some task

Agents may be involved in auctions with other agents

Agents may need to deal with other agents that deliberately try to 'harm' them

Examples:

1. Internet agent that takes part in auctions involving other agents (and people)

2. Swarm of UAVs (unmanned autonomous vehicles) that co-operate to destroy an enemy

Co-operation, coalitions, auctions, negotiation, communication, social ability etc. for multi-agent systems are major agent research issues

Example Environment Types

	Solitaire	Backgammon	Internet shopping	Taxi
Observable	Yes	Yes	No	No
Deterministic	Yes	No	Partly	No
Episodic	No	No	No	No
Static	Yes	Semi	Semi	No
Discrete	Yes	Yes	Yes	No
Single-agent	Yes	No	Yes (except auctions)	No

The environment type largely determines the agent design

The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

Agent Programs

agent = architecture + program

(architecture is physical; agent program implements the agent function)

- Simple reflex agents
- Reflex agents with state
- Goal-based agents
- Utility-based agents

All these can be turned into *learning agents* (or *adaptive agents*) by adding a learning component

Table-driven Agent

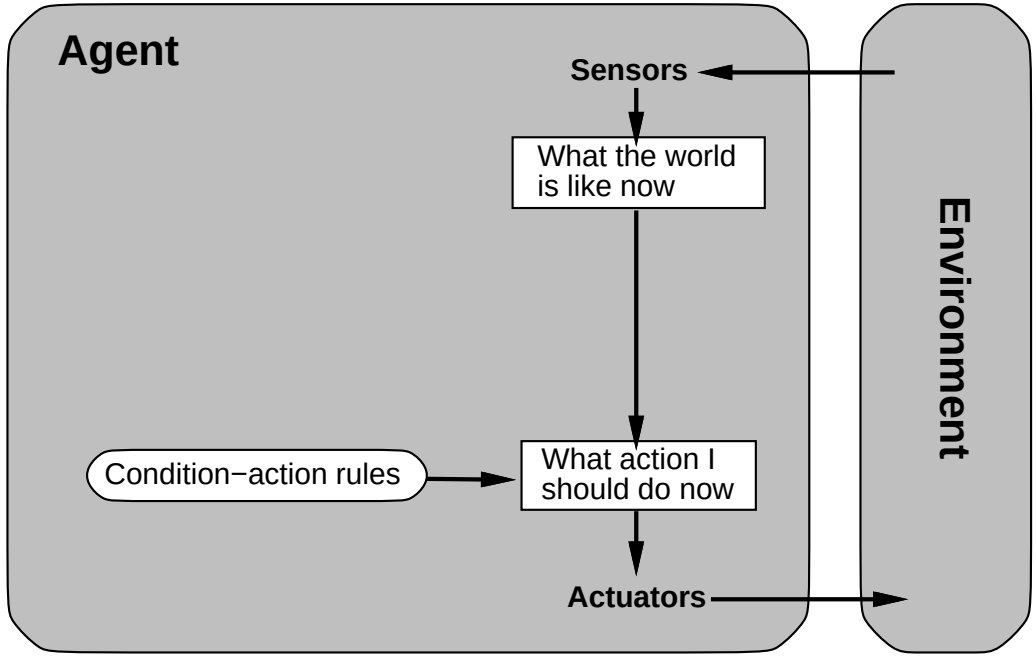
function Table-Driven-Agent(*percept*) **returns** an action
  static : *percepts*, a sequence of percepts, initially empty
    table, a table of actions, indexed by percept sequences, initially fully specified

  append *percept* to the end of *percepts*
  action ← Lookup(*percepts*, *table*)
  return *action*

Except for the most trivial of tasks, the table-driven approach is utterly infeasible because of the size of the table

We want to construct agents that are rational using small amounts of code (not gigantic tables)

Simple Reflex Agent



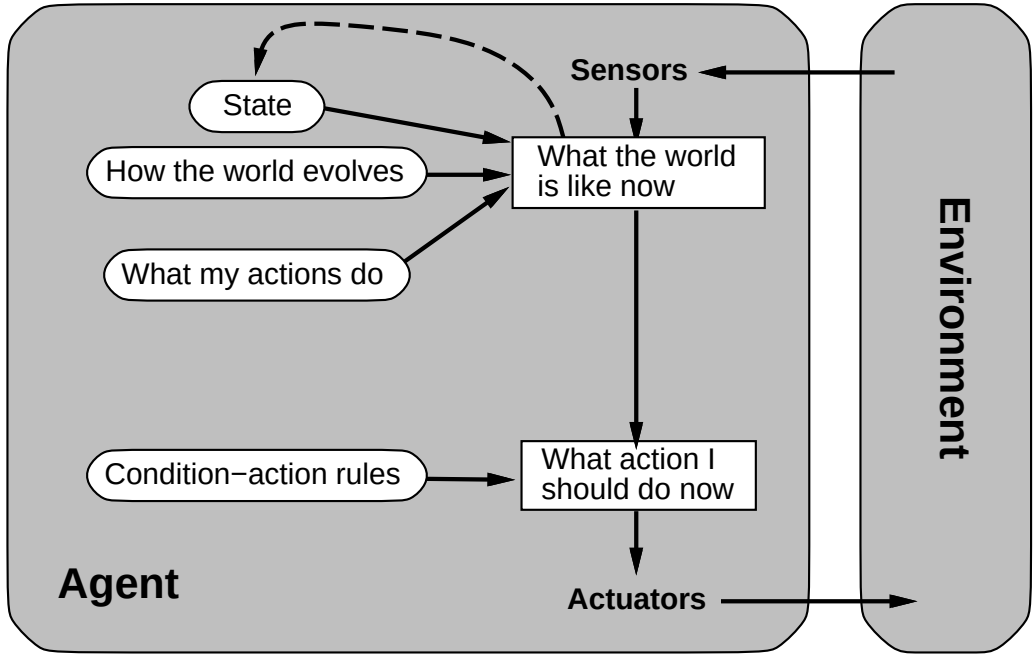
Simple Reflex Agent 2

function Simple-Reflex-Agent(*percept*) **returns** an action
  static : *rules*, a set of condition-action rules

  state ← Interpret-Input(*percept*)
  rule ← Rule-Match(*state*, *rules*)
  action ← Rule-Action(*rule*)
  return *action*

A simple reflex agent will work only if the correct decision can be made on the basis of solely the current percept – that is, only if the environment is fully observable

Model-based Reflex Agent

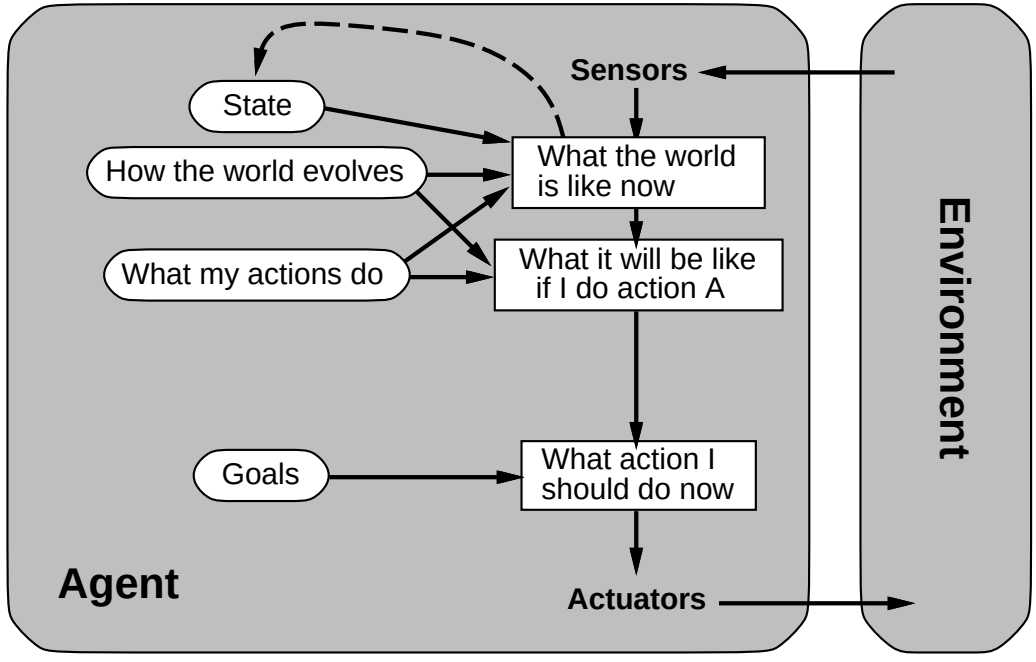


Model-based Reflex Agent 2

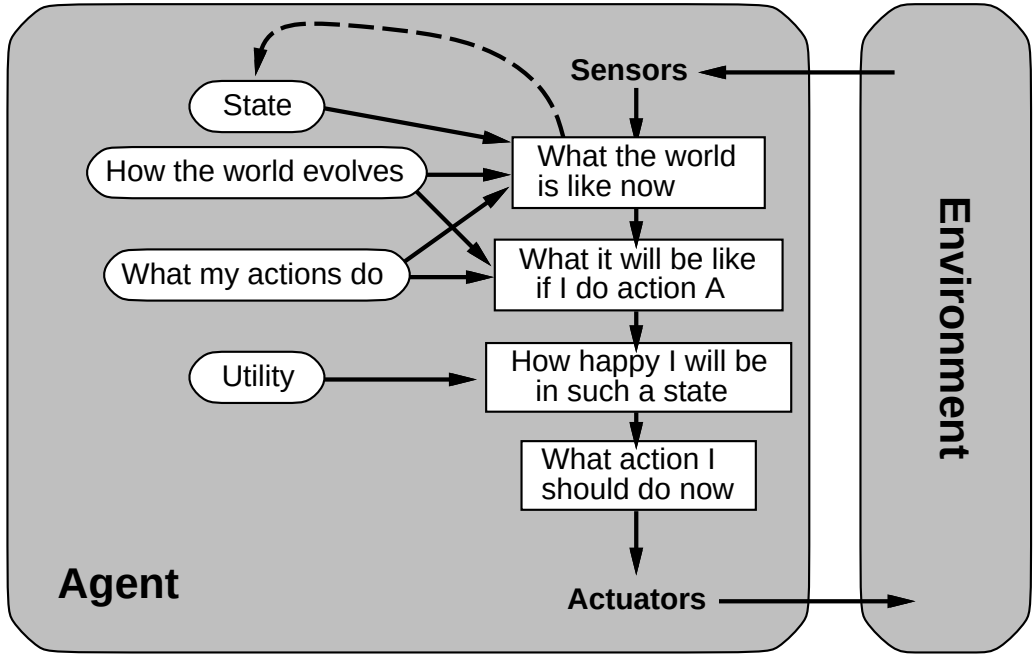
function Reflex-Agent-With-State(*percept*) **returns** an action
  static : *state*, a description of the current world state
    rules, a set of condition-action rules
    action, the most recent action, initially none

  state ← Update-State(*state*, *action*, *percept*)
  rule ← Rule-Match(*state*, *rules*)
  action ← Rule-Action(*rule*)
  return *action*

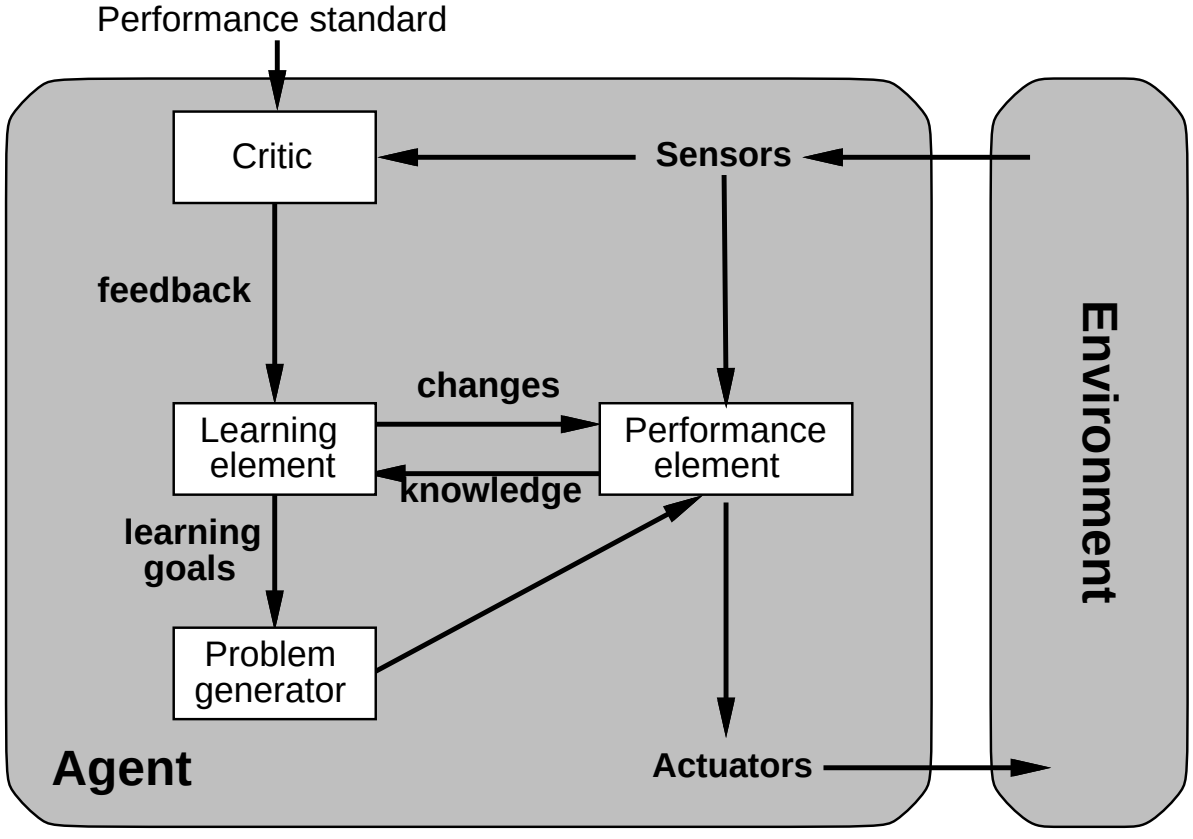
Model-based Goal-based Agent



Model-based Utility-based Agent



Learning Agent



Exploitation vs. Exploration

An important issue for learning agents is *exploitation* versus *exploration*

Exploitation: using what the agent has learned so far to select actions

Exploration: trying actions just to see what happens in the hope of learning more successful behaviours

In practice, agents must do some exploration otherwise they may be stuck in a subset of environment states having low(er) utility

It even makes sense in some applications to choose actions randomly!

Typically, agent explore more in the early stages of deployment and exploit more in later stages

Personalisation

Consider an agent that interacts with a particular user (a *user agent*)

For example, the agent may mediate interactions between the user and the Internet
(Web search, recommenders for TV, movies, etc.)

It is desirable that the agent gets to know the user's interests and preferences

These can be learned from a sequence of training examples obtained by interactions between the user and the agent

BDI Agents

BDI stands for Beliefs, Desires and Intentions

This approach is based on theories of practical reasoning that originated in philosophy, but recently have been taken up by computer scientists as a basis for agent architectures

Beliefs are what the agent believes to be the case – they may not be true! (True beliefs are usually called knowledge)

Desires are states of the environment that the agent would like to achieve

Intentions are desires that the agent is currently actively trying to achieve

BDI Agents 2

There is a theory of practical reasoning (for people) that involves iterating through the following cycle:

get percept
update beliefs
update desires
choose intention
select action
put action

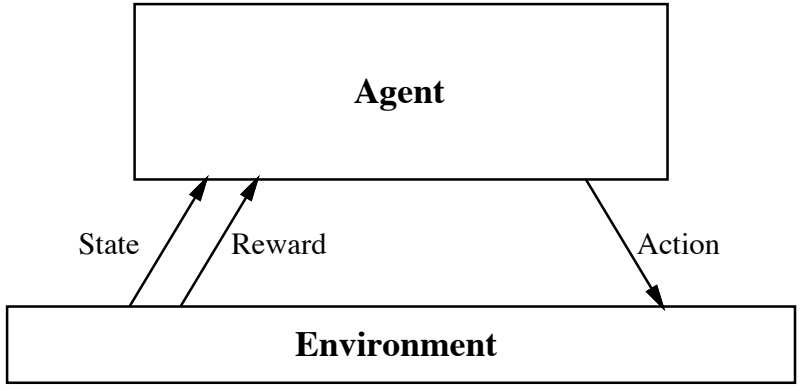
A number of existing widely-used agent platforms are based on the BDI approach

Markov Decision Processes

Assume

- finite set of states S
- set of actions A
- at each discrete time agent observes state $s_t \in S$ and chooses action $a_t \in A$
- then receives immediate reward r_t
- and state changes to s_{t+1}
- Markov assumption: $s_{t+1} = \delta(s_t, a_t)$ and $r_t = r(s_t, a_t)$
  - i.e., r_t and s_{t+1} depend only on *current* state and action
  - functions δ and r may be nondeterministic
  - functions δ and r not necessarily known to agent

Reinforcement Learning



$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \ldots$$

Goal: Learn to choose actions that maximize

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots \text{ , where } 0 \leqslant \gamma < 1$$

Summary

Agents interact with *environments* through *actuators* and *sensors*

The *agent function* describes what the agent does in all circumstances

The *performance measure* evaluates the environment sequence

A *rational* agent maximizes expected performance

Agent programs implement agent functions

Environments are categorized along several dimensions:
observable? *deterministic*? *episodic*? *static*? *discrete*? *single-agent*?

Summary 2

There are several basic agent architectures:
reflex, *reflex with state*, *goal-based*, *utility-based*

Learning can be added to any basic architecture and is indeed essential for satisfactory performance in many applications.

Rationality requires a learning component – it is necessary to know as much about the environment as possible before making a rational decision.

When studying the various subfields (such as knowledge representation and reasoning, planning, learnining, and so on) of AI later, remember to keep in mind the *whole agent* view of AI. The individual subfields are interesting, but it's even more interesting to put them all together into an integrated system.

Outline of History

- 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)

Outline of History 2

- 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)

Fields that AI draws upon

- Philosophy
- Mathematics
- Economics
- Neuroscience
- Psychology
- Computer engineering
- Control theory and cybernetics
- Linguistics

The Turing Test

- Proposed in "Computing Machinery and Intelligence", 1950
- Operational definition of AI
- Program has a conversation (via online typed messages) with an interrogator for five minutes. The interrogator then has to guess if the conversation is with a program or a person; the program passes the test if it fools the interrogator 30% of the time
- Eliza program (Weizenbaum)
- Loebner Prize

Philosophical Issues

- Weak AI hypothesis: machines can act as if they were intelligent
- Strong AI hypothesis: machines can actually be intelligent
- Can a machine be conscious?

Can AI Succeed?

- Many people (especially philosophers and scientists in other fields) have argued against the strong AI hypothesis
- Lucas: Gödel's incompleteness theorem
- Dreyfus: "What Computers Can't Do" "What Computers Still Can't Do"
- Penrose: "The Emperor's New Mind" "Shadows of the Mind"
  An AI system is just a Turing machine; therefore it can never be intelligent/conscious
  People are intelligent/conscious because of some purported quantum gravity mechanism
- Searle's Chinese Room

What If We Do Succeed?

Let an ultraintelligent machine be defined as a machine that can far surpass all the intellectual activities of any man however clever. Since the design of machines is one of these intellectual activities, an ultraintelligent machine could design even better machines; then there would unquestionably be an "intelligence explosion", and the intelligence of man would be left far behind. Thus the first ultraintelligent machine is the *last* invention that man need ever make, provided that the machine is docile enough to tell us how to keep it under control.

I.J. Good (1965)

And Now We Enter The Realm Of Science Fiction ... The Singularity

This "intelligence explosion" is now called the *singularity*.

> "The Singularity" is a phrase borrowed from the astrophysics of black holes. The phrase has varied meanings; as used by Vernor Vinge and Raymond Kurzweil, it refers to the idea that accelerating technology will lead to superhuman machine intelligence that will soon exceed human intelligence, probably by the year 2030. The results on the other side of the "event horizon," they say, are unpredictable.

http://www.kurzweilai.net