ARTIFICIAL INTELLIGENCE & AGENTS

What is AI?

- So, what do you think?

What is AI?

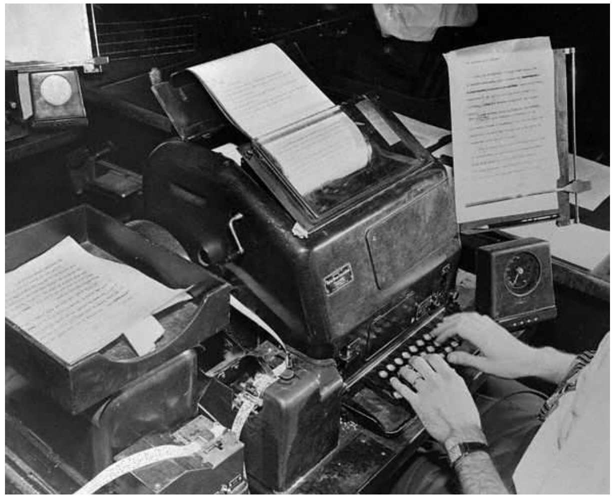
- AI is both science and engineering:
  - the *science* of understanding intelligent entities — of developing theories which attempt to explain and predict the nature of such entities;
  - the *engineering* of intelligent entities.
- Both of these aspects are important to the field.
- They are, to some extent, interdependent.

- Four views of AI:
  1. AI as *acting humanly*
     - as typified by the Turing test
  2. AI as *thinking humanly*
     - cognitive science.
  3. AI as *thinking rationally*
     - as typified by logical approaches.
  4. AI as *acting rationally*
     - the intelligent agent approach.

Acting Humanly

- A problem that has greatly troubled AI researchers: *when can we count a machine as being intelligent*?
- Most famous response due to Alan Turing, British mathematician and computing pioneer:

- Human interrogates entity via teletype for 5 minutes.



- If, after 5 minutes, human cannot tell whether entity is human or machine, then the entity must be counted as intelligent
- (The guy on the right is Joshua Lederberg, another AI pioneer)

- No program has yet passed Turing test!
  (Loebner competition & prize.)
- A program that succeeded would need to be capable of:
  - natural language understanding & generation;
  - knowledge representation;
  - learning;
  - automated reasoning.
- Note no *visual* or *aural* component to basic Turing test — augmented test involves video & audio feed to entity.

Thinking Humanly

- Try to understand how the mind works — how do we think?
- Two possible routes to find answers:
  - by *introspection* — we figure it out ourselves!
  - by *experiment* — draw upon techniques of psychology to conduct controlled experiments. ("Rat in a box"!)
- The discipline of *cognitive science*: particularly influential in *vision*, *natural language processing*, and *learning*.

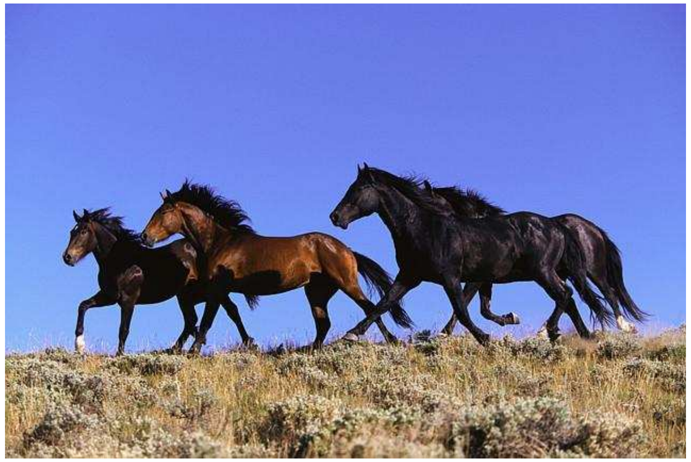
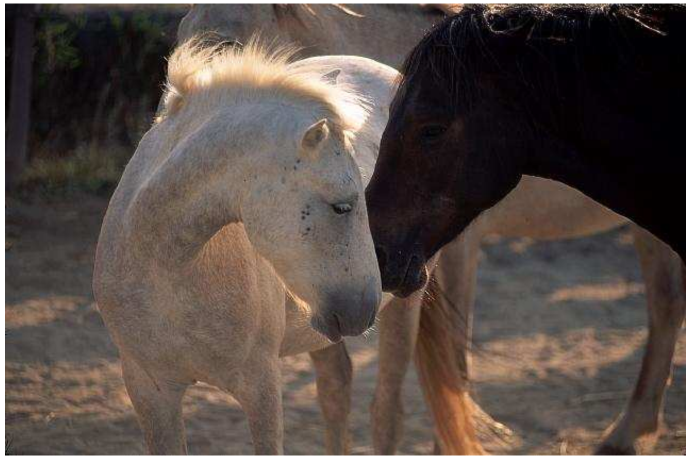
Thinking Rationally

- Trying to understand how we *actually* think is one route to AI — but how about how we *should* think.
- Use *logic* to capture the *laws of rational thought* as *symbols*.
- *Reasoning* involves shifting symbols according to well-defined rules (like algebra).
- Result is *idealised* reasoning.
- Logicist approach is theoretically attractive.

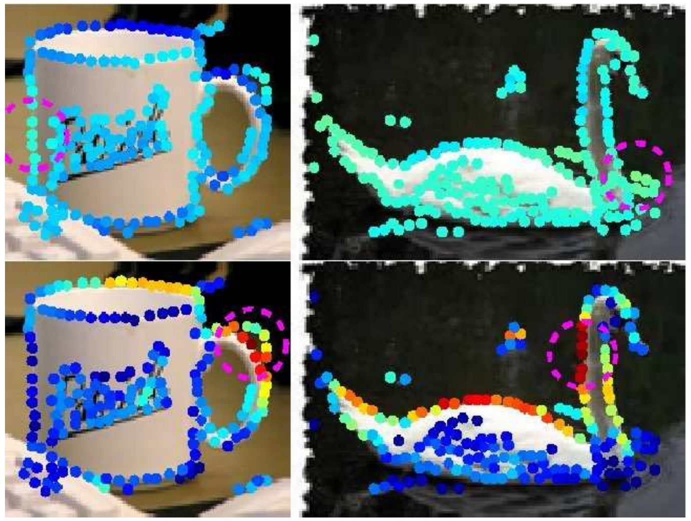
- Lots of problems:
  - *transduction* — how to map the environment to symbolic representation;
  - *representation* — how to represent real world phenomena (time, space, . . .) symbolically;
  - *reasoning* — how to do symbolic manipulation *tractability* — so it can be done by real computers!
- We are still a long way from solving these problems.
- In general logic-based approaches are unpopular in AI at the moment.

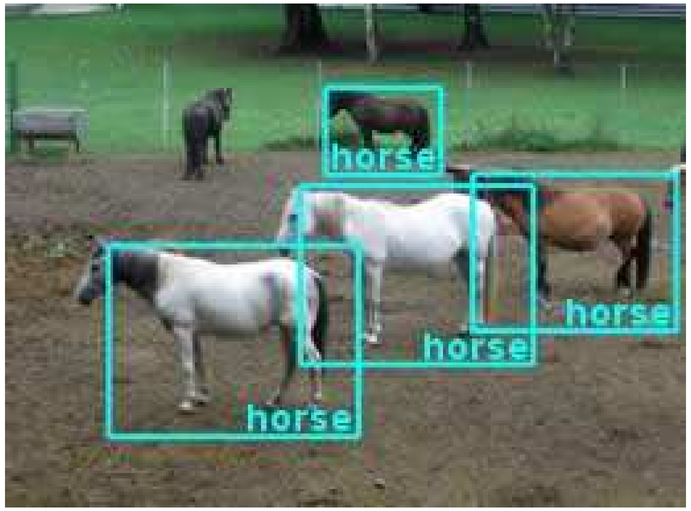
The transduction problem

- How many horses do you see?

- Locating objects in pictures is relatively easy.





- But identifying objects much harder.

Acting Rationally

- Acting rationally = acting to achieve one's goals, given one's beliefs.
- An *agent* is a system that perceives and acts; intelligent agent is one that acts rationally w.r.t. the goals we delegate to it.
- Emphasis shifts from designing *theoretically best* decision making procedure to best decision making procedure possible in circumstances.
- Logic may be used in the service of finding the best action — not an end in itself.

- Achieving *perfect* rationality — making the best decision theoretically possible — is not usually possible, due to *limited resources*:
  - limited time;
  - limited computational power;
  - limited memory;
  - limited or uncertain information about environment.
- The trick is to *do the best with what you've got*!
- This is easier than doing perfectly, but still tough.

Brief History of AI

Gestation, 1943–56:

- McCulloch & Pitts (1943)
  artificial neural net — proved equivalent to Turing machine;
- Shannon, Turing (1950)
  chess playing programs
- Marvin Minsky (1951)
  first neural net computer — SNARC
- Dartmouth College (1956)
  term "AI" coined by John McCarthy
  Newell & Simon presented LOGIC THEORIST program

Early enthusiasm, 1952–69:

- Major centres established:

Minsky, MIT

McCarthy, Stanford

Newell & Simon, CMU

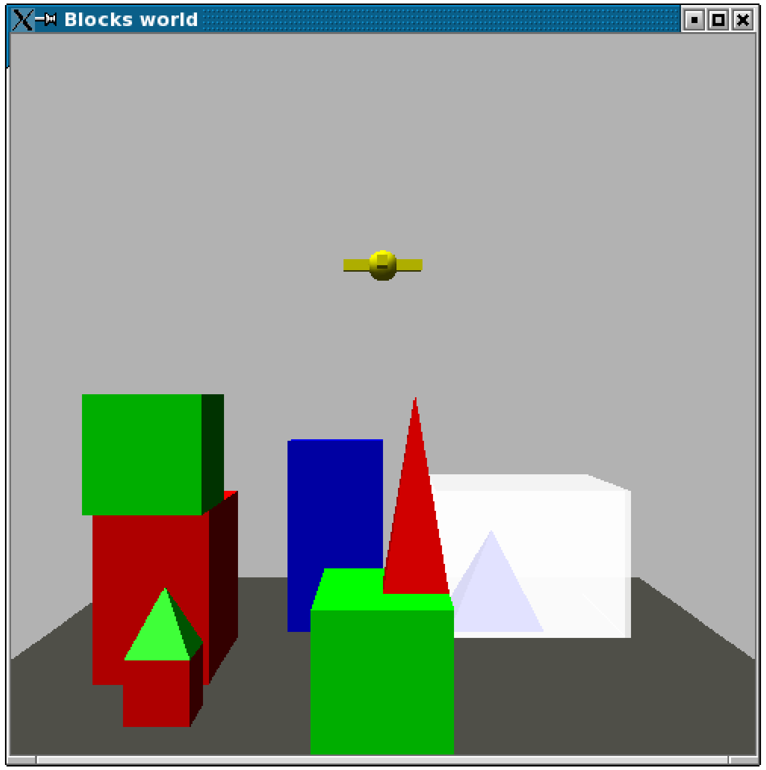
- For the next 20 years these places dominated the field.

- Lots of progress.
- Programs written that could:
  - plan,
  - learn,
  - play games,
  - prove theorems,

  in general, *solve problems*.
- Major feature of the period were *microworlds* — toy problem domains.

  Example: blocks world.

- Planners simulated manipulating simple shapes like this:



- Even some simple natural language handling:

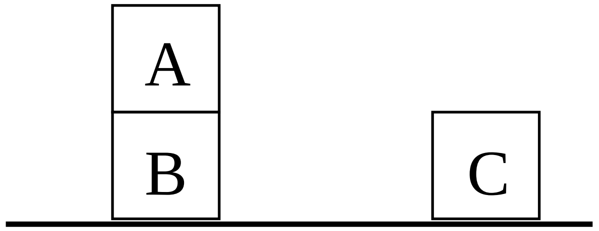
```
Person: GRASP THE PYRAMID.
Computer: I DON'T UNDERSTAND WHICH PYRAMID YOU MEAN.
Person: FIND A BLOCK WHICH IS TALLER THAN THE ONE
        YOU ARE HOLDING AND PUT IT INTO THE BOX.
Computer: BY "IT", I ASSUME YOU MEAN THE BLOCK WHICH
          IS TALLER THAN THE ONE I AM HOLDING.
Computer: OK. (does it)
Person: WHAT DOES THE BOX CONTAIN?
Computer: THE BLUE PYRAMID AND THE BLUE BLOCK.
Person: WHAT IS THE PYRAMID SUPPORTED BY?
Computer: THE BOX.
Person: HOW MANY BLOCKS ARE NOT IN THE BOX?
Computer: FOUR OF THEM.
```

- The promise was "It'll scale, honest. . . "

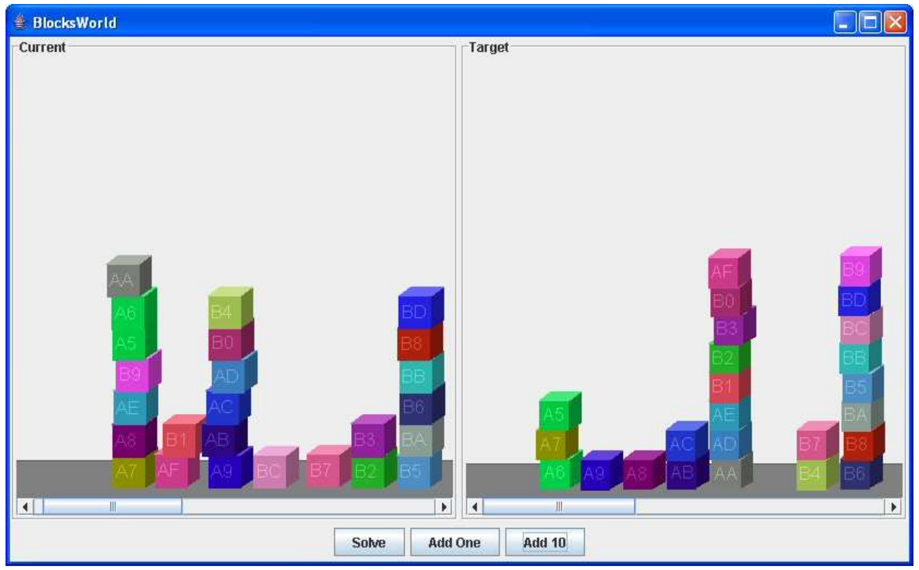
A dose of reality, 1966–1973:

- Techniques developed on microworlds *would not* scale.
- Implications of *complexity theory* — developed in late 1960s, early 1970s — began to be appreciated:
  - *brute force techniques will not work*.
  - *works in principle does not mean works in practice*.
- Lots of early programs did symbol manipulation without any real understanding of the domain they were in.

- Techniques that worked for problems this:



Wouldn't work for problems like this:



Knowledge based systems, 1969–1979:

- General purpose, brute force techniques don’t work, so use *knowledge rich* solutions.
- Early 1970s saw emergence of *expert systems* as systems capable of exploiting knowledge about tightly focussed domains to solve problems normally considered the domain of experts.
- Ed Feigenbaum’s *knowledge principle*:

  > [Knowledge] is power, and computers that amplify that knowledge will amplify every dimension of power

- Expert systems success stories:
  - MYCIN — blood diseases in humans;
  - DENDRAL — interpreting mass spectrometers;
  - R1/XCON — configuring DEC VAX hardware;
  - PROSPECTOR — finding promising sites for mineral deposits;
- Expert systems emphasised *knowledge representation*: rules, frames, semantic nets.
- Problems:
  - the knowledge elicitation bottleneck;
  - marrying expert system & traditional software;
  - breaking into the mainstream.

AI makes money, 1980–present

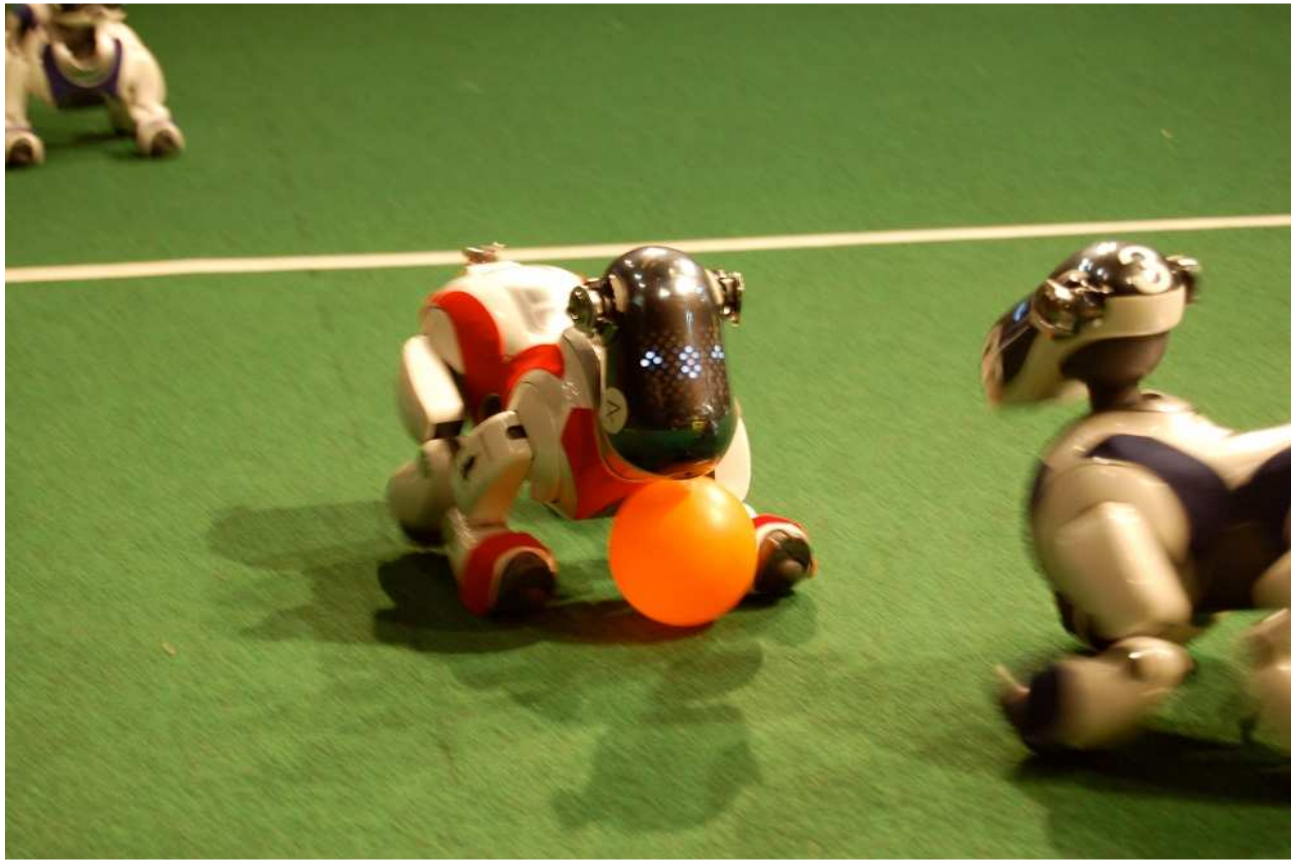
- R1 was the first commercial expert system.
- Led to a boom in expert systems companies.
  - Like an earlier dot com boom
- Most of those companies failed to deliver increased productivity for their customers.
- They went bust.
- AI suffered from all the broken promises, but didn't go away.

- "The best small computer in the world"

AI and the scientific method, 1987–present

- Early AI was all about writing cool programs to demonstrate what could be done.
- About rejecting the limitations of existing fields.
- Now it is more about identifying hypotheses and empirically evaluating them.
  - I can show that my machine learning method will converge by running experiments and showing that it reliably converges.
- Or, alternatively, about proving theorems about techniques.
  - I can show that my machine learning method will converge by proving that it will do so, under reasonably assumptions.
- Distinction between *neat* and *scruffy* AI.

- Connection back to other fields.
  - Probability/statistics
  - Control theory
  - Economics
  - Operations research
- New industries spawned.
  - Data mining is machine learning.
- Low-key successes.
  - Expert systems embedded in Windows
- Increased emphasis on shared datasets, common problems, competitions.



Intelligent agents, 1993–present:

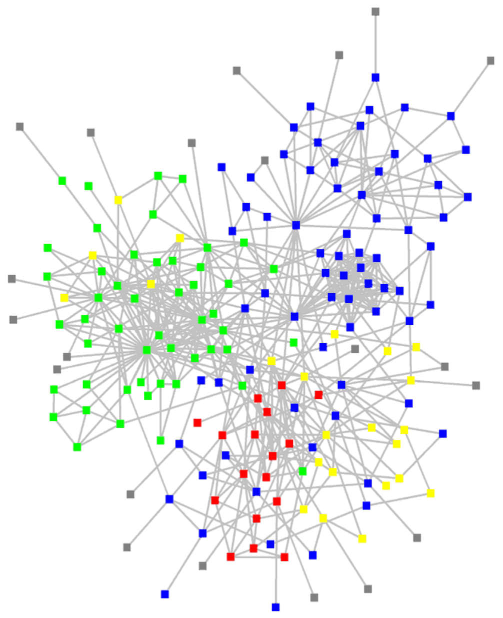
- Emphasis on understanding the interaction between *agents* and *environments*.
- AI as *component*, rather than as end in itself.
  - "Useful first" paradigm — Etzioni (NETBOT, US$35m)
  - "Raisin bread" model — Winston.
- More about interaction between components, emergent intelligence, and doing well enough to be useful.
- We will concentrate on this view in the class.

- Good enough to be useful.

Large datasets, 2001–present

- Change in emphasis from *algorithm* to *data*.
  - Perhaps with enough data, we don't have to be smart to be good.
- For example, how to fill in a gap in a picture.
  - Perhaps you have a picture of your ex in front of a great landscape.
- Poll lots of similar pictures for a matching piece.
  - Moving from 10,000 images to 2,000,000 led to a big jump in performance.
- Wisdom of the crowd.

- The growth of the internet makes this much easier in 2010 than it was in 1990. (Why?)



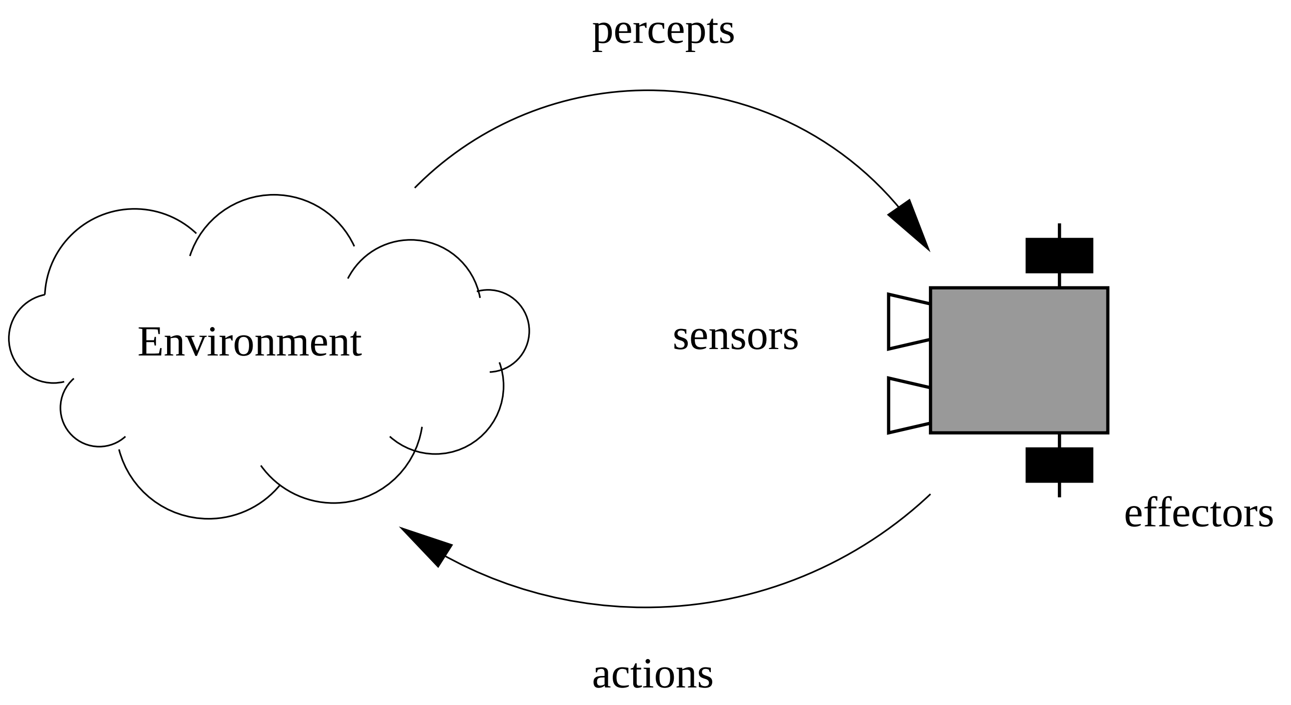
- Currently a lot of interest in mining data from social networks.
- If people are connected, they are similar in some sense.

Intelligent Agents

- An *agent* is a system that:
  - is *situated* in an environment,
  - is capable of *perceiving* its environment, and
  - is capable of *acting* in its environment

  with the goal of satisfying its design objectives.

- Pictorially:



- The task is to program the agent to convert percepts to actions.

- Human "agent":
  - *environment*: physical world;
  - *sensors*: eyes, ears, . . .
  - *effectors*: hands, legs, . . .
- Software agent:
  - *environment*: (e.g.) UNIX operating system;
  - *sensors*: `ls`, `ps`, . . .
  - *effectors*: `rm`, `chmod`, . . .
- Robot:
  - *environment*: physical world;
  - *sensors*: sonar, camera;
  - *effectors*: wheels.

gle
intel
Red Bull
MDV
MOHR DAVIDOW VENTURES
Powered by Diesel.
NXP

What to do?

Those who do not reason
Perish in the act.
Those who do not act
perish for that reason
(W H Auden)

- The key problem we have is *knowing the right thing to do*.
- Knowing what to do can *in principle* be easy: consider all the alternatives, and choose the "best".
- But any time-constrained domain, we have to make a decision *in time for that decision to be useful*!
- A tradeoff.

- *Ideal rational agent*:

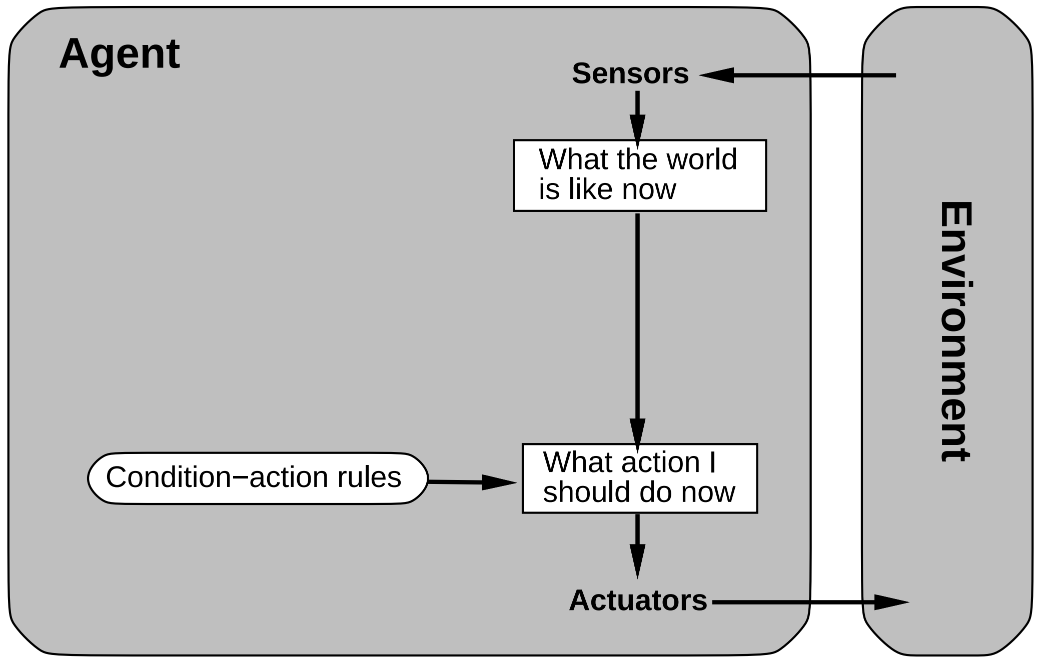
  For each percept sequence, an ideal rational agent will act to maximise its expected performance measure, on the basis of information provided by percept sequence plus any information built in to agent.
- Note that this does not preclude performing actions to *find things out*.
- More precisely, we can view an agent as a function:

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

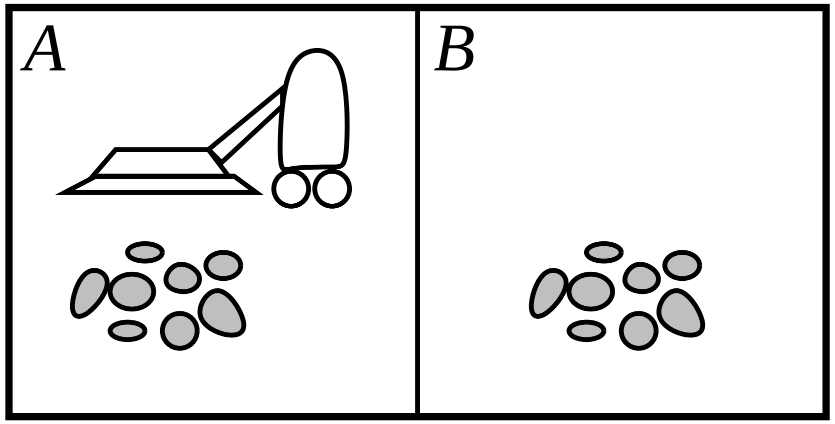
  from sequences of percepts P to actions A.

Simple reflex agent

- A simple agent maps percepts directly to actions:



- In the “vacuum world”:



- A simple agent may suffice:

```
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
```

- A more general version of this program, which works for the agent architecture given above, is on the next slide.

```
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
    return action
```

Fully observable versus partially observable

- A *fully observable* environment is one in which the agent can obtain complete, accurate, up-to-date information about the environment's state.
- Such an environment is also called *accessible*.
- Most moderately complex environments (including, for example, the everyday physical world and the Internet) are only *partially observable*.
- Such environments are also known as *non-accessible*
- The more observable an environment is, the simpler it is to build agents to operate in it.

Deterministic versus non-deterministic

- A *deterministic* environment is one in which any action has a single guaranteed effect — there is no uncertainty about the state that will result from performing an action.
- The physical world can to all intents and purposes be regarded as non-deterministic.
- The textbook calls non-deterministic environments *stochastic* if we quantify the non-determinism using probability theory.
- Non-deterministic environments present greater problems for the agent designer.

Episodic versus sequential

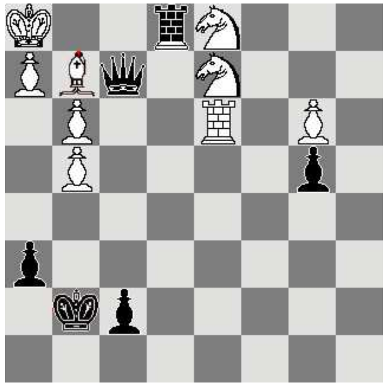
- In an *episodic* environment, the performance of an agent is dependent on a number of discrete episodes, with no link between the performance of an agent in different scenarios.
- An example of an episodic environment would be an assembly line where an agent had to spot defective parts.
- Episodic environments are simpler from the agent developer's perspective because the agent can decide what action to perform based only on the current episode — it need not reason about the interactions between this and future episodes.
- Environments that are not episodic are called either *non-episodic* or *sequential*. Here the current decision affects future decisions.
- Driving a car is sequential.

Static *vs* dynamic

- A *static* environment is one that can be assumed to remain unchanged except by the performance of actions by the agent.
- A *dynamic* environment is one that has other processes operating on it, and which hence changes in ways beyond the agent's control.
- The physical world is a highly dynamic environment.
- One reason an environment may be dynamic is the presence of other agents.

Discrete *vs* continuous

- An environment is *discrete* if there are a fixed, finite number of actions and percepts in it.
- The textbook *gives* a chess game as an example of a discrete environment, and taxi driving as an example of a continuous one.



Summary

- This lecture has looked at:
  - The history of AI
  - The notion of intelligent agents
  - A classification of agent environments.
- Broadly speaking, the rest course will cover the major techniques of AI, with special reference to agents.
- The techniques we'll look at will start with those applicable to simple environments and move towards those suitable for more complex environments.