

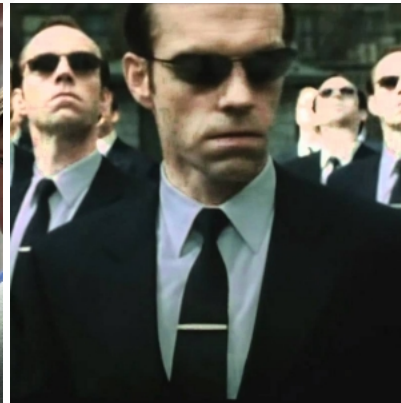
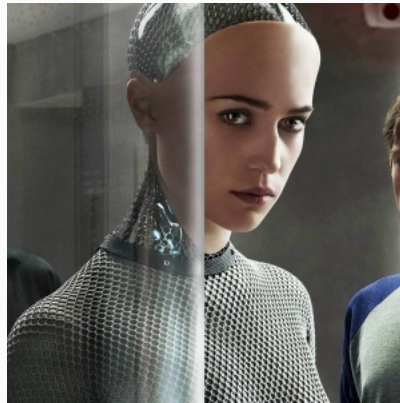
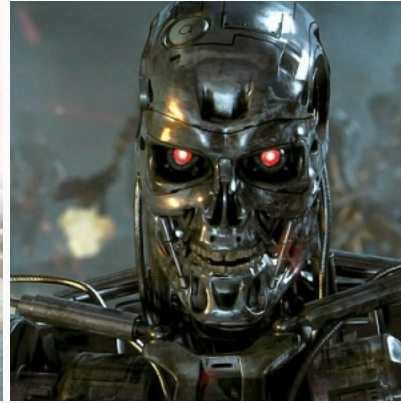
Introduction to Artificial Intelligence

Lecture 1: Foundations

Introduction

(Chapter 1)

AI \neq Science fiction

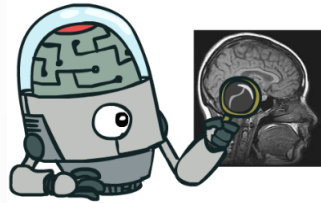


What is AI?

What is AI?

Artificial intelligence is the science of making machines or programs that:

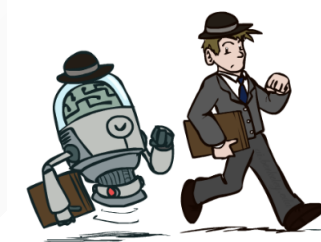
Think like people



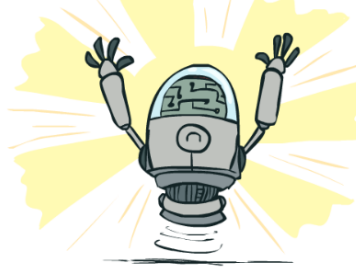
Think rationally



Act like people



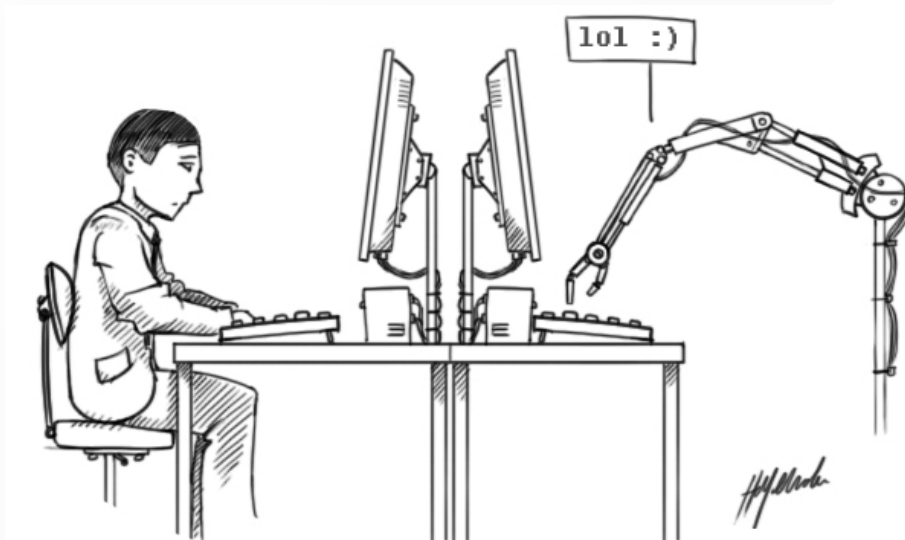
Act rationally



Acting humanly

The Turing test (the Imitation Game)

A computer passes the test if a human operator, after posing some written questions, cannot tell whether the written responses come from a person or from a computer.



Can machines think? (Alan Turing, 1950).

Abilities for passing the test

An agent would not pass the Turing test without the following requirements:

- natural language processing
- knowledge representation
- automated reasoning
- machine learning
- computer vision (total Turing test)
- robotics (total Turing test)

Despite being proposed almost 70 years ago, the Turing test is still relevant today.

Limitations of the Turing test

- Tends to focus on **human-like errors**, **linguistic tricks**, etc.
- It seems more important to study the **principles** underlying intelligence than to replicate an exemplar.

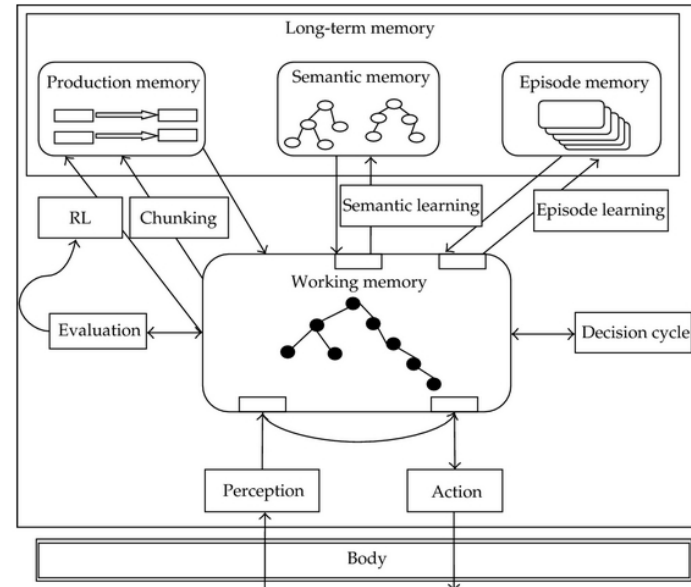


Aeronautics is not defined as the field of making machines that fly so exactly like pigeons that they can fool even other pigeons.

Thinking humanly

Cognitive science

- Study of the **human mind** and its processes. The goal of cognitive science is to form a theory about the structure of the mind, summarized as a comprehensive **computer model**.
- A **cognitive architecture** usually follows human-like reasoning and can be used to produce testable predictions (time of delays during problem solving, kinds of mistakes, learning rates, etc).



The modern SOAR cognitive architecture, as a descendant of the Logic Theorist (Alan Newell, Herbert Simon, 1956).

Limitations of cognition for AI

- In linguistics, the argument of **poverty of the stimulus** states that children do not receive sufficient input to generalize grammatical rules through linguistic input alone.
 - A baby hears too few sentences to deduce the grammar of English before he speaks correctly.
- (Controversial) Therefore, humans must be **biologically pre-wired** with **innate knowledge** for representing language.



How do we know what we know? (Noam Chomsky, 1980).

This suggests that it may not be possible to implement a fully functioning computer model of the human mind without background knowledge of some sort. This is a huge technical **obstacle**, as accessing this knowledge would require reverse-engineering the brain.

Thinking rationally

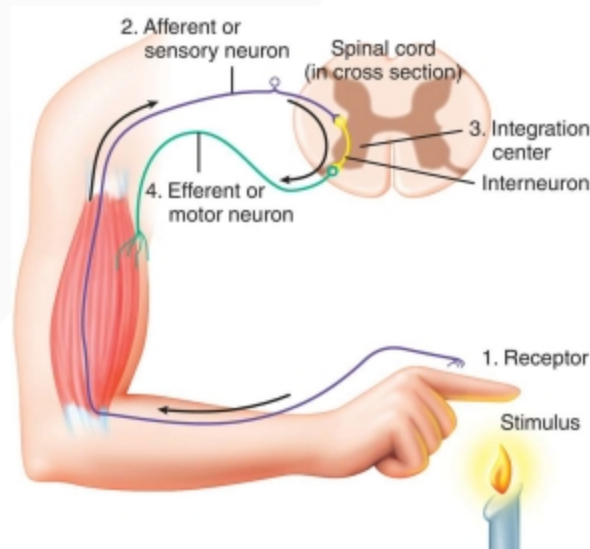
The logical approach

- The rational thinking approach is concerned with the study of **irrefutable reasoning processes**. It ensures that all actions performed by a computer are formally **provable** from inputs and prior knowledge.
- The "laws of thought" were supposed to govern the operation of the mind. Their study initiated the field of **logic** and the **logician tradition** of AI (1960-1990).
- Studied in depth in **Knowledge representation** (Prof. Pascal Gribomont).

```
/* Example of automated reasoning in Prolog */  
mortal(X) :- human(X).  
human(socrate).  
  
?- mortal(socrate).  
yes.
```

Limitations of logical inference

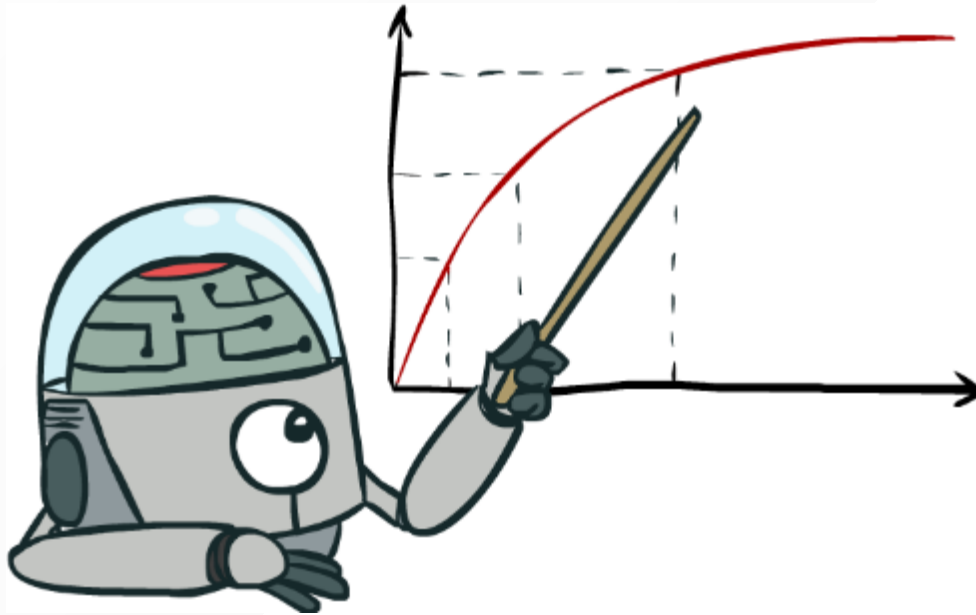
- Representation of **informal** knowledge is difficult.
- Hard to define provable **plausible** reasoning.
- **Combinatorial explosion** (in time and space).
- Logical inference is part of intelligence. It does not cover everything:
 - e.g., might be no provably correct thing to do, but still something must be done;
 - e.g., reflex actions can be more successful than slower carefully deliberated ones.



Pain withdrawal reflexes do not involve inference.

Acting rationally

- A **rational agent** acts so as to achieve the best (expected) outcome.
 - Correct logical inference is just one of several possible mechanisms for achieving this goal.
 - Perfect rationality cannot be achieved due to computational limitations! The amount of reasoning is adjusted according to available resources and importance of the result.
 - The brain is good at making rational decisions but not perfect either.
- Rationality only concerns **what** decisions are made (not the thought process behind them, human-like or not).
- Goals are expressed in terms of the **performance** or **utility** of outcomes. Being rational means maximizing its expected performance.
 - The standard of rationality is general and mathematically well defined.
- In this course, we will study general principles of rational agents and the components for constructing them.



In this course, Artificial intelligence = **Maximizing expected performance**

AI prehistory

- **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.
- **Psychology:** adaptation, phenomena of perception and motor control, psychophysics.
- **Economics:** formal theory of rational decisions.
- **Linguistics:** knowledge representation, grammar.
- **Neuroscience:** plastic physical substrate for mental activity.
- **Control theory:** homeostatic systems, stability, simple optimal agent designs.

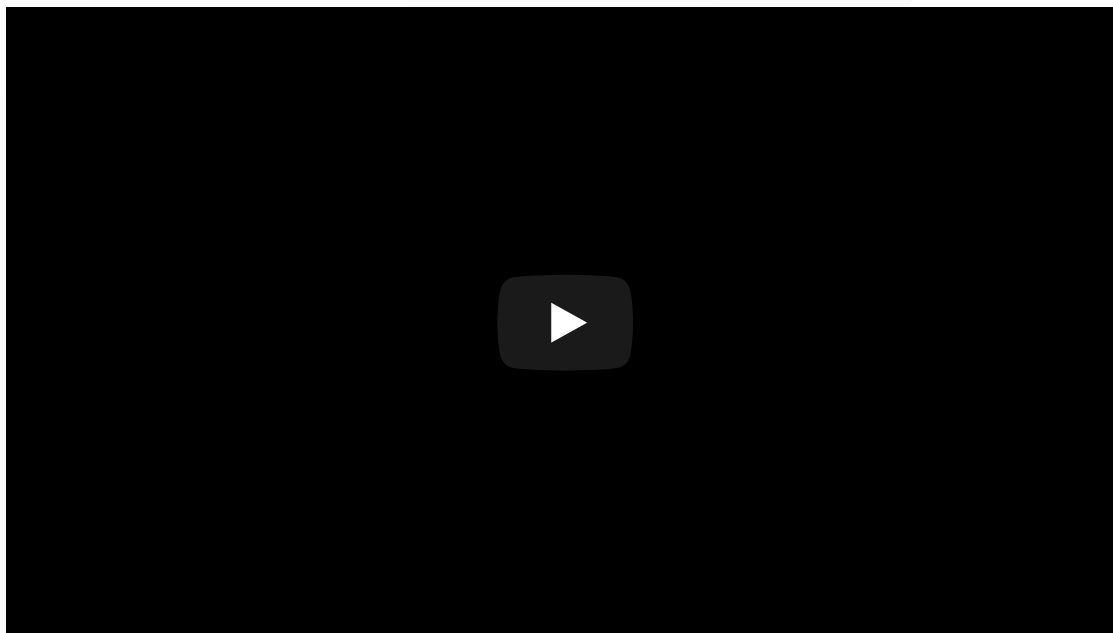
A short history of AI

1940-1950: Early days

- 1943: McCulloch and Pitts: Boolean circuit model of the brain.
- 1950: Turing's "Computing machinery and intelligence:."

1950-1970: Excitement and expectations

- 1950s: Early AI programs, including Samuel's checkers program, Newell and Simon's Logic Theorist and Gelernter's Geometry Engine.
- 1956: Dartmouth meeting: "Artificial Intelligence" adopted.
- 1958: Rosenblatt invents the perceptron.
- 1965: Robinson's complete algorithm for logical reasoning.
- 1966-1974: AI discovers computational complexity.



A short history of AI

1970-1990: Knowledge-based approaches

- 1969: Neural network research almost disappears after Minsky and Paper's paper.
- 1969-1979: Early development of knowledge-based systems.
- 1980-1988: Expert systems industrial boom.
- 1988-1993: Expert systems industry busts (AI winter).

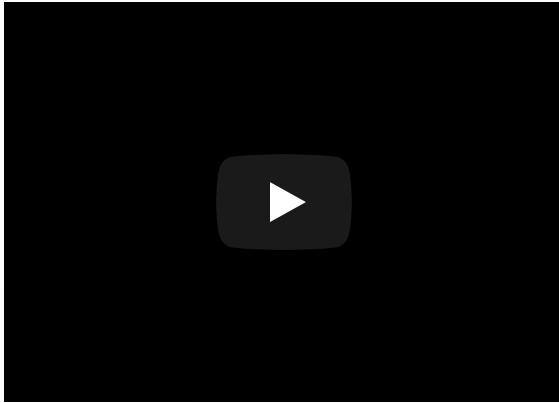
1990-Present: Statistical approaches

- 1985-1995: The return of neural networks.
- 1988-: Resurgence of probability, focus on uncertainty, general increase in technical depth.
- 1995-2010: New fade of neural networks.
- 1995-: Complete intelligent agents and learning systems.
- 2000-: Availability of very large datasets.
- 2010-: Availability of fast commodity hardware (GPUs).
- 2012-: Resurgence of neural networks with deep learning approaches.

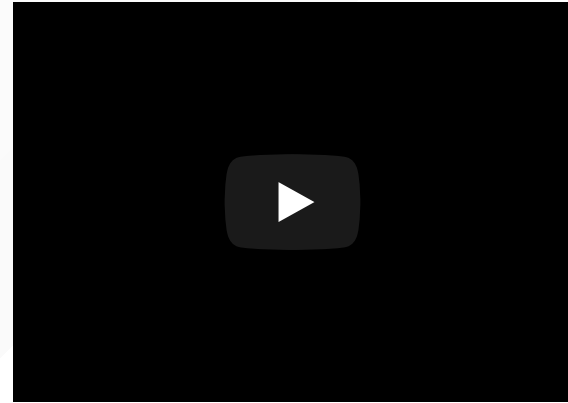
What can AI do at present?

- Translate spoken Chinese to spoken English, live?
- Answer multi choice questions, as good as an 8th grader?
- Converse with a person for an hour?
- Play decently at Chess? Go? Poker? Soccer?
- Buy groceries on the web? in a supermarket?
- Prove mathematical theorems?
- Drive a car safely on a parking lot? in New York?
- Perform a surgery?
- Identify skin cancer better than a dermatologist?
- Write a funny story?
- Paint like Vangogh? Compose music?
- Show common sense?

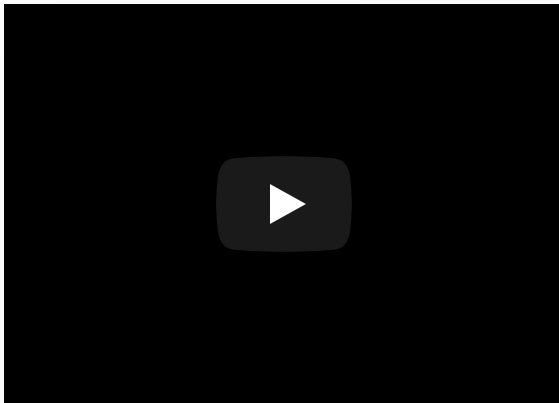
Games



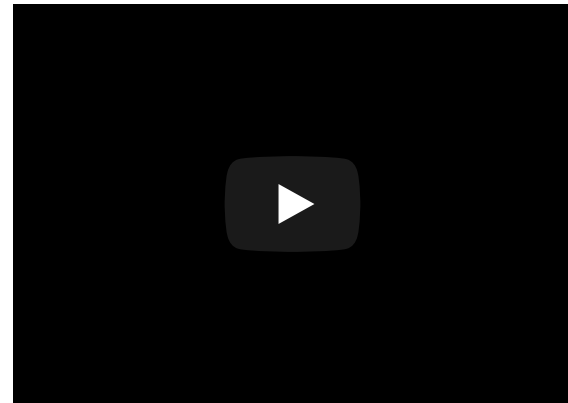
Deep Blue



Playing Atari games

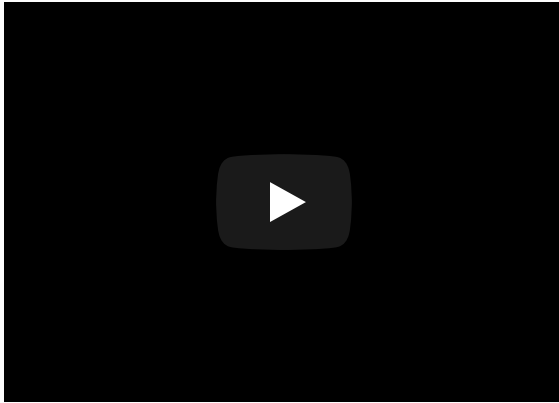


Alpha Go

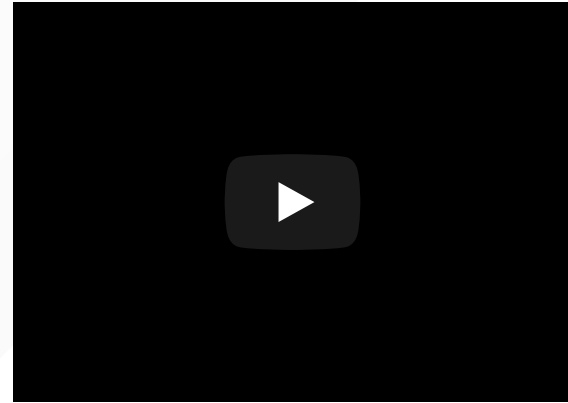


Starcraft

Natural language

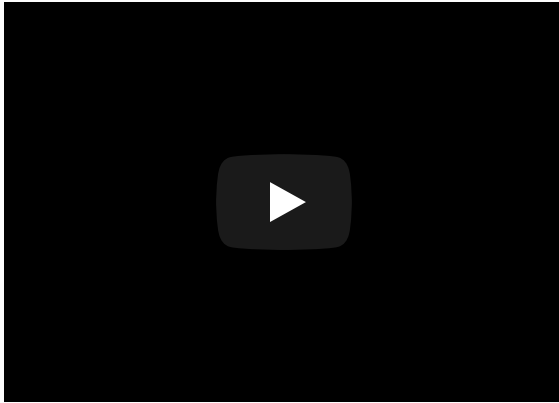


Speech translation and synthesis

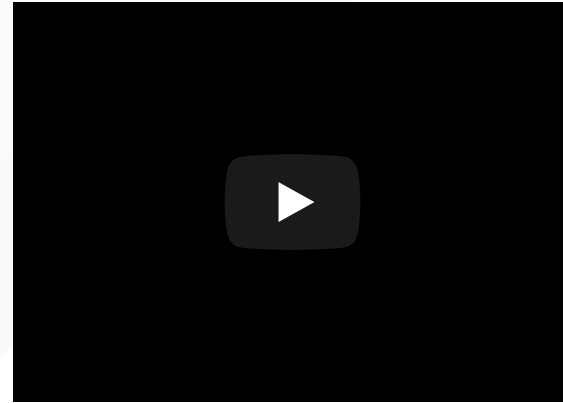


Question answering systems

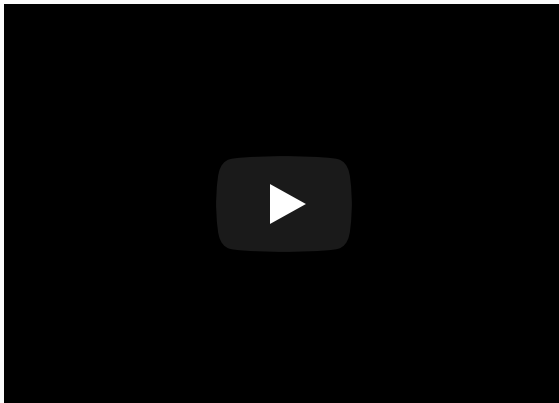
Vision



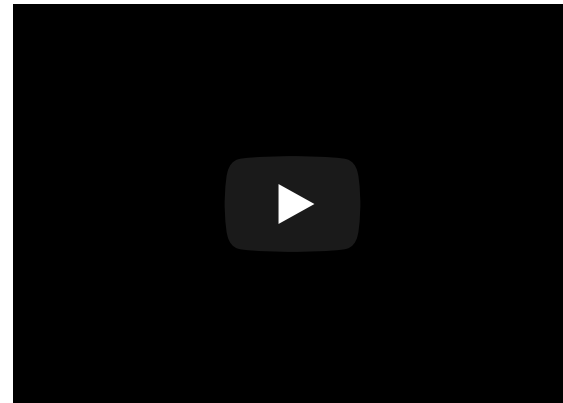
Semantic segmentation



Generating image descriptions

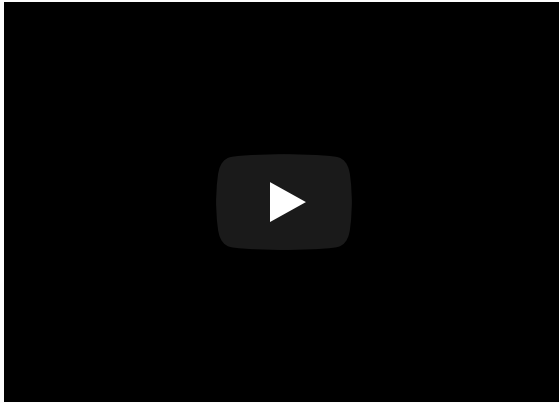


Pose estimation

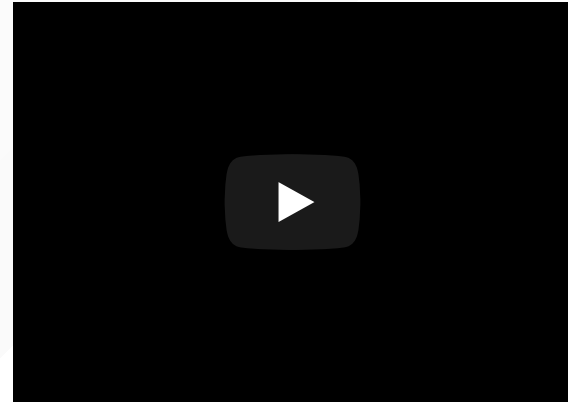


Detecting skin cancer

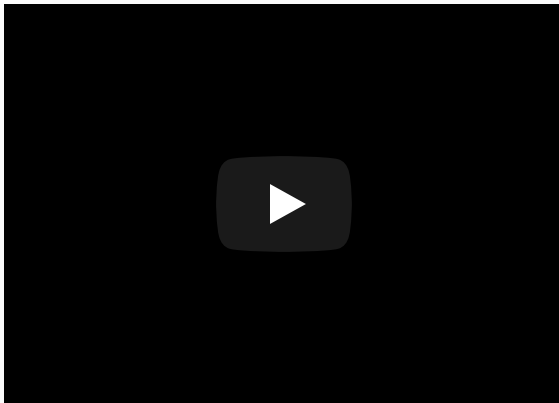
Robotics



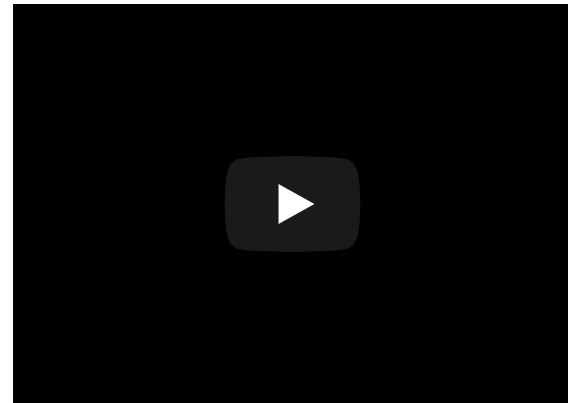
Autonomous cars



Playing soccer

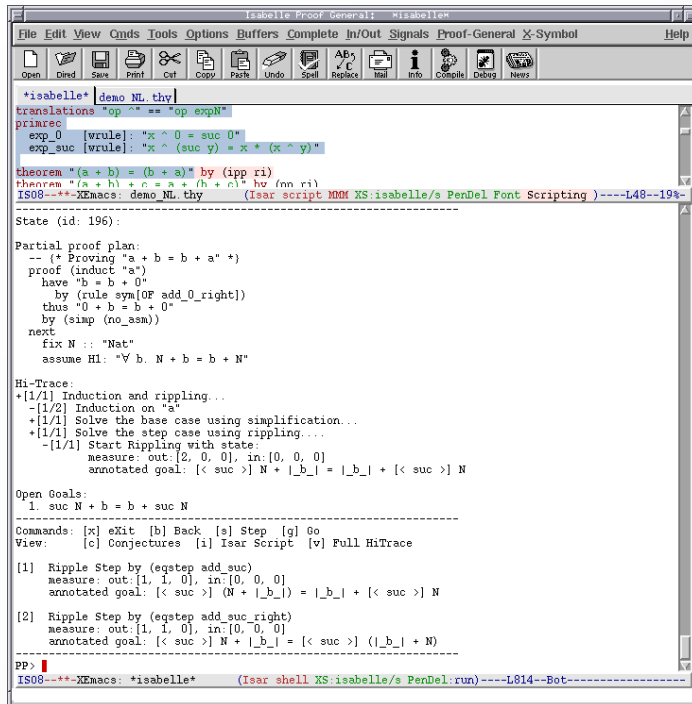


Learning to walk



Folding laundry

Logic



```
*isabelle* demo ML.thy
translations "op ^" == "op expN"
primrec
  exp_0 [wrule]: "x ^ 0 = suc 0"
  exp_suc [wrule]: "x ^ (suc y) = x * (x ^ y)"

theorem "(a + b) = (b + a)" by (ippr ri)
theorem "(a + b) + c = a + (b + c)" by (ippr ri)
IS08---**XEmacs: demo ML.thy (Isar script MMM XS:isabelle/s PenDel Font Scripting)----L48--194-

State (id: 196):

Partial proof plan:
-- (* Proving "a + b = b + a" *)
proof (induct "a")
  have "b = b + 0"
    by (rule sym[OF add_0_right])
  thus "0 + b = b + 0"
    by (simp (no_asm))
next
  fix N :: "Nat"
  assume H1: "∀ b. N + b = b + N"

Hi-Trace:
+ [1/1] Induction and rippling...
- [1/2] Induction on "a"
+ [1/1] Solve the base case using simplification...
+ [1/1] Solve the step case using rippling...
- [1/1] Start Rippling with state:
  measure: out [2, 0, 0], in [0, 0, 0]
  annotated goal: [< suc >] N + |b_| = |b_| + [< suc >] N

Open Goals:
1. suc N + b = b + suc N

Commands: [x] eXit [b] Back [s] Step [g] Go
View: [c] Conjectures [i] Isar Script [v] Full HiTrace

[1] Ripple Step by (egstep add_suc)
  measure: out [1, 1, 0], in [0, 0, 0]
  annotated goal: [< suc >] (N + |b_|) = |b_| + [< suc >] N

[2] Ripple Step by (egstep add_suc_right)
  measure: out [1, 1, 0], in [0, 0, 0]
  annotated goal: [< suc >] N + |b_| = [< suc >] (|b_| + N)

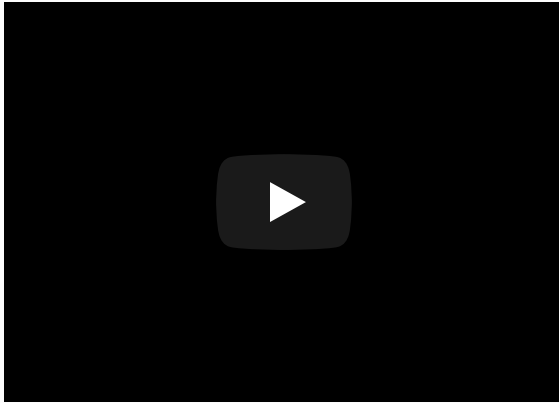
PP>
IS08---**XEmacs: *isabelle* (Isar shell XS:isabelle/s PenDel:run)----L814--Bot-----
```

Automated Theorem Prover

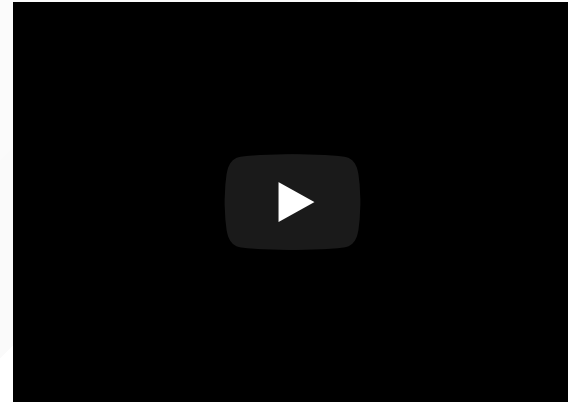


Formal software verification

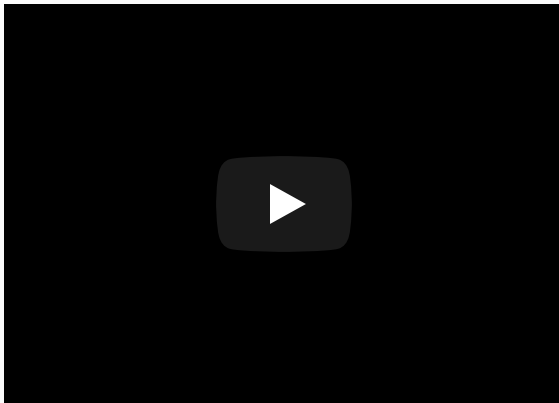
Decision making



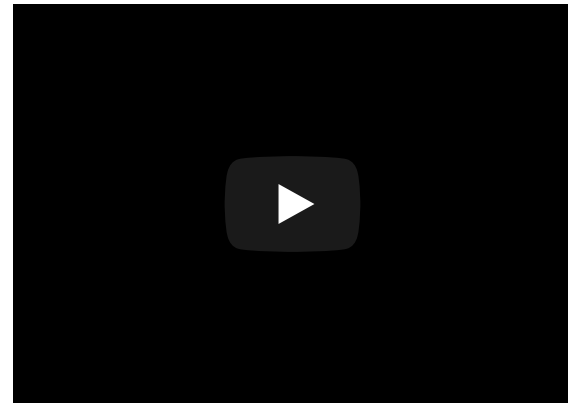
Search engines



Fraud detection



Recommendation systems

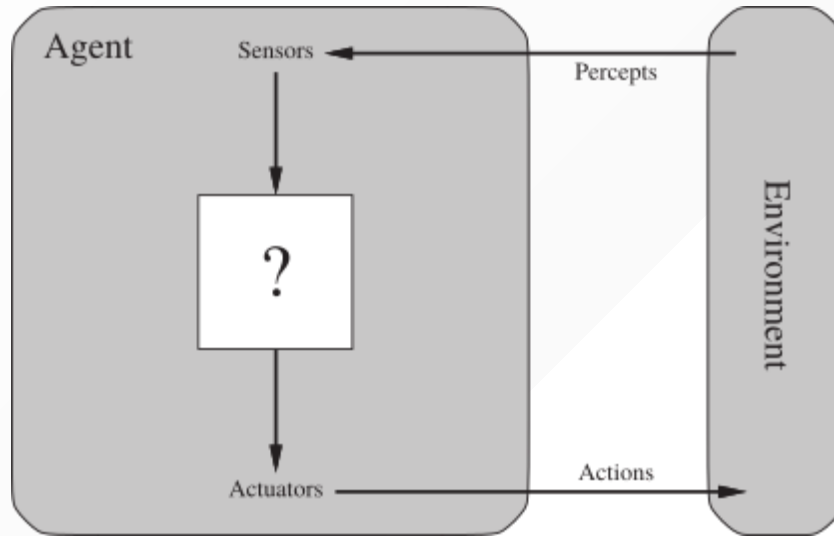


Sorting packages (routing, planning)

Intelligent agents

(Chapter 2)

Agents and environments

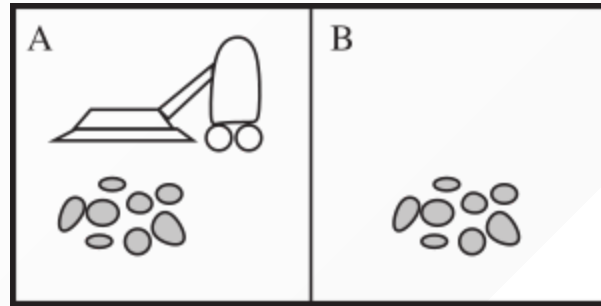


- An **agent** is an entity that **perceives** its environment through sensors and take **actions** through actuators.
- The agent behavior is described by the **agent function**, or **policy**, that maps percept histories to actions:

$$f : \mathcal{P}^* \rightarrow \mathcal{A}$$

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

Vacuum-cleaner world



- **Percepts:** location and content, e.g. $[A, \textit{Dirty}]$
- **Actions:** *Left, Right, Suck, NoOp*

A vacuum-cleaner agent

Partial tabulation of a simple vacuum-cleaner agent function:

Percept sequence	Action
<i>[A, Clean]</i>	<i>Right</i>
<i>[A, Dirty]</i>	<i>Suck</i>
<i>[B, Clean]</i>	<i>Left</i>
<i>[A, Dirty]</i>	<i>Suck</i>
<i>[A, Clean], [A, Clean]</i>	<i>Right</i>
<i>[A, Clean], [A, Dirty]</i>	<i>Suck</i>
<i>(...)</i>	<i>(...)</i>

A vacuum-cleaner agent

An implementation of the agent function:

```
def program(percept):  
    location, status = percept  
    if status == "dirty":  
        return "suck"  
    elif location == "A":  
        return "right"  
    elif location == "B":  
        return "left"
```

The agent in its environment:

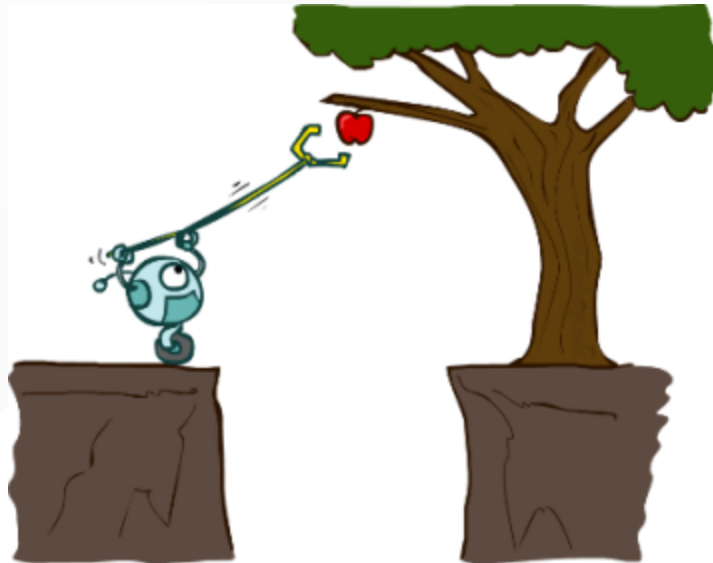
```
environment = Environment()  
agent = Agent(program)  
  
while True:  
    percept = environment.percept()  
    action = agent.program(percept)  
    environment.execute(action)
```

The optimal vacuum-cleaner?

- What is the **right** agent function? How to formulate the goal of the vacuum-cleaner agent?
 - 1 point per square cleaned up at time t ?
 - 1 point per clean square per time step, minus one per move?
 - penalize for $> k$ dirty squares?
- Can it be implemented in a **small** agent program?

Rational agents

- Informally, a **rational agent** is an agent that does the "right thing".
- A **performance measure** evaluates a sequence of environment states caused by the agent's behavior.
- A rational agent is an agent that chooses whichever action that **maximizes** the **expected** value of the performance measure, given the percept sequence to date.



Rational agents

- Rationality \neq omniscience
 - percepts may not supply all relevant information.
- Rationality \neq clairvoyance
 - action outcomes may not be as expected.
- Hence, rational \neq successful.
- However, rationality leads to [exploration](#), [learning](#) and [autonomy](#).

Performance, environment, actuators, sensors

The characteristics of the performance measure, environment, action space and percepts dictate techniques for selecting rational actions.

These characteristics are summarized as the **task environment**.

Example 1: an autonomous car

- **performance measure**: safety, destination, legality, comfort, ...
- **environment**: streets, highways, traffic, pedestrians, weather, ...
- **actuators**: steering, accelerator, brake, horn, speaker, display, ...
- **sensors**: video, accelerometers, gauges, engine sensors, GPS, ...

Performance, environment, actuators, sensors

Example 2: an Internet shopping agent

- **performance measure**: price, quality, appropriateness, efficiency
- **environment**: current and future WWW sites, vendors, shippers
- **actuators**: display to user, follow URL, fill in form, ...
- **sensors**: web pages (text, graphics, scripts)

Environment types

- Fully observable vs. partially observable
 - Whether the agent sensors give access to the complete state of the environment, at each point in time.
- Deterministic vs. stochastic
 - Whether the next state of the environment is completely determined by the current state and the action executed by the agent.
- Episodic vs. sequential
 - Whether the agent's experience is divided into atomic independent episodes.
- Static vs. dynamic
 - Whether the environment can change, or the performance measure can change with time.
- Discrete vs. continuous
 - Whether the state of the environment, the time, the percepts or the actions are continuous.
- Single agent vs. multi-agent
 - Whether the environment include several agents that may interact with each other.
- Known vs unknown
 - Reflects the agent's (or its designer's) state of knowledge of the "law of physics" of the environment.

Examples of environments

Are the following task environments fully observable? deterministic? episodic? static? discrete? single agents? Known?

- Crossword puzzle
- Chess, with a clock
- Poker
- Backgammon
- Taxi driving
- Medical diagnosis
- Image analysis
- Part-picking robot
- Refinery controller
- The real world

Agent programs

The job of AI is to design an **agent program** that implements the agent function. This program will run on an **architecture**, that is a computing device with physical sensors and actuators.

$$agent = program + architecture$$

Implementation

Agent programs can be designed and implemented in many ways:

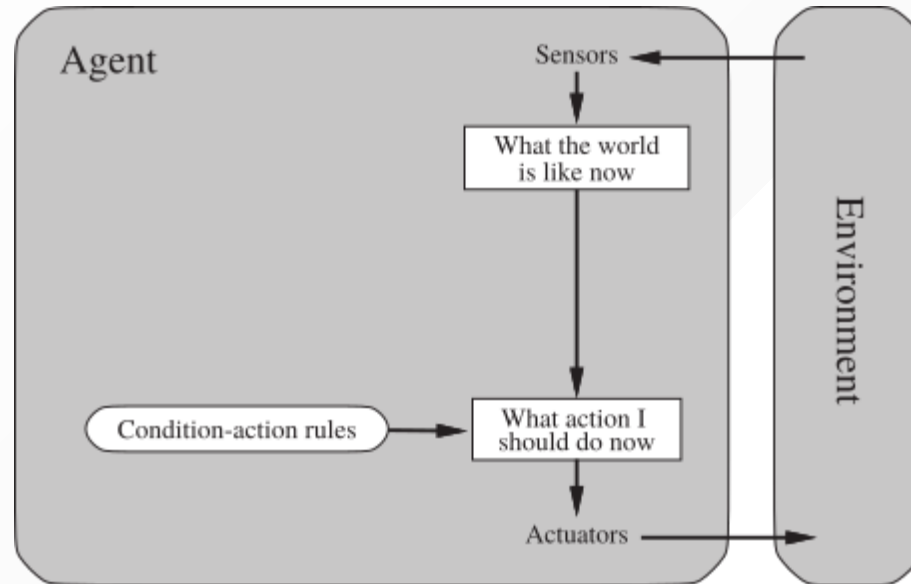
- with tables
- with rules
- with search algorithms
- with learning algorithms

Table-driven agents

```
def TableDrivenAgentProgram(table):  
    percepts = []  
    def program(percept):  
        percepts.append(percept)  
        action = table[percepts]  
        return action  
    return program
```

- A **table-driven agent** determines its next action with a table that contains the appropriate action for every possible percept sequence.
- **Design issue:** one needs to anticipate all sequence of percepts and how the agent should respond.
- **Technical issue:** the lookup table will contain $\sum_{t=1}^T |\mathcal{P}|^t$ entries.
 - Example (autonomous car): using a 30fps 640x480 RGB camera as sensor, this results in a table with over $10^{2500000000000}$ entries for an hour of driving.

Simple reflex agents



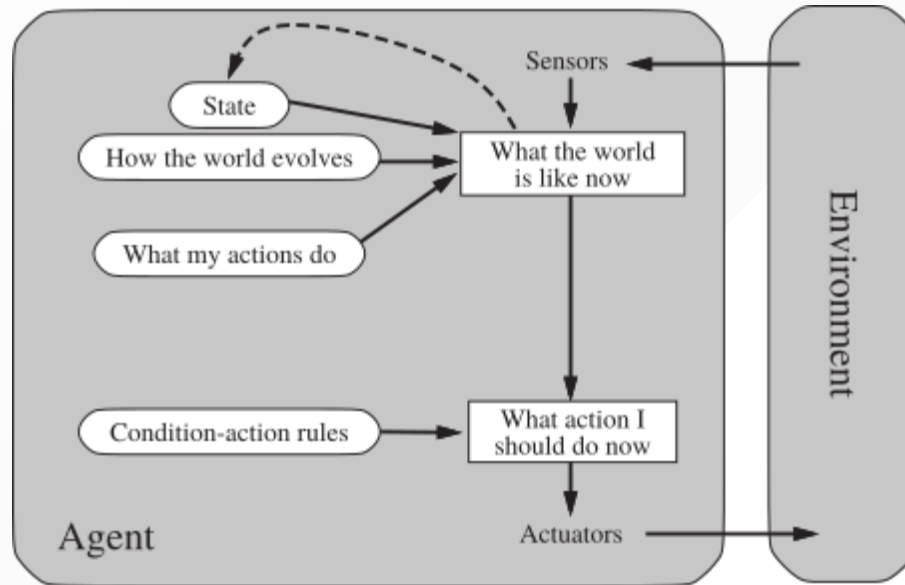
- **Simple reflex agents** select actions on the basis of the current percept, ignoring the rest of the percept history.
- They implement **condition-action rules** that match the current percept to an action.

Simple reflex agents

```
def SimpleReflexAgentProgram(rules, interpret_input):  
    def program(percept):  
        state = interpret_input(percept)  
        rule = rule_match(state, rules)  
        action = rule.action  
        return action  
    return program
```

- Rules provide a way to **compress** the function table.
 - Example (autonomous car): If a car in front of you slow down, you should break. The color and model of the car, the music on the radio or the weather are all irrelevant.
- Simple reflex agents are simple but they turn out to have **limited intelligence**.
- They can only work in a **Markovian** environment, that is if the correct decision can be made on the basis of only the current percept. In other words, if the environment is fully observable.

Model-based reflex agents

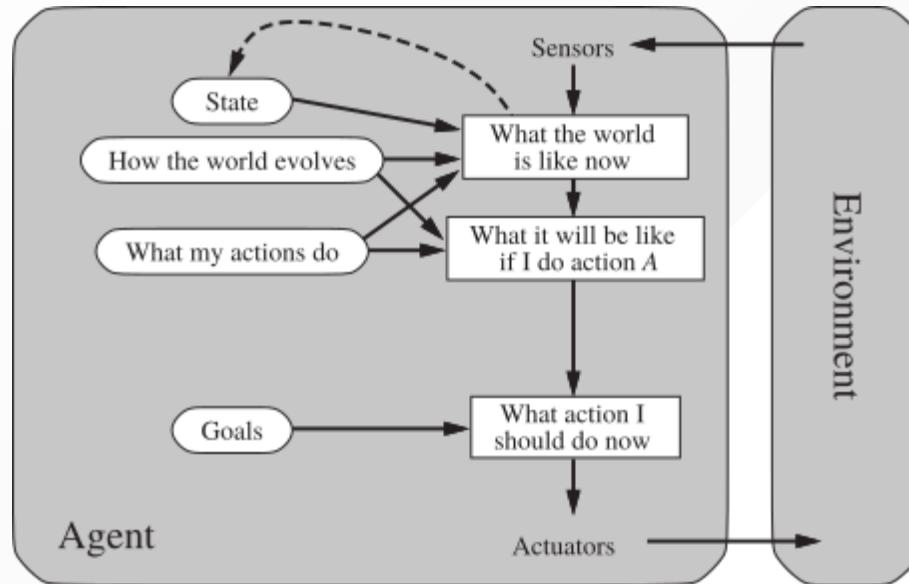


- **Model-based agents** handle partial observability of the environment by keeping track of the part of the world they cannot see now.
- The internal state of model-based agents is updated on the basis of a **model** which determines:
 - how the environment evolves independently of the agent;
 - how the agent actions affect the world.

Model-based reflex agents

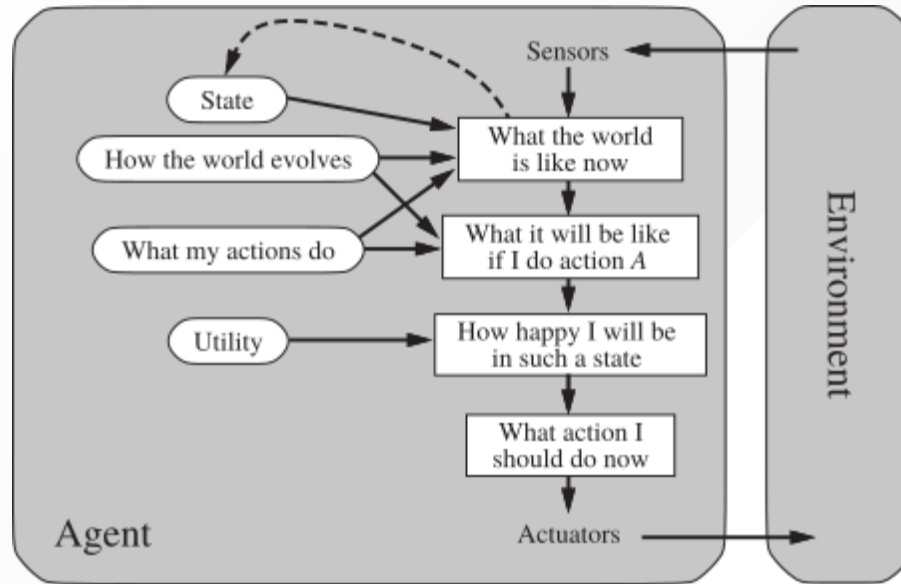
```
def ModelBasedReflexAgentProgram(rules, update_state, model):  
    def program(percept):  
        program.state = update_state(program.state,  
                                     program.action,  
                                     percept,  
                                     model)  
  
        rule = rule_match(program.state, rules)  
        action = rule.action  
        return action  
    program.state = program.action = None  
    return program
```

Goal-based agents



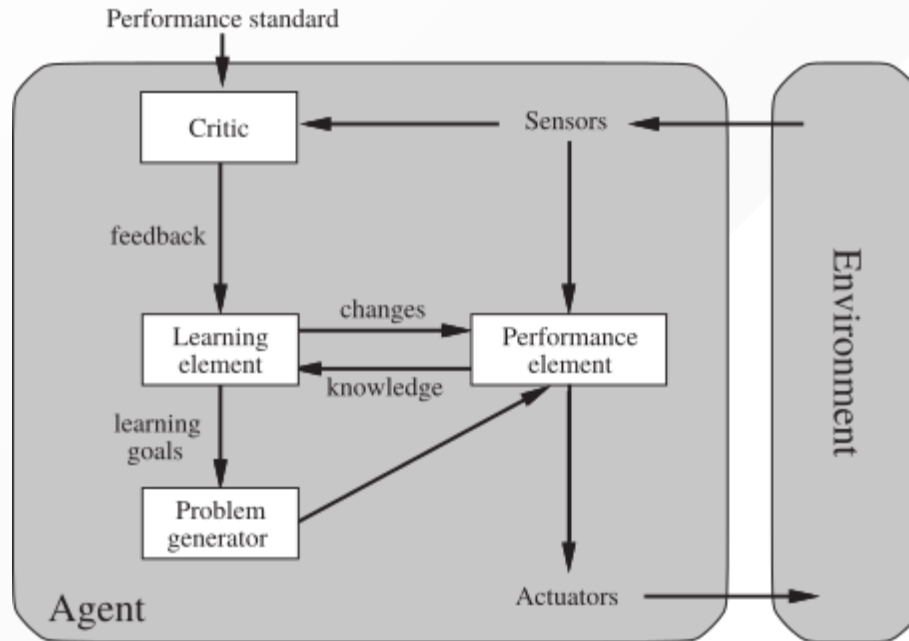
- Principle: i) generate possible sequences of actions, ii) predict the resulting states and iii) assess **goals** in each.
 - Example (autonomous car): Has the car arrived to destination?
- A **goal-based agent** chooses an action that will achieve the goal.
 - More general than rules. Goals are rarely explicit in condition-action rules.
 - Finding action sequences that achieve goals is difficult. **Search** and **planning** are two strategies.

Utility-based agents



- **Goals** are often not enough to generate high-quality behavior.
 - Example (autonomous car): There are many ways to arrive to destination, but some are quicker or more reliable.
 - Goals only provide binary assessment of performance.
- A **utility function** scores any given sequence of environment states.
 - The utility function is an internalization of the performance measure.
- A rational utility-based agent chooses an action that **maximizes the expected utility of its outcomes**.

Learning agents



- **Learning agents** are capable of **self-improvement**. They can become more competent than their initial knowledge alone might allow.
- They can make changes to any of the knowledge components by:
 - learning how the **world** evolves;
 - learning what are the **consequences** of actions;
 - learning the utility of actions through **rewards**.

A learning autonomous car

- **Performance element:**
 - The current system for selecting actions and driving.
- The **critic** observes the world and passes information to the **learning element**.
 - E.g., the car makes a quick left turn across three lanes of traffic. The critic observes shocking language from the other drivers and informs bad action.
 - The learning element tries to modifies the performance element to avoid reproducing this situation in the future.
- The **problem generator** identifies certain areas of behavior in need of improvement and suggest experiments.
 - E.g., trying out the brakes on different surfaces in different weather conditions.

Summary

- An **agent** is an entity that perceives and acts in an environment.
- The **performance measure** evaluates the agent's behavior. **Rational agents** act so as to maximize the expected value of the performance measure.
- **Task environments** includes performance measure, environment, actuators and sensors. They can vary along several significant dimensions.
- The **agent program** effectively implements the agent function. Their designs are dictated by the task environment.
- **Simple reflex agents** respond directly to percepts, whereas **model-based reflex agents** maintain internal state to track the world. **Goal-based agents** act to achieve goals while **utility-based agents** try to maximize their expected performance.
- All agents can improve their performance through **learning**.

References

- Turing, Alan M. "Computing machinery and intelligence." *Mind* 59.236 (1950): 433-460.
- Newell, Allen, and Herbert Simon. "The logic theory machine--A complex information processing system." *IRE Transactions on information theory* 2.3 (1956): 61-79.
- Chomsky, Noam. "Rules and representations." *Behavioral and brain sciences* 3.1 (1980): 1-15.