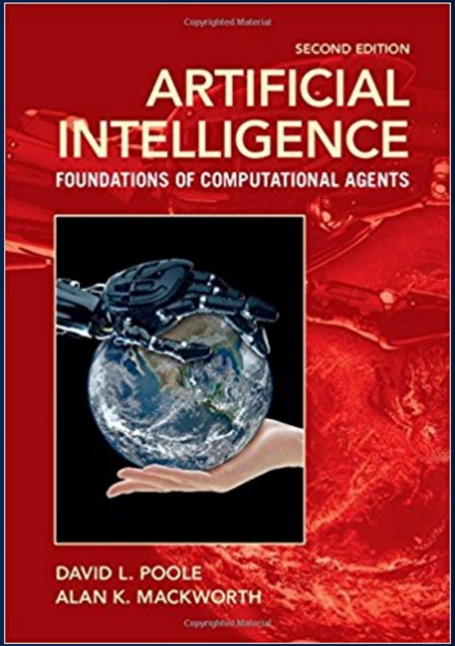


AI Fundamentals: agents

Maria Simi

AI as building intelligent computational agents

LESSON 2: ARTIFICIAL INTELLIGENCE AND AGENTS

(AI-FCA, CHAPTER 1)

Summary

1. AI is the enterprise of building **intelligent computational agents**
2. Dimensions of complexity in the design of intelligent computational agents
3. The agent architecture and behaviour in time
4. Hierarchical control

[AIFCA 2nd Edition]

AI means building computational agents

Artificial intelligence, or **AI**, is the field that studies the synthesis and analysis of **computational agents** that act intelligently.

An **agent** is something that acts in an environment/it does something.

We are interested in what an agent does; that is, how it **acts**. We judge an agent by its actions.

An agent acts **intelligently** when

- ✓ what it does is appropriate given the circumstances and its goals
- ✓ it is flexible to changing environments and changing goals
- ✓ it learns from experience
- ✓ it makes appropriate choices given its perceptual and computational limitations.

Computational agents

A **computational** agent is an agent whose decisions about its actions can be explained in terms of computation and implemented on a physical device

- ✓ The central **scientific goal** of AI is to understand the principles that make intelligent behavior possible in natural or artificial systems.
- ✓ The central **engineering goal** of AI is the **design** and **synthesis** of useful, intelligent artefacts, agents, that are useful in many applications.

This is done by

- ✓ the **analysis** of natural and artificial agents;
- ✓ formulating and testing hypotheses about what it takes to construct intelligent agents;
- ✓ designing, building, and experimenting with computational systems that perform tasks commonly viewed as requiring intelligence.

The term "Artificial Intelligence"

- ✓ Artificial vs natural intelligence

  Artificial Intelligence **is not** the opposite of real Intelligence (not *fake* vs real).

  Intelligence cannot be *fake*. If an artificial agent behaves intelligently, it is intelligent. It is only the external behavior that defines intelligence (weak AI).

  Artificial intelligence is real intelligence created artificially.
- ✓ Turing test: only external behavior counts
- ✓ Winograd schemas [proposed by Levesque] as a test of intelligence:
  - *The city councilmen refused the demonstrators a permit because they feared violence*. Who feared violence?
  - *The city councilmen refused the demonstrators a permit because they advocated violence*. Who advocated violence?

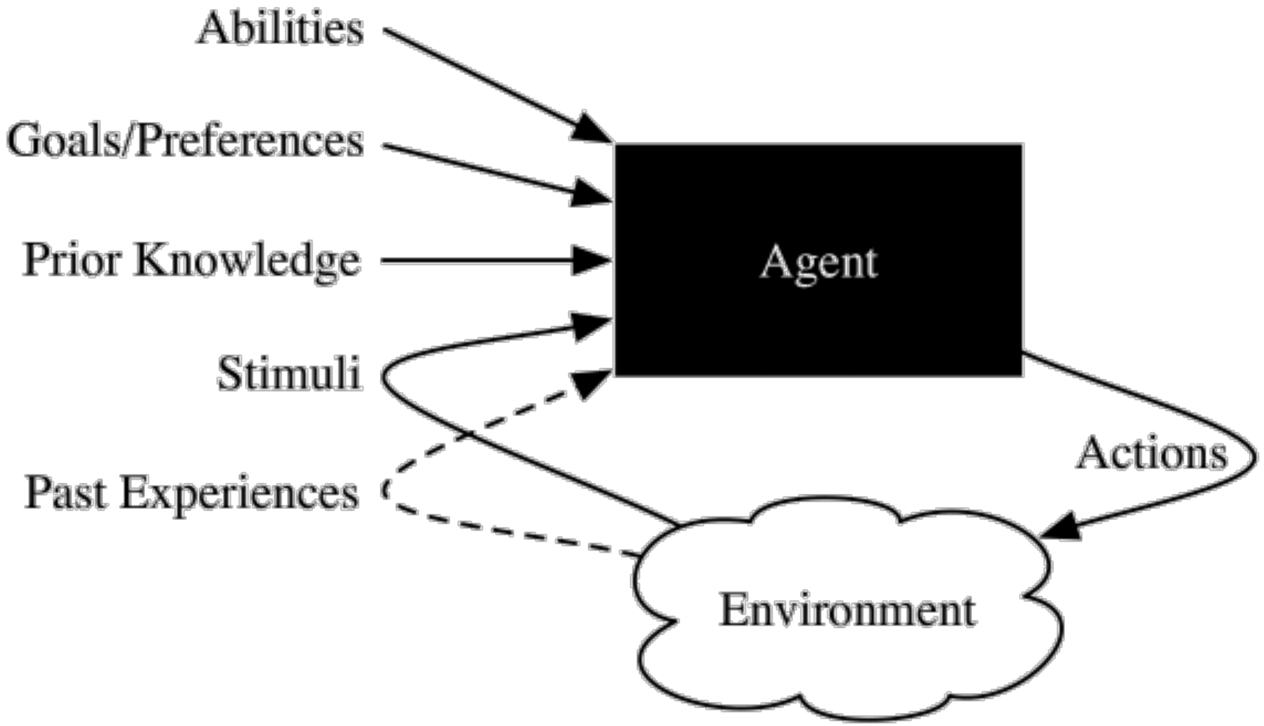
Human intelligence

The obvious naturally intelligent agent is the human being. Human intelligence comes from three main sources:

1. **biology**: Humans have evolved into adaptable animals that can survive in various habitats.
2. **culture**: Culture provides not only language, but also useful tools, useful concepts, and the wisdom that is passed from parents and teachers to children. Language, which is part of culture, provides distinctions in the world that should be noticed for learning.
3. **life-long learning (experience)**: Humans learn throughout their life and accumulate knowledge and skills.

Another form of intelligence is **social intelligence**, the one exhibited by communities and organizations.

Agents Situated in Environments



Belief state

Purposive agents

Design process for agents

Three aspects of computation that must be distinguished:

1. **Design time computation**, that goes into the design of the agent
2. **Offline computation**, that the agent can do before acting in the world
3. **Online computation,** the computation that is done by the agent as it is acting.

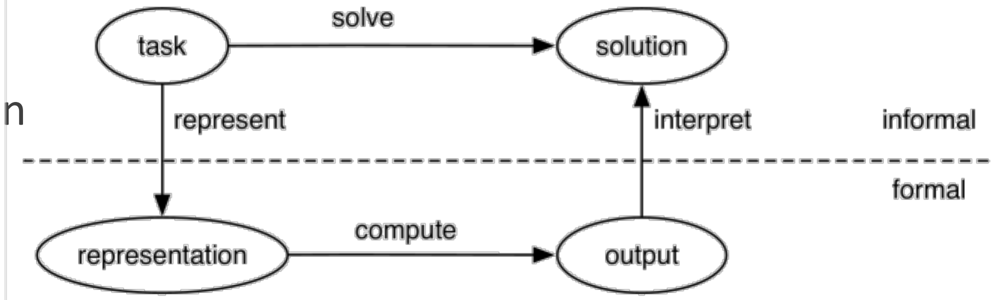
Designing an intelligent agent that can adapt to complex environments and changing goals is a major challenge.

To reach this ultimate goal, two strategies are possible:

- simplify environments and build complex reasoning systems for these simple environments;
- build simple agents for natural/complex environments, simplifying the tasks

Steps in the design process

1. Define the **task**: specify *what* needs to be computed
2. Define what constitutes a **solution** and its quality**:** optimal solution, satisficing solution, approximately optimal solution, probable solution.
3. Choose a **formal representation** for the task; this means choosing how to **represent knowledge** for the task, including representations suitable for learning. Level of abstraction (next).
4. Compute an **output**
5. **Interpret** output as a solution



Levels of abstraction

A model of the world is a symbolic representation of the beliefs of the agents about the world. It is necessarily an **abstraction**

More abstract representations are simpler and human-understandable. But they may be not effective enough.

Low level descriptions are more detailed and accurate but introduce complexity.

Multiple level of abstractions are possible (hierarchical design).

Two levels that are usually considered in design:

1. The knowledge level: what the agent knows and its goals
2. The symbol level: the internal representation and reasoning algorithms

Agent design space: dimensions of complexity

1. Degree of modularity
2. Planning Horizon: *how far ahead to plan*
3. Representation: *how to describe the world*
4. Computational limits: *real agents have limited computational resources*
5. Learning: *how to learn from experience*
6. Uncertainty, in both perception and the effects of actions
7. Preference: *the structure of goals or preferences*
8. Number of Agents
9. Interaction

Modularity

Modularity is the extent to which a system can be decomposed into interacting modules and it is a key factor for reducing complexity.

In the modularity dimension, an agent’s structure is one of the following:

- **flat** – there is no organizational structure
- **modular** – the system is decomposed into interacting modules that can be understood on their own
- **hierarchical** – the system is modular, and the modules themselves are decomposed into simpler modules. The agent reasons at **multiple levels of abstraction**.

  Example: planning a trip

Planning Horizon

The planning horizon dimension is how far ahead in time the agent plans.

According to this dimension an agent is one of the following:

- **Non-planning agent:** does not look at the future.
- **Finite horizon planner**: agent that looks for a fixed finite number of stages. Greedy if only looks one step ahead.
- **Indefinite horizon planner** is an agent that looks ahead some finite, but not predetermined, number of stages.
- **Infinite horizon planner** is an agent that keeps planning forever. Ex. stabilization module of a legged robot

Representation

The representation dimension concerns how the state of the world is described. A state of the world specifies the agent’s internal state (its belief state) and the environment state.

From simple to complex:

- **Atomic states**, as in problem solving.
- **Feature-based representation:** set of propositions that are true or false of the state, properties with a set of possible values. (PROP, CSP, most machine learning).
- **Individuals and relations** (often called relational representations). Representations at the expressive level of FOL (or contractions)

Computational limits

An agent must decide on its best action within time constraints or other constraints in computational resources (memory, precision, …)

The computational limits dimension determines whether an agent has

- **perfect rationality**, where an agent is able to reasons about the best action without constraints
- **bounded rationality**, where an agent decides on the best action that it can find given its computational limitations.

An **anytime algorithm** is an algorithm where the solution quality improves with time. To take into account bounded rationality, an agent must decide whether it should act or keep reasoning for a better decision.

Learning

Learning is necessary when the designer does not have a good model of the problem.

The **learning dimension** determines whether

- knowledge is given in advance, or
- knowledge is learned (from data or past experience).

Learning typically means finding the best model that fits the data and produces a good predictive model.

Modelling formalisms and approaches are dealt in this course.

All the issues concerned with learning are dealt in the **Machine Learning** course.

Uncertainty

Uncertainty is divided into two dimensions:

- uncertainty in sensing/perception (**fully observable**, **partially observable** states)
- uncertainty about the effects of actions (**deterministic**, **stochastic**).

  When the effect is stochastic, it means that we have a probability distribution over the states resulting from the actions.

We will deal with uncertainty in Section III.

Preference

The preference dimension considers whether the agent has goals or richer preferences:

- A **goal** is either an *achievement* goal, which is a proposition to be true in some final state, or a *maintenance* goal, a proposition that must be true in all visited states.
- **Complex preferences** involve trade-offs among the desirability of various outcomes, perhaps at different times.
  - ✓ An **ordinal preference** is where only the ordering of the preferences is important.
  - ✓ A **cardinal preference** is where the magnitude of the values matters. States are evaluated by **utility functions**.

Number of agents

The **number-of-agents** dimension considers whether the agent explicitly considers other agents:

- **Single agent reasoning** means the agent assumes that there are no other agents in the environment or that all other agents are “part of nature”, and so are **non-purposive** (they do not pursue their own goals).
- **Multiple agent reasoning** (or **multi-agent** reasoning) means the agent takes the reasoning and actions of other agents into account.
  - This occurs when there are other intelligent agents whose goals or preferences depend, in part, on what the agent does or when the agent must communicate with other agents.

Interaction

The interaction dimension considers whether the agent does

- **offline reasoning:** the agent determines what to do before interacting with the environment, or
- **online reasoning:** the agent must determine what action to do while interacting in the environment and needs to make timely decisions.

More sophisticated agents reason while acting; this includes long-range strategic reasoning as well as reasoning for reacting in a timely manner to the environment.

Prototypical Applications

1. **Autonomous delivery robot**
2. Diagnostic assistant
3. Tutoring system
4. Trading agent
5. Smart house

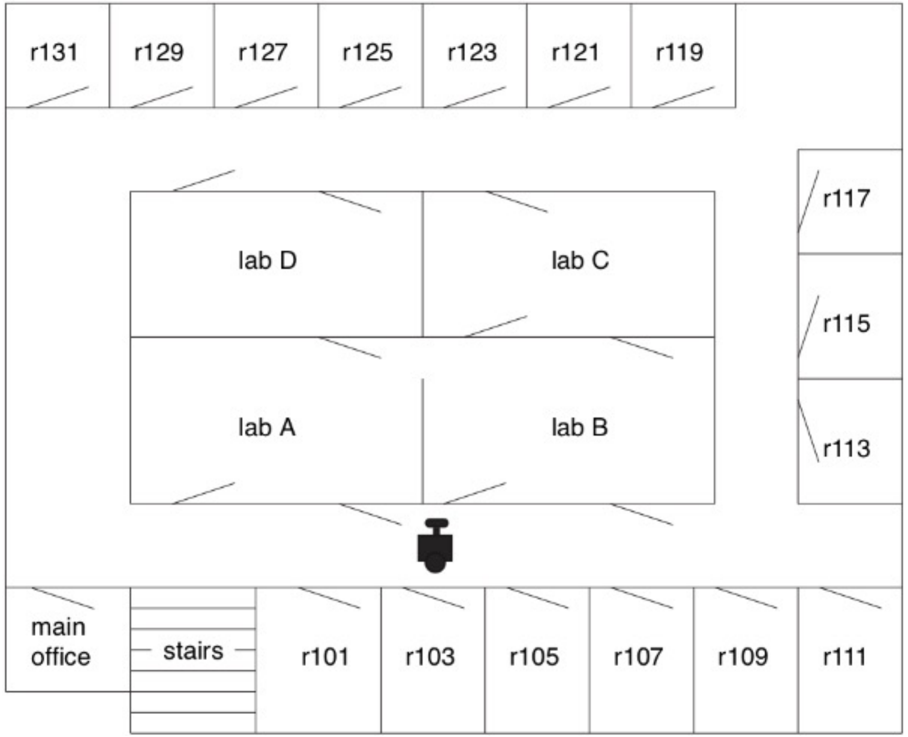
The delivery robot

An autonomous delivery robot moves around a building delivering packages and coffee.

It has sensors to avoid obstacles, the ability to carry objects, and wheels to move around.

This delivery agent should be able to:

- ✓ find paths
- ✓ allocate resources
- ✓ receive requests from people [in NL]
- ✓ make decisions about priorities
- ✓ deliver packages/coffee
- ✓ avoid injuring people or itself.



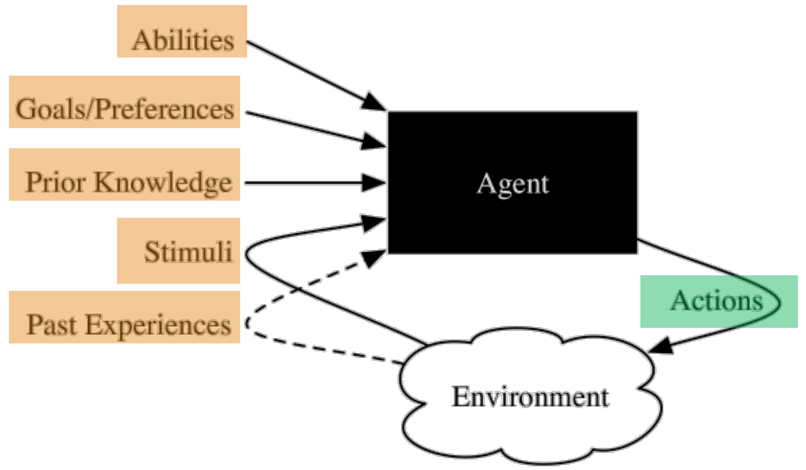
Delivery robot as a black box

Input:

- Prior knowledge, provided by the designer, about the environment and agent's **capabilities**
- **Past experience** about the environment
- **Stimuli** from sensors or other input devices: infrared, cameras, touch, whiskers, voice input, keyboards …
- **Goals**: what needs to be delivered and when. Preferences.

Output:

- Wheel controls, gripper movements, speech, video displays …



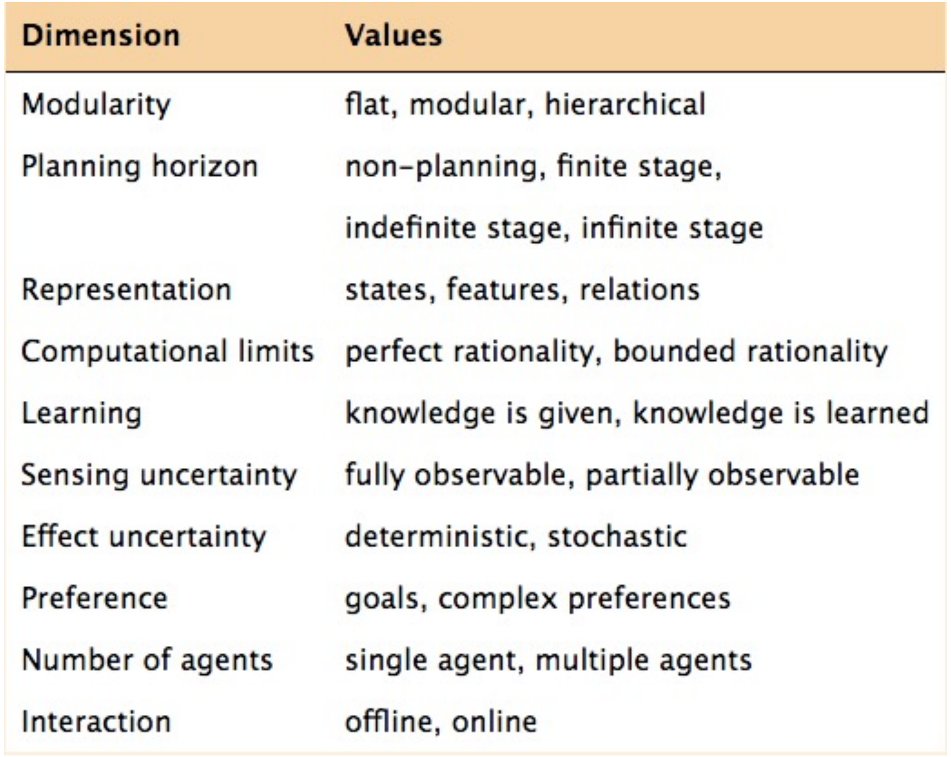
Delivery Robot: dimensions of complexity

Simplest scenario: flat system, finite set of atomic states, no uncertainty, simple recognition of goal states, no preferences => planning as graph search

Complexity may arise in all dimensions:

- ✓ Modularity: it may be necessary to build a hierarchy of modules (see next).
- ✓ Representation: states represented as set of features, or a set of related objects => planning and problem solving can exploit specialized techniques
- ✓ Goals may be more complex and conflicting, with preferences and trade-offs
- ✓ Uncertainty in sensors
- ✓ Uncertainty in the result of actions.
- ✓ Multiple robots in the same environments that need to cooperate and compete for plugs
- ✓ There is a lot to learn from experience on how to improve performance.
- ✓ If the environment is unpredictable you cannot plan in advance, you have to be able to adapt

Summary



Dimension	Values
Modularity	flat, modular, hierarchical
Planning horizon	non-planning, finite stage, indefinite stage, infinite stage
Representation	states, features, relations
Computational limits	perfect rationality, bounded rationality
Learning	knowledge is given, knowledge is learned
Sensing uncertainty	fully observable, partially observable
Effect uncertainty	deterministic, stochastic
Preference	goals, complex preferences
Number of agents	single agent, multiple agents
Interaction	offline, online

Agent architectures

LESSON 2: AGENT FUNCTIONS, AGENT ARCHITECTURES, HIERARCHICAL CONTROL (AI-FCA: CHAPTER 2)

Agents

An agent is something that interacts with an environment, receiving information through its **sensors** and acts in the world through **actuators** (effectors)

- A robot is an artificial purposive agent with a physical body (**embodied**).
- A computer program is a software agent living in a digital environment.

We will turn now our attention to:

- how an intelligent agent can perceive, reason, and act over time in an environment
- the internal structure of an agent
- in particular the notion of hierarchical control

Agent systems

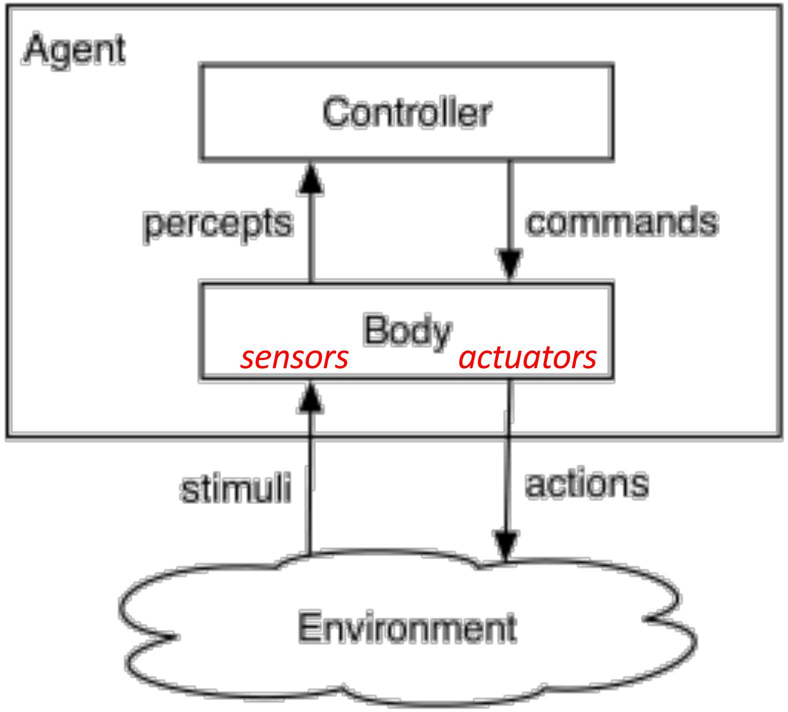
An agent is made up of a **body** and a **controller**. The controller receives percepts from the body and sends commands to the body.

A body includes **sensors** that convert **stimuli** into percepts and **actuators** that convert **commands** into actions.

Both sensor and actuators can be sources of uncertainty

The controller is the brain of an agent.

An **agent system** includes an agent and the environment in which it acts.



Agents in time

Agents act in time. T is a set of time points. Assume for simplicity that T has a start (0) is totally ordered, discrete, and each t has a next time $t+1$

- **Percept trace**/stream: a function of time into percepts (past, present, future)
- **Command trace**: a function of time into commands (past, present, future)
- Agent **history** at time t: percepts up to t and commands up to t−1

- **Causal transduction**: a function from history to commands
  - ✓ The term *transduction* comes from Finite State Transducers, where both new states and commands are emitted.
  - ✓ "*Causal*" because only previous and current percepts and previous commands can be considered
- A controller ideally implements a causal transduction

Example: trading agent

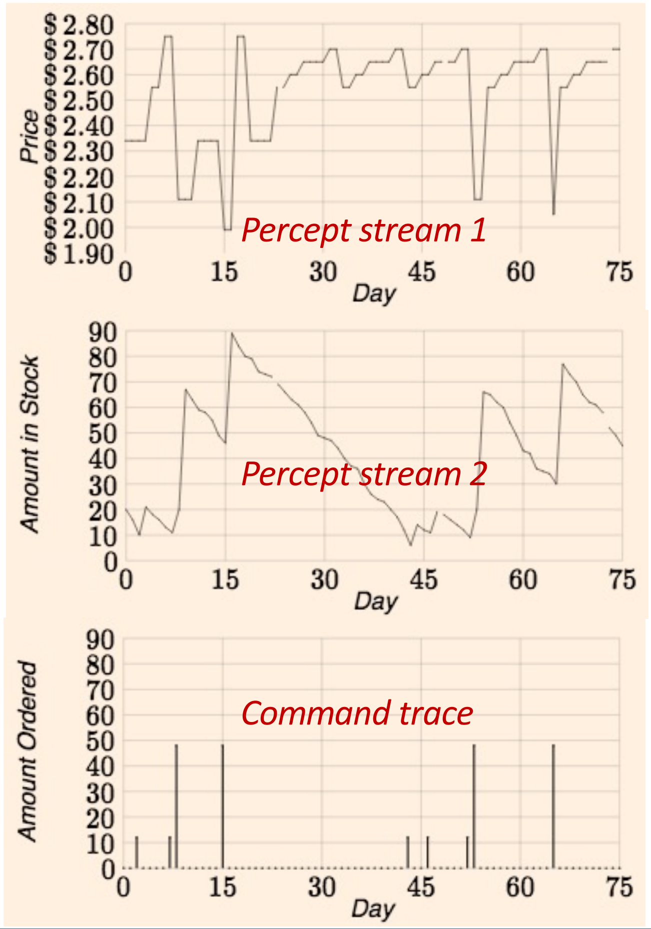
Our trading agent monitors the price of some goods, e.g. toilet paper, in order to decide when to buy and how many rolls to buy.

How much toilet paper the agent should buy depends on the price history, the history of the commodity in stock and the past history of buying.

The picture shows

- the percept stream for *price*
- for percept stream for *commodity in stock*
- the command trace

A causal transduction would map history at time *t* to a specific buy command



Command/agent function

However, complete history is usually not available … only the memory of it

The *belief state* of an agent at time t is all the information the agent *remembers* from the previous times.

The behavior of an agent can then be described by two functions:

A **belief state transition** function:

$remember: S \times P \rightarrow S$ $\quad$ *S is the set of belief states, P the set of percepts*

A **command** function:

$command: S \times P \rightarrow C$ $\quad$ *C is the set of commands*

The controller implements the *remember* and *command functions* (an approximation of a causal transduction).

Example: trading agent

Suppose we decide that the correct amount to buy depends on the average price in the last 20 days.

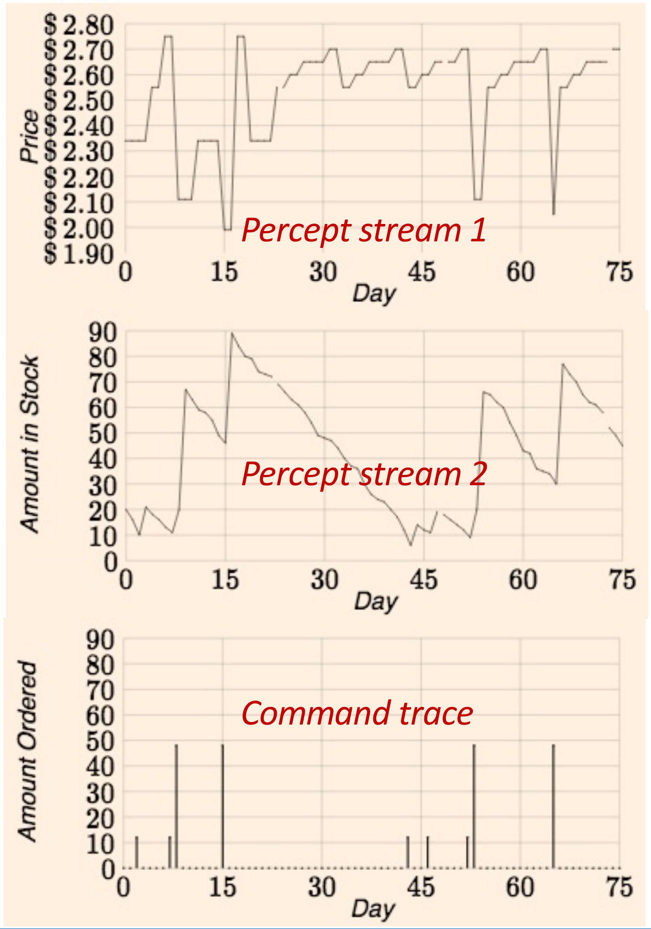
- A **causal transducer** may recompute the average price in an updated 20-days window

$$average := average + (new - old)/20$$

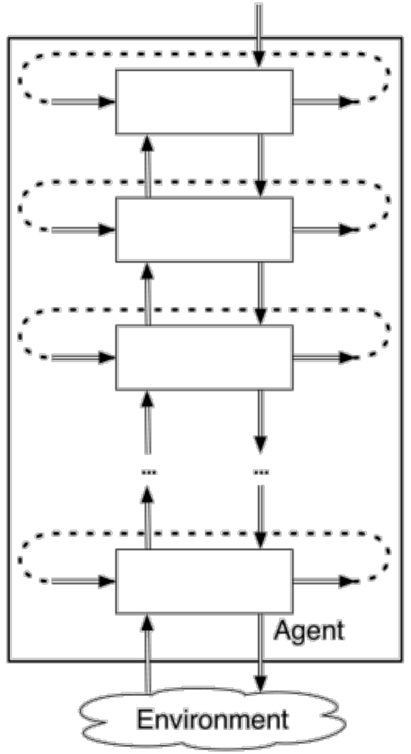
- **Command function** may be based on maintaining in memory a single quantity *ave* which is only an approximation of *average*

$$ave := ave + (new - ave)/20$$

We have a **finite state controller** when there are a finite number of belief states.

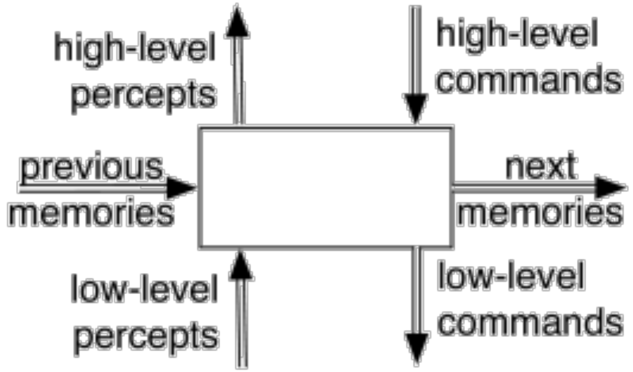


Hierarchical control



With a single controller … *it is difficult to reconcile the slow reasoning about complex high-level goals with the fast reaction that an agent needs for lower-level tasks such as avoiding obstacles*.

Reminds of Kahneman distinction between System 1 and System 2



3 inputs
3 outputs

Functions for layered control

Functions to be implemented at each layer: (l : lower; h : higher)

$remember: S \times P_l \times C_h \rightarrow S$	*belief state transition function*
$command: S \times P_l \times \mathrm{C}_h \rightarrow C_l$	*command function*
$tell: S \times P_l \times C_h \rightarrow P_h$	*higher-percept function*

where:

- S is the belief state of the level
- C_h is the set of commands from the higher layer
- P_l is the set of percepts from the lower layer
- C_l is the set of commands for the lower layer
- P_h is the set of percepts for the higher layer

Hybrid systems

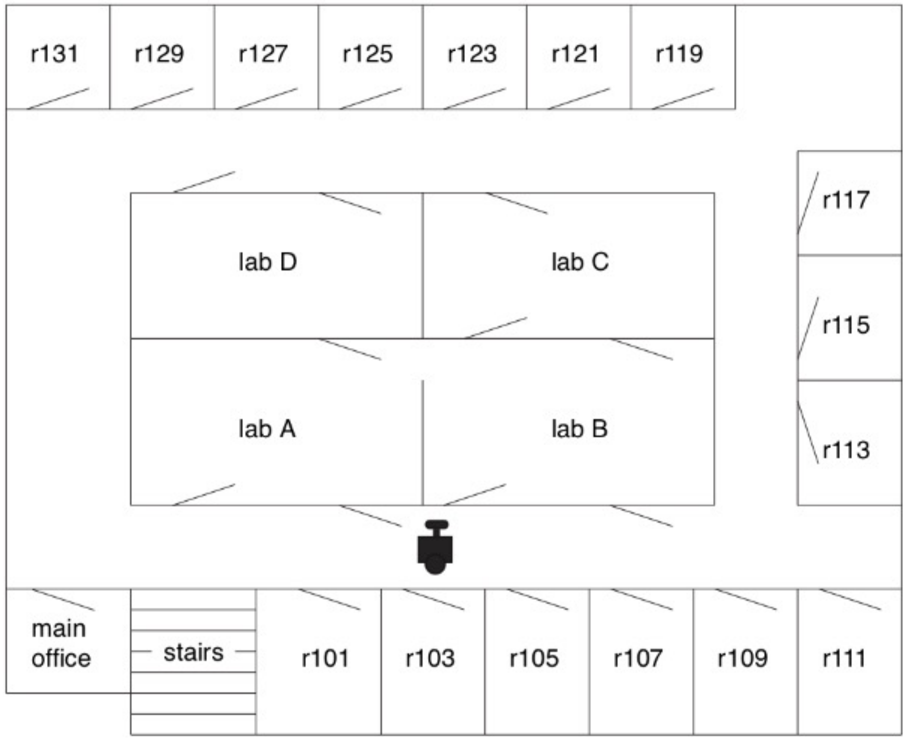
- high-level reasoning, is often discrete and qualitative, often implemented with logic based reasoning systems
- low-level reasoning is often continuous and quantitative, implemented with differential and integral calculus

A controller that reasons in terms of both discrete and continuous values is called a **hybrid system**.

Example: delivery robot

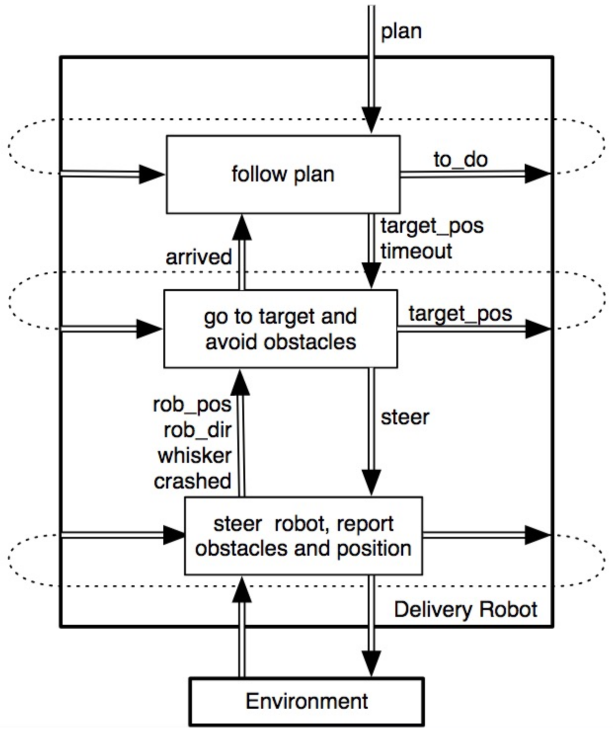
- The robot is given the task of visiting a sequence of named locations , avoiding obstacles.
- It has wheels and three actions: go straight, go right, go left. Its velocity doesn’t change. It can only set the steering angle.
- It has a position sensor, it knows position and orientation, but no map.
- It has a single whisker sensor pointing forward and to the right. The robot can detect if the whisker hits an object.
- Obstacles can be moved dynamically and new locations can be created.

The delivery robot scenario



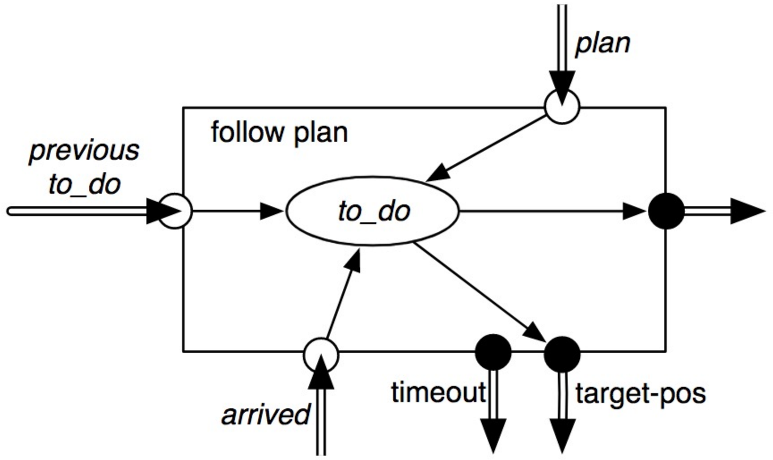
A decomposition for the Delivery Robot

Three layers



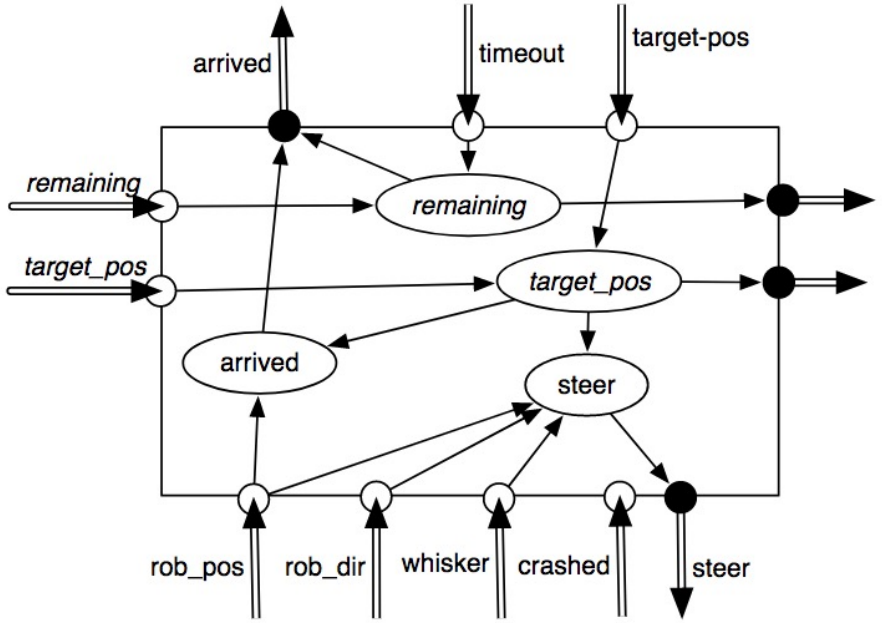
Top layer

- The top layer is given a plan which is a sequence of named locations to visit in sequence
- The top layer tells the middle layer the coordinates of the target position for the next location to visit and a *timeout*
- It has to remember the current goal position and the locations still to visit.
- When the middle layer reports the robot has arrived, the top layer takes the next location from the list of positions to visit, and makes it the new target position.



The middle layer

Go to target position and avoid obstacles



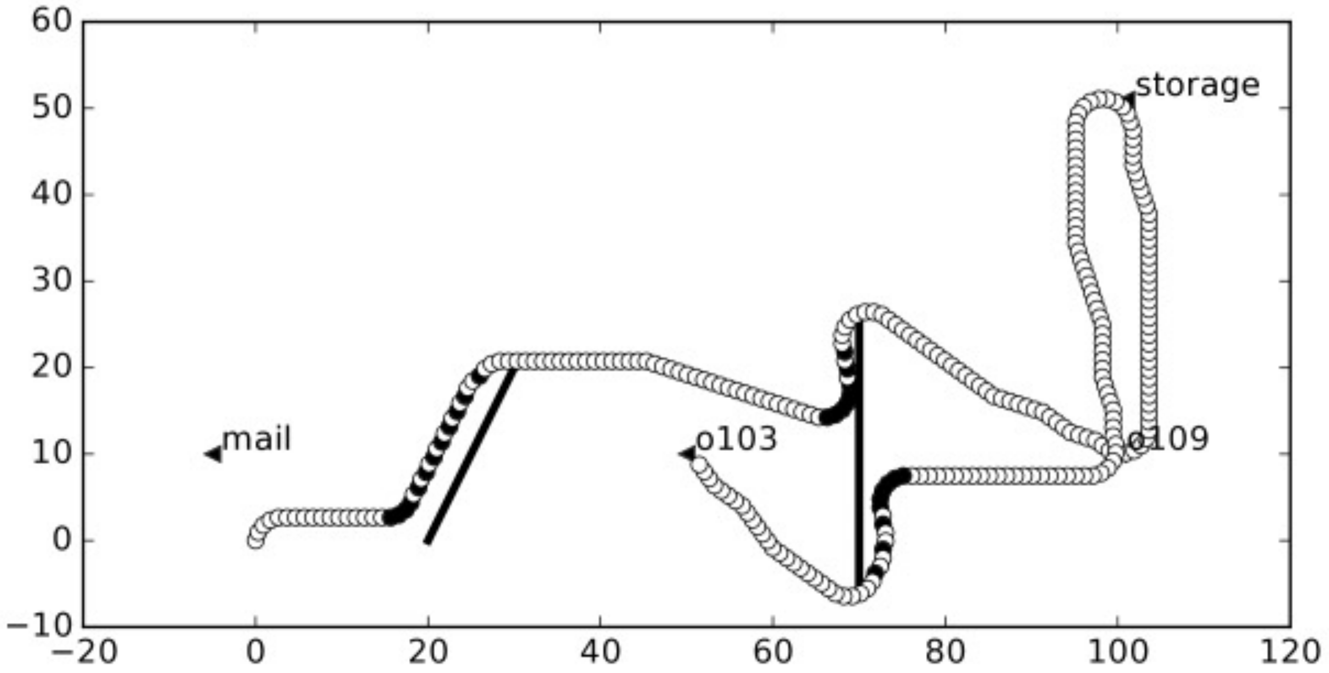
The lowest level is in charge of steering and reporting robot position and whether there are obstacles

A controller for the middle layer

```
given $timeout$ and $target\_pos$:
    $remaining \leftarrow timeout$
    while not $arrived()$ and $remaining \neq 0$
        if $whisker\_sensor = on$
            then $steer := left$
        else if $straight\_ahead(rob\_pos, robot\_dir, target\_pos)$
            then $steer := straight$
        else if $left\_of(rob\_pos, robot\_dir, target\_pos)$
            then $steer := left$
        else $steer := right$
        $do(steer)$
        $remaining \leftarrow remaining - 1$
    tell upper layer $arrived()$    # true or false
```

A simulation for the Delivery Robot

Plan: [$goto$(o109), goto(storage), goto(o109), $goto$(o103)]



Knowledge requirements

The notion of **belief state** is quite general, most agents need to keep a **model** of the word and update it while acting.

Two extremes:

1. The agent possess a very good predictive model: it does not need to use perceptions to update the model
2. Purely **reactive systems** do not have a model, and decide only on the basis of perceptions

In the general case the agent uses a combination of **prediction** and **sensing:**

1. In Bayesian reasoning (under uncertain information) the estimation of the next belief state is called **filtering** (part IV of the course).
2. In alternative, more complex models of the world can be kept and updated, for example through vision and image processing.

Knowledge and action

Knowledge of a specific domain may also be represented explicitly and used to decide the action.

The **knowledge base** contains general rules and specific/contingent facts in declarative form.

The KB is built offline, built by designers or learned from data.

A **Domain Ontology** gives meaning to symbols used to represent knowledge.

Knowledge may be then updated and used to decide actions during operation.

We will talk about **knowledge based systems** in section 2 of the course.

Conclusions

- We introduced a vision of AI as an integrated science whose goal is to build **intelligent computational agents**.
- The design of an intelligent system involves several dimensions of complexity that will be further addressed in the course.
- The design of complex AI systems may require working at different levels of abstractions, possibly with different techniques at different levels.

References

[AI-FCA] David L. Poole, Alan K. Mackworth. *Artificial Intelligence: foundations of computational agents*, Cambridge University Press, 2017. http://artint.info/2e/html/ArtInt2e.html (online version)

Chapter 1: Artificial Intelligence and agents

Chapter 2: Agent architectures and hierarchical control

Next

1. We will review the AI paradigm of *problem solving* as search in state space
2. We will focus to the techniques developed for CSP problems, which adopt a featured representation for states.

Your turn

1. Read chapter 1 and 2 of the online book “**Artificial Intelligence: Foundations of Computational Agents”** by Poole and Macworth http://artint.info/2e/html/ArtInt2e.html
2. Do the quiz Your Turn 2 on Moodle (Two simple questions about agents).