Introduction to Multi-Agent Systems

Michal Jakob
Agent Technology Center, Dept. of Computer Science and Engineering, FEE, Czech Technical University

AE4M36MAS Autumn 2013 - Lect. 1

Selected illustrations taken Russel and Norvig – Artificial Intelligence: Modern Approach

General Information

Lecturers: Prof. Michal Pěchouček and Dr. Michal Jakob

Tutorials: Branislav Bošanský and Michal Čáp

13 lectures and 13 tutorials 24th September – 17th December

Course web page: https://cw.felk.cvut.cz/doku.php/courses/ae4m36mas/start

Recommended reading:

- J. M. Vidal: Multiagent Systems: with NetLogo Examples (available on-line)
- Y. Shoham and K. Leyton-Brown: Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations (available on-line)
- Russel and Norvig: Artificial Intelligence: Modern Approach
- M. Wooldridge: An Introduction to MultiAgent Systems
- V. Marik, O. Stepankova, J. Lazansky a kol.: Umela inteligence (3)

Course Requirements and Grading

Total 100 pts – 40 pts projects + 60 pts final exam

Semestral projects – 40 pts:

- Project #1 (9 pts) – due end of October (TBC)
- Project #2 (14 pts) – due end of November (TBC)
- Project #3 (17 pts) – due at least a weak before you want a course assessment (end of semester)

Final exam – 60 pts

At least 50% points from each part required to pass

Outline of Lecture 1

1. Motivational Introduction
2. Defining Agency
3. Specifying Agents
4. Agent Architecturess

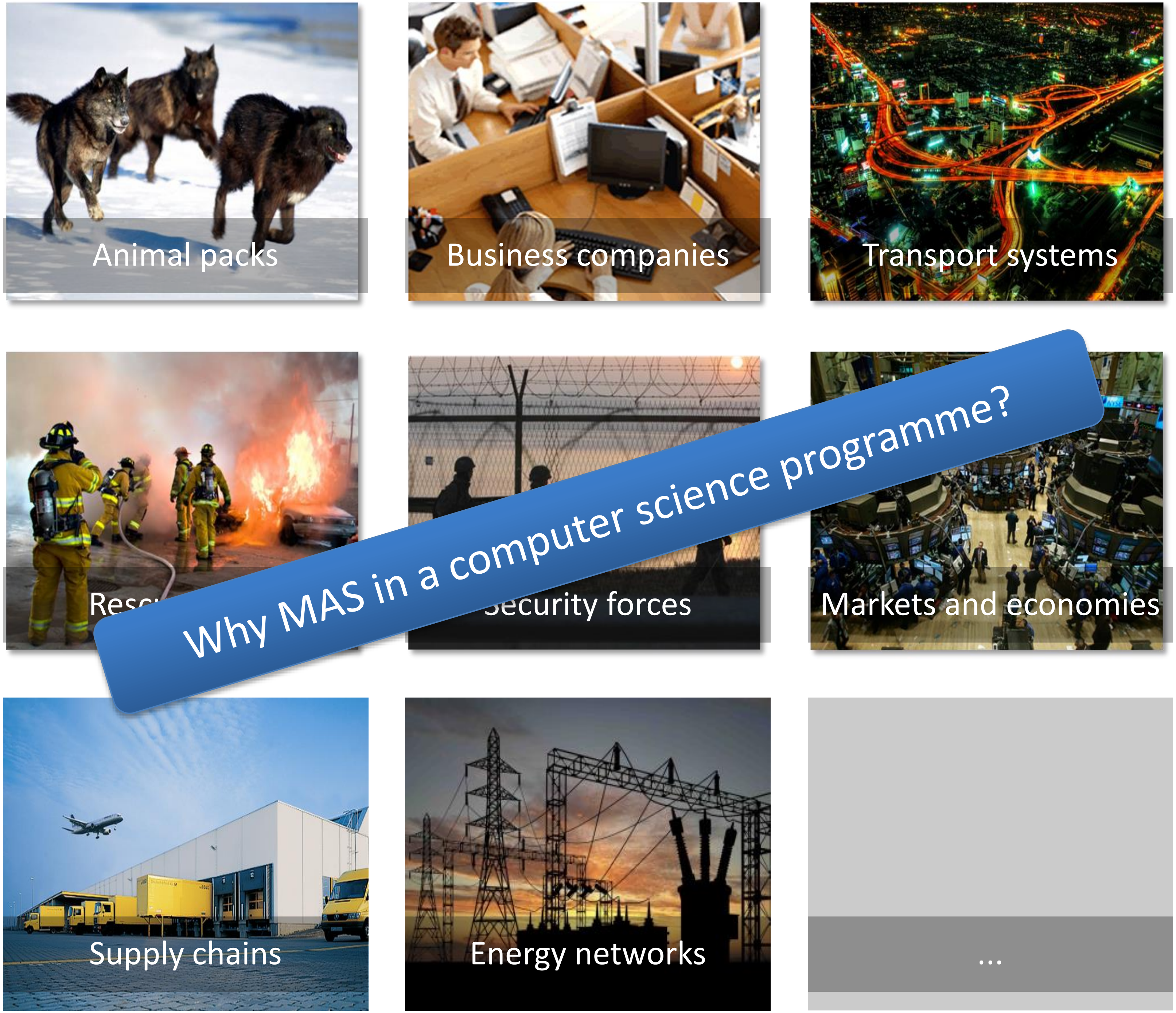
Introduction to Multiagent Systems

Motivational Introduction

Autonomous Agents and Multiagent Systems (MAS)

Multiagent system is a collection of multiple **autonomous agents**, each acting towards its **objectives** while all **interacting** in a **shared environment**, being able to **communicate** and possibly **coordinate** their actions.

Autonomous agent ~ intelligent agent (see later).


Animal packs
Business companies
Transport systems
Resc
ecurity forces
Markets and economies
Supply chains
Energy networks
...
Why MAS in a computer science programme?

Trends in Computing

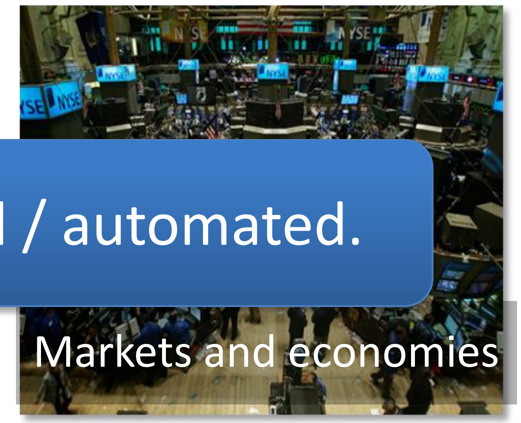
Ubiquity: Cost of processing power decreases dramatically (e.g. Moore's Law), computers used everywhere

Interconnection: Formerly only user-computer interaction, nowadays distributed/networked machine-to-machine interactions (e.g. Web APIs)

Complexity: Elaboration of tasks carried out by computers has grown

Delegation: Giving control to computers even in safety-critical tasks (e.g. aircraft or nuclear plant control)

Human-orientation: Increasing use of metaphors that better reflect human intuition from everyday life (e.g. GUIs, speech recognition, object orientation)



MAS are increasingly computerized / automated.

...

Goals for MAS

Develop formal models, data structures and algorithms to

1. **Understand** how MASes operate.
2. **Design** new MASes or **improve the** behavior of existing MASes.

Multiagent Systems Engineering

Novel paradigm for building robust, scalable and extensible control, planning and decision-making systems

- socially-inspired computing
- self-organized teamwork
- collective (artificial) intelligence

MAS become increasingly relevant as the connectivity and intelligence of devices grows!

Systems of the future will need to be good at teamwork

New Challenges for Computer Systems

Traditional design problem: *How can I build a system that produces the correct output given some input?*

- Each system is more or less isolated, built from scratch

Modern-day design problem: *How can I build a system that can operate independently on my behalf in a networked, distributed, large-scale environment in which it will need to interact with different other components pertaining to other users?*

- Each system is **built into** an existing, persistent but constantly evolving *computing ecosystem* – it should be robust with respect to changes
- No single owner and/or central authority

Topics in Multiagent Systems

How should agent’s **objectives** be specified?

How should agent’s **control logic** be implemented so that the agents acts towards its objectives?

What **languages** should agents use to communicate their beliefs and aspirations?

Which **protocols** should agents use to negotiate and agree/choose if there are multiple options (as there always are)?

How should agents in a **team decompose and allocate tasks** so as to effectively achieve team’s common goal?

How should the agent **maximize its utility** in the presence of other competing and possible hostile agents?

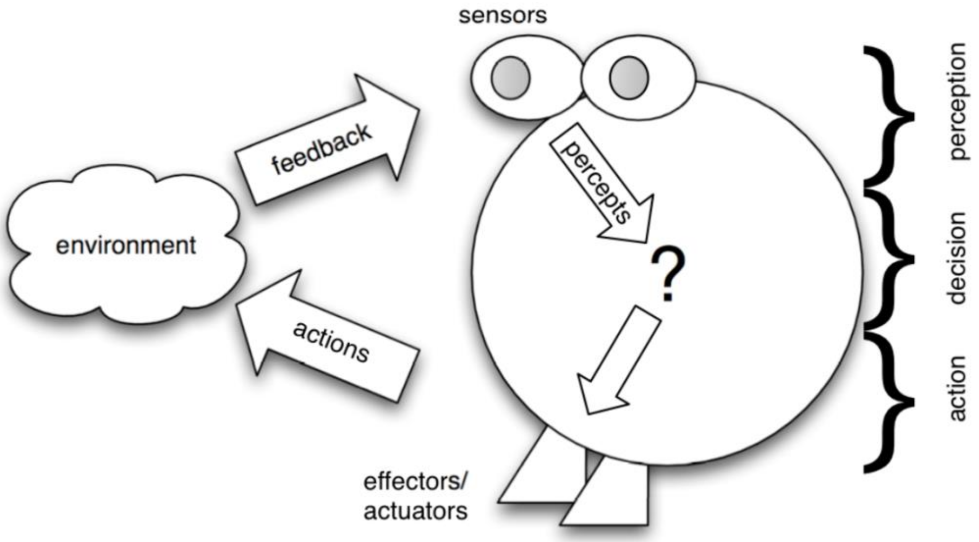
Which **voting mechanisms** are robust against manipulation?

. . .

Introduction to Multi-Agent Systems

Defining Agency

What is Agent?



Definition (Russell & Norvig)

- An agent is anything that can perceive its environment (through its sensors) and act upon that environment (through its effectors)

Focus on **situatedness** in the environment (**embodiment**)

The agent can only influence the environment but not fully control it (sensor/effector failure, non-determinism)

What is Agent? (2)

Definition (Wooldridge & Jennings)

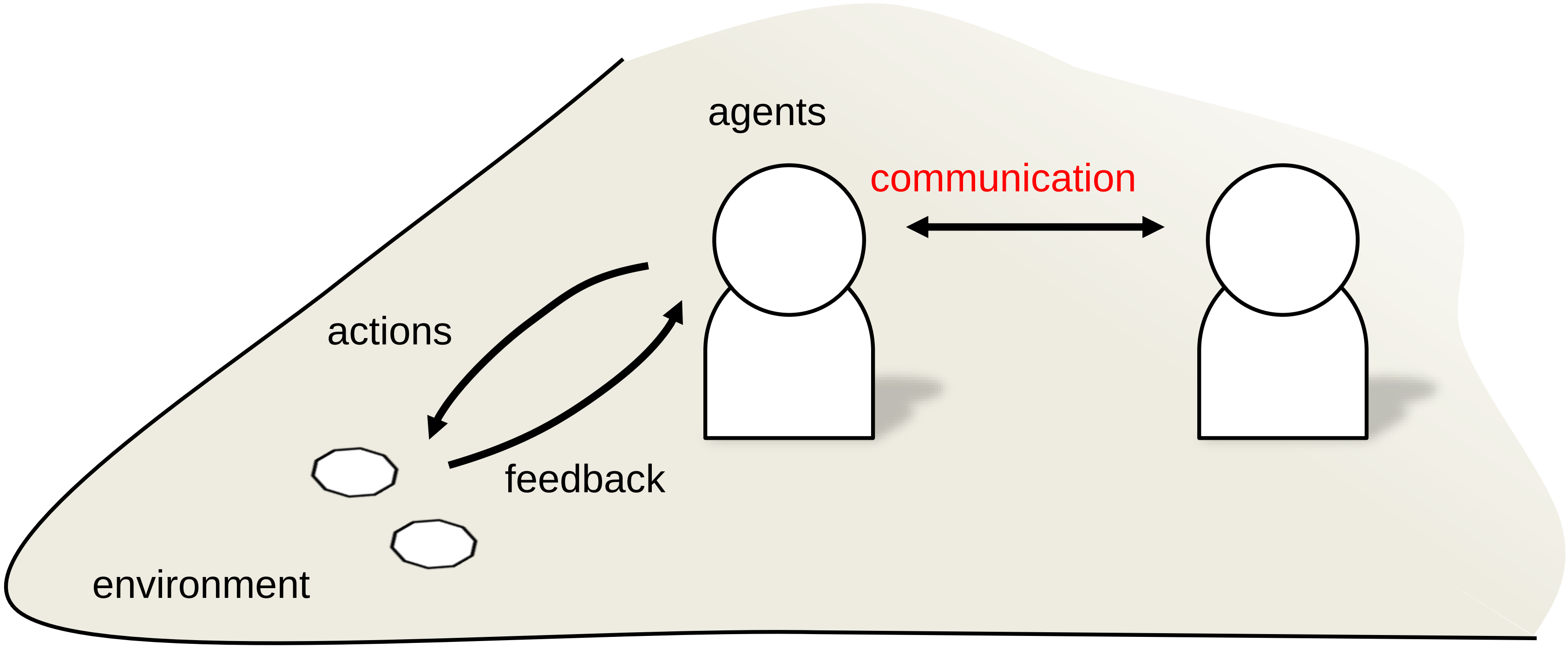
- An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to **meet its design objectives/delegated goals.**

Adds a second dimension to agent definition: the relationship between agent and designer/user

- agent is capable of independent action
- agent action is purposeful

Autonomy is a central, distinguishing property of agents

Multiagent Systems



Autonomous Agent Properties

autonomous – the agent is self goal-directed and acts without requiring user initiation and guidance; it can choose its own goal and the way to achieve it; its behavior is determined by its experience; we have no direct control over it

reactive – the agent maintains an ongoing interaction with its environment, and responds to changes that occur in it

proactive – the agent generates and attempts to achieve goals; it is not driven solely by events but takes the initiative

Autonomous Agent Properties

sociable – the agent interacts with other agents (and possibly humans) via cooperation, coordination, and negotiation; it is aware and able to reason about other agents and how they can help it achieve its own goals

- **coordination** is managing the interdependencies between actions of multiple agents (not necessarily cooperative)
- **cooperation** is working together as a team to achieve a shared goal
- **negotiation** is the ability to reach agreements on matters of common interest

Agents vs. Objects

An agent has unpredictable behaviour as observed from the outside

- unless its simple reflexive agent

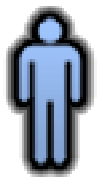
An agent is *situated* in the environment

Agent communication model is *asynchronous*

Objects do it for free; agents do it because they want to

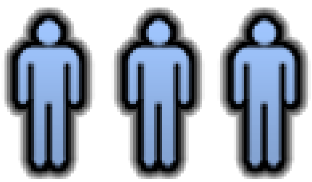
Types of Agent Systems

single-agent



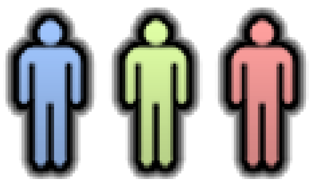
multi-agent

cooperative



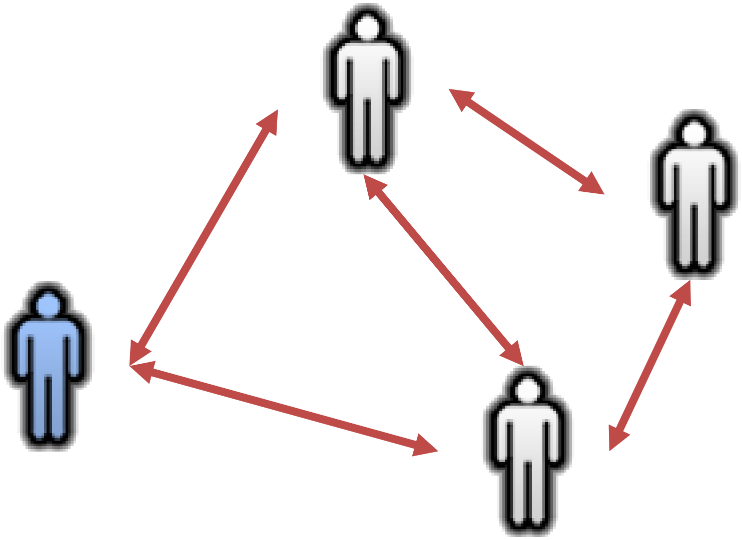
single shared utility

competitive



multiple different utilities

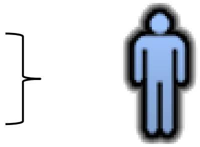
Micro vs. Macro MAS Engineering



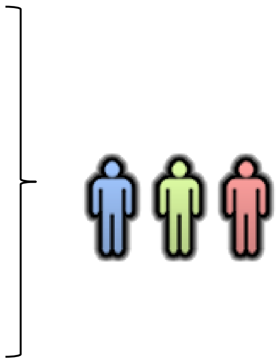
1. The **agent design problem (micro perspective)**:
   How should agents act to carry out their tasks?
2. The **society design problem (macro perspective)**:
   How should agents interact to carry out their tasks?

Course Content

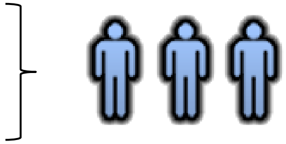
- Agent architectures (inc. BDI architecture)
- Logics for MAS



- Non-cooperative game theory
- Coalition game theory
- Mechanism design
- Auctions
- Social choice



- Distributed constraint reasoning (satisfaction and optimization)



Introduction to Multiagent Systems

Specifying Agents

Agent Behavior

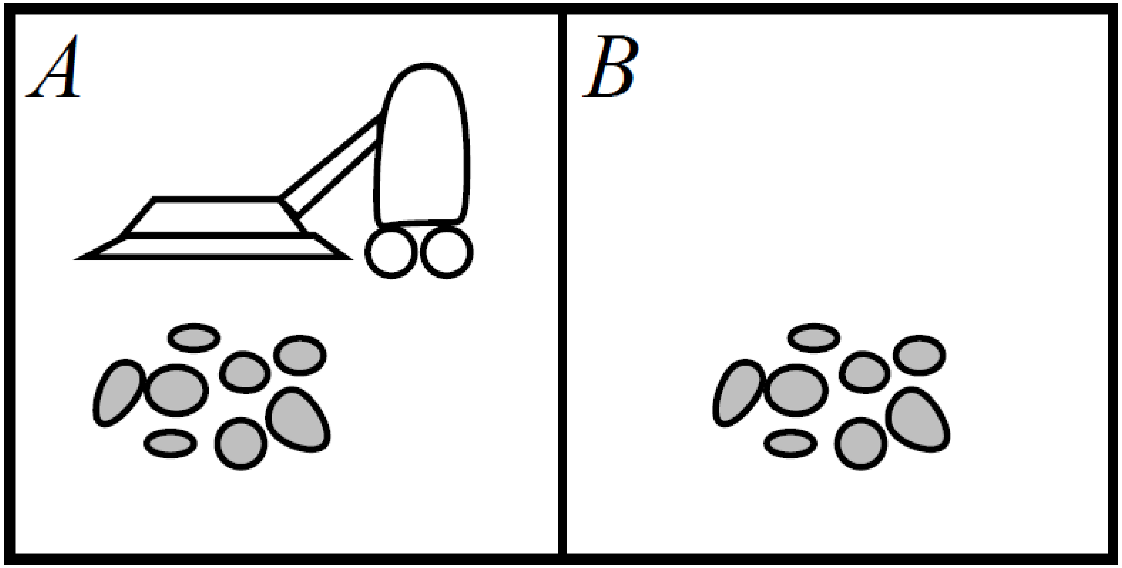
$$f : \mathscr{P} \mapsto \mathscr{A}$$

Agent's behavior is described by the **agent function** that maps percept sequences to actions

The **agent program** runs on a physical architecture to produce *f*

Key questions: What is the right function? Can it be implemented in a small agent program?

Example: Vacuum Cleaner World



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

Actions: Left, Right, Suck, NoOp

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
…	…
[A,Clean], [A,Clean], [A,Clean]	Right
[A,Clean], [A,Clean], [A, Dirty]	Suck
…	…

Is this a good agent function?

Rational Behavior

Definition (Russell & Norvig)

- Rational agent chooses whichever action maximizes the expected value of the performance measure given the percept sequence to date and whatever bulit-in knowledge the agent has.

Rationality is relative and depends on four aspects:

1. performance measure which defines the degree of success
2. percept sequence (complete perceptual history)
3. agent's knowledge about the environment
4. actions available to the agent

Rational ≠ omniscient, rational ≠ clairvoyant => rational ≠ successful

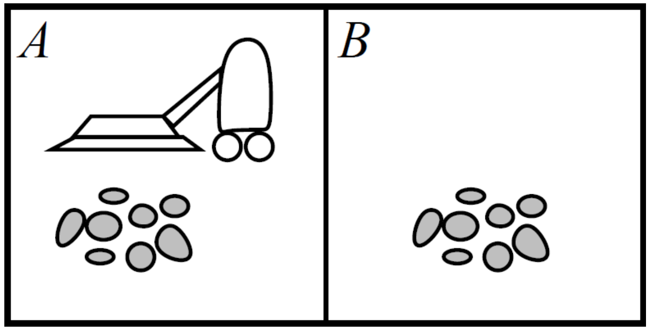
Specifying Task Environments

To design a rational agent, we must specify the **task environment (PEAS)**

1. Performance measure
2. Environment
3. Actuators
4. Sensors

Task environments define problems to which rational agents are the solutions

Rationality of Vacuum Cleaner Agent



Agent programme:
Cleans a square if it is dirty and moves to the other square if not. **Is it rational?**

PEAS:

- The performance measure awards one point for each clean square at each time step, over a "lifetime" of 1000 time steps.
- The "geography" of the environment is known a priori but the dirt distribution and the initial location of the agent are not. Clean squares stay clean and sucking cleans the current square. The Left and Right actions move the agent left and right except when this would take the agent outside the environment, in which case the agent remains where it is.
- The only available actions are Left, Right, and Suck.
- The agent correctly perceives its location and whether that location contains dirt.

Yes, we can prove no other agent does better.

PEAS Examples

Agent	Performance measure	Environment	Actuators	Sensors

Properties of Environments

Fully observable vs. partially observable – can agents obtain complete and correct information about the state of the world?

Deterministic vs. stochastic – Do actions have guaranteed and uniquely defined effects?

Episodic vs. sequential – Can agents decisions be made for different, independent episodes?

Static vs. dynamic – Does the environment change by processes beyond agent control?

Discrete vs. continuous – Is the number of actions and percepts fixed and finite?

Single-agent vs. multi-agent – Does the behavior of one agent depends on the behavior of other agents?

Example Environments

	Solitaire	Backgammon	Internet shopping	Taxi
Observable				
Deterministic				
Episodic				
Static				
Discrete				
Single-agent				

True or false?

An agent that senses only partial information about the state cannot be perfectly rational.

There exists a task environment in which every agent is rational.

Every agent function is implementable by some program/machine combination.

Every agent is rational in an unobservable environment.

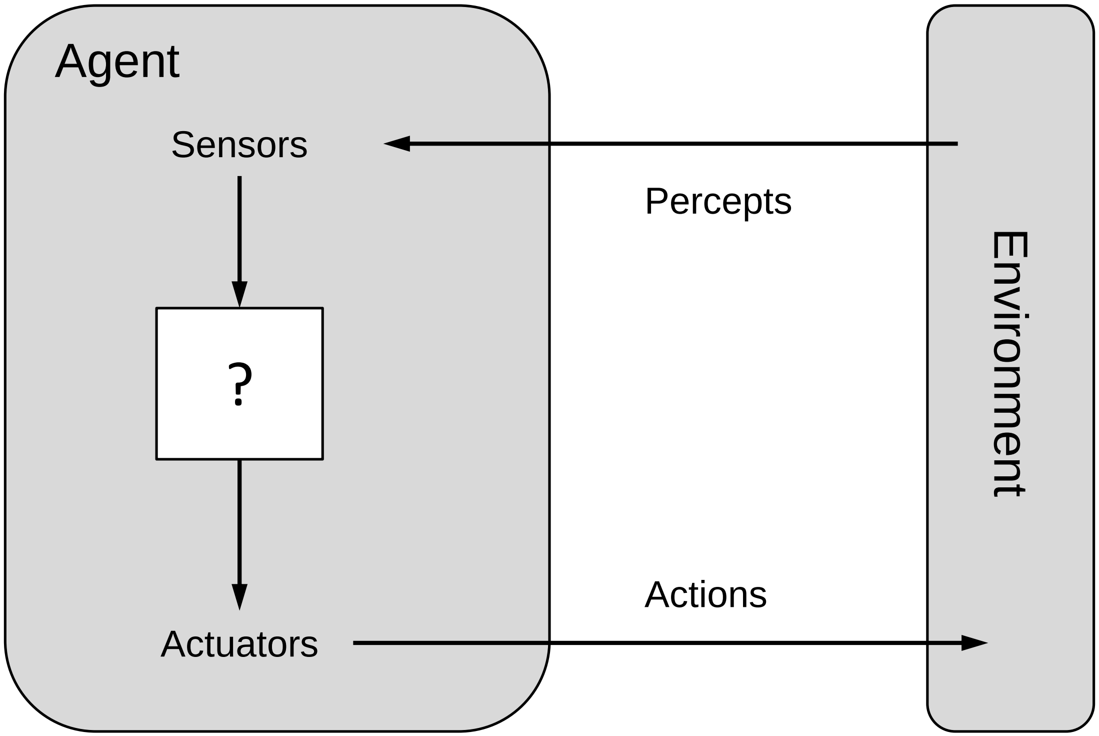
A perfectly rational poker-playing agent never loses.

Introduction to Agents

Agent Architectures

Implementing the Agent

How should one implement the agent function?



Concern 1: Rationality

Concern 2: Computability and tractability

Hierarchy of Agents

The key challenge for AI is to find out how to write programs that produce rational behavior from a small amount of code rather than from a large number of table entries.

4+1 basic types of agents in the order of increasing capability:

1. simple reflex agents
2. model-based agents with state
3. goal-based agents
4. utility-based agents
5. (learning agents)

There is a link between the complexity of the task and the minimum agent architecture required to implement a rational agent.

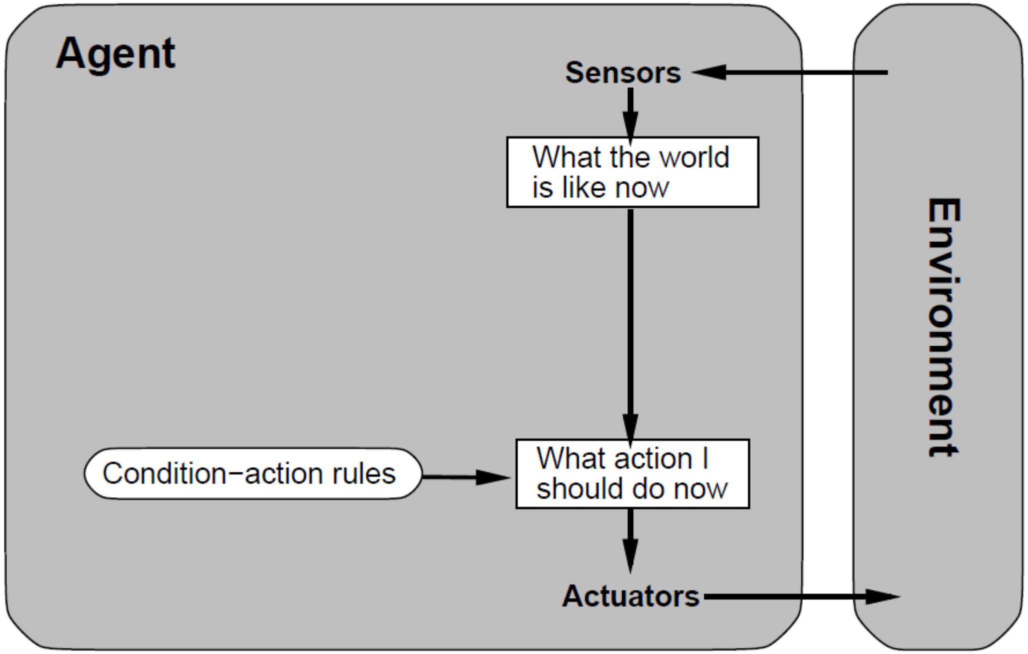
Running Example: Robotic Taxi

Task specification

- Performance measure: the overall profit (= passenger revenues - fines)
- Environment: road network with traffic signs, passengers
- Actions (actuators): driving between junctions, picking up and dropping out passengers
- Percepts (sensors): current GPS location, junction layout, traffic signs, passengers

Simple Reflex Agents

Simple reflex agent chooses the next action on the basis of the current percept only.



Simple Reflex Agent

Condition-action rules provide a way to present common regularities appearing in input/output associations

- Ex.: `if car-in-front-is-braking then initialize-braking`

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

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

Simpe Reflex Agent for Robotic Taxi

Simple program:

- If a passenger at your location => pickup the passenger
- Otherwise: Continue in the left-most direction possible

More sophisticated program:

- Turn-directions depend on the current GPS location (can implement specific fixed route through the city)

Issues with Reflex Agents

Robotic taxi

- driving to a given destination
- respecting traffic signs (e.g. speed limits)
- getting stuck in loops

In general: Reflex agents are simple but of limited intelligence – the only work if

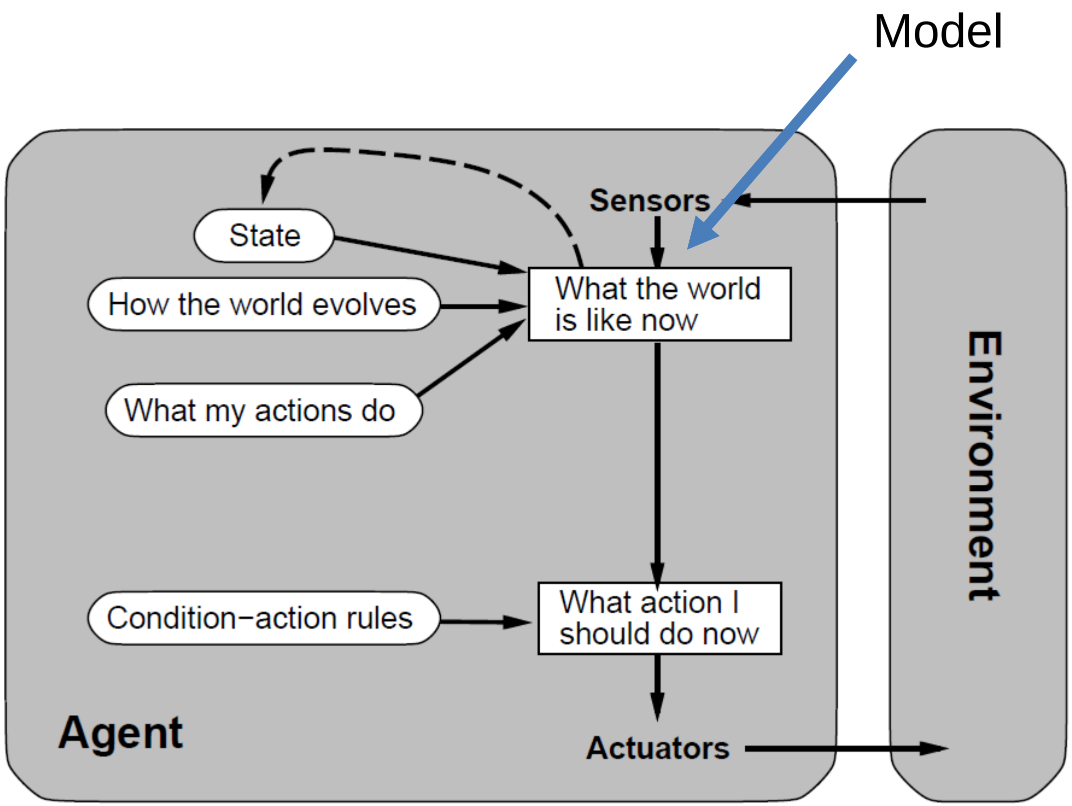
1. the environment is fully observable and
2. the decision can be made based solely on the current percept

If the above not the case => suboptimal action choices, infinite loops.

=> It can be advantageous to **store information about the world** in the agent.

Model-based Reflex Agent

Keeps track of the world by extracting relevant information from percepts and storing it in its memory.



Model-based Reflex Agent

```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              model, a description of how the next state depends on current state and action
              rules, a set of condition–action rules
              action, the most recent action, initially none

  state ← UPDATE-STATE(state, action, percept, model)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

Model-based Reflex Taxi Agent

States tracked in the model

- passengers' destinations
- traffic signs
- visited locations (to avoid cycles)
- pickup locations (=> learning)

Issues with Model-based Agents

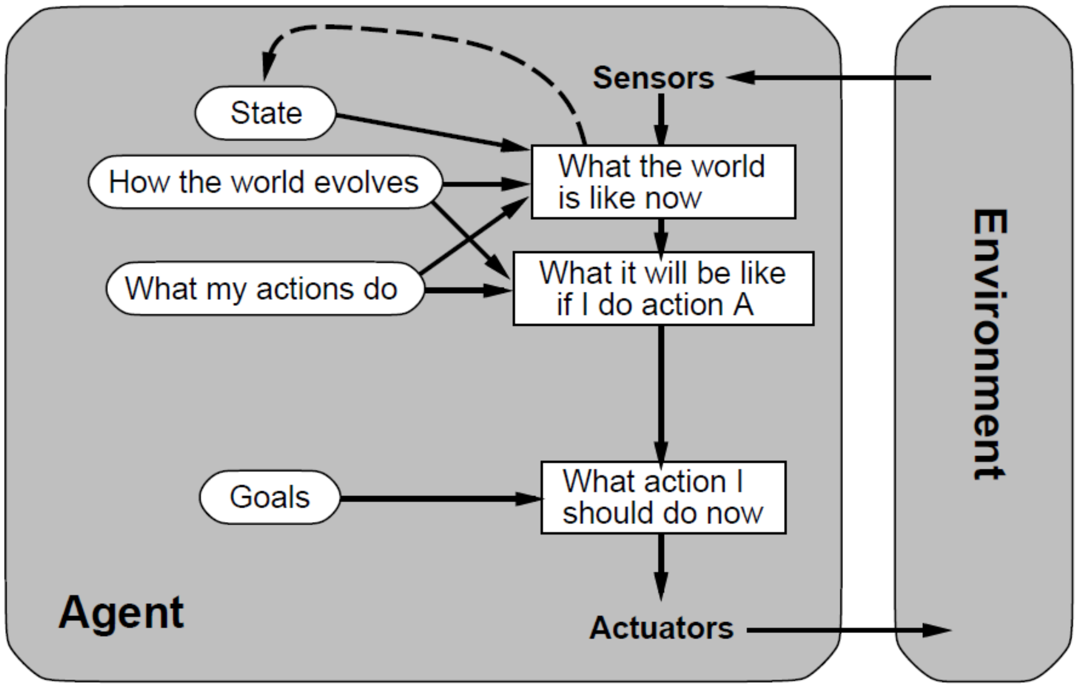
Taxi agent: Hot to get to a destination?

- Always move towards the destination location => can end-up in dead end streets
- Hard-code routes between all locations
  - memory demanding and of limited intelligence
  - e.g. requires reprogramming the agent if street network changes

Cause:

- *whats* and *hows* tightly coupled (impossible to tell the agent what to do)
- the agent does not anticipate the effects of its actions (only finds out the result after having executed the action)

Goal-based Agents



Goal-based agents are more flexible

Problem: goals are not necessarily achievable by a single action:

→ **search and planning**

Goal-based Taxi Agent

Uses planning

- Uses a map to find a sequence of movement actions that brings the taxi to the destination reliable

Issue

- will not choose the fastest route
- will not balance revenue vs. fees/fines

Cause: goals alone are not sufficient for decision making:

1. there may be multiple ways of achieving them;
2. agents may have several conflicting goals that cannot be achieved simultaneously.

Utility-based Agents

Goals only a very crude (binary) distinction between "happy" and "unhappy" states.

We introduce the concept of **utility**:

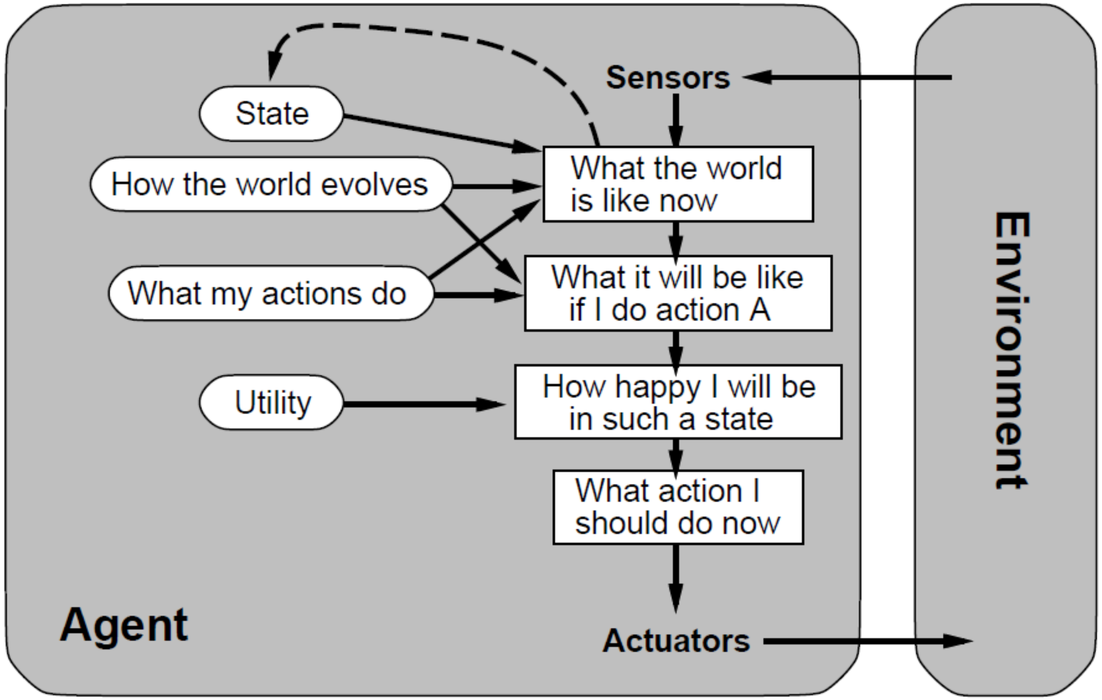
- utility is a function that maps a state onto a real number; it captures "quality" of a state
- if an agent prefers one world state to another state then the former state has higher utility for the agent.

Utility can be used for:

1. choosing the best plan
2. resolving conflicts among goals
3. estimating the successfulness of an agent if the outcomes of actions are uncertain.

Utility-based Agents

Utility-based agent use the utility function to choose the most desirable action/course of actions to take



Utility-based Taxi Agent

Uses **optimizing** planning

- searches for the plan that leads to the maximum utility

There are still issues

- irreducible preference orderings
- non-deterministic environment (→ Markov decision processes)

Summary

Multiagent systems approach ever more important in the increasingly interconnected world where systems are required to cooperate flexibly

→ "socially-inspired computing"

Intelligent agent is autonomous, proactive, reactive and sociable.

Agents can be cooperative or competitive (or combination thereof).

There are different agent architectures with different capabilities and complexity.

Related reading:

- Russel and Norvig: Artificial Intelligence: A Modern Approach – Chapter 2
- Wooldrige: An Introduction to Multiagent Systems – Chapters 1 and 2

→ **Next:** Belifef-Desire-Intention Architecture