# Introduction to Multi-Agent Systems

Michal Jakob & Michal Pechoucek
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 Karel Horak

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

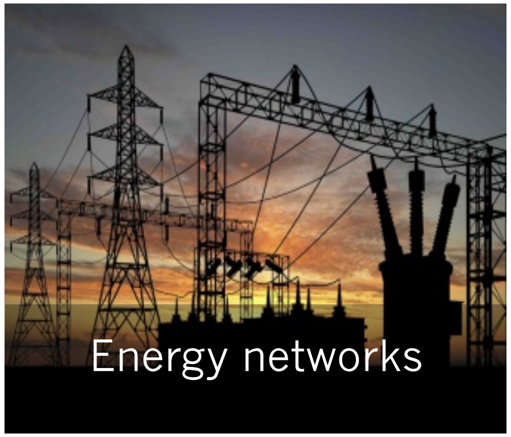
## Motivational Introduction

## 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).

## Why MAS in a computer science programme?



...

## Why Intelligent Agents?

1992: computers everywhere

- lots of computerized data
- computer driven manufacturing, production planning, diagnostics

## Why Intelligent Agents?

1992: computers everywhere

- lots of computerized data
- computer driven manufacturing, production planning, diagnostics

⬇

- AI: expert systems, automated planning, machine learning

## Why Intelligent Agents?

1992: computers everywhere

Y2K: internet everywhere

- data provisioning via internet, search (Google from 1998, in 2001 3B of docu)
- an explosion of internet shopping (Amazon from 1995, Ebay from 1996)

# Why Intelligent Agents?

1992: computers everywhere

Y2K: internet everywhere

- data provisioning via internet, search (Google from 1998, in 2001 3B of docu)
- an explosion of internet shopping (Amazon from 1995, Ebay from 1996)

↓

- parallel computing (map-reduce)
- statistical data analysis and machine learning
- networking, servers

## Why Intelligent Agents?

1992: computers everywhere

Y2K: internet everywhere

NOW: internet of everything

- mobile computing
- cloud computing
- wireless enabled devices

## Why Intelligent Agents?

1992: computers everywhere

Y2K: internet everywhere

NOW: internet of everything

- mobile computing
- cloud computing
- wireless enabled devices

↓

- Intelligent Agents and Multiagent systems

# Why Intelligent Agents?

1992: computers everywhere

Y2K: internet everywhere

NOW: internet of everything

- mobile computing
- cloud computing
- wireless enabled devices

↓

- Intelligent Agents and Multiagent

<u>Latest trends in computing</u>

**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)

# Agents briefly

multi-agent system is a decentralized multi-actor (software) system, often geographically distributed whose behavior is defined and implemented by means of complex, peer-to-peer interaction among autonomous, rational and deliberative entities.

autonomous agent is a special kind of a intelligent software program that is capable of highly autonomous rational action, aimed at achieving the private objective of the agent – can exists on its own but often is a component of a multi-agent system – agent is autonomous, reactive, proactive and social

agent researchers study problems of integration, communication, reasoning and knowledge representation, competition (games) and cooperation (robotics), agent oriented software engineering, …

agent technology is software technology supporting the development of the autonomous agents and multi-agent systems
agent-based computing is a special research domain, subfield of computer science and artificial intelligence that studies the concepts

**autonomous agent** is a special kind of a intelligent software program that is capable of highly autonomous rational action, aimed at achieving the private objective of the agent – can exists on its own but often is a component of a multi-agent system – agent is autonomous, reactive, proactive and social

**agent researchers** study problems of integration, communication, reasoning and knowledge representation, competition (games) and cooperation (robotics), agent oriented software engineering, …

**agent technology** is software technology supporting the development of the autonomous agents and multi-agent systems
agent-based computing is a special research domain, subfield of computer science and artificial intelligence that studies concepts of autonomous agents

**multi-agent application** is a software system, functionality of which is given by interaction among autonomous software/hardware/human components.

- but also a monolithic software application that is autonomously operating within a community of autonomously acting software applications, hardware systems or human individuals

## Key properties of Intelligent Agent

Autonomy: Agent is fully accountable for its given state. Agent accepts requests from other agents or the environment but decides individually about its actions

Reactivity: Agent is capable of near-real-time decision with respect to changes in the environment or events in its social neighbourhood

Intentionality: Agent maintain long term intention. the agent meets the designer's objectives. It knows its purpouse and executes even if not requested.

Rationality: Agent is capable of intelligent rational decision making. Agent can analyze future course of actions and choose an action which maximizes his utility

Social capability: Agent is aware of the either (i) existence, (ii) communication protocols, (iii) capability, services provided by the other agents. Agent can reason about other agents.

individually about its actions

**Reactivity**: Agent is capable of near-real-time decision with respect to changes in the environment or events in its social neighbourhood

**Intentionality**: Agent maintain long term intention. the agent meets the designer’s objectives. It knows its purpouse and executes even if not requested.

**Rationality**: Agent is capable of intelligent rational decision making. Agent can analyze future course of actions and choose an action which maximizes his utility

**Social capability**: Agent is aware of the either (i) existence, (ii) communication protocols, (iii) capability, services provided by the other agents. Agent can reason about other agents.

- **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*

## 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

## Multiagent Design Problem

**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

**Multiagent 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

# Types of Agent Systems

**single**-agent

**multi**-agent

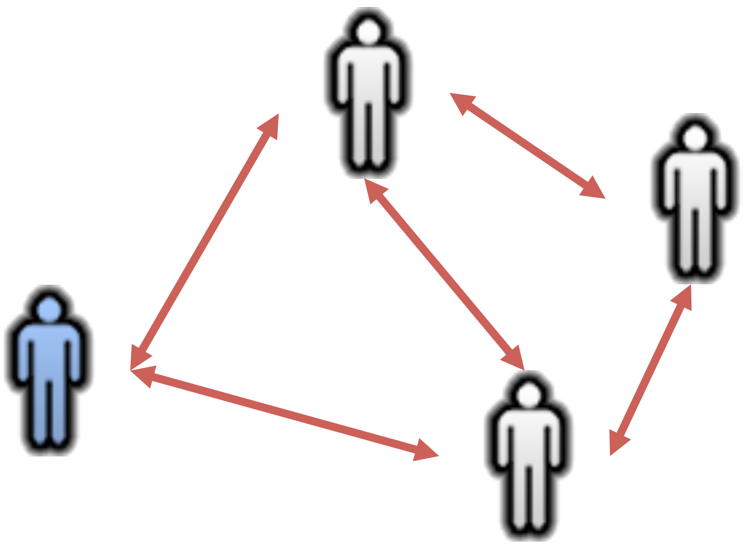
**cooperative**

*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?

# Opportunities for MAS Deployment

Agent-based computing have been used:

1. **Design paradigm** – the concept of decentralized, interacting, socially aware, autonomous entities as underlying software paradigm (often deployed only in parts, where it suits the application)
2. **Source of technologies** – algorithms, models, techniques architectures, protocols but also software packages that facilitate development of
multi-agent systems
3. **Simulation concept** – a specialized software technology that allows simulation of natural multi-agent systems, based on (1) and (2).

# Opportunities for MAS Deployment

Agent-based computing have been used:

1. **Design paradigm** – the concept of decentralized, interacting, socially aware, autonomous entities as underlying software paradigm (often deployed only in parts, where it suits the application)

⬇

Agent Oriented Software Engineering – provide designers and developers with a way of structuring an application around autonomous, communicative elements, and lead to the construction of software tools and infrastructures to support this metaphor

# Opportunities for MAS Deployment

Agent-based computing have been used:

1. Design paradigm – the concept of decentralized, interacting, socially aware, autonomous entities as underlying software
2. Source of technologies – algorithms, models, techniques architectures, protocols but also software packages that facilitate development of multi-agent systems

Multi-Agent Techniques – provide a selection of specific computational techniques and algorithms for dealing with collective of computational processes and complexity of interactions in dynamic and open environments.

# Opportunities for MAS Deployment

Agent-based computing have been used:

1. Design paradigm – the concept of decentralized, interacting, socially aware, autonomous entities as underlying software
2. Source of technologies – algorithms, models, techniques architectures, protocols but also software packages that facilitate development of multi-agent systems
3. Simulation concept – a specialized software technology that allows simulation of natural multi-agent systems, based on (1) and (2).

⬇

Multi-Agent Simulation – provide expressive models for representing complex and dynamic real-world environments, with the emphasis on capturing the interaction related properties of such systems

# Intelligent Agents Applications

Manufacturing and production

Traffic and logistics

Robotics, autonomous systems

Air traffic and space

Security applications

Energy and smart grids

## 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?

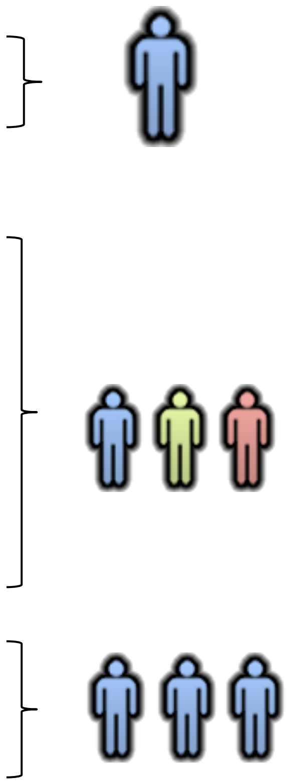
How should the agent **maximize its utility** in the presence of other competing and possible hostile agents?

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

. . .

## Course Content

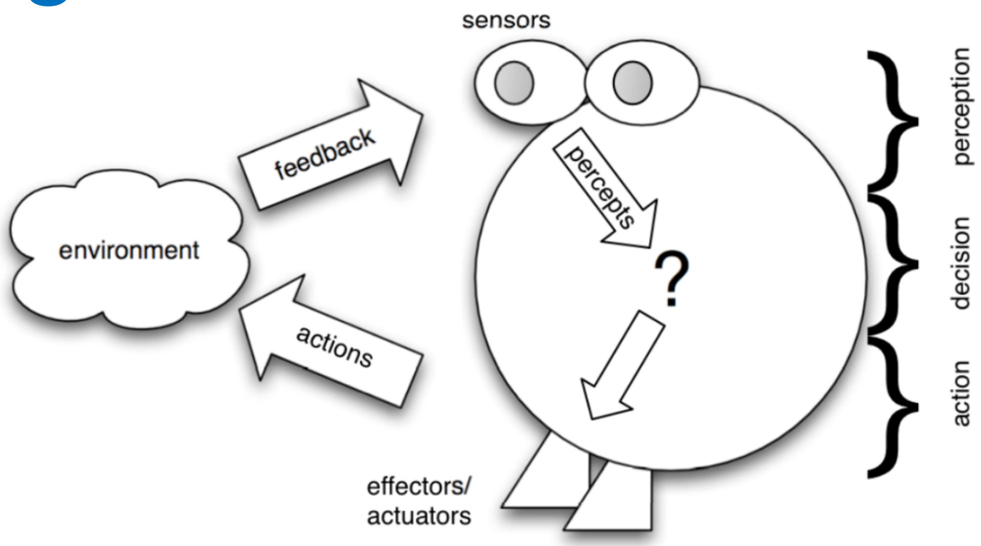
- Agent architectures (inc. BDI architecture)
- Logics for MAS
- Non-cooperative game theory
- Coalition game theory
- Mechanism design
- Auctions
- Social choice
- Distributed constraint reasoning (satisfaction and optimization)



Introduction to 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

Introduction to Multiagent Systems

## Specifying Agents

## Agent Behavior

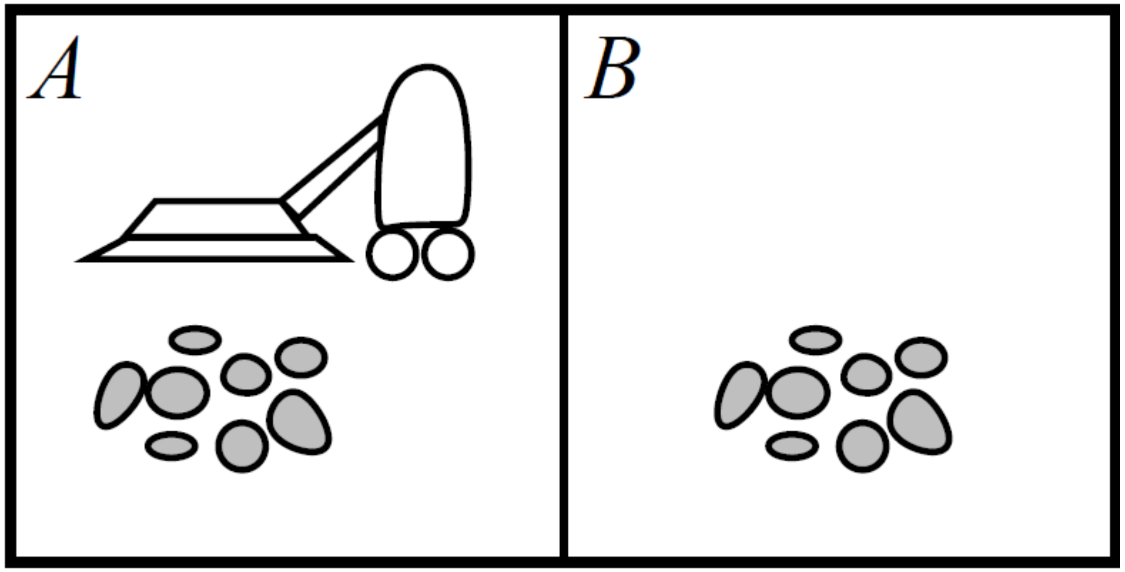
$$f : \mathscr{P} \mapsto \mathscr{A}$$

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 for the degree of success
2. percept sequence (complete perceptual history)
3. agent’s knowledge about the environment
4. actions available to the agent

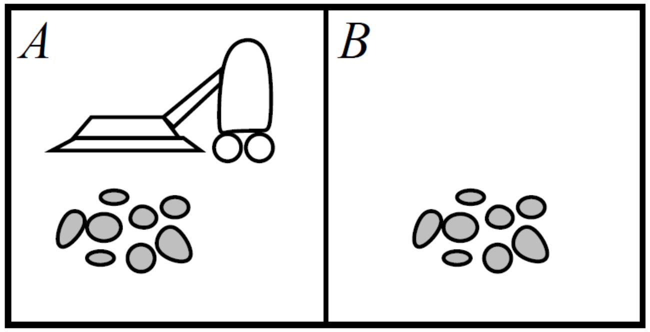
## Specifying Task Environments

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 | Shopping | Taxi |
|---|---|---|---|---|
| Observable | | | | |
| Deterministic | | | | |
| Episodic | | | | |
| Static | | | | |
| Discrete | | | | |
| Single-agent | | | | |

## 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 for the degree of success
2. percept sequence (complete perceptual history)
3. agent's knowledge about the environment
4. actions available to the agent

# Rationality

The agents rationality is given by the choice of actions based on expected utility of the outcome of the action. The rational agent selects an action a that provides the maximal expected outcome:

$$a = \arg\max_{l \in \mathcal{L}} \sum_{p_i : o_i \in l} p_i u(o_i)$$

Bounded Rationality: capability of the agent to perform rational decision (to choose the lottery providing maximal expected outcome) given bounds on computational resources:

- bounds on time complexity
- bounds on memory requirements

Calculative Rationality: capability to perform rational choice earlier than a fastest change in the environment can occur.

## Rationality

Let us have a community of agents $A_j \in \mathcal{A}$ each choosing to play an action $a_j$, executing the lottery $l_j$. providing the agents with the utility $u(a_i)$.

- **Self-interested rational agent:** selects the action that optimizes its individual utility

$$a = \arg\max_{l \in \mathcal{L}} \sum_{p_i : o_i \in l} p_i U(o_i)$$

- **Cooperative rational agent:** selects the action that optimizes collective utility of the whole team:

$$a = \arg\max_{l \in \mathcal{L}} \sum_{\forall a_j \in \mathcal{A} - a} \sum_{p_{i,j} : o_{i,j} \in l_j} p_{i,j} u(o_{i,j}) + \sum_{p_i : o_i \in l} p_i u(o_i)$$

## 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.

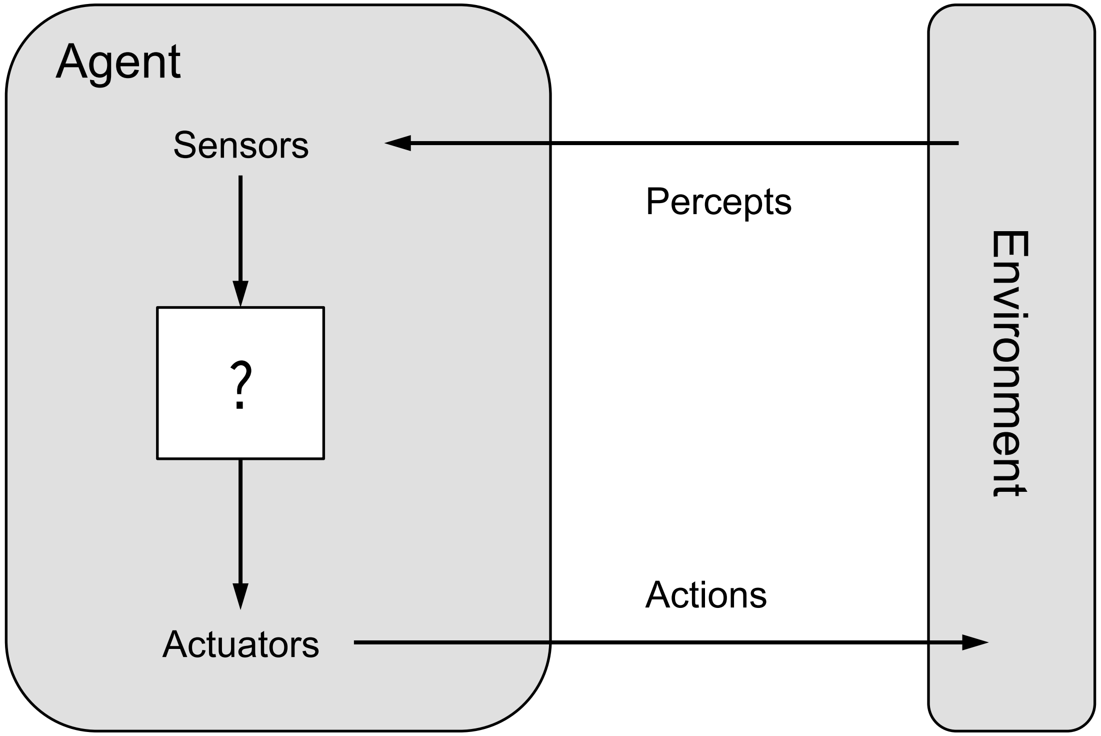
A perfectly rational poker-playing agent never loses.

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.

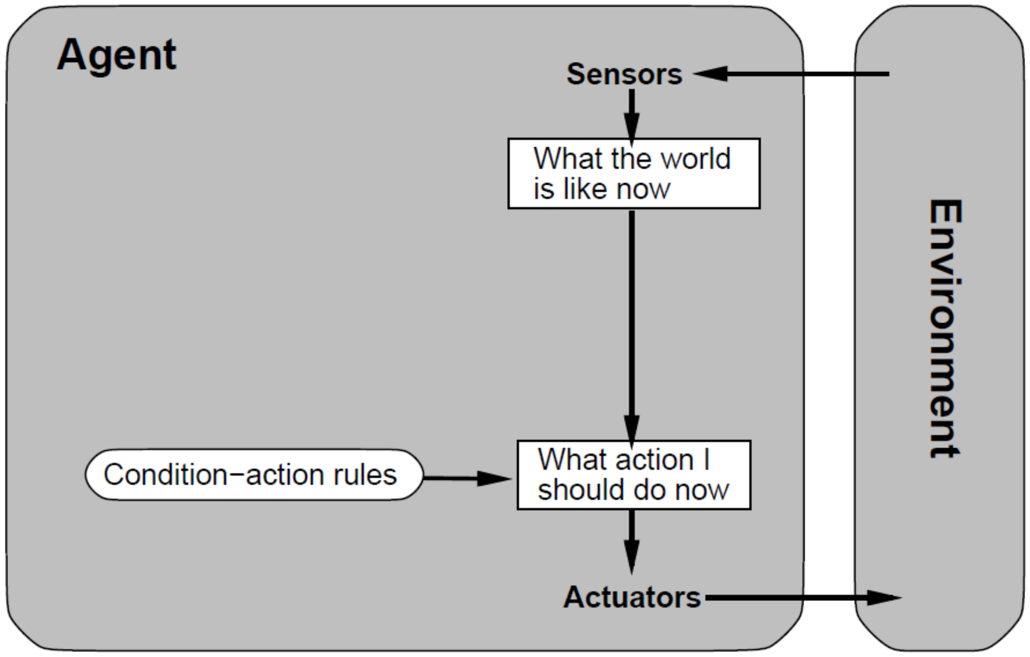
## Running Example: Robotic Taxi

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

- example:

```
If car-in-front-is-braking
=> initialize-braking
```

```
Function SIMPLE-REFLEX-AGENT(percept)
    state <= INTERPRET-INPUT(percept)
    rule <= RULE-MATCHING(state, rules)
    action <= rule.ACTION
    Return action

Function RULE-MATCHING(state, rules) ...
```

## Simpe Reflex Agent for Robotic Taxi

Simple program:

```
If a passenger at your location
=> pickup the passenger
else 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

Reflex agents are simple but of limited intelligence, rational if

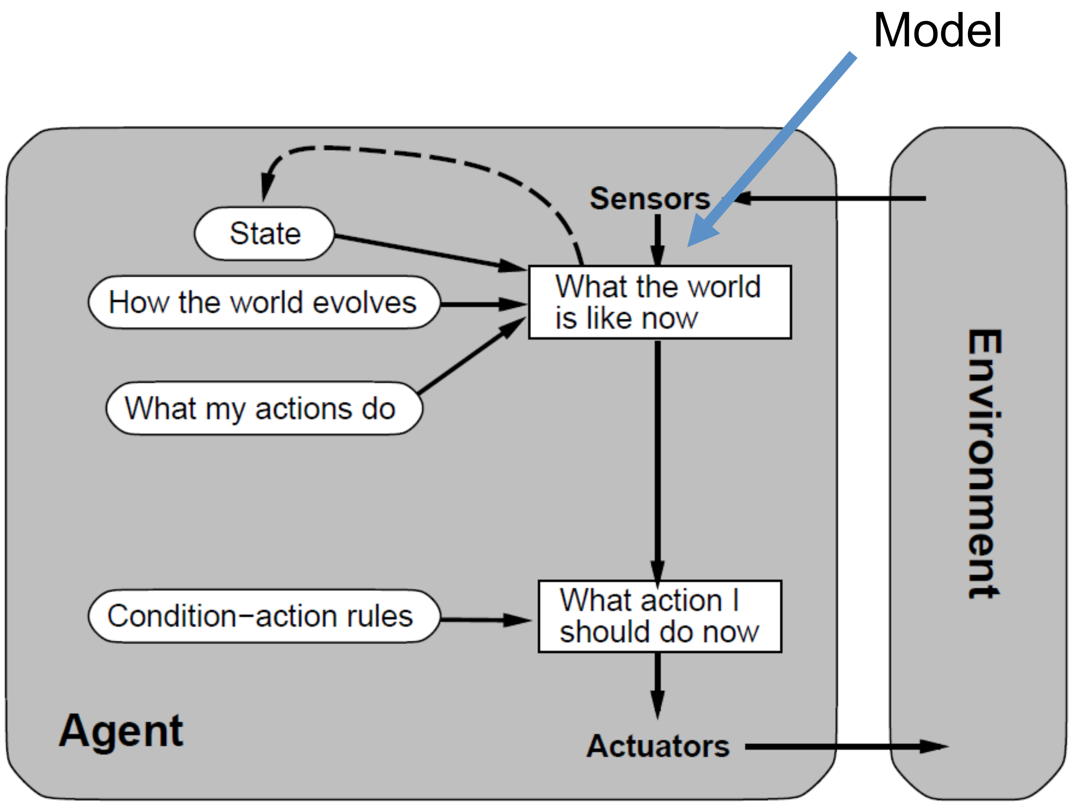
1. the environment is fully observable and
2. the decision can be made based solely on the current percept

Otherwise leads to suboptimal action choices, infinite loops.

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

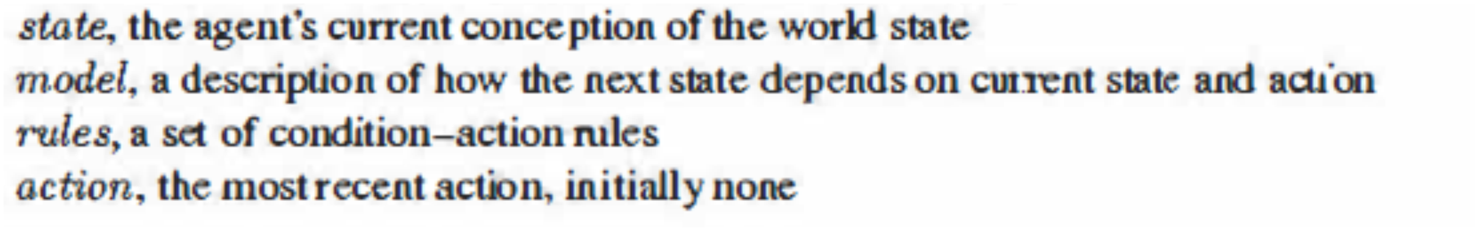
# Model-based Reflex Agent

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



## Model-based Reflex Agent

```
Function SIMPLE-REFLEX-AGENT(percept)
    state <=
          UPDATE-STATE(state, action, percept, model)
    rule <= RULE-MATCHING(state, rules)
    action <= rule.ACTION
    Return action
```



*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

# 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

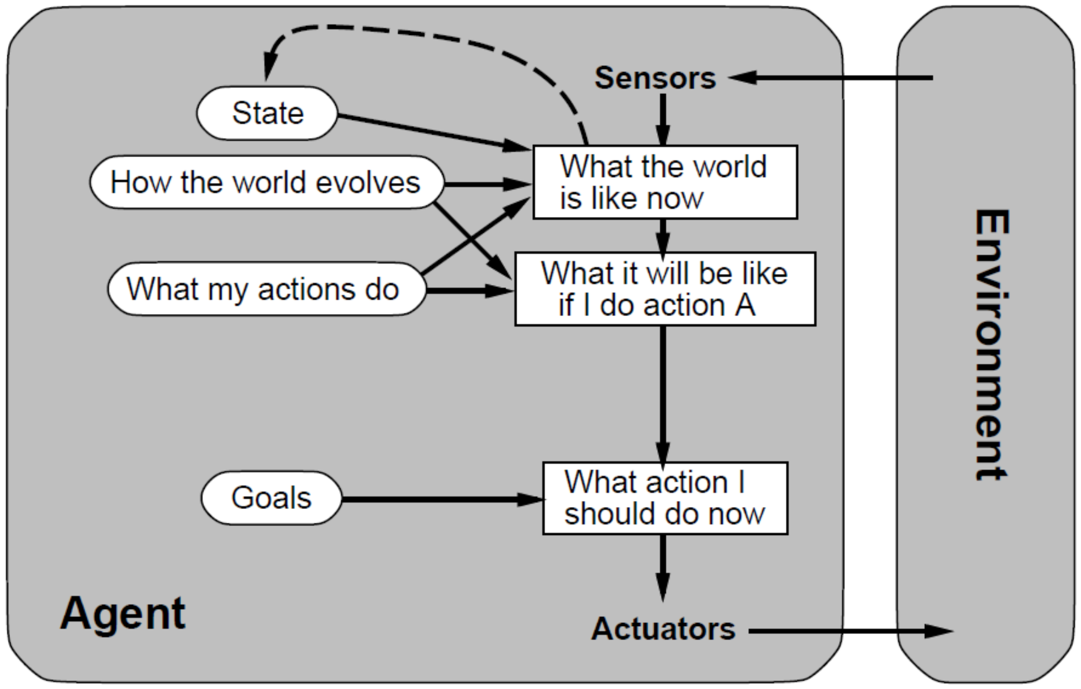
Taxi agent: Hot to get to a destination?

- 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


Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

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.

## Telling the Agent What to Do

Previous types of agents have the behavior hard-coded in their rules – there is no way to tell them what to do

Fundamental aspect of autonomy: we want to tell agent what to do but not how to do it!

We can specify:

- action to perform – *not interesting*
- (set of) goal state(s) to be reached → goal-based agents
- a performance measure to be maximized → utility-based agents

## Utility-based Agents

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

We introduce the concept of **utility**:

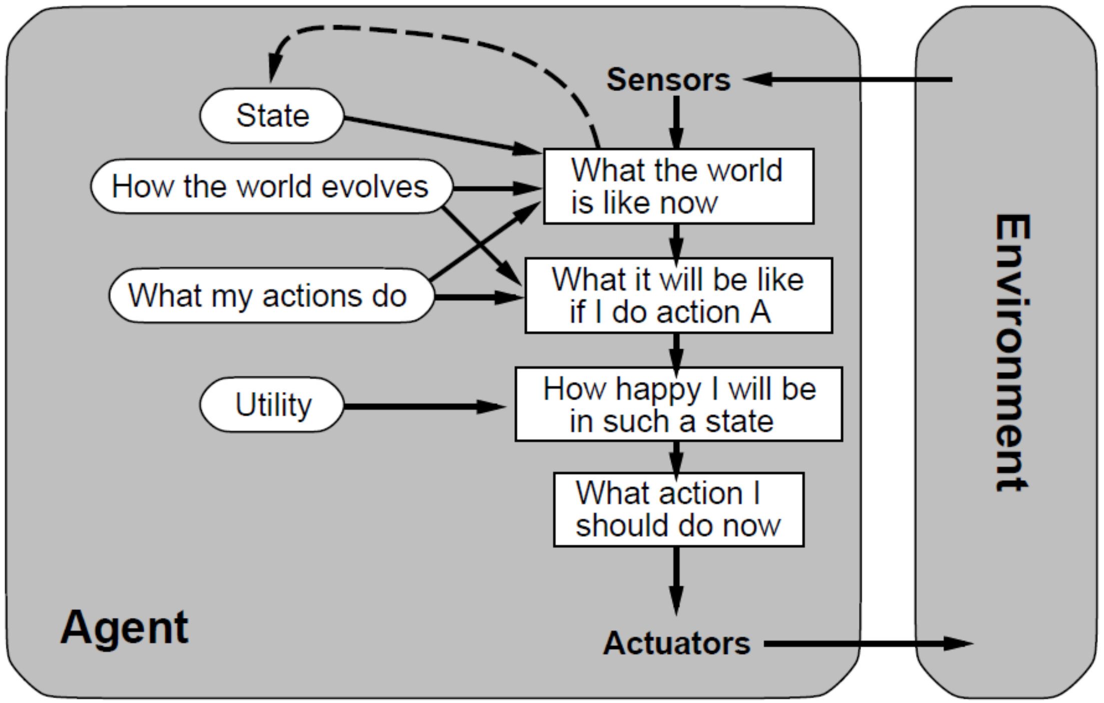
- utility is a function that maps a state onto a real number; it captures “quality” of a state
- if an agent prefers one world state to another state then the former state has higher utility for the agent.

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


Sensors
State
How the world evolves
What the world is like now
What my actions do
What it will be like if I do action A
Utility
How happy I will be in such a state
What action I should do now
Agent
Actuators
Environment

## 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

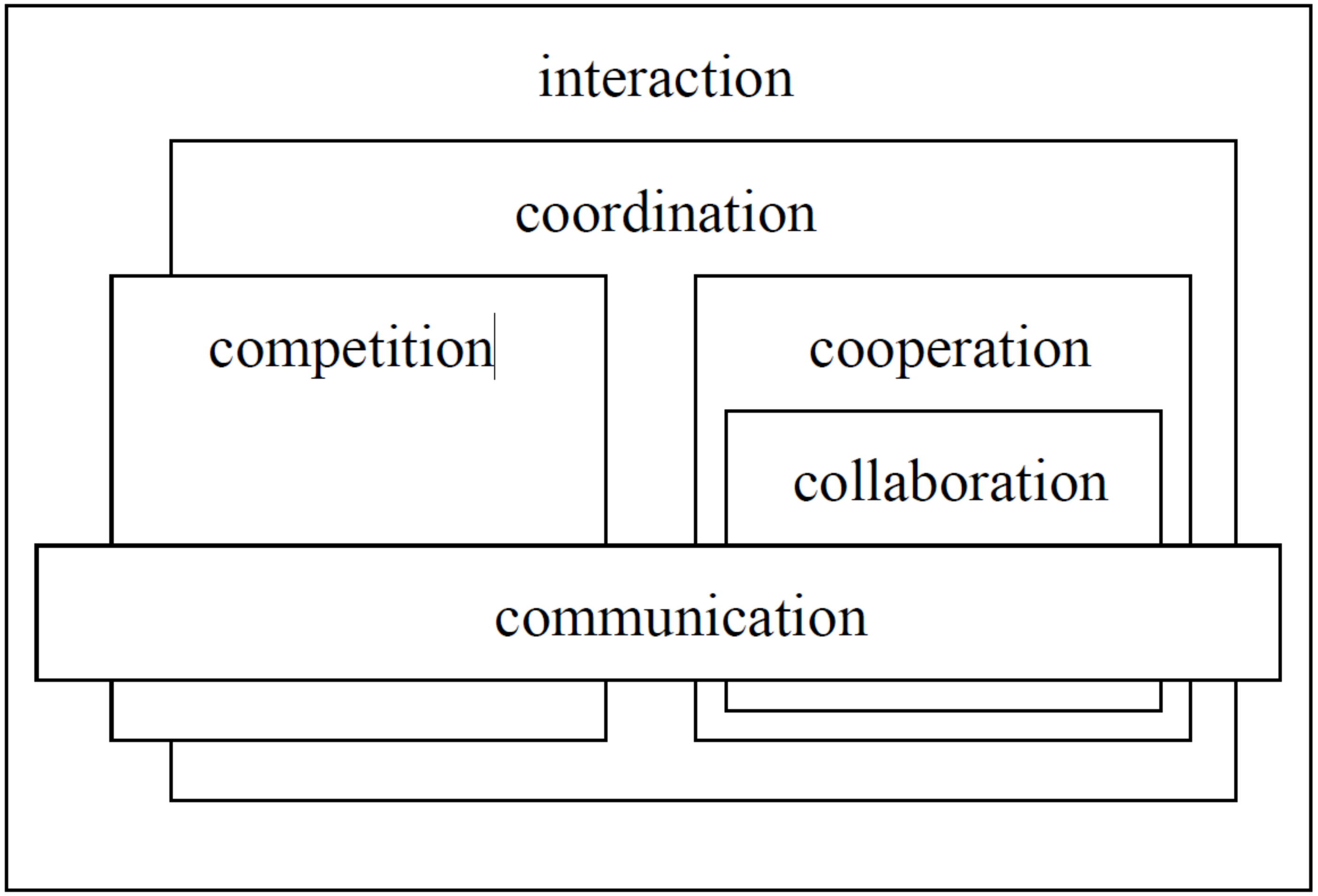
→ **Next:** Belifef-Desire-Intention Architecture

## History / Roots

## Course Schedule

1. Introduction to multi-agent systems
2. Modeling agents and their behavior in formal logic
3. Intelligent agent architectures, Belief-Desire-Intention architecture
4. Non-cooperative game theory, normal form games, prisoners dilemma
5. Nash equilibrium, Stackelberg equilibrium, security games
6. Solution concepts, extended form games, game tree search
7. Social choice and voting
8. Cooperative game theory, coalition formation, team work
9. Auctions
10. Distributed coordination
11. Agent-based simulations
12. Applications of multi-agent systems

## Typology of Interaction



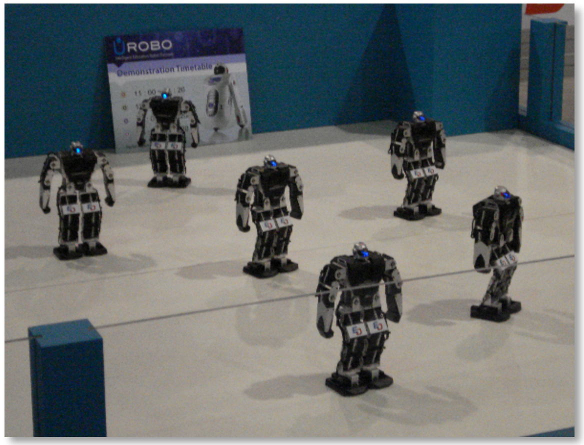
## Natural MAS

Business companies

Markets and economies

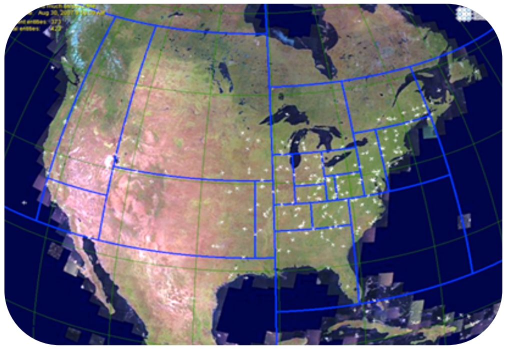
Transport systems

Distributed software systems

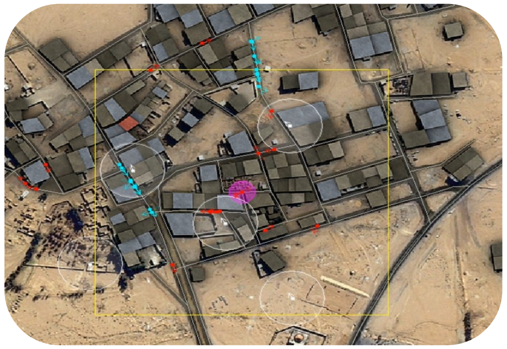


Robotic teams

## Application Areas (at ATG)



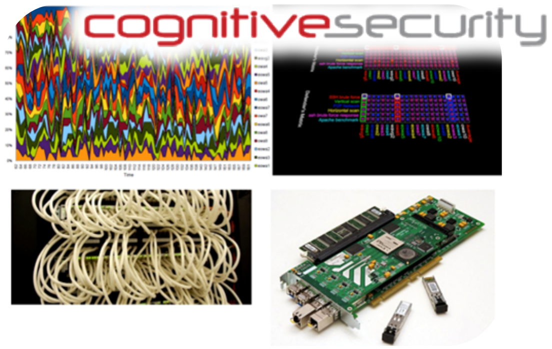
Air Traffic Management



Tactical Operations

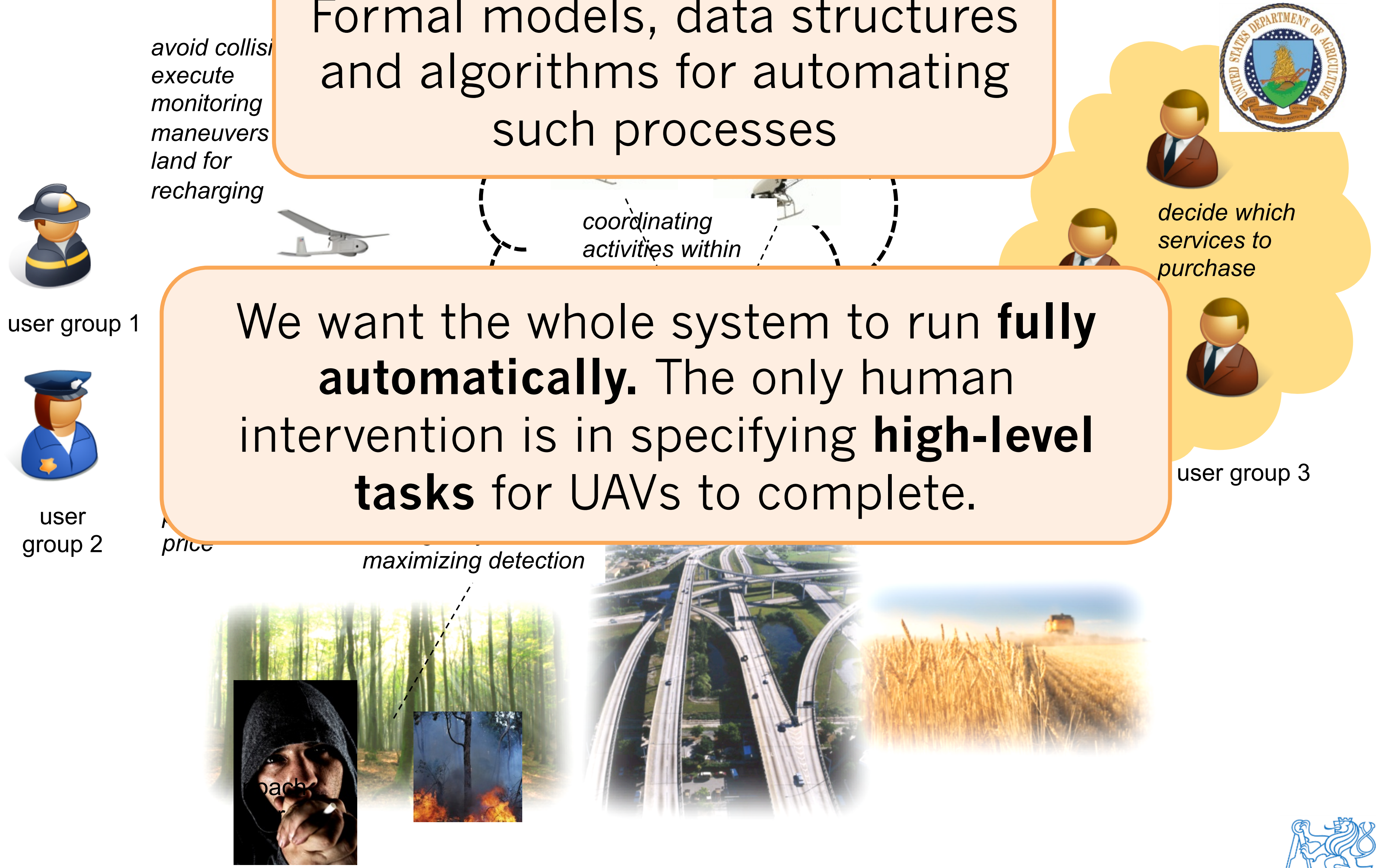
Autonomous Aerial Vehicles

Physical/ Critical Infrastructure Security



Cybersecurity and Steganography

Intelligent Transport Systems



Formal models, data structures and algorithms for automating such processes

We want the whole system to run **fully automatically.** The only human intervention is in specifying **high-level tasks** for UAVs to complete.

# Agent-control architecture and programming languages

Goal: Developing robust controllers capable of ex complexa activities in a dynamic, non-deterministic environment

- E.g. Avoiding collisions, executing monitoring maneuvers, land for recharging

Challenges

- Modularizing the agent into modules
- Describing the control logic in a compact form
- Handling concurrency, interruptions, complex plans, communications, …

# Coalition Formation

Goal: Forming and incentivizing teams that have highest value

- E.g. Determining which assets should form a team and how they should split payment for executing a task

Challenges

- determining right coalitions (centralized vs. decentralized)
- defining payments within coalitions

## Distributed Coordination

Goal: Coordinating assignment of tasks / resource so that constraints are met and an objective function maximized

- E.g. choosing which areas / targets should be tracked by whom so that coverage / tracking duration is maximized

Challenges:

- primarily algorithmic: efficient scalable algorithms that can handle many costraints
- distributed algorithms (due to communication limitations or privacy issues)

## Auctions

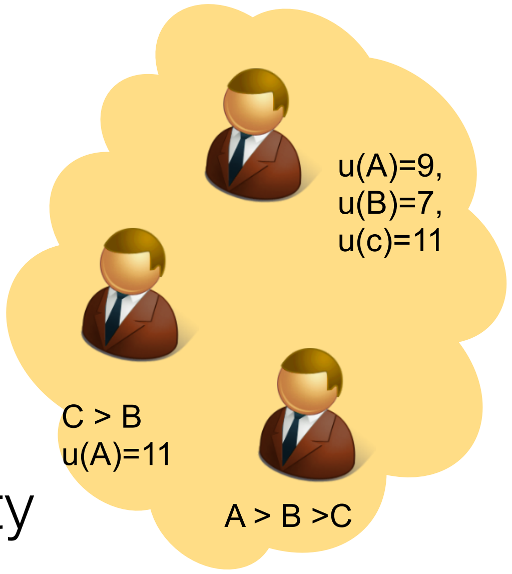
Goal: Allocate a scarce resource and determine payments so that profit is maximized

- E.g.: matching UAV teams with task issuers – which team should execute which task and for how much

Challenges

- representations: single vs. multi-attribute, single vs. multi-unit, single vs. multi-item
- protocols: bidding rules, market clearing rules, information dissemination rules
- bidding strategies
- centralized vs. distributed

# Social Choice / Negotiation



Goal: Agree on a single choice between multiple agents with different preferences

E.g.: choosing between monitoring crop quality or looking for forest fires

Challenges

- define what’s best: egalitarian, utilitarian, Nash bargaining solution, pareto efficiency, independence of irrelevant alternatives, non-dictatorship
- protocols to find the best:
  - the number of iterations / deadlines, stopping rules
  - with or without trusted third party
  - monotonic concesion protocol

## Non-cooperative Game Theory

Goal: Acting strategically in the presence of other rational agents

- E.g. deciding where to check for intruders assuming the intruders know they are going to be checked

Challenges

- defining good strategies: Nash equilibrium, minimax, …
- finding a good / best strategy
- various extensions: partial observation, sequential interactions, uncertainty about the objectives of the opponent, …

## MAS Applications – Problem Characteristics

**Non-cooperative/Competetive setting** with no trusted central authority and unwillingness of agents to share information

**Distributed setting** with inability (or ineffectiveness) to create and maintain shared global knowledge

**Highly dynamic environments** where fast reaction and frequent replanning is necessary

## Applications of MAS [TODO: visualize]

manufacturing and logistics – production planning, inventory management, supply chain/network management, procurement

markets – automated trading/auctioning, auction mechanism analysis and design, strategy modeling, market modeling

ubiquitous computing – context-enabled personal assistance, user modeling, privacy management, power-consumption optimization

robotic teams – team planning and coordination, optimum team composition, coalition formation, information fusion

utility networks – smart grid management, virtual powerplants, smart appliances, consumption modeling

## Applications of MAS

computer and communication networks – load balancing, intrusion detection, bandwidth management, monitoring

transport – intelligent transport system, cooperative driving, ridesharing

security and defense – mission planning and execution, optimum patrolling and surveillance, opponent modeling, vulnerability assessment

computer games and computer animation - game AI, behavioral animation, NPC implementation

policy-making support – modeling of various socio-economical phenomena (crime, migration, urban growth)