Introduction to Multi-Agent Systems

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

AE4M36MAS Autumn 2012 - Lect. 1

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

General Information

- Lecturers: prof. Michal Pěchouček, Michal Jakob
- Tutorials: Branislav Bošanský and Jan Hrnčíř
- 13 lectures and 13 tutorials
- Course web page: https://cw.felk.cvut.cz/doku.php/courses/ae4m36mas/start
- Recommended reading:
  - Russel and Norvig: Artificial Intelligence: Modern Approach
  - 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)
  - 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:
  - Project #1 (9 pts) – due end of October
  - Project #2 (14 pts) – due end of November
  - Project #3 (17 b.) – due at least a weak before you want a course assessment (end of semester)
- Final exam – 60 pts
- At least 50% points required from each part

Introduction to Multiagent systems

Introduction

Trends in Computing

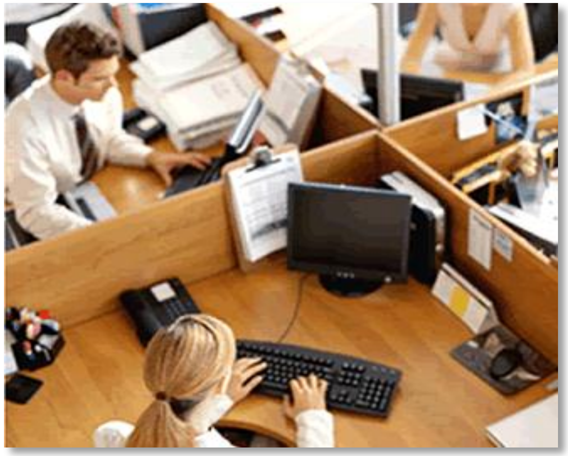
- **Ubiquity**: Cost of processing power decreases dramatically (e.g. Moore's Law), computers used everywhere
- **Interconnection**: Formerly only user-computer interaction, nowadays distributed/networked machine-to-machine interactions (e.g. Web APIs)
- **Complexity**: Elaboration of tasks carried out 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)

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

Multiagent Systems (MAS)

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



Human teams and companies

Markets and economies

Transportation networks

Distributed software systems

Communication networks

Robotic teams

Multi-Agent System 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

Application Areas (at ATG)

Air Traffic Management

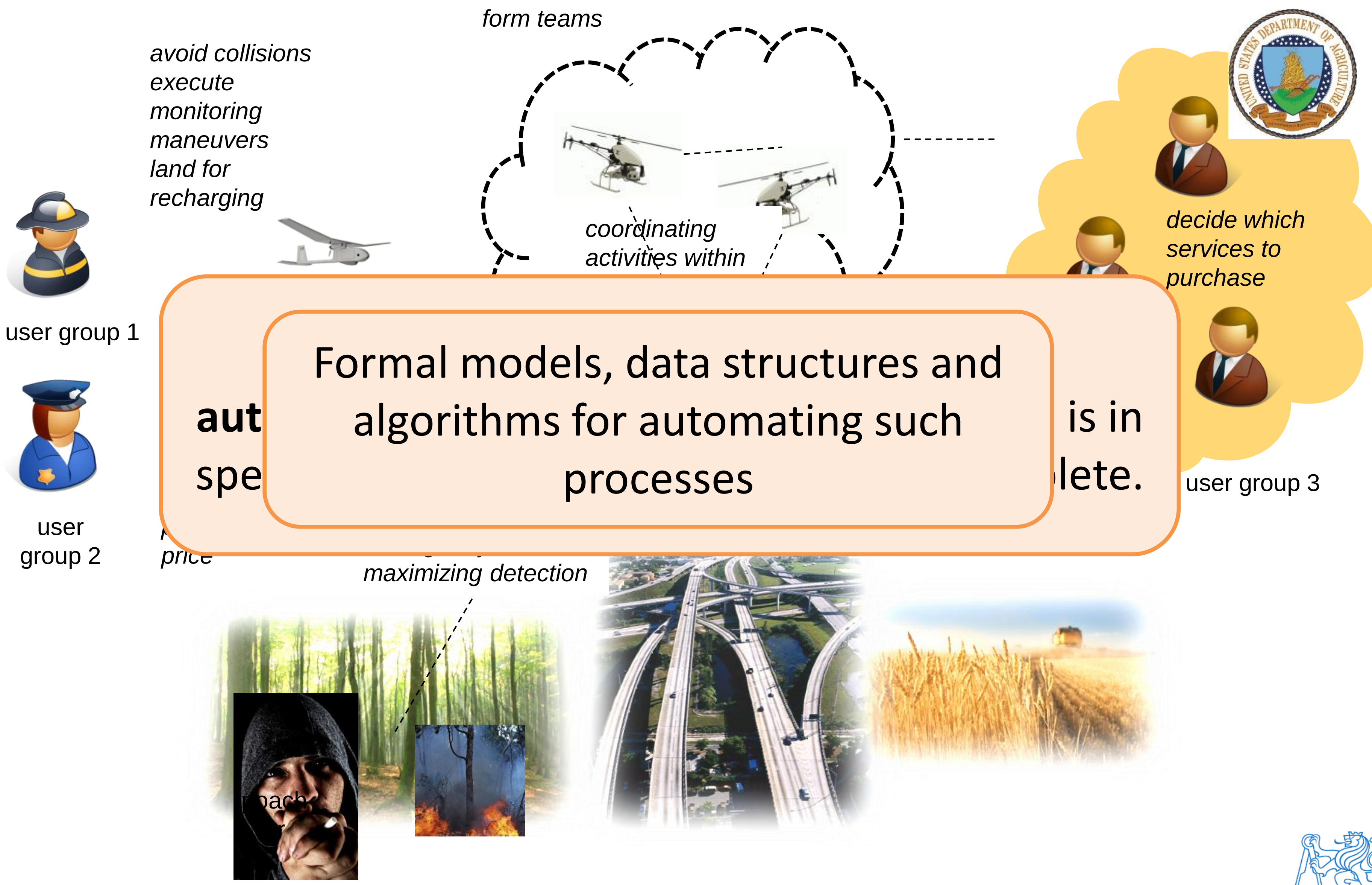
Tactical Operations

Autonomous Aerial Vehicles

Physical/ Critical Infrastructure Security

Cybersecurity and Steganography

Intelligent Transport Systems



form teams
avoid collisions
execute
monitoring
maneuvers
land for
recharging
coordinating
activities within
decide which
services to
purchase
user group 1
user
group 2
user group 3
maximizing detection
Formal models, data structures and algorithms for automating such processes

Agent-control architecture and programming languages

- Goal: Developing robust controllers capable of executing complex 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

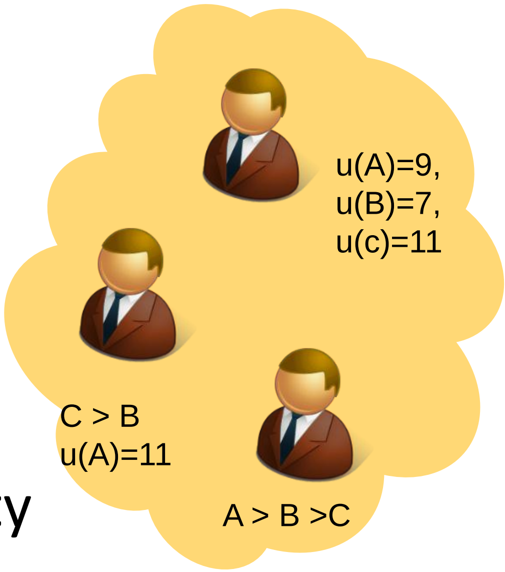
Distributed Coordination

- 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, …

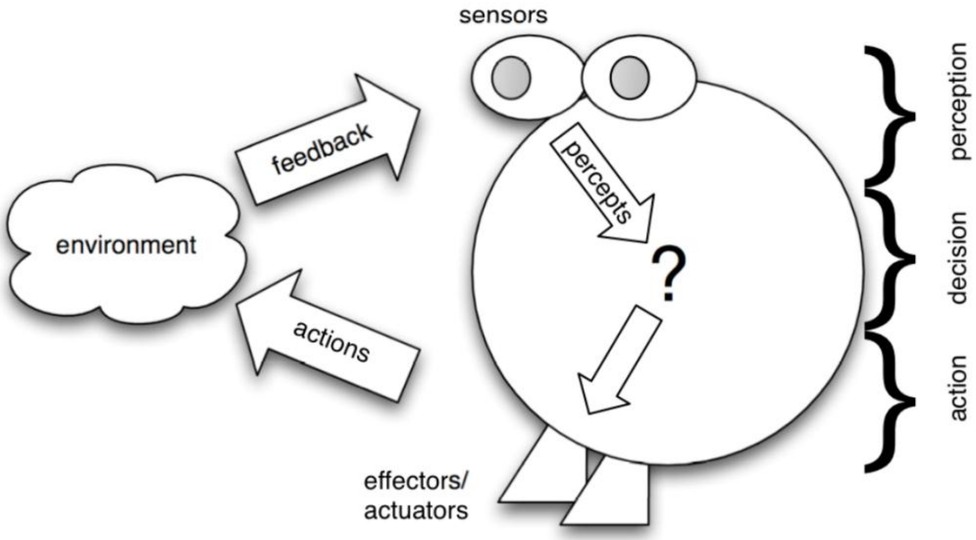
Course Schedule

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

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 thatis 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

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
- Systems of the future will need to be good at teamwork

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*

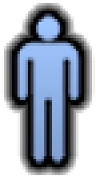
Types of Agent Systems

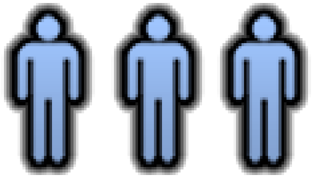
single-agent

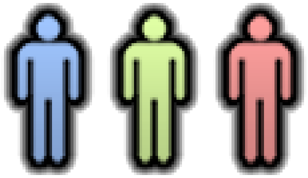
multi-agent

cooperative

competitive



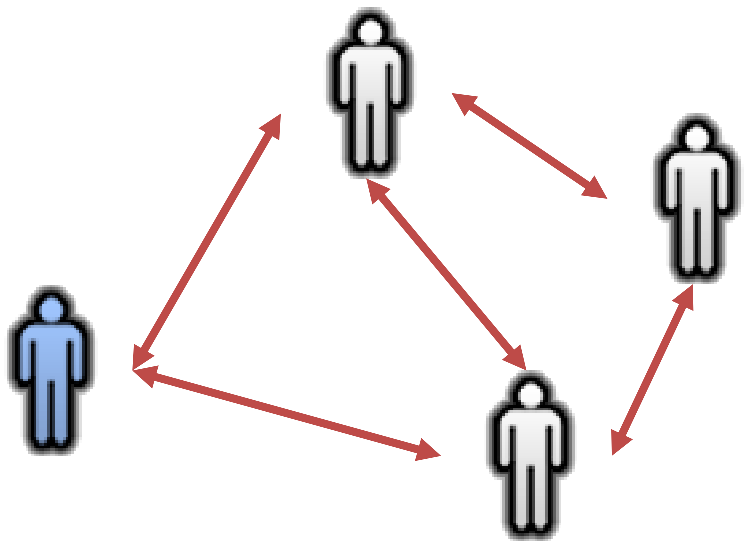




single shared utility

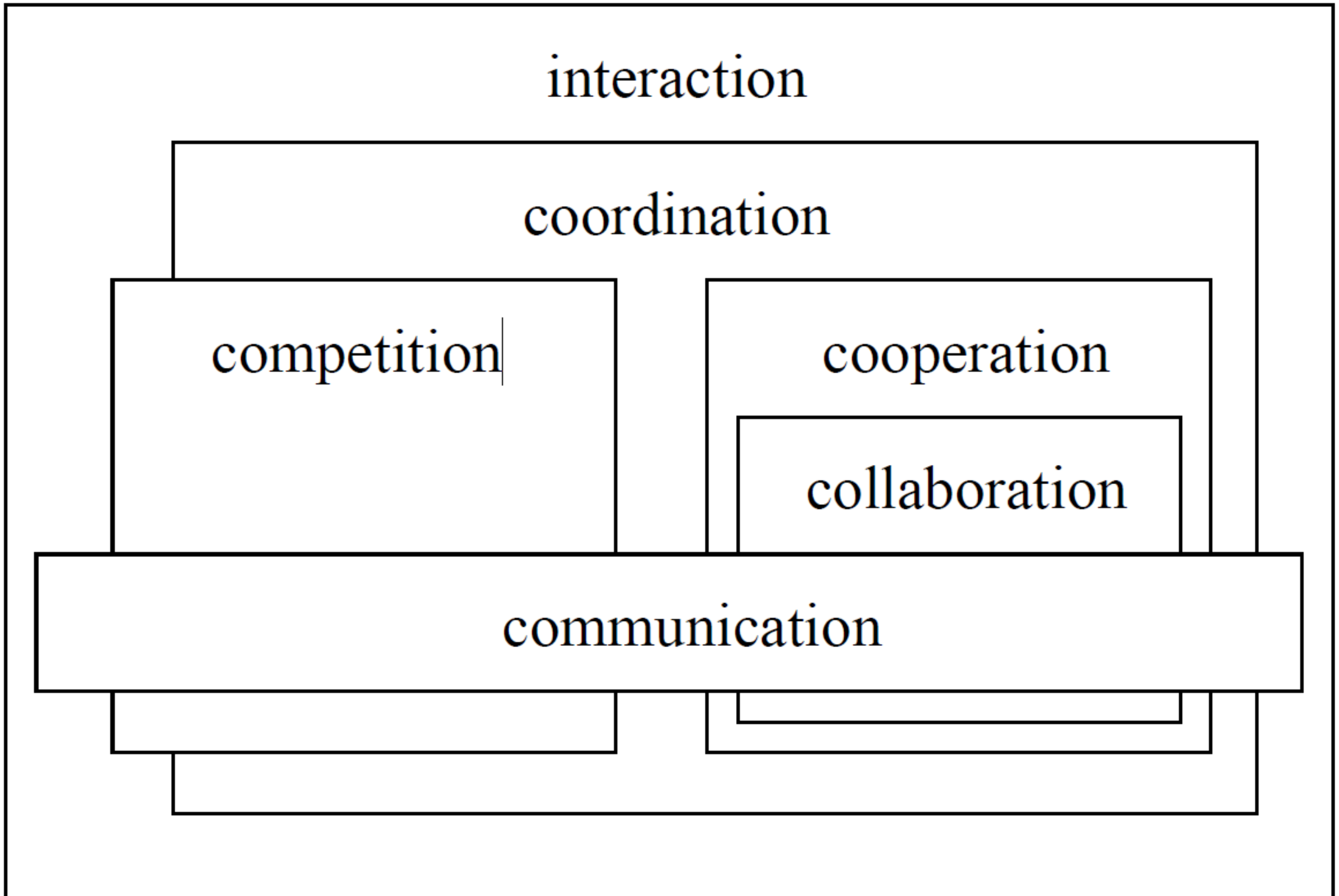
multiple different utilities

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?

Typology of Interaction



Introduction to Multiagent Systems

Specifying Agents

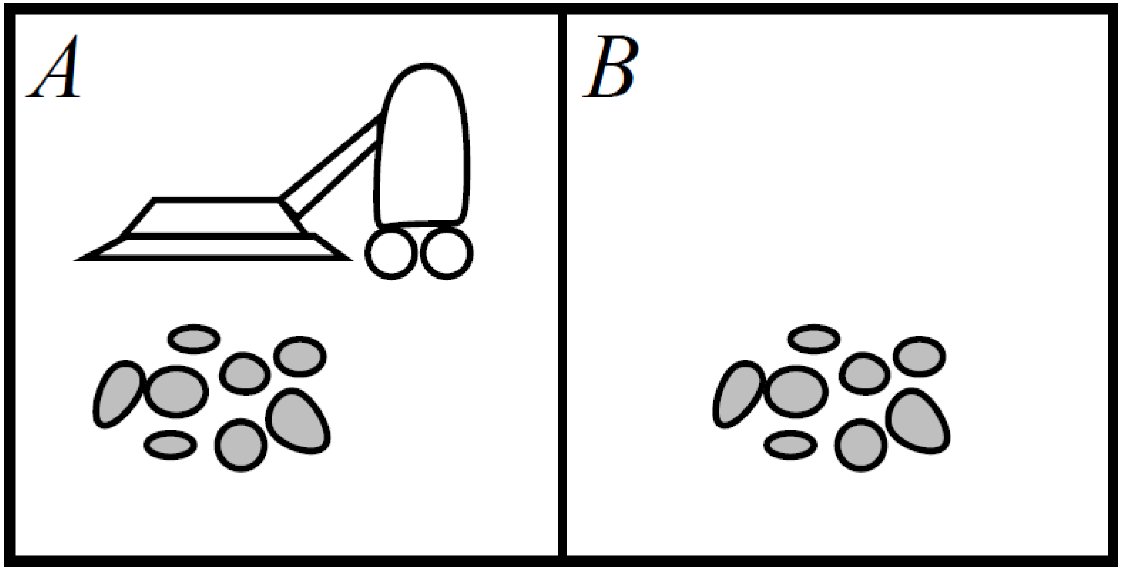
Agent Behavior

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

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

- 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
…	…

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

PEAS Examples

Agent	Performance measure	Environment	Actuators	Sensors
Taxi driver	safe, fast, legal, comfortable trip, maximize profits	roads, other traffic, pedestrians, customers	steering, accelerator, brake, signal, horn, display	cameras, sonar, speedometer, GPS, engine sensors, keyboard
Part picking robot	percentage of parts in correct bins	conveyor belt with parts, bins	jointed arm and hand	camera, joint angle sensors
Trading agent	maximum profit over a defined period	electronic trading platform	API for placing trading orders	current and historic prices, current orders
Refinery controller	maximize purity, yield, safety	refinery operators	valves, pumps, heaters, displays	temperature, pressure, chemical 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

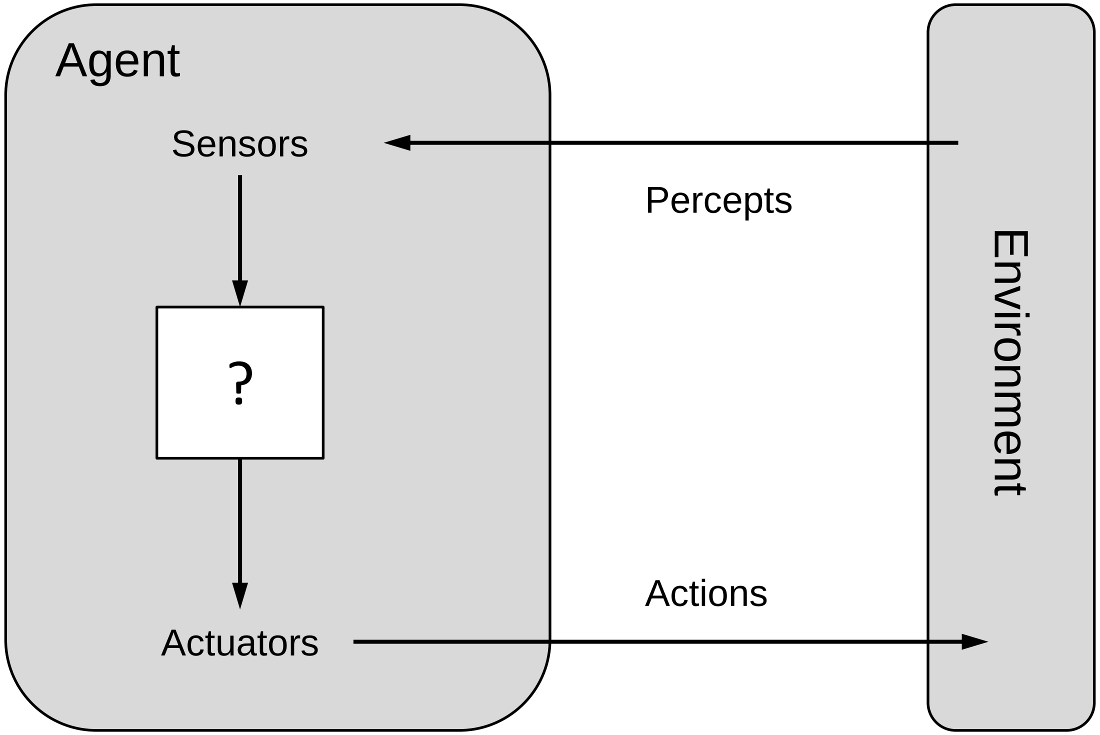
	Solitaire	Backgammon	Internet shopping	Taxi
Observable	No	Yes	No	No
Deterministic	Yes	No	Partly	No
Episodic	No	No	No	No
Static	Yes	Semi	Semi	No
Discrete	Yes	Yes	Yes	No
Single-agent	Yes	No	Yes (except auctions)	No

Introduction to Agents

Agent Architectures

Implementing the Agent

- How should one implement the agent function?
  - So that the resulting behavior is (near) rational.
  - So that its calculation is computationally tractable.

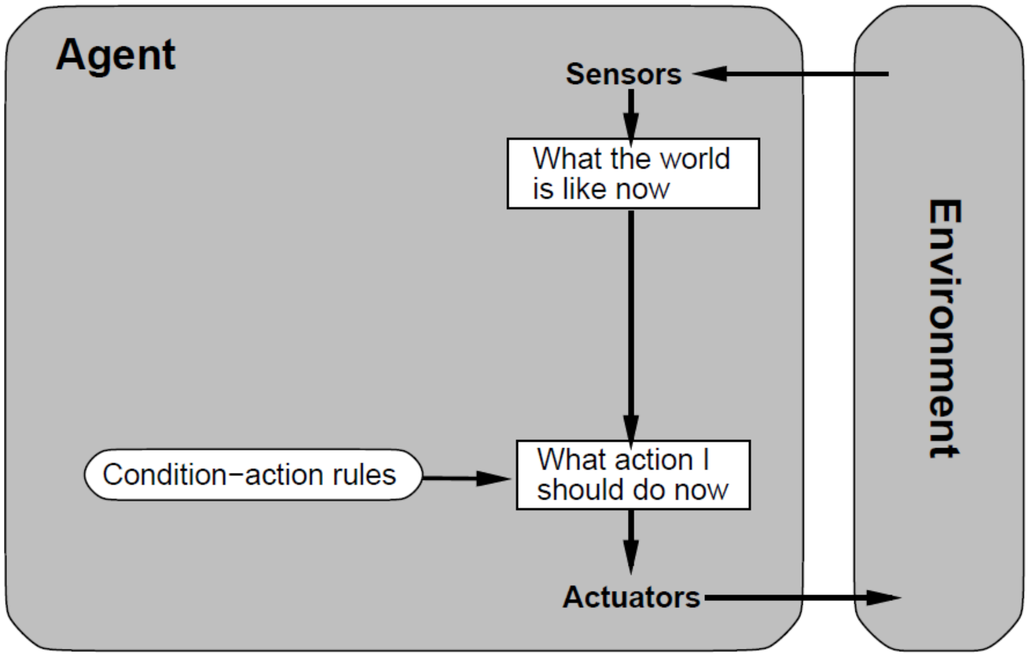


Hierarchy of Agents

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.

- Four basic types of agent in the order of increasing capability:
  1. simple reflex agents
  2. model-based agents with state
  3. goal-based agents
  4. utility-based agents

Simple Reflex Agents

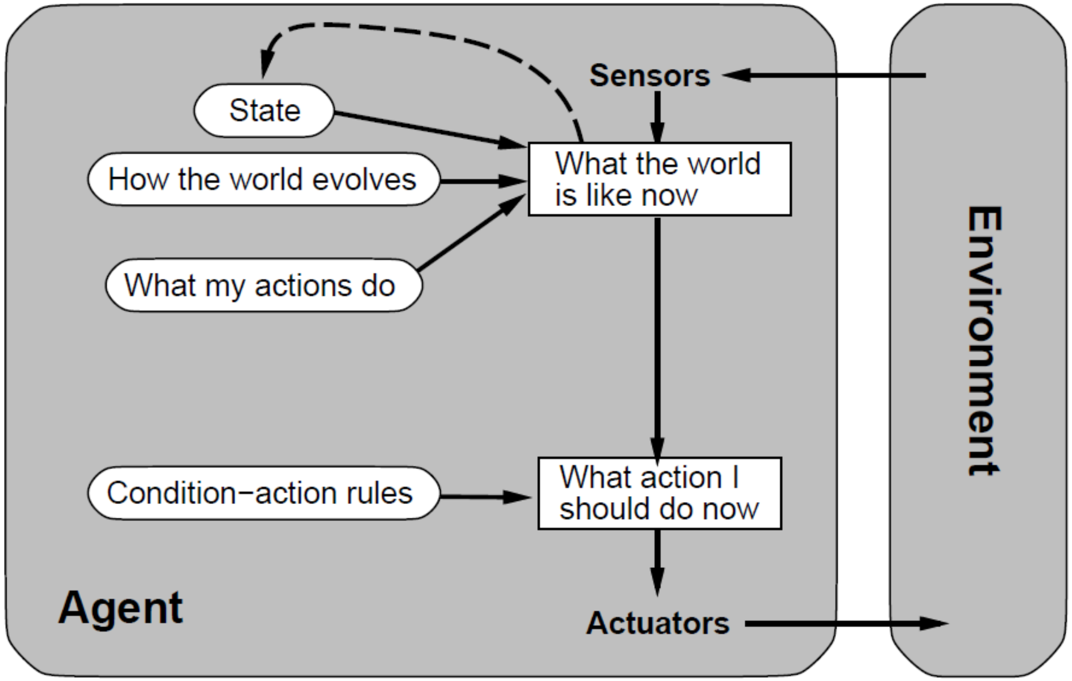


- Simple reflex agent chooses the next action on the basis of the current percept
  - Condition-action rules provide a way to present common regularities appearing in input/output associations
  - Ex.: `if car-in-front-is-braking then initialize-braking`

Adding State / Model

- Reflex agents are simple but of limited intelligence
- Only work if environment is fully observable and the decision can be made based solely on the current percept
- If 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

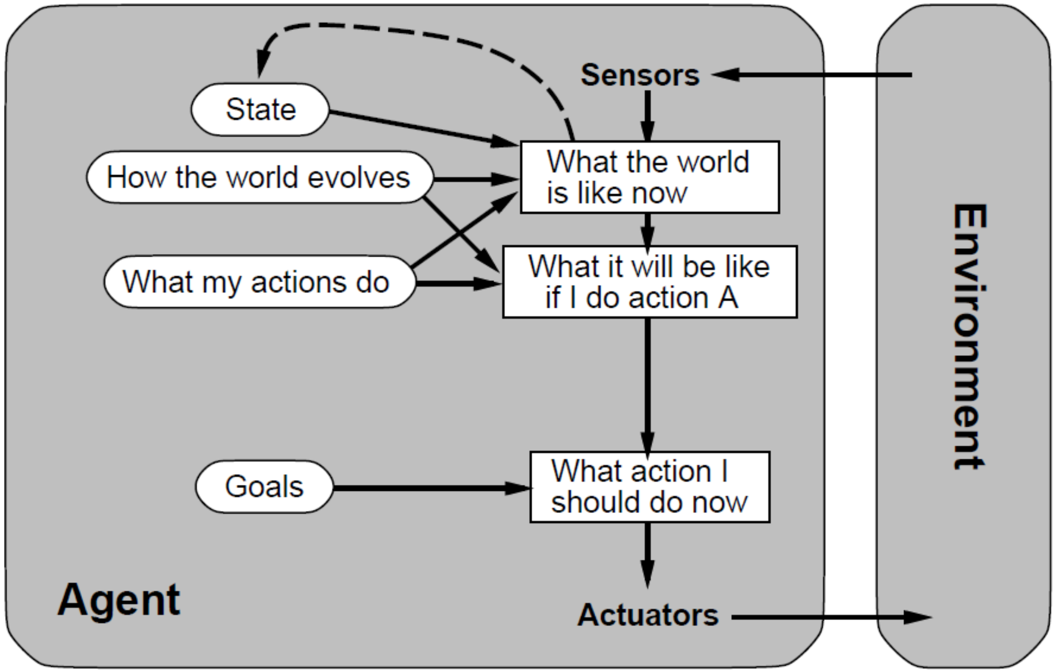


- Keeps track of the world by extracting relevant information from percepts and storing it in its memory
  - models: (1) how the world evolves, (2) how agent's actions affect the world

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**

Goal-based Agents



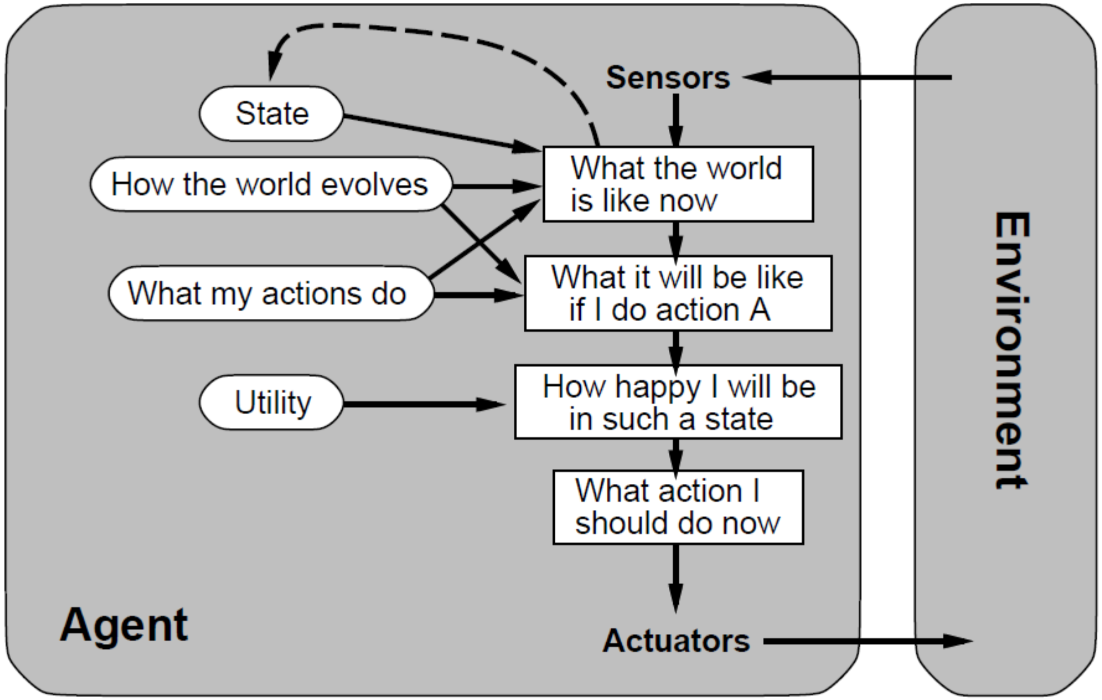
- **Problem**: goals are not necessarily achievable by a single action:

  → **search and planning** are subfields of AI devoted to finding actions sequences that achieve the agent's goals

Towards 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

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