LECTURE 3

SUMMARY

1. Agent design 1
Example: Simple Vacuum Cleaning Agent 1
2. Environment 3
3. Mobile agents ([1], Section 1.5) 6
4. Agent programming languages ([1], Section 1.5) 8

1. Agent design

- when designing agent-based systems, we should consider **PAGE(S)** – Perception, Action, Goal, Environment.
    - conceptually, the environment (**E**) is the set of the possible environment states (**S**). …thus, we will add an additional **S**

Example: **Simple Vacuum Cleaning Agent**

- we have a small robotic agent that will clean up a room
- the room is entirely surrounded by walls
- the robot is equipped with a sensor, and a vacuum cleaner that can be used to suck up dirt.
- the robot has a definite orientation (one of north , south, east, or west)
- the robot is capable of sucking up dirt, moving forward and turning right
- the robot is able to sense if it is over any dirt and it can detect if a wall is directly in front of it
- the robot does not know the location of the dirt ahead of time.
- E.g. R starts from (1,1) facing South

```
  1   2   3   4
 +---+---+---+---+
1|R  |   |   |   |
 +---+---+---+---+
2|   |*  |   |   |
 +---+---+---+---+
3|   |   |   |   |
 +---+---+---+---+
4|   |   |   |*  |
 +---+---+---+---+
Location: (1,1)   Facing: SOUTH
```

Conceptual modeling (PAGE)

- **Agent**
    - robot
- **State**
    - a configuration of the map

- **Environment**
  - grid world
  - set of possible states (configurations of the map)
- **Action**
  - Suck, forward, turn
- **Perception**
  - dirt
  - wall
- **Goal**
  - to search and remove dirt within the room

Possible extensions

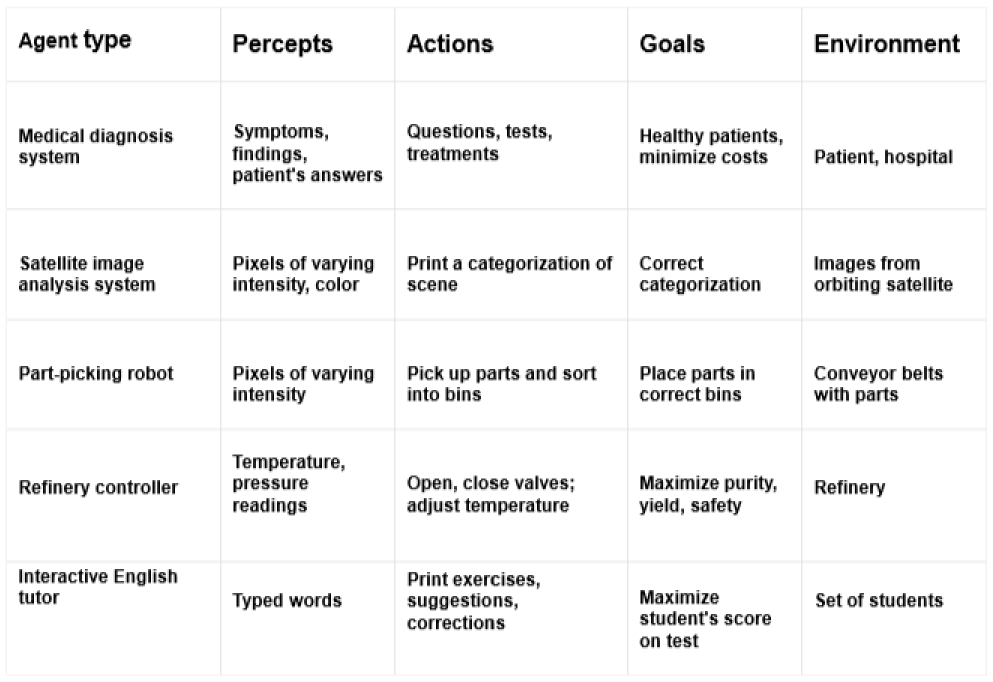
- a more complex environment, with obstacles
  - the agent sensor detects if there is an obstacle in front of it or if it has accidentally bumped into an obstacle
- another action is available for the agent: ShutOff (the robot shuts off the vacuu cleaner)
- the robot has a more complex goal
    - minimize the power
    - return in its initial position after it has removed all dirt
    - **learn** to clean the room (RL)
- multiple robot are cleaning the room (**MAS**)

Note

For implementing the simulation of the Simple Vacuum Cleaning World

- a logic based architecture may be chosen for the agent ([1], Section 1.4.1)
  - logic programming language: Prolog
- a reactive/deliberative architecture

Examples of agents in different types of applications



Agent type	Percepts	Actions	Goals	Environment
Medical diagnosis system	Symptoms, findings, patient's answers	Questions, tests, treatments	Healthy patients, minimize costs	Patient, hospital
Satellite image analysis system	Pixels of varying intensity, color	Print a categorization of scene	Correct categorization	Images from orbiting satellite
Part-picking robot	Pixels of varying intensity	Pick up parts and sort into bins	Place parts in correct bins	Conveyor belts with parts
Refinery controller	Temperature, pressure readings	Open, close valves; adjust temperature	Maximize purity, yield, safety	Refinery
Interactive English tutor	Typed words	Print exercises, suggestions, corrections	Maximize student's score on test	Set of students

#	Agent	P	A	G	E	S
(1)	Simple vacuum cleaning	- dirt - wall	- suck up dirt - go forward - turn right 90°	Clean the room	Grid-like environment	Current configuration of the room
(2)	Autonomous driver	Images	- turn left - turn right …	Drive autonomously on a highway	Highway	Current configuration of the highway
(3)	Backgammon player	Board state	specific moves	Play backgammon	All possible board states	Board state
(4)	Softbot	Web page	file manipulation commands	Collect info on a subject	Internet	Web page

Softbots (Software Robots)

- Etzioni (1993, 1996)
    - determine the e-mail address of a person
    - search papers for a person
    - AI planning techniques
- MARVIN (1998) - retrieve medical information on Internet

2. Environment

Environment

- determines the interaction between the "outside world" and the agent
    - the "outside world" is not necessarily the "real world"
    - it may be a real or virtual environment the agent lives in

Properties of the environment

A. Accessible vs inaccessible.

- An accessible environment is one in which the agent can obtain complete, accurate, up-to-date information about the environment's state.
- Most moderately complex environments (including, for example, the everyday physical world and the Internet) are inaccessible.
- The more accessible an environment is, the simpler it is to build agents to operate in it.

  E.g.

  Accessible – (3)

  Inaccessible – (1), (2)

B. Deterministic vs non-deterministic.

- A deterministic environment is one in which any action has a single guaranteed effect — there is no uncertainty about the state that will result from performing an action.
- In a non-deterministic environment, a probability distribution function is required
- In a non-deterministic environment, there is a stochastic process acting on it.
  - The physical world can be regarded as non-deterministic.
  - Non-deterministic environments present greater problems for the agent designer.

  E.g.

  Deterministic – (1)

  Non-deterministic – (2), (3)

C. Episodic vs non-episodic.

- In an episodic environment, the performance of an agent is dependent on a number of discrete episodes, with no link between the performance of an agent in different scenarios.
  - The agent can decide what action to perform based only on the current episode (no need to reason about the interaction between the current episode and the future ones).
- In a non-episodic (sequential) environment, the current decision could affect all future decisions "*short term actions can have long term consequences*".
  - the agent has to plan his actions
  - planning
- Episodic environments are simpler from the agent developer's perspective because the agent can decide what action to perform based only on the current episode — it need not reason about the interactions between this and future episodes.

  E.g.

  Episodic – (1)

  Sequential – (2), (3)

D. Static vs dynamic.

- A static environment is one that can be assumed to remain unchanged except by the performance of actions by the agent.
- A dynamic environment is one that has other processes operating on it, and which hence changes in ways beyond the agent's control. The physical world is a highly dynamic environment.

  E.g.

  Static – (1), (2)

Dynamic – (3)

E. Discrete vs continuous.

- An environment is discrete if there are a fixed, finite number of actions and percepts in it.
- *How do we **represent** or **abstract** or **model** the world?*

  E.g.

  Discrete – (1), (3)
  Continuous – (2)

Markovian vs non-Markovian.

- *Markov processes* (probabilities and statistics)
  - A process satisfies the Markov property of one can make predictions about the future state of the process based only on the current state (without reference to the history).
  - The conditional probability distribution of future states depends only upon the present state.
- **Markovian agent environment**
  - The environment response at time *t*+1 depends only on the state and action at time *t* $\Rightarrow$ the environment's dynamics can be defined by specifying $Pr(s_{t+1} = s' | s_t, a_t)$.
  - $Pr(s_{t+1} = s' | s_t, a_t) = Pr(s_{t+1} = s' | s_t, a_t, s_{t-1}, a_{t-1}, \ldots s_0, a_0)$

Examples

- The *weather evolution* is a Markov process.
- The chess game is Markovian.
- A *machine breakdown* is non-Markovian, as it depends on the entire lifecycle of the machine, not only on the present condition.
- An example of a stochastic process that is non-Markovian: *An un contains two red balls and one green ball. One ball was drawn yesterday, one ball was drawn today and one ball will be drawn tomorrow. All off the drawns are without replacement.*
  - Suppose you know that the today's ball was red, but you have no information about yesterday's ball $\Rightarrow$ the chance that the tomorrow's ball will be red is 0.5.
  - If you know that the yesterday's ball was red, then you are guaranteed to get a green ball tomorrow.

➢ The agent may keep track (in its internal memory) of the current state of the environment (considering all the received percepts)
  - and thus it may act as if its environment is a **Markovian** one.

- **Single agent vs. Multiagent**
  - image analysis, crossword puzzle, taxi driving, chess
  - *An agent operating by itself in an environment*
  - *Does the other agent interfere with my performance measure?*
  - *Interaction and collaboration among the agents?*
    - competitive, cooperative

Task Environment

- has to be specified before the design of the agent, besides **PAGE**
- **PEAS**
  - **Performance measure**
  - **Environment**
  - **Actuators**
    - actions performed through actuators
  - **Sensors**
    - perception through sensors

Examples

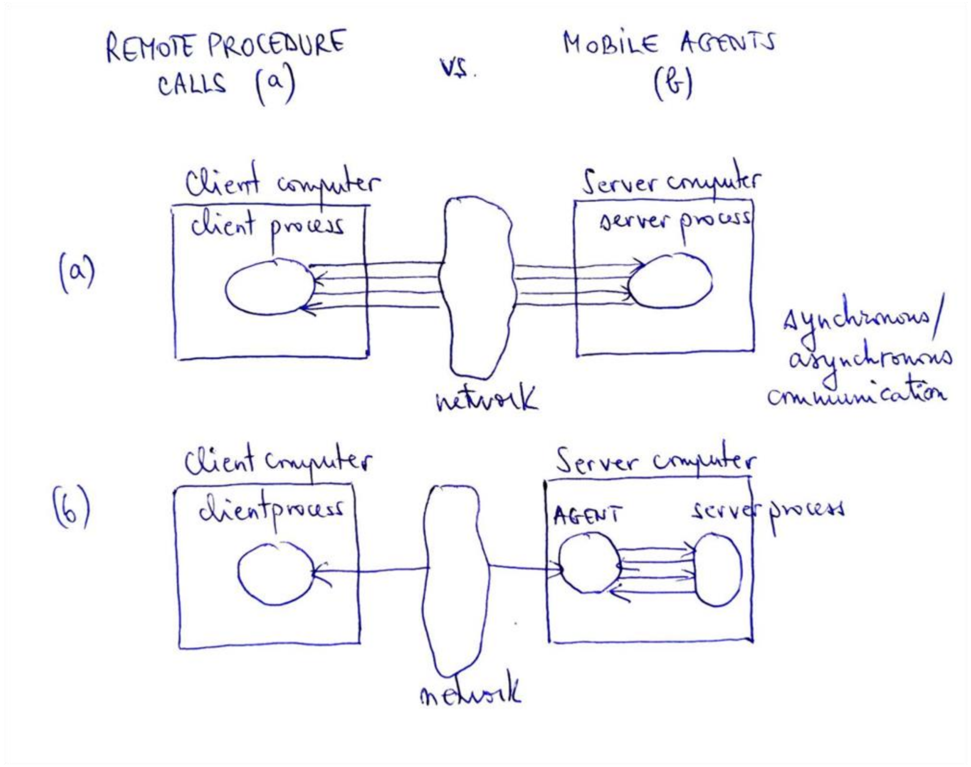
- **Agent = part-picking robot**
  - **P**: percentage of parts in correct bins
  - **E**: conveyor belt with parts, bins
  - **A**: jointed arm and hand
  - **S**: camera, joint angle sensors
- **Agent = taxi driver**
  - **P**: safe, fast, legal, comfortable trip, maximize profits
  - **E**: roads, other traffic, pedestrians, customers
  - **A**: Steering wheel, accelerator, brake, signal, horn
  - **S**: Cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard

3. Mobile agents ([1], Section 1.5)

- a **mobile agent** infrastructure offers an attractive approach to transforming public networks into computing platforms.
- the agents are *mobile*, they are able to move from one place to another
  - ➢ their program and state are encoded and transmitted across the network to another plc, where the execution restarts.

A **mobile agent**

- ➢ consists of a program code, persistent internal state and other attributes
  - travel plan, movement history, access privilege information
  - moves in a network from host to host to accomplish its task
  - is based on the **remote programming paradigm**
    - as opposed to the traditional **remote procedure calling** paradigm of *distributed computing*.
    - the image bellow illustrates the main differences between **remote procedure calls** (RPC) and **mobile agents**.



  - facilitates communicating applications involving distributed data and knowledge sources and services
  - can combine knowledge and data from the client and server and perform inference on the server, where the data and computing resources are located
  - can facilitate real-time interaction with a server
- TELESCRIPT – a language-based environment for constructing mobile agent systems.
- JADE framework – offers the functionality of building mobile agents: https://www.iro.umontreal.ca/~vaucher/Agents/Jade/Mobility.html

Mobile agent platforms provide
- host-independent execution environment for mobile agent programs
- standard communication languages using which agents and servers can engage in dialogs
- Sample mobile agent platforms
  - *Voyager* (written in Java, for DM and knowledge discovery - KD, bioinformatics)
  - *JavaMob* (in Java, uses CORBA for managing distributed objects, for DM and KD).

Mobile agent facility (MAF) is an attempt to standardize certain aspects of mobile agent platforms to facilitate interoperability among mobile agent platforms.

Why mobile agents?
- low-bandwidth networks
- efficient use of network resources

4. Agent programming languages ([1], Section 1.5)

- 1993, Yoav Shoham proposed a new programming paradigm, called **agent oriented programming** (**AOP**)
- the key idea of AOP
    - direct programming of agents in terms of mentalistic notions (**beliefs**, **desires**, **intentions**)
    - AOP is viewed as a kind of **post-declarative programming**
        - **what** to do, not how to do it
        - we state our goals and let the built-in mechanism figure out what to do in order to achieve them
    - AOP
        - specialization of OOP (the actors variant)
            - Scala
        - concurrent OOP with asynchronous communication
- AGENT0 programming language
    - Lisp-like syntax
- **AgentSpeak** programming language
    - 1996, Rao
    - Prolog-like syntax
    - BDI architecture
    - Jason framework is an interpreter for an extended version of AgentSpeak
    - AgentSpeak(PL) integrates the concept of probabilistic beliefs through the use of Bayesian Networks, to core BDI programming concepts
- Concurrent MetateM
    - Michael Fisher, 1993
    - direct execution of logical formulae (temporal logic)

Bibliography

[1] Weiss, G. (Ed.): Multiagent Systems: *A Modern Approach to Distributed Artificial Intelligence*, MIT Press, 1999 (available at www.cs.ubbcluj.ro/~gabis/weiss/Weiss.zip) [Ch. 1]

[2] Șerban Gabriela, *Sisteme Multiagent în Inteligenta Artificiala Distribuita. Arhitecturi si Aplicatii*, Editura RisoPrint, Cluj-Napoca, 2006

[3] Czibula Gabriela, *Sisteme inteligente. Instruire automată*, Editura RisoPrint, Cluj-Napoca, 2008