Agent and Multi-Agent Systems: architectures and reasoning

Concepts and Definitions

Wassila Ouerdane

13.06.2022

CentraleSupélec- SAFRAN AI Training

Organization of the course

Link for the course material:

`https://github.com/wassilaOuerdane/SAFRAN-IA-Promo3.git`

Organization

Dates	
13/06/2022	Course 1: Agent and MAS: Concepts and Definitions (Course+PW)
14/06/2022	Course 2 : Multi-Agent based Simulation (Course+PW)
15/06/2022	Course 3 : Interaction mechanisms: models and Implementation (Course+PW)

Motivations: Multi-Agent Systems and AI

Artificial Intelligence (AI)

"Designing and building machines that do things that would require intelligence if performed by humans" [Marvin Minsky].

Source: https://humanoides.fr/robot-appris-jouer-echecs-tout-seul/

Artificial Intelligence (AI)

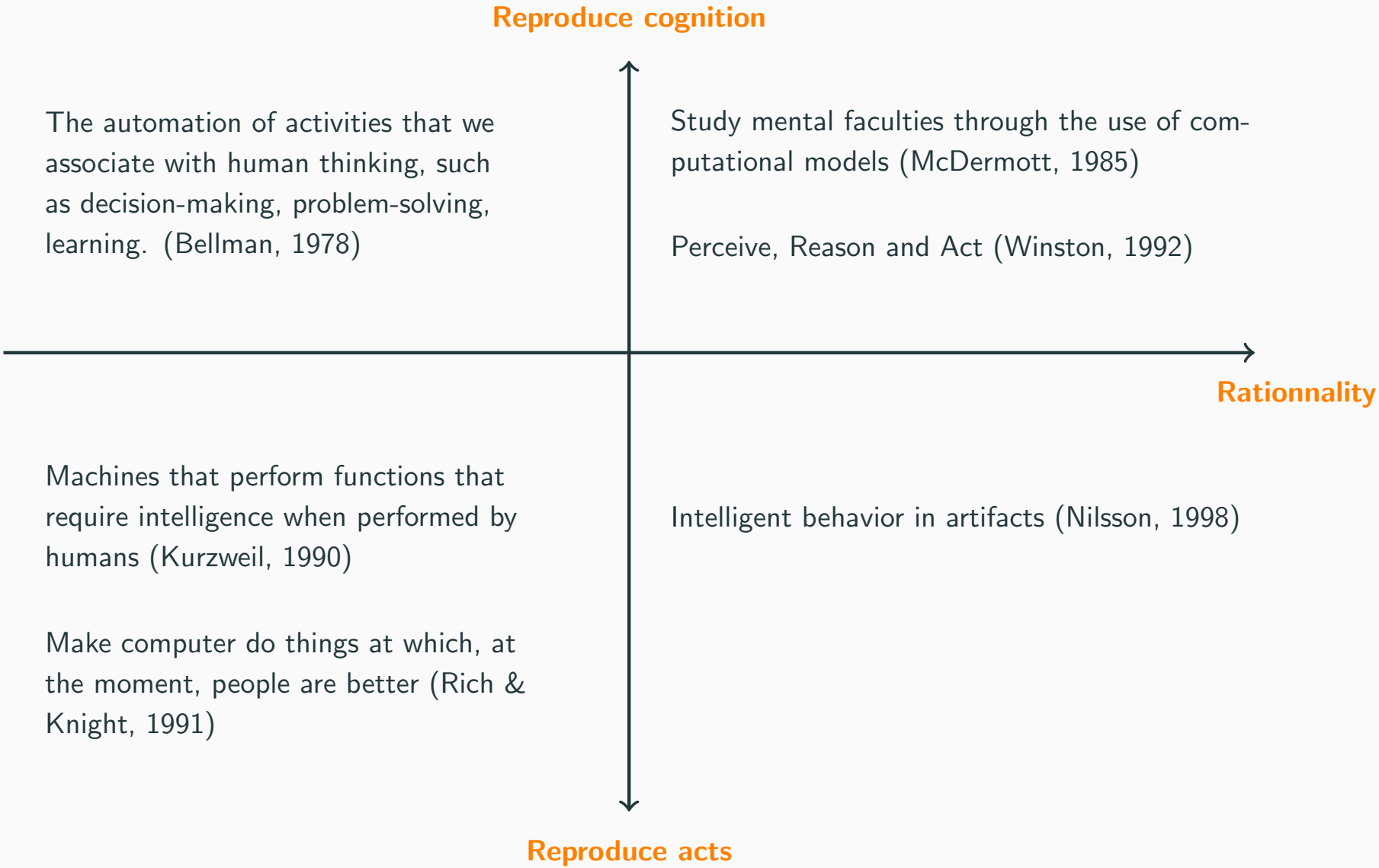
"Designing and building machines that do things that would require intelligence if performed by humans" [Marvin Minsky].

Source: https://humanoides.fr/robot-appris-jouer-echecs-tout-seul/

AI is not about

- ✗ Building intelligent machines
- ✗ Building machines that think

Artificial Intelligence: several definitions!



Thinking humanly: cognitive science

The human brain = an information processing machine. Systems should solve problems the same way humans do.

- Study of human brain and human behavior.
- Requires scientific theories of internal activities of the brain.
- Reproduction of these scientific theories and comparison with human behavior.
- Validation :
  - Predicting and testing the behavior of human subjects (top-down : cognitive sciences).
  - Direct identification from neurological data (bottom-up : Cognitive neuroscience).

Acting like an human : The Turing test (1950)

Idea

The Imitation game : do not define AI but test it !

Principle

An Interrogator converses with a man and a machine via a text-based channel. If the interrogator fails to guess which one is the machine, then the machine is said to have passed the Turing Test.

Requirements

NLP, Knowledge Representation, Automatic Reasoning, Learning, Interaction

Thinking rationally : Laws of Thought

- Rational thinking is based on logic inferences (Logics)
  $\rightsquigarrow$ Notations for statements about all kinds of objects in the world and the relations among them.
- Main difficulties :
  - Not all intelligent behavior is mediated by logical deliberation.
  - Not easy to take informal knowledge and state it in the formal logical notations (uncertainty).
  - Big difference between solving a problem in principle and solving it in practice (complexity).

Acting rationally : The rational agent approach

- Rational behavior :
  - doing the right thing : that which is expected to maximize goal achievement given the available information.
- Does not necessarily involve thinking (e.g. : blinking reflex) but in the service of a rational action, a goal.
- Goals are expressed in terms of the utility of outcomes.
- Being rational means maximizing your expected utility
- Rather than *Artificial Intelligence*, we could speak of Computational rationality.

Artificial Intelligence: several definitions!

There is more than one approach to AI...

Machine Learning and Neural Networks exist since the 1960s!

Artificial Intelligence: several definitions!

There is more than one approach to AI...

Machine Learning and Neural Networks exist since the 1960s!

1. Rule based systems — *explicit representation of knowledge*
   - Ask a domain expert **how** he makes the decision
   - Model this decision process in a computer program
   - Formal models (Knowledge Representation and Reasoning)
   - **Pros**: explainable AI (not always)
   - **Cons**: the bottomless pit of exceptions...

Artificial Intelligence: several definitions!

There is more than one approach to AI...

Machine Learning and Neural Networks exist since the 1960s!

1. Rule based systems — *explicit representation of knowledge*
   - Ask a domain expert **how** he makes the decision
   - Model this decision process in a computer program
   - Formal models (Knowledge Representation and Reasoning)
   - **Pros**: explainable AI (not always)
   - **Cons**: the bottomless pit of exceptions...

2. ML based systems — *knowledge implicit in data*
   - Ask an expert **what is important** to make the decision (*features*)
   - Find a corpus of (possibly annotated) data that cover the situation
   - Write a ML program and train it on the data
   - **Pros**: success on many difficult AI tasks
   - **Cons**: not personalized, hardly explainable, not always possible (data required)

Artificial Intelligence: several definitions!

MAS in all this?

- ✓ Reproduce cognition
- ✓ Rational behaviours (but not only)
- ✓ Rule-based models with explicit knowledge (but not only)

➜ **Reproduce collective** intelligence

Agents achieve things together, the result **emerges** from their interactions

Table of contents

1. Motivations: Multi-Agent Systems and AI

2. A brief history of MAS

3. What is an agent?

4. Properties and Defintions

5. The mesa platform for Python

13.09.2022 (PM)

A brief history of MAS

A brief history of Multiagent Systems

- Multiagent Systems appeared in the 1990s
- At the frontier between:
  - software engineering,
  - distributed computing,
  - artificial intelligence.

  with some contributions from human sciences!

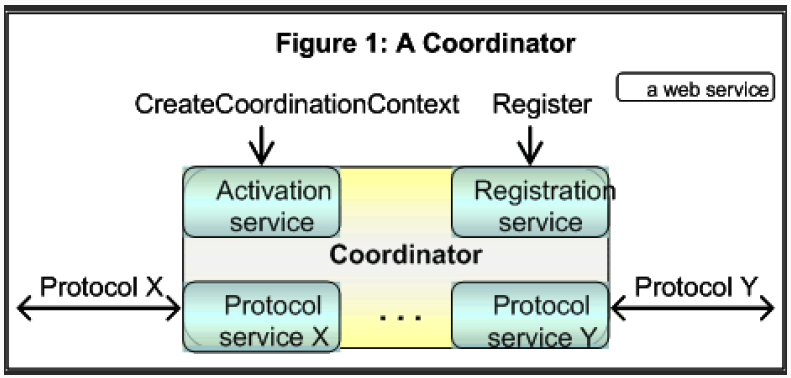
MAS as seen by a software engineer

- Improve the speed, the reusability, the readability, the security of code, . . . ?
- *From Object to Service*:
  - A piece of code, possibly a thread, should be provided with some machine-understandable interface;
  - Allows other code to use it without a priori knowledge of its operational content.
- Software engineers rely on **interaction protocols** to define how the different services should be assembled, how the information has to be exchanged between them, so as to produce some expected result.

MAS as seen by a software engineer

Multiagent system

- Platforms to support the development of such service-oriented software
- Coordination entities capable to interpreting protocols.



The OASIS Web Services Coordination standard (version 1.1, 2007)

What is important (for a software engineer)?

- Make it possible to assemble separate components or threads by designing well-formed interaction protocols

MAS as seen by a distributed systems engineer

- Different processes located on different physical systems
- All connected by some network infrastructure
- Interact with each other using some message passing mechanism

➜ The agents (or processes) can run on one single computer but:

- they have independent *memory spaces*,
- they have independent *runtimes* ($\rightsquigarrow$ They cannot rely on synchronous interactions).

Counter-example (synchronous code)

```
1 def compute(x,y,z):
2     ... computation here ...
3     return result
4
5 def calling_function():
6     ... beginning of code ...
7     res = compute(1,3,'hello')
8     ... end of code ...
```

calling function waits for the compute function to end before the code continues

MAS as seen by a distributed systems engineer

On the contrary, DS consider situation in which several problems can occur:

- The called process can not be running, or even not be accessible in the system;
- The called process can not answer, whatever the reason;
- If an answer arrives, there is no guarantee on the delay before it arrives;

All these problems assume a perfectly functioning infrastructure

$\rightsquigarrow$ programming a piece of software by taking into account the fact that the other half of the code is run in a completely asynchronous manner.

MAS as seen by a distributed systems engineer

In Distributed systems, several problems can occur:
even when assuming a perfectly operational infrastructure

- The called process can not be running, or even not be accessible in the system;
- The called process can not answer, whatever the reason;
- When it answers, there is no guarantee on the delay before it arrives;

$\rightsquigarrow$ programming a piece of software by taking into account the fact that **the other piece of code run in a completely asynchronous manner**.

MAS as seen by a distributed systems engineer

Consequences

- Don't wait for the answer: agents must carry on their activity while a possible answer is computed by another agent;
- Use timeouts: agents must deal with the absence of answers.

What is important (for a DS engineer)?

- Deal with asynchronous interactions and possible errors

and support the distribution of the system among different physical supports

(optional: most MAS architectures run on a single machine)

MAS as seen by an Artificial Intelligence engineer

- Artificial Intelligence is about having machine solve problems (through computation) that human beings solve with their intelligence.
- A subfiled of AI: Planning
  - design of actions and changes
  - solve problems that consist in finding a sequence of actions to go from a given state to another.
- The STanford Research Institute Problem Solver (STRIPS) is one of the first (and most famous) planner.

MAS as seen by an Artificial Intelligence engineer

```
1
2  Initial state: At(A), Level(low), BoxAt(C), BananasAt(B)
3      Goal state:    Have(bananas)
4
5      Actions:
6                  // move from X to Y
7                  _Move(X, Y)_
8                  Preconditions:  At(X), Level(low)
9                  Postconditions: not At(X), At(Y)
10
11                 // climb up on the box
12                 _ClimbUp(Location)_
13                 Preconditions:  At(Location), BoxAt(Location), Level(low)
14                 Postconditions: Level(high), not Level(low)
15
16                 // climb down from the box
17                 _ClimbDown(Location)_
18                 Preconditions:  At(Location), BoxAt(Location), Level(high)
19                 Postconditions: Level(low), not Level(high)
20
21                 // move monkey and box from X to Y
22                 _MoveBox(X, Y)_
23                 Preconditions:  At(X), BoxAt(X), Level(low)
24                 Postconditions: BoxAt(Y), not BoxAt(X), At(Y), not At(X)
25
26                 // take the bananas
27                 _TakeBananas(Location)_
28                 Preconditions:  At(Location), BananasAt(Location), Level(high)
29                 Postconditions: Have(bananas)
```

A solution: Move(A,C), MoveBox(C,B), ClimbUp(B), TakeBananas(B) (automatically is an NP-hard problem).

MAS as seen by an Artificial Intelligence engineer

From Single Intelligenec to Collective Intelligence

- BDI (Belief, Desire, Intention) logic: modeling actions and changes, planning

  $\rightsquigarrow$ programming systems that *simulate* human decision making
  with action selection based on its perception of the environment

- Social Animals Behaviour: solution *emerges* from the interactions between the individuals

  $\rightsquigarrow$ programming systems that interact with each other to compute a solution.
  e.g. Ant Colony Optimisation algorithm (Marco Dorigo, 1992)

What is important (for an AI engineer)?

- Solve problems in a distributed manner, using interactions and automated reasoning

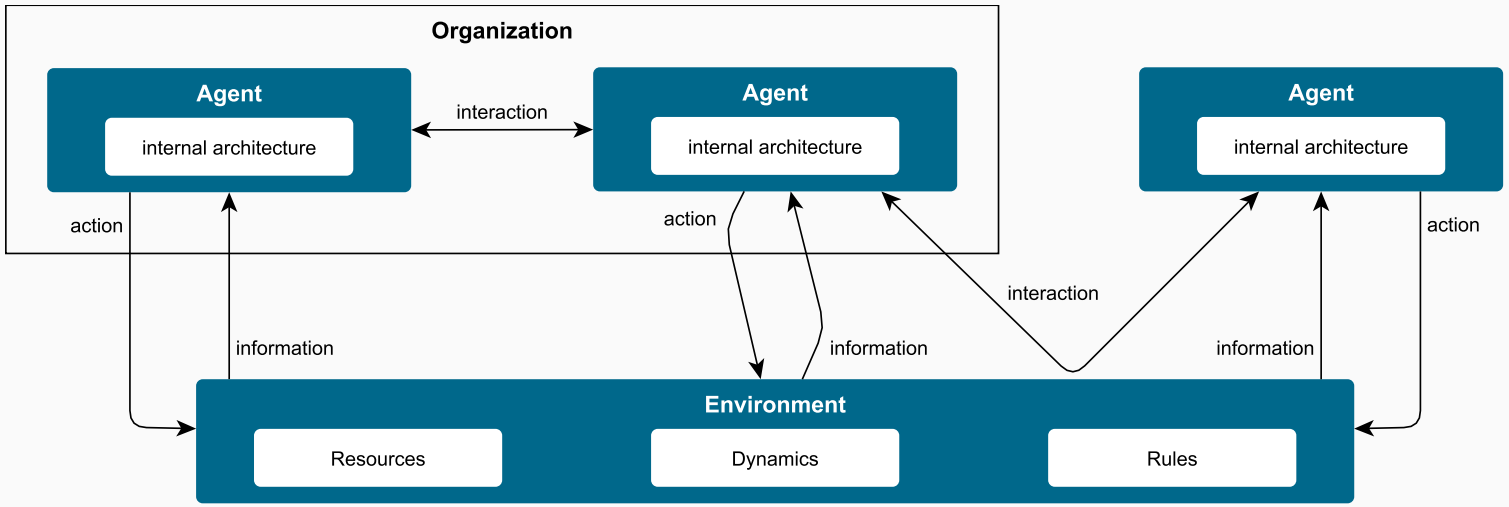
To be remembered

An agent is :

- a software process (or part of a process, or possibly attached to some physical components e.g. in robotics. . .)
- with some encapsulated data (i.e. other agents/processes can't modify directly this data),
- capable of providing a predefined set of services (its actions)
- and capable of reasoning about its actions and the other agents' actions and to coordinate with the other agents so as to participate in a **collective problem solving**.

To be remembered

- A MAS is a group of agents that share a common environment and that act in a distributed manner (the agents runtime are supposed to be asynchronous) to solve a problem for a user.
- A MAS always has a predefined purpose, and it is important to define the expected outcomes, should it be a solution to a concrete problem (e.g. a path in a TSP problem) or a set of observable properties in a multi-agent based simulation.



Some Practical Application of MAS

MAS Applied to Fault Detection

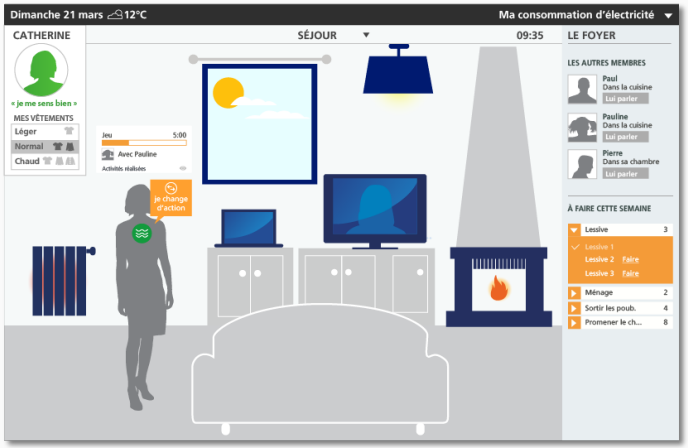
- SurferLab is a joint research lab formed by Bombardier Transport and the university of Valencienne, France.
- MAS is used for diagnosis of trains with the Jade platform.
- The multi-agent systems is capable to detect faulty cars so as to correct them faster[1].

[1] Antoine Le Mortellec, Joffrey Clarhaut, Yves Sallez, Thierry Berger, Damien Trentesaux, Embedded holonic fault diagnosis of complex transportation systems, Engineering Applications of Artificial Intelligence, Volume 26, Issue 1, 2013

Some Practical Application of MAS

MAS Applied to the Study of Electrical Consumption

- The R&D branch of EDF has been working on a MAS for the simulation of electrical consumption in households. This applied research project is called SMACH[2].
- The MAS platform is used for the generation of realistic load curves in answer to prospective studies: market research & pricing, new production means, new consumption habits, etc.



[2] https://www.youtube.com/watch?v=Hyc7XX4vEjw&t=17s

Some Practical Application of MAS

MAS Applied to Taxi Fleet Management

- The company Magenta Technology has developed in 2008 a MAS-based technology for the management of taxi fleets.
- This software solution is now being used by many taxi companies, such as Green Tomato Cars or Blackberry Cars, two major UK companies in London, UK.

Some Practical Application of MAS

MAS for Crowd Simulation in Movies

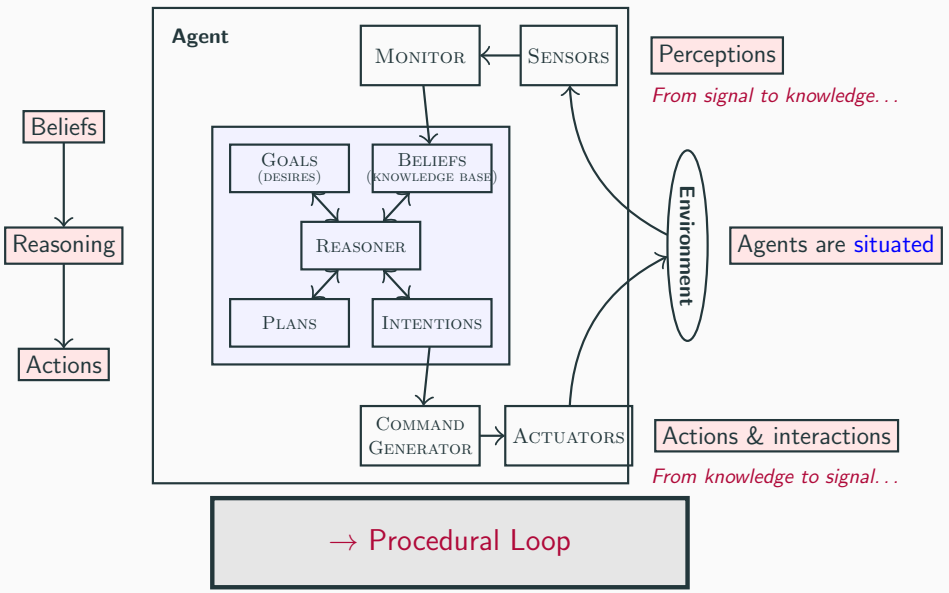
- **Massive Software** was originally developed for use in Peter Jackson's The Lord Of The Rings film trilogy.
- Massive has become the leading software for crowd related visual effects and autonomous character animation.
- It combines MultiAgent Systems with Graphical Design.

Some Practical Application of MAS

MASA

What is an agent?

PRS architecture [Georgeff and Lansky 87]



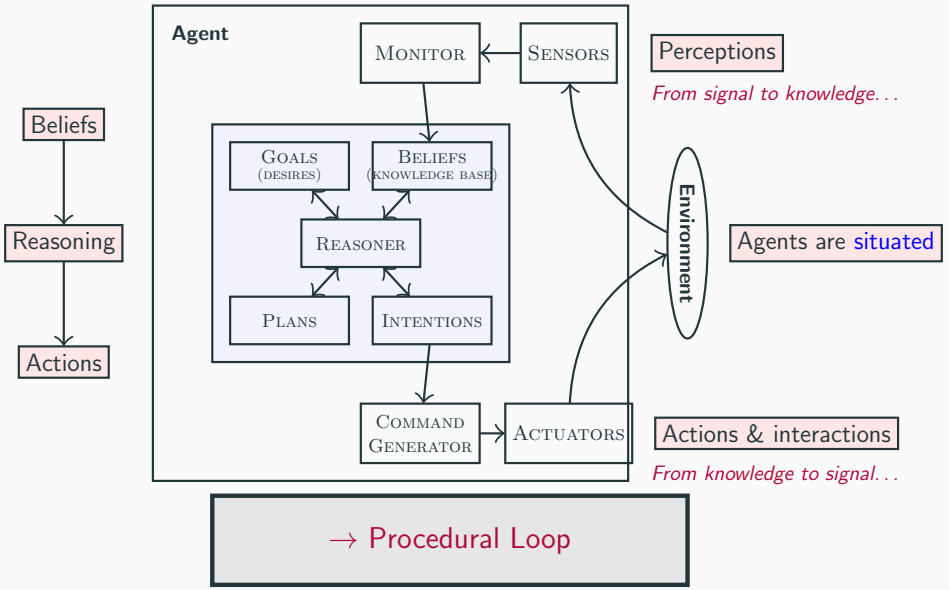
- PRS: Procedural Resoning System
- The environment: characterized by a (partially) observable state:
  - it can be modified,
  - some of the variable values can be observed by the agents.

  $\rightsquigarrow$ a set of observation functions and a set of modification functions.
- Agent: sensors (to access the observation function) + actuators (to access the modification functions).

PRS architecture [Georgeff and Lansky 87]

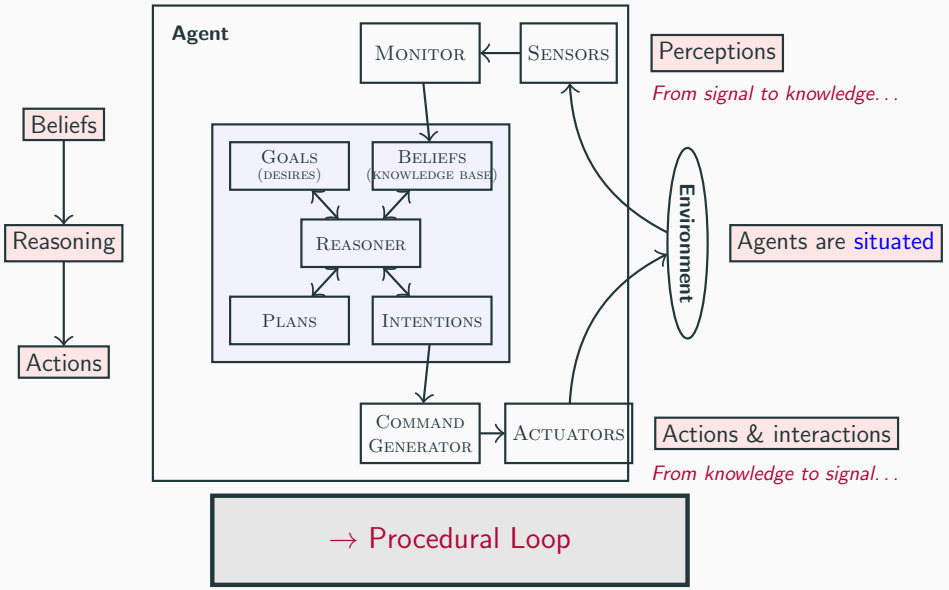
```
environment_variable = 0

def operation_increase_variable():
    global environment_variable
    environment_variable += 1

def perception_get_variable():
    global environment_variable
    return environment_variable

def agent(preferred_value):
    v = perception_get_variable() # this is a sensor
    while (v<preferred_value):
        operation_increase_variable() # this is an actuator
        v = perception_get_variable()

```

PRS architecture [Georgeff and Lansky 87]



- Procedural Loop: an agent runs continuously in three steps:
  1. Perception of the environment
  2. Deliberation = selection of the action (i.e. transforms the result of the perception into internal variables, called *beliefs*. It then combines these with a *set of goals* and a *set of plans* so as to decide for some *intention* that will be turn into a *concrete action* in the environment.)
  3. Action on the environment

PRS architecture [Georgeff and Lansky 87]



- Procedural Loop: an agent runs continuously in three steps:
  1. Perception of the environment
  2. Deliberation = selection of the action (i.e. transforms the result of the perception into internal variables, called *beliefs*. It then combines these with a *set of goals* and a *set of plans* so as to decide for some *intention* that will be turn into a *concrete action* in the environment.)
  3. Action on the environment

To keep in mind: an agent

- Is situated in an environment: it can perceive and act;
- Continuously performs a procedural loop: perception, deliberation, action.

An agent: remark

There is not "the" definition of an agent

[Russel and Norvig,1995]

"An agent is anything that can be viewed as perceiving its environment through *sensors* and acting upon that environment through *effectors*."

An agent: remark

There is not "the" definition of an agent

[Russel and Norvig,1995]

"An agent is anything that can be viewed as perceiving its environment through *sensors* and acting upon that environment through *effectors*."

[Wooldridge & Jennings, 1995]

"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"

...

Agent Models

- Reactive agents: do not make use of any internal variable: they directly wire all perception to a specific action (or set of actions), hence their name: they perceive and react.

Agent Models

- Reactive agents: do not make use of any internal variable: they directly wire all perception to a specific action (or set of actions), hence their name: they perceive and react.
- Cognitive agents:
  - add internal variables and data
  - a first type: internal variables (beliefs) to store its perception $\rightsquigarrow$ it has a memory of what it perceived, and it can make decisions based on this memory.

STRIPS example

`Move(A,C), MoveBox(C,B), ClimbUp(B), TakeBananas(B)`

Agent Models

- Reactive agents: do not make use of any internal variable: they directly wire all perception to a specific action (or set of actions), hence their name: they perceive and react.
- Cognitive agents:
  - add internal variables and data
  - a first type: internal variables (beliefs) to store its perception $\rightsquigarrow$ it has a memory of what it perceived, and it can make decisions based on this memory.
  - must be capable of handling with action failures.

STRIPS example

`Move(A,C), MoveBox(C,B), ClimbUp(B), TakeBananas(B)`

Agent Models

- Reactive agents: do not make use of any internal variable: they directly wire all perception to a specific action (or set of actions), hence their name: they perceive and react.
- Cognitive agents:
  - add internal variables and data
  - a first type: internal variables (beliefs) to store its perception $\rightsquigarrow$ it has a memory of what it perceived, and it can make decisions based on this memory.
  - must be capable of handling with action failures.
  - Note that "explicit planning" is not mandatory. Agent simply checks its current state of beliefs and search for a possible action. The "plan" is then hard-wired in the code. Different levels of "reasoning" about the beliefs: simple deduction, belief revision, diagnosis, ...

STRIPS example

`Move(A,C), MoveBox(C,B), ClimbUp(B), TakeBananas(B)`

Agent Models

- Reactive agents: do not make use of any internal variable: they directly wire all perceptions to a specific action (or set of actions), hence their name: they perceive and react.
- Cognitive agents:
  - add internal variables and data
  - a first type: internal variables (beliefs) to store its perception $\rightsquigarrow$ it has a memory of what it perceived, and it can make decisions based on this memory.
  - must be capable of handling with action failures.
  - Note that "explicit planning" is not mandatory. Agent simply checks its current state of beliefs and search for a possible action. The "plan" is then hard-wired in the code. Different levels of "reasoning" about the beliefs: simple deduction, belief revision, diagnosis, ...
- "explicit planning": agents that have explicit goals and that perform any sort of computation (from simple plan selection in a predefined list to complex planing algorithms such as SATPLAN) $\rightsquigarrow$ Rational agents.

To be remembered

According to the PRS model that is at the core of agent modeling, an *agent*:

- is situated in the environment (it can perceive and act);
- might have some internal data called beliefs and these are encapsulated within the agent (they are not accessible to other agents or to the environment);
- runs with a procedural loop (perceive, decide for an action, act) in an asynchronous w.r.t other agents;
- in the decision phase, it can do any sort of reasoning, from simple reaction (reactive agents) to more complex reasoning on the beliefs (cognitive agents);
- agents that have *explicit goals* and *planning procedures* in the deliberation phase are called rational agents.

Agent and MAS: Exercices

Open the notebook: IntroductionMAS_2022_Subject.ipynb.

Let us begin with two very simple agents in an environment

```
1  from time import sleep
2
3  class Environment:
4      def act(self,message):
5          print(message)
6      def perceive(self):
7          pass
8
9  class Agent:
10     def __init__(self,name,env):
11         self.name = name
12         self.env = env
13     def procedural_loop(self):
14         while True:
15             self.env.act("Agent "+self.name+" says hello!")
16             sleep(0.1)
17
18 class Runtime:
19     def __init__(self):
20         e = Environment()
21         (Agent("Alice",e)).procedural_loop()
22         (Agent("Bob",e)).procedural_loop()
23
24 Runtime()
25
```

Agent and MAS: Exercices

Questions

- Why does this not behave as a multiagent system?
- What is the problem?

Agent and MAS: Exercices

Questions

To overcome the above limitation, two solutions can be considered.

The first one is to write a **scheduler**, i.e. a piece of code that calls the procedural loops of all agents, one after the other.

- Modify the previous code so that Runtime creates two agents and calls a single-step procedural loop for all agents.

Agent and MAS: Exercices

Questions

Do we have a multi-agent system yet?

Agent and MAS: Exercices

Questions

Write a new multi-agent system in which the environment has some variable that the agents can perceive. Write two reactive agents: the first one increases the variable by a random value when it is even, the other one when it odd.

Concurrent Modifications

Writing asynchronous MAS with threads requires to consider possible concurrent modifications. In the general case, you must consider a possible interruption of the following sequence of code:

```
1 self.b = self.env.perceive()     # in Agent.procedural_loop()
2 if condition_on_b:                # in AgentX.act()
3     x = randint(1,4)              # in Environment.increase() -- 4
      lines
4     print("Agent " + name + " increase the value by "+ str(x))
5     self.v = self.v+
6     print("  --> " + str(self.v))
7
```

Concurrent Modifications

Writing asynchronous MAS with threads requires to consider possible concurrent modifications. In the general case, you must consider a possible interruption of the following sequence of code:

```
1 self.b = self.env.perceive()      # in Agent.procedural_loop()
2 if condition_on_b:                # in AgentX.act()
3     x = randint(1,4)              # in Environment.increase() -- 4
      lines
4     print("Agent " + name + " increase the value by "+ str(x))
5     self.v = self.v+
6     print("  --> " + str(self.v))
7
```

Questions

- What do you expect to be a problem? Explain why.
- What is a possible solution?

Properties and Defintions

Properties of an environment [Russel &Norvig 2003]

- 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? Independent episodes (ML algorithms can be used to build Markov Decision Processes representations of the MAS behavior.)?
- Static vs. dynamic – Does the environment change by processes beyond agent control (e.g. ant colony simulation in which pheromones dropped by the agents in the environment need to spread and evaporate with time)?
- Discrete vs. continuous – Is the number of actions and percepts fixed and finite?

Properties of the Agents in the MAS

Autonomous agents

The independence of one agent's behavior (i.e. action selection) with respect to some part of the system.

1. **At the agent level**, pro-action is the first degree of autonomy (i.e. independently from any specific signal from the environment).
   e.g. an agent that plays hide and seek and that begins seeking after ending a countdown.
2. **At the interaction level**, autonomy consists in having agents that do not necessarily accept other agent's requests (ignoring messages because they are considered irrelevant; not answering immediately while it could have; agent answers but refuses the request; etc.).
3. **At the MAS level**, *there is no* autonomy: agents are not autonomous w.r.t. the system. On the contrary, they perform what they are designed for. They follow well-defined protocols so that the global task is achieved. Otherwise, there is some misconception in the system.

Properties of the Agents in the MAS

Loosely vs tightly coupled?

1. **In tightly coupled MAS**, the developer has a complete view of the MAS, of possible actions performed by other agents. In other words, it can use some of this information to design action mechanism in a more efficient manner.
2. **In a loosely coupled MAS**, you must make no assumption on the design of other agents in the MAS. This means that you must define very robust behaviours to avoid deadlocks, infinite loops or unnecessary waiting times in communication.

Properties of the Agents in the MAS

Open MAS?

- a specific kind of system in which you assume that agents can enter and leave the system at any time during the execution.
- requires programmers to design specific mechanism to ensure that messages are not left unanswered, that some feature in the system will not disappear when not expected, etc.
- Concrete examples of open MAS are ant colonies or prey-predator simulations

Properties of the Agents in the MAS

Distributed systems?

1. Conceptual level: it implements distribution principles: encapsulated data, accessed via perception and action methods only, using a *procedural loop* that interleaves the perceptions and actions of agents.
   $\rightsquigarrow$ Netlogo or Gamma
2. Software level: agents get separated in different threads or processes (depending on the language and implementation). These threads generally run on a single computer.
   $\rightsquigarrow$ Repast Symphony
3. Physical level: specific code to connect the abstract actions and perceptions to physical operations, message passing, etc. (e.g. a network of smart sensors that exchange information with each other)
   $\rightsquigarrow$ The Java Agent Development Environment (Jade) , The Robot OS (ROS)

The mesa platform for Python
13.09.2022 (PM)

The mesa platform for Python

Open Notebook: money_model_Subject.ipynb

- Mesa is a Python framework for agent-based modeling.
- The running example will be the implementation of a simple Agent-based Computational Economics (ACE) model that simulates distributions of money, income, and wealth in society using statistical physics[3].

[3] https://arxiv.org/abs/cond-mat/0211175