Advanced Techniques in Artificial Intelligence

Curso 2021-2022

German Rigau
german.rigau@ehu.eus

Grado en Ingeniería en Informática

Topics

- Intelligent Agents
- Multiagent Systems
- Planning

1 Intelligent Agents

1. Introduction
2. Evolution of Agents
3. Architectures for Agents

1.1 Introduction

Due to unexpected system failures, a space probe approaching Saturn becomes disoriented and loses contact with its Earth base.

Instead of disappearing in the deep space, the probe recognises what has occurred in a crucial failure, diagnoses the problem, corrects it, reorients it and makes contact with the base again.

NASA Deep Space 1 –DS1– 1998

1.1 Introduction

Ruritania's main airport air traffic control system suddenly fails, leaving flights in its vicinity unchecked.

Distributed Vehicle Monitoring Testbed DVMT, 1991

OASIS 1992 Sydney

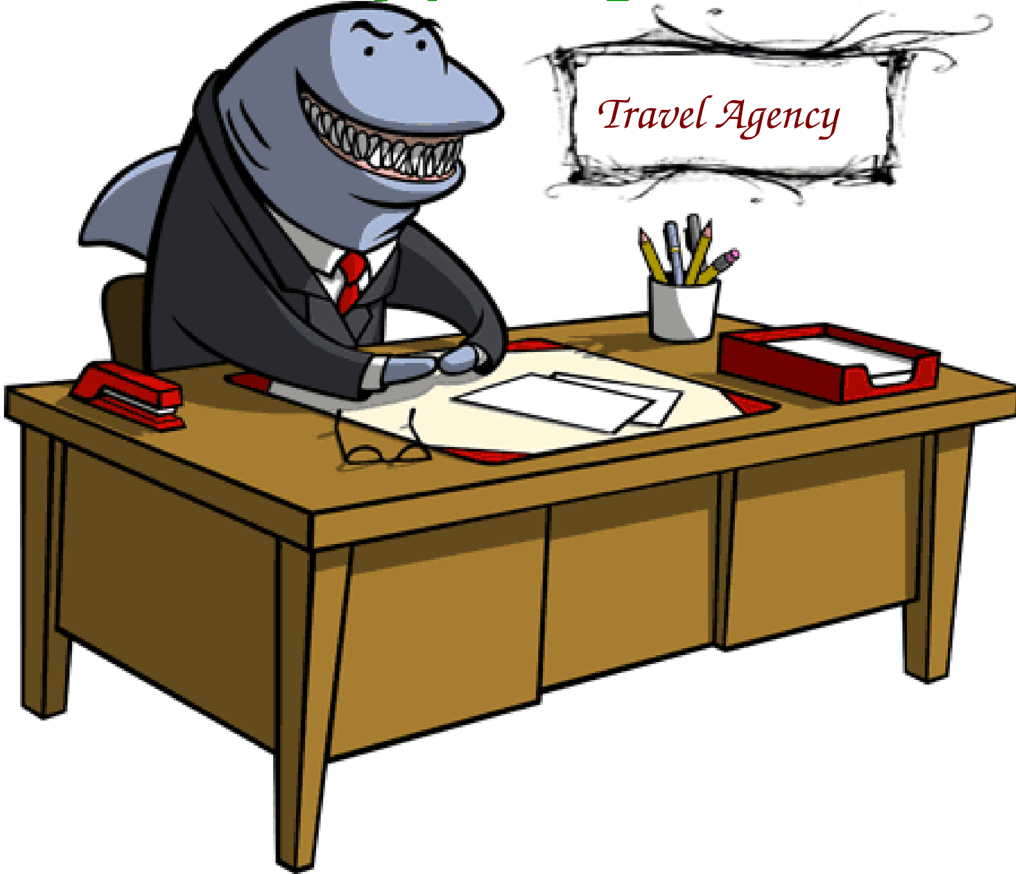
Fortunately, the air traffic control systems at neighbouring airports recognise their failure and cooperate to track and manage the affected flights.

The potentially disastrous situation ends without incident.

1.1 Introduction

After a hideous course you need a holiday somewhere sunny and funny. After specifying your requirements to your phone, it talks to several websites that sell flights, hire hotel rooms and rent cars.

After negotiating hard with them on your behalf, your phone shows you a perfect holiday package!!



1.1 Introduction

- Agents can help you: negotiate prices, search for cheaper products, organize trips,...
- Types of agents: tourist agents, commercial agents, stock brokers, judicial agents,...
- What is an agent in general?

1.1 Introduction

- Characteristics of **Agent Technology**:
- Collecting works from three decades of computer engineering and AI.
- Fusion of three streams:
  - Software Engineering (SE)
  - Artificial Intelligence (AI)
  - Distributed Systems (DS)

1.1 Introduction

- From **SE** (Object technology):
  - Encapsulation, independence
  - Messages between objects (communication)
  - Classes, inheritance

1.1 Introduction

- From **AI**:
  - Knowledge Representation:
    - rules, frames, logic ...
  - Reasoning, ...
  - Learning, ...
  - Perception, vision, language, ...

  - "Intelligent" Agent Approach:
    - Sensors
    - Smart Process
    - Effectors (or actuators)

1.1 Introduction

- From **Distributed Systems**:
    - Distribution of data and processes
    - Connectivity, Networks, Protocols
    - Interoperability
    - Internet

    - Especially Multi-Agent Systems!

1.1 Introduction

- Concept of Agent:
    - There is no commonly accepted definition.
    - (Wooldridge & Jennings, 1995)
        - Any computational process located in an environment and capable of performing autonomous actions in that environment to achieve its objectives.

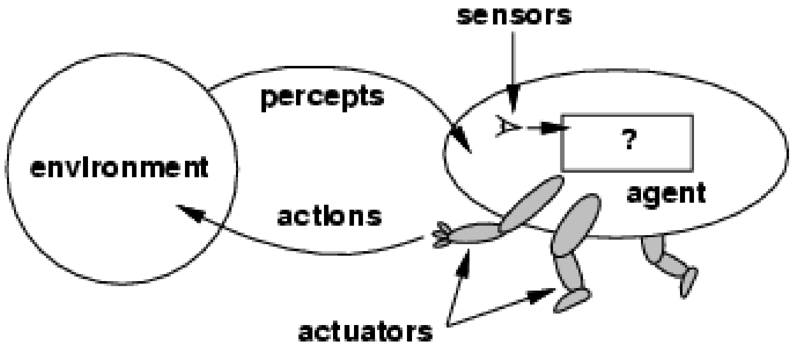
1.1 Introduction

- Characteristics of Agents (1):
  - **Autonomy**: ability to act without direct human intervention or other agents.
  - **Reactivity**: an agent is immersed in a certain environment (habitat), from which it perceives stimuli and reacts in a pre-established time.
  - **Initiative**: an agent must not only react to changes in their environment, but must have a proactive nature, taking the initiative to act guided by the objectives that must meet.
  - **Rationality**: it has specific objectives and always tries to carry them out.

1.1 Introduction

- Characteristics of Agents (2):
    - **Sociability**: ability to interact with other agents, using some language of communication between agents.
    - **Mobility**: ability to move in a computer network.
    - **Truthfulness**: does not communicate false information intentionally.
    - **Benevolence**: has no contradictory objectives and always tries to perform the task that is requested.

1.1 Introduction



- Sensors: to perceive the environment
- Actuators: to modify the environment

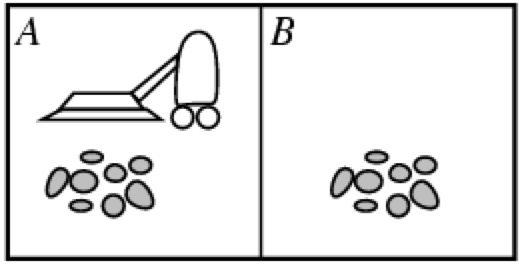
- *What can a sensor be? What an actuator?*

- Actions are a function of the history of perceptions

$$[f: P^* \rightarrow A]$$

1.1 Introduction

- The (simple) world of a vacuum cleaner



- Percibe: place and content, e.g. [A, Dirty]
- Actions: Left, Right, Aspirate, NoAct.

- *What should be the behavior of a rational agent?*

1.1 Introduction

- An agent should try to "do the right thing", depending on what he perceives and the actions he can take.

- Performance measure: an objective criterion for measuring the success of an agent's behavior
    - For example, a vacuuming agent could be the amount of garbage collected, the amount of time spent, the amount of electricity consumed, the amount of noise generated, etc.

1.1 Introduction

- For each possible sequence of perceptions, a rational agent must select an action that maximizes its performance measure, taking into account the evidence provided by the sequence of perceptions and all the knowledge that incorporates the agent.
- Rationality is different from omniscience (absolute knowledge)
- Agents can carry out actions in order to obtain useful information (information gathering, exploration of the environment)
- An agent is autonomous if his behavior is determined by his own experience (with the ability to learn and adapt)

1.1 Introduction

- Example (**PEAS**):
  - *autonomous taxi, driverless taxi* …

- **P**erformance Measure:
  - security, comfort, speed, legality, maximize profits, …
- **E**nvironment:
  - streets, traffic lights, traffic, pedestrians, customers, …
- **A**ctuators:
  - steering wheel, accelerator, brake, signal, horn, …
- **S**ensors:
  - cameras, sonar, speedometer, GPS, motor sensors, microphones, …

Types of Environments

- Proposed by (Russell and Norvig, 2010):
  - observable vs. partially observable;
  - deterministic vs. non deterministic;
  - episodic vs non episodic;
  - static vs dynamic;
  - discreet vs. continuous.

Observable vs. partially observable

- An environment is observable if an agent can obtain information
  - complete
  - correct
  - updated

  about their status.
- Thus, the sensors of an agent provides access to the complete state of the environment at each instant of time.

- The more observable an environment, the easier it is to build agents that can operate on it.
- Most real-life environments are not accessible.

Deterministic vs. no deterministic

- An environment is deterministic if any action has a single effect on it, and there is no uncertainty about the resulting state.
- Thus, the next state of the environment is completely determined by the current state and the action performed by the agent.

- Non-deterministic environments are more problematic
- The physical world, for all intents and purposes, can be considered as non-deterministic.
- In complex environments, while essentially deterministic, predicting the effect of an action may be too complex to be feasible.

Deterministic vs. no deterministic

- Frameworks for physical simulation:
    - http://www.ode.org
    - http://opensimulator.org
    - http://gazebosim.org
    - http://www.mujoco.org/
    - https://dartsim.github.io/
    - https://home-platform.github.io/
    - https://github.com/clic-lab/chalet
    - http://virtual-home.org/
    - http://gibsonenv.stanford.edu/
    - ...

Episodic vs. no episodic

- An environment is episodic if the behavior of the agent can be divided into sequences of perception-action not related to each other (episodes).
- Thus, the agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action), and the choice of action in each episode depends only on the episode itself.

- Episodic environments are easier for developers because the agent can decide what action to perform only on the basis of the current episode;
- The agent do not need to remember previous episodes or reason about the next ones.

Static vs. dynamic

- An environment is static if we can assume that it remains unchanged (except for the actions of the agents themselves).
- Thus, the environment does not change while the agent is deliberating.
- The environment is semi-dynamic if it does not change over time, but the agent's behavior does.

- Dynamic environments are more difficult for the developer because other entities can interfere with the actions of the agent.
- Many real-life environments are very dynamic:
  - The real world,
  - Internet …

Discrete vs. continuous

- An environment is discrete if there is a fixed, finite number of actions and perceptions in it.
- Thus, the environment can be described by a limited number of clearly defined perceptions and actions.

- Chess describes a discrete environment.
- The driving of a taxi is in a continuous environment.

- Of course, discrete environments are much easier for developers.

1.1 Introduction

- System Based on Agents
  - It uses the agents as an abstraction, but still being modeled in terms of agents, can be implemented without any software structure corresponding to them.
- Multi-Agent Systems
  - It is designed and implemented with several agents interacting with each other, to achieve the desired functionality.

1.1 Introduction

- Advantages of agent technology:
  - Improves functionality and quality.
  - Lower cost (reusability).
  - Reduces maintenance.
  - Easy integration with other technologies (web, DBs, components, ...)
  - They simplify the work of engineers (agent patterns).

1 Intelligent Agents

1. Introduction
2. Evolution of Agents
3. Architectures for Agents

1.2 Evolution of Agents

- Beginnings: (1975-1980): First works in the area of Artificial Intelligence (AI), ...
- Distributed IA (80s): Blackboard architecture, network of contracts (negotiation), organization and scientific societies, …
- Consolidation (90s): Congresses and scientific publications, prototypes of industrial interest, mobile agents, agent-oriented programming, …
- Super-human results (2010s-): deep neural networks, reinforcement learning, …
- Singularity? (??): self-improvement cycles up to surpassing all human intelligence …

- *What conferences (national and international) are there on multi-agent systems?*

1.2 Evolution of Agents

- Types of agents:
  - According to its individual characteristics:
  - **Reactive** agents, simple tasks in an event-reaction cycle.
  - **Cognitive** agents, complex tasks (reasoning, planning or learning) in a percention-assimilation-reasoning-acting cycle.

1.2 Evolution of Agents

- Types of agents:
  - According to its interaction mode:
  - **Agent-agent**: ACL and KQML languages, and communication protocols RMI, CORBA, SOAP, HTML, ...
  - **Agent-environment**: DBs, servers, libraries, sensors …
  - **Agent-person**: natural language (voice or text), sensors, semi-standard languages, graphics, … interface agents

1.2 Evolution of Agents

- Types of agents:
  - According to its social behaviour:
  - **Individual** agents
  - **Cooperative** agents, roles, responsibilities, common plans, norms, conflict resolution, special agents, negotiation ...

1.2 Evolution of Agents

- Types of agents:
  - According to its use:
  - **Domain of application**: electronic commerce, telecommunications, economy (bag), administration, leisure and entertainment, ...
  - **Task performed**: monitoring, diagnosis, information search, systems control, simulation, ...

Intelligent Agent

- An intelligent agent is a system capable of autonomous and flexible actions in some environments.

- Flexible means (Wooldridge and Jennings, 1995):
    - reactive
    - proactive
    - social

Reactivity, Proactivity, Sociability

- Reactivity is the ability of an agent to perceive its environment, and to respond in a timely manner to the changes that occur in it, in order to meet its design goals.
- Proactivity is the ability of an agent to take the initiative in order to meet their design goals.
- Sociability is the ability of an agent to interact with other agents in order to meet their design goals.
- Interacting means cooperating, coordinating, negotiating.

Reactivity, Proactivity, Sociability

- Very difficult (indeed, an open research problem) if an agent is required to be reactive, proactive and social simultaneously.
- We are looking for a balance between:
  - Planning achievable goals
  - Pursuing the objectives
  - Reacting to changes in the environment
  - Recognizing the opportunities of the moment
  - Interacting with other agents
  - ...
- How should an agent distribute his resources and time between these goals?
- Difficult even for humans!

AI vs. DAI

AI	**DAI**
a **unique** agent	**multiple** agents
Intelligence: Property of **one** agent	Intelligence: Property of **multiple** agents
Cognitive process of a **unique** agent	**Social process** from **multiple** agents

MAS vs classical DAI

- **DAI** (Distributed AI):
    - A particular problem is divided into smaller problems. These subproblems have a common knowledge. A solution method is provided for the every subproblem.
- **MAS** (Multi-Agent System):
    - Several agents coordinate their knowledge and actions. The solution method is not provided.

- Currently DAI is used as a synonym for MAS.

Agents vs. Objects

- Objects:
  - a state (encapsulated): control over an internal state
  - capabilities for passing messages to other objects
- Java:
  - Private and public methods.
  - Objects know their own state, but without having full control over their behavior.
  - An object cannot prevent others from using their public methods.

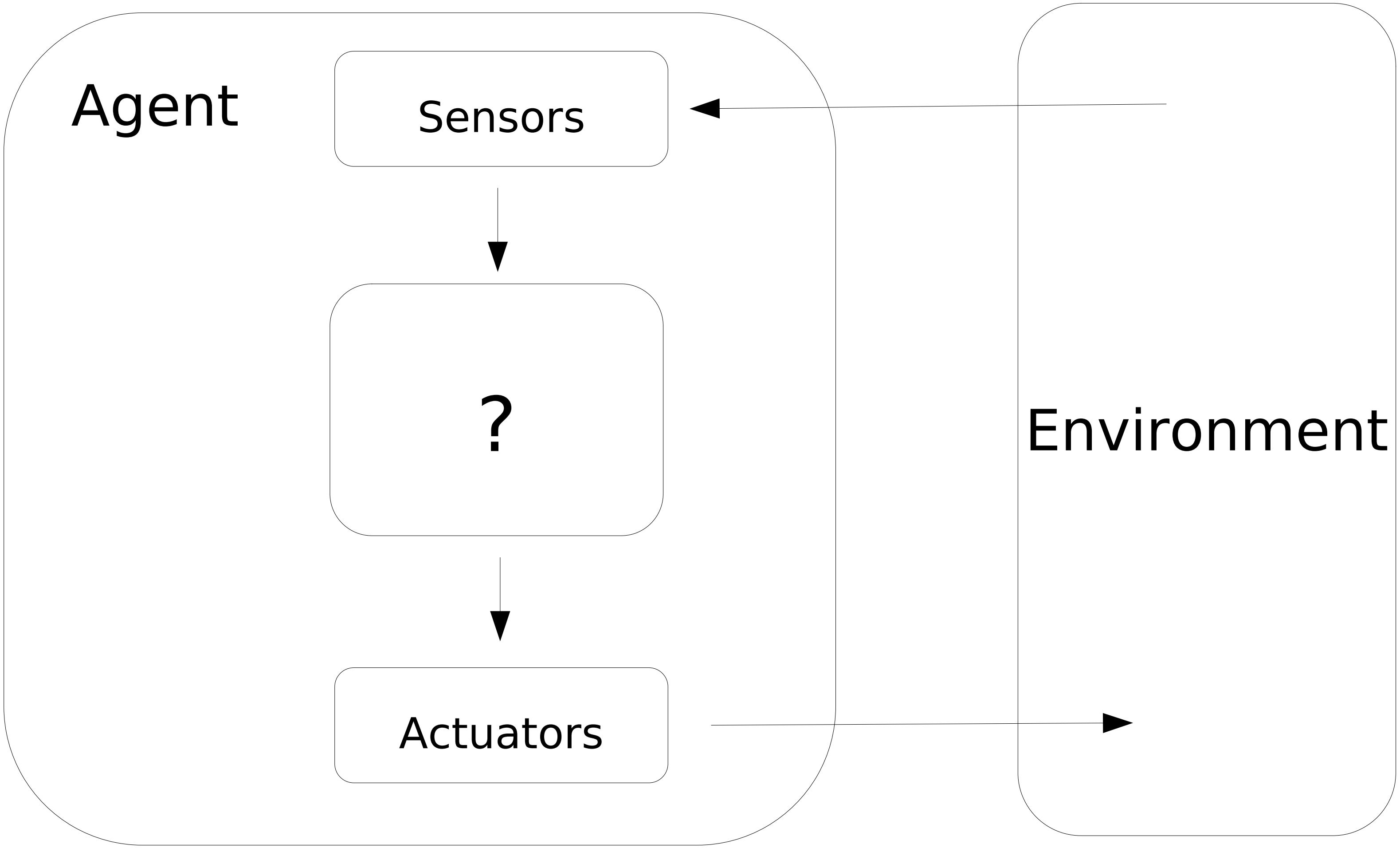
Agents vs. Objects

- **Agents**:
    - Agents communicate with other agents and ask them to execute actions for them.
    - Objects always do what they are asked, agents don't.
    - For objects there is no analogy to being reactive, proactive or social.
    - MAS are multi-threaded or multi-process: each agent can have its running thread
    - For objects, only the system as a whole has one thread.

1.3 Architectures for Agents

- Reactive Architectures
- Deliverative Architectures
- Hibrid Architectures

1.3 Architectures for Agents



Agent

Sensors

?

Actuators

Environment

1.3 Architectures for Agents

- The Architecture:
    - It determines the mechanisms that the agent uses to react to stimuli, to act, to communicate, etc.
    - Specifies how the agent's internal structure is: how it is decomposed into sets of modules that interact with each other to achieve the desired functionality
    - groups techniques and algorithms

1.3 Architectures for Agents

- Scenario with a **single** agent ...

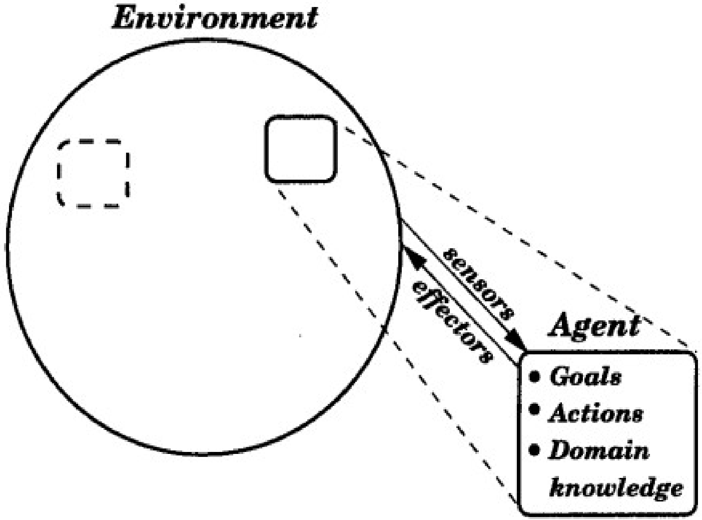


Figure 2: A general single-agent framework. The agent models itself, the environment, and their interactions. If other agents exist, they are considered part of the environment.

1.3 Architectures for Agents

- Complete scenario with **multiple** agents ...

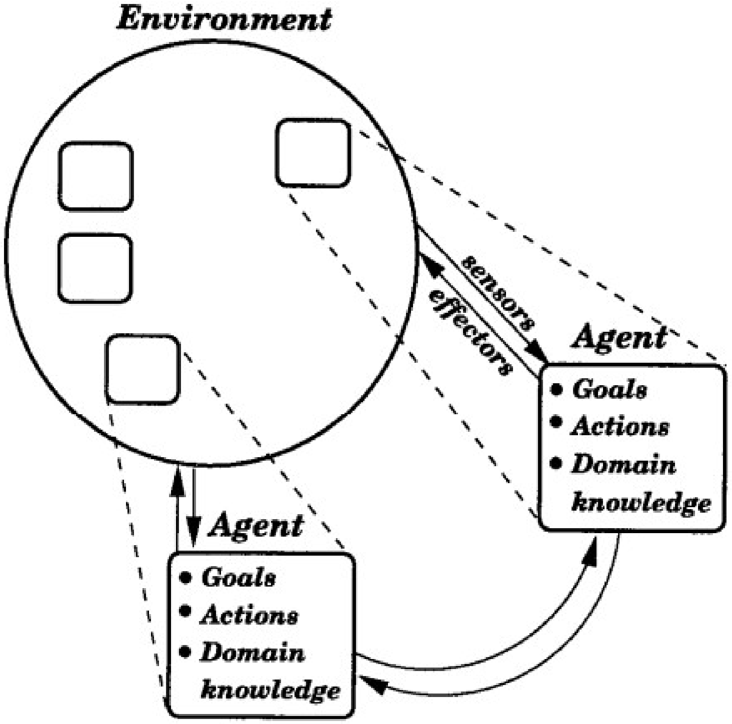
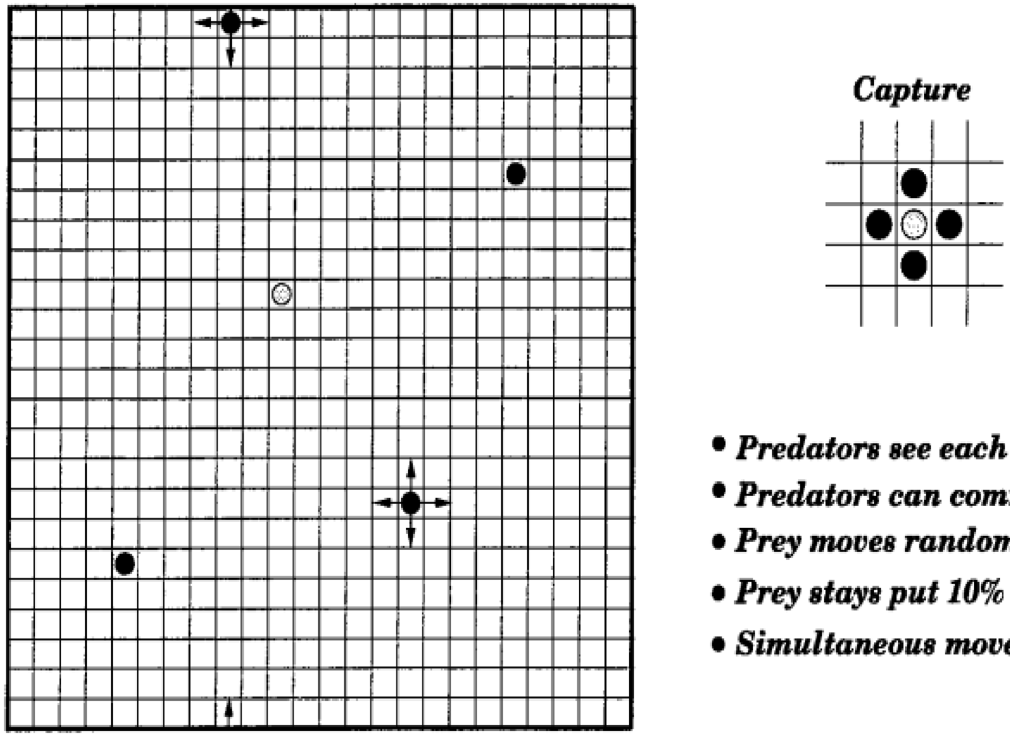


Figure 3: The fully general multiagent scenario. Agents model each other's goals and actions; they may also interact directly (communicate).

1.3 Architectures for Agents

- Hunting scenario with a prey and multiple predators ...



Orthogonal Game in a Toroidal World

- ***Predators see each other***
- ***Predators can communicate***
- ***Prey moves randomly***
- ***Prey stays put 10% of time***
- ***Simultaneous movements***

Figure 4: A particular instantiation of the pursuit domain. Predators are black and the prey is grey. The arrows on top of two of the predators indicate possible moves.

1.3 Architectures for Agents

- Hunting scenario with a single agent ...

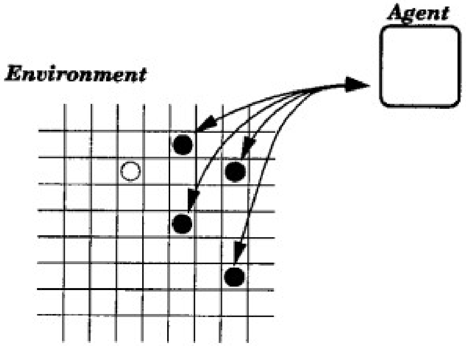


Figure 5: The pursuit domain with just a single agent. One agent controls all predators and the prey is considered part of the environment.

1.3 Architectures for Agents

- Hunting scenario with multiple homogeneous agents but without communication …

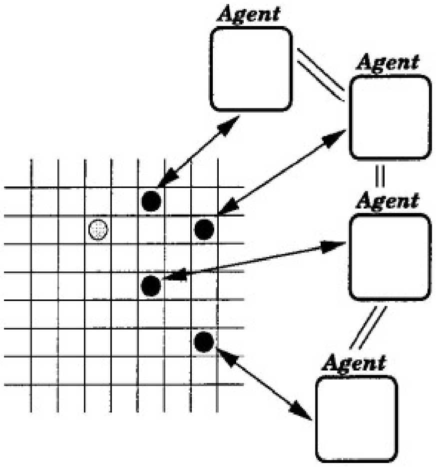


Figure 6: The pursuit domain with homogeneous agents. There is one identical agent per predator. Agents may have (the same amount of) limited information about other agents' internal states.

1.3 Architectures for Agents

- Hunting scenario with a multiple heterogeneous agents but without communication ...

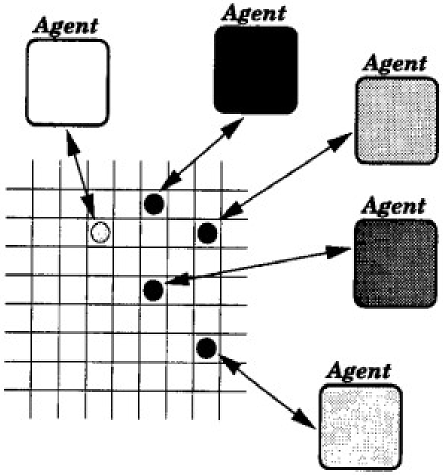


Figure 8: The pursuit domain with heterogeneous agents. Goals and actions may differ among agents. Now the prey may also be modeled as an agent.

1.3 Architectures for Agents

- Hunting scenario with multiple heterogeneous agents but with communication …

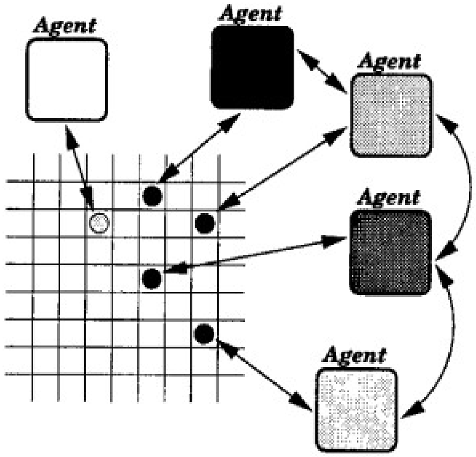


Figure 10: The pursuit domain with communicating agents. Agents can still be fully heterogeneous but now the predators can communicate with one another.

1.3 Architectures for Agents

- Which is the best option?
  - Homogeneous vs. Heterogeneous
  - Communication vs. No communication
- According to what criteria?
  - Results
  - Cost/Dificulty of implementing

Reactive architectures

- Idea:
  - Intelligent behaviour arises from the interaction of agents with their environment.
  - Intelligence emerges by combining simple behaviours and multiple interactions.

Reactive architectures

- Subsumption architecture (Brooks 1991)
  - Agent's behaviour is aimed towards reaching a goal
  - A behaviour is the result of many individual actions.
  - Associating actions to situations
  - The rules are of the form:
    - Given a situation -> Perform an action

Reactive architectures

- Subsumption architecture (Brooks 1991)
  - Several behaviours can be triggered simultaneously. How to choose between them?
  - A subsumption hierarchy allows to prioritize behaviors by structuring them in layers.
  - Upper layers represent more general behaviours.

Reactive architectures

- Example: Exploring a planet.
  - A distant planet contains gold. Several stand alone vehicles are available. The samples should be taken to a mother ship that landed on the planet. It is not known where the gold is. Due to the topography of the planet there is no connection between vehicles.
- Field gradient
  - The mother ship sends radio signals.

Reactive architectures

- Behaviour rules

- (1) **IF** detects an obstacle **THEN** change direction
- (2) **IF** (samples on board AND at the base) **THEN** drop samples
- (3) **IF** (samples on board AND not on base) **THEN** follow the gradient
- (4) **IF** detect samples **THEN** collect samples
- (5) **IF** true **THEN** take a random path

- Following this strict order (Subsumption hierarchy):
  - 1 < 2 < 3 < 4 < 5

Reactive architectures

- Pros:
  - Simplicity, economy (computational requirements), robust to failures and elegant.
  - Immediate response, …
- Cons:
  - Decisions based on local information (with global effects)
  - Difficult to design purely reactive agents who can learn from experience ...
  - The relationship between agents, environments and behavior is not completely clear ...
  - Agents with ≤ 10 behaviors are feasible. But the more layers, the more complicated it is to understand what is happening.

Deliberative architectures

- Idea:
  - Model (symbolic representation) of the environment, explicitly represented.
  - Planning system as logical reasoning mechanisms based on pattern matching and symbolic manipulation.
  - Based on classical planning theory:
    - Given an initial state they are able to generate plans to reach the target state.

Reasoning

$$\begin{array}{l} A \rightarrow B \\ A \\ \hline B \end{array}$$

Reasoning

$$\begin{array}{l} A \text{ -> } B \\ A \\ \hline B \end{array}$$

$$\begin{array}{l} A \text{ -> } B \\ A \\ \hline ? \end{array}$$

Deduction

$$\begin{array}{l} ? \\ A \\ \hline B \end{array}$$

Induction

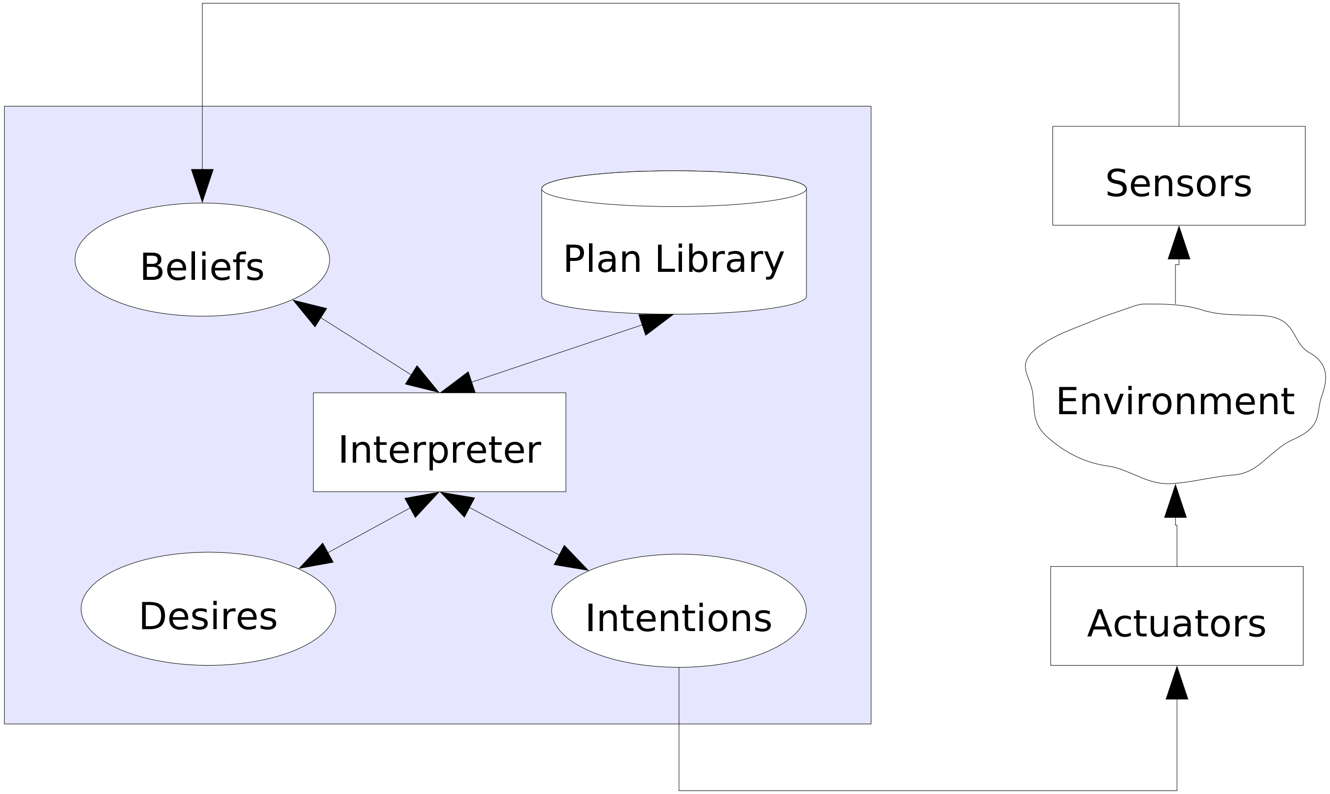
$$\begin{array}{l} A \text{ -> } B \\ ? \\ \hline B \end{array}$$

Abduction

Deliberative architectures

- **BDI** architectures are based on the assumption that the mind (mental state) of agents consists of:
    - **B**eliefs: what the agent believes to be true about the world (information).
    - **D**esires: state (s) of the world that agents want to establish (motivation).
    - **I**ntentions: what the agent really intends to do and how to do it (deliberation).
- The world for an agent is the other agents, the environment, and the agent itself.

Deliberative architectures



Deliberative architectures

- BDI allows for the interaction between two forms of reasoning:
  - Goal-based (means to an end)
  - Assessment of competing possibilities
- Addressing the problem of limited resources

Deliberative architectures

- Goal-based (means to an end)

- from the AI sub-field that deals with planning
- Given: an initial state, a set of final states (ends), as well as a description of actions (means or capabilities)
- Goal: to find a sequence of actions (plan) that goes from the initial state to the final state

Deliberative architectures

- Goal-based reasoning
- Example:
  - Initial state:
    - Being at home, having a picture, nails, not having nor hammer.
  - Final State:
    - the picture is framed and placed on the wall
  - Plan:
    1. go to a store
    2. acquire a frame and a hammer
    3. go home
    4. frame the picture
    5. use a hammer and nails to hang the picture on the wall

Deliberative architectures

- Assessment of competing possibilities

- from decision theory
- Given some competing possibilities
- The possibilities are considered and one of them is selected
- The selection is based on the utility function of the agent taking into account their beliefs (what the agent knows) and desires (what the agent wants)

Deliberative architectures

- Assessment of competing possibilities
- Example:
  - Desire: enjoy a meal
  - Possibility 1: go to Paco's cantine
  - Possibility 2: go to a luxury restaurant
  - Beliefs: I have no funds
  - Decision: go to Paco's cantine

Real-time applications (deliberative)

1) The environment is non-deterministic, that is, in each moment the environment can evolve in several ways.
2) The system is non-deterministic, that is, potentially at each moment there are different actions to be performed.
3) The system may have several different objectives simultaneously
4) The best actions/procedures to achieve the objectives depend on the situation of the environment and are independent of the internal state of the system.
5) The environment can only be detected locally.
6) The speed of deliberation and the agent actions are limited by the speed at which the environment evolves.

Real-time applications (deliberative)

- The characteristics:
  - 4) the best action depends on the environment state and is independent of the internal state of the system,
  - 1) non-deterministic environment, and
  - 5) local detection implies that
- it is necessary that there be some component of the system that can represent information about the state of the world.

~> Beliefs!

Real-time applications (deliberative)

- The characteristics:
  - 3) different simultaneous objectives and
  - 5) local detection implies that
- it is necessary that the system also has information on the objectives to be fulfilled.

~> Desires!

Real-time applications (deliberative)

- Idea: reconsider the choice of actions at each step.
- Dilemma: this is potentially very expensive and the chosen action could possibly be invalid when selected.
- Assumption: it is possible to limit the frequency of the review and achieve a balance between too much or too insufficient reconsideration. Recall characteristic 6 (reasonable frequency of calculations and actions).
- Implication: It is necessary to include a component of the system that represents the currently chosen course of action.

~> Intentions!

Knowledge Bases (deliberatives)

- Beliefs:
  - It is usually stored on a belief database.
    - *I'm a computer student.*
    - *I'm in my fourth year, first term*.
- Desires:
  - They are usually stored on a database of desires.
    - *I want to graduate in computer science*.
- Plans:
  - Recipes on how to reach the goals. Usually, somehow structured, for example, nested actions and stored on a library of plans.
    - *Attend a lot of classes.*
    - *Perform a lot of assigments.*
    - *Overcome a lot of exams.*

Knowledge Bases (deliberatives)

- Language for knowledge representation, e.g. Prolog

- Beliefs:
  - study(me, informatics).
  - course(me, 4), term(me, 1).

  Desires:
  - grade(me, informatics).
- Plan:
  - [attend(me, ATAI), attend(me, ...),...]

BDI v1 agent control iteration

```
while true do
    observe the world;
    update the internal world model;
    decide what intent to pursue next;
    reason to get a plan for the intention;
    execute the plan;
end
```

- Decide: carefully considering all options.
- Planning: once committed to do something, how to reach the goal?
- Replanning: What if during the execution of the plan, things are running out of control and the original plan fails?

BDI v2 agent control iteration

```
Set<Belief> beliefs = initBeliefBase();
while ( true ) {
    Percept percept = getNextPercept();
    beliefs = beliefRevision(beliefs, percept);
    Set<Intention> intentions = deliberation(beliefs);
    Plan plan = generatePlan(beliefs, intentions);
    execute(plan);
}
```

BDI v2 agent control iteration

- The agent's internal state is a triplet (B, D, I)
- *Intentions* are the most important thing.
- *Beliefs* and *intentions* generate desires.
- Desires may be incompatible with each other.
- The intentions are recalculated based on the current intentions, desires and beliefs.
- Intentions should persist, normally.
- Beliefs are constantly updated and therefore generate new desires.
- From time to time *intentions* must be reexamined.

BDI v3 agent control iteration

```
Set<Belief> beliefs = initBeliefBase();
Set<Intention> intentions = initIntentionBase();
while ( true ) {
    Percept percept = getNextPercept();
    beliefs = beliefRevision(beliefs, percept);
    Set<Desire> desires = findOptions(beliefs,intentions);
    intentions = filter(beliefs,desires,intentions);
    Plan plan = generatePlan(beliefs, intentions);
    execute(plan);
}
```

BDI v3 agent control iteration

- Now we have some initial intentions.
- The deliberation has been divided into two components:

  1) Generate options (desires).

  2) Filter the right intentions.
- Intentions may be in a stack (e.g. priorities).

- But, there is no way to re-plan if something goes wrong!

BDI v4 agent control iteration

```
Set<Belief> beliefs = initBeliefBase();
Set<Intention> intentions = initIntentionBase();
while ( true ) {
    Percept percept = getNextPercept();
    beliefs = beliefRevision(beliefs,percept);
    Set<Desire> desires = findOptions(beliefs,intentions);
    intentions = filter(beliefs,desires,intentions);
    Plan plan = generatePlan(beliefs,intentions);
    while( !plan.isEmpty() ) {
        Action head = plan.removeFirst();
        execute(head);
        percept = getNextPercept();
        beliefs = beliefRevision(beliefs,percept);
        if ( !sound(plan,intentions,beliefs) ) {
            plan = generatePlan(beliefs,intentions);
        }
    }
}
```

BDI v4 agent control iteration

- But … what is a plan?
- A π plan is a list of primitive actions. They lead us, through their successive application, from the initial state to the target state.

BDI architectures

- Classes of agents:
  - Intrepid:
    - Do not stop to reconsider intentions
    - Low temporal and computational cost
    - Suitable for environments that do not change quickly
  - Cautious:
    - Constantly stop to reconsider intentions
    - They exploit new possibilities
    - Suitable for rapidly changing environments
  - Meta-control?
    - Who determines when to be bold or cautious?

BDI architectures

- Pros:
  - Intuitive Model, it is possible to recognize the processes to decide what to do and how to do it.
  - Functional decomposition, which determines the class of subsystems needed to create the agent
- Cons:
  - The biggest difficulty, as always, is knowing how to implement these functions efficiently.
  - Difficult to balance an agent behavior that has both initiative and reactivity

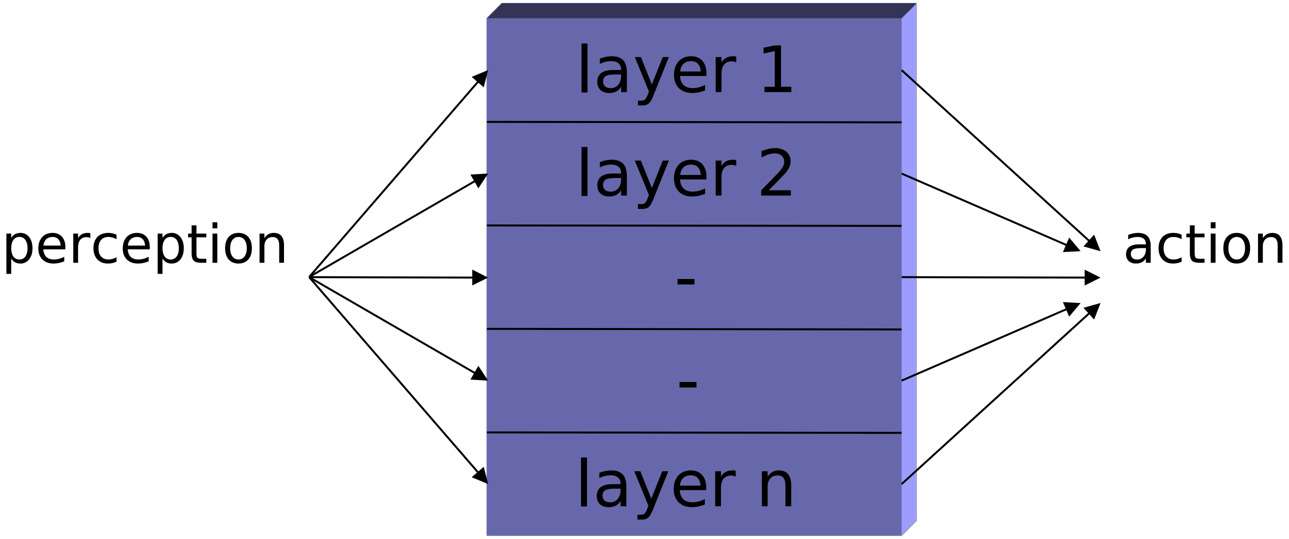
Hybrid Architectures

- Architectures formed by two or more subsystems:
- Reactive:
    - To process stimuli that do not need deliberation.
- Deliberative:
    - Symbolic model of the world
    - Generates plans: determines actions to be carried out to satisfy the local and cooperative objectives of the agents

- Layered Structure: *Horizontal* and *Vertical*

Hybrid Architectures

- Layered structure: Horizontal
    - Each layer is directly connected to sensors and actuators
    - Contributes with suggestions to action to act



perception

layer 1

layer 2

-

-

layer n

action

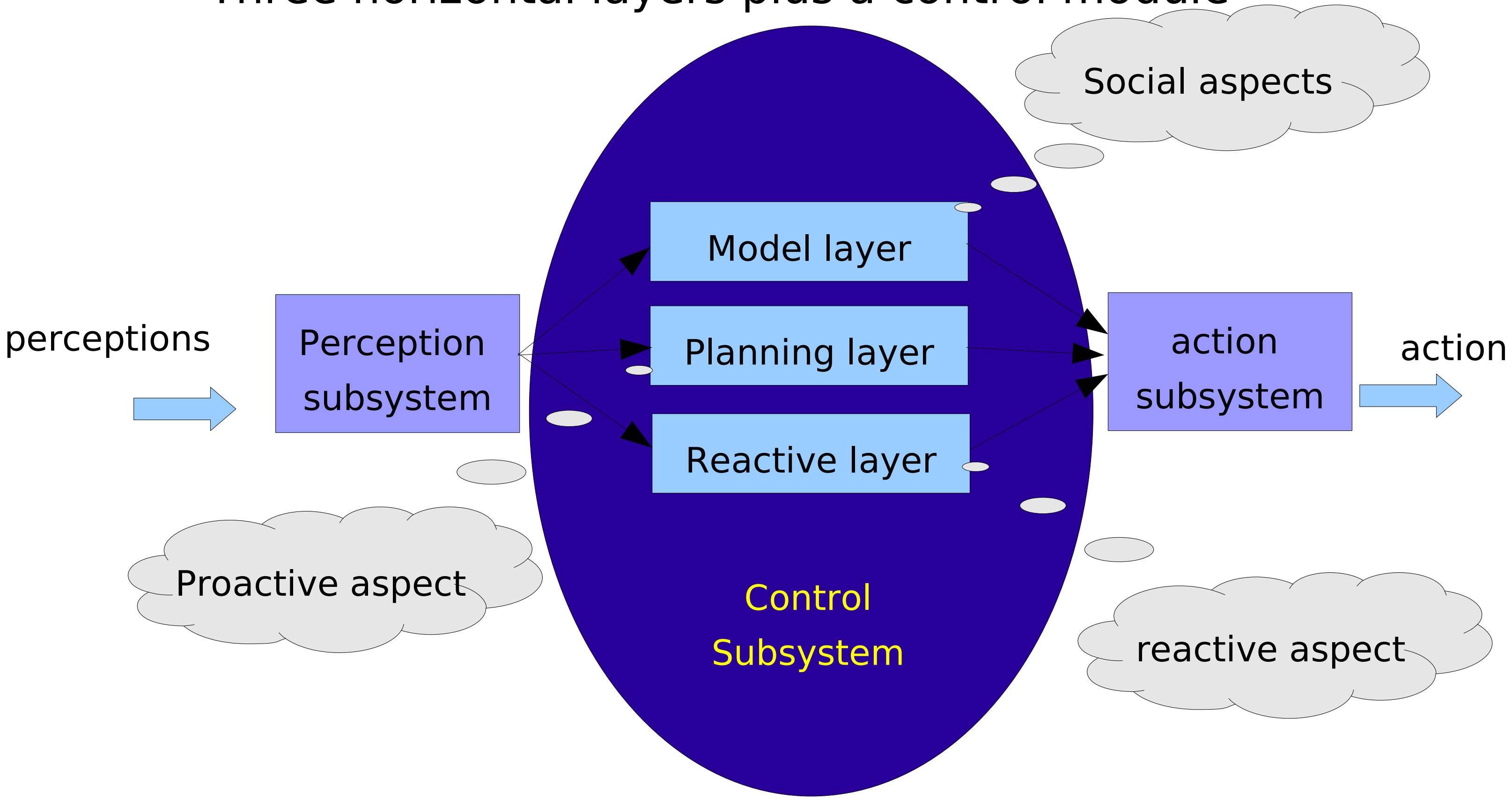
Hybrid Architectures

- Layered structure: Horizontal

- Pros:
  - Simplicity, n different behaviors -> n layers.
- Cons:
  - Coherence? mediating function that decides which layer has control of the agent,
    - Ensures consistency,
    - Bottleneck:
      - n layers with m possible actions ->
      - m^n interactions to consider!

Hybrid Architectures

- Example of layered structure: Horizontal
  - TOURINGMACHINES (Ferguson, 1992)
  - Three horizontal layers plus a control module

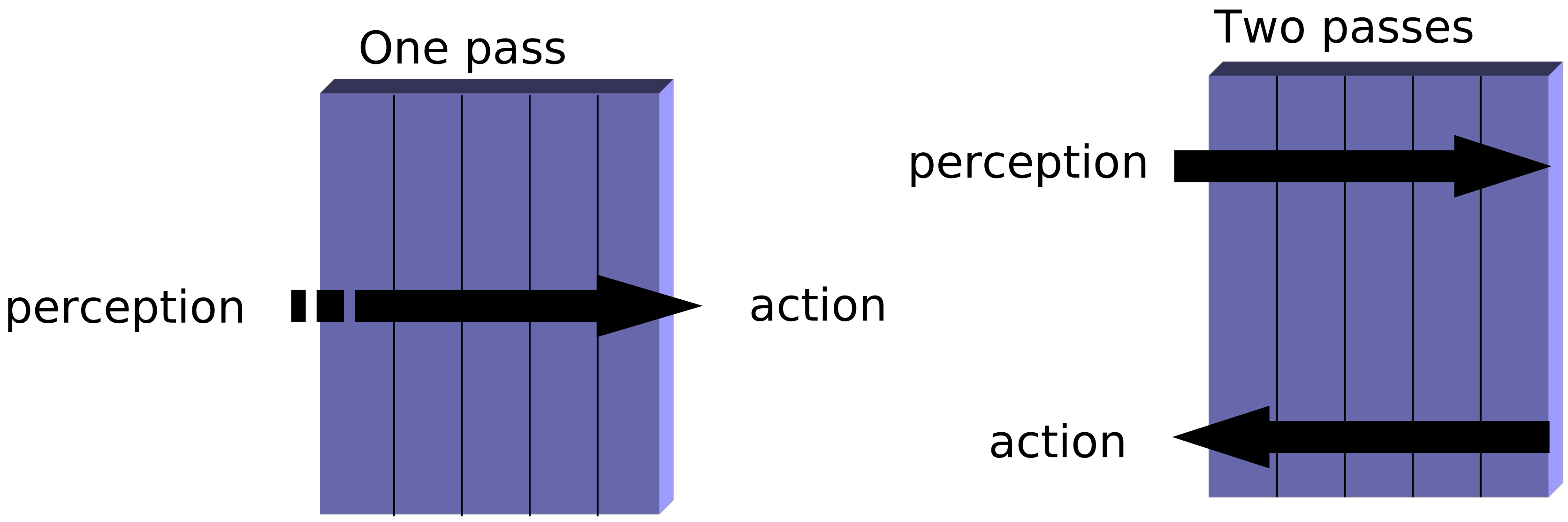


Hybrid Architectures

- Example of layered structure: Horizontal
  - Reactive layer:
    - more or less immediate responses to changes in the environment, implemented with action-situation rules
  - Planning layer:
    - represents the agent's initiative, contains a skeleton library of plans, called schemes. Plans structured to decide what to do.
  - Layer modeling:
    - represents the entities of the environment
  - Control system:
    - decides which layer has control over the agent to avoid conflicts, implemented with control rules that can suppress the inputs and inhibit the outputs

Hybrid Architectures

- Layered structure: Vertical
    - Sensors and the actuators are connected to one layer



Hybrid Architectures

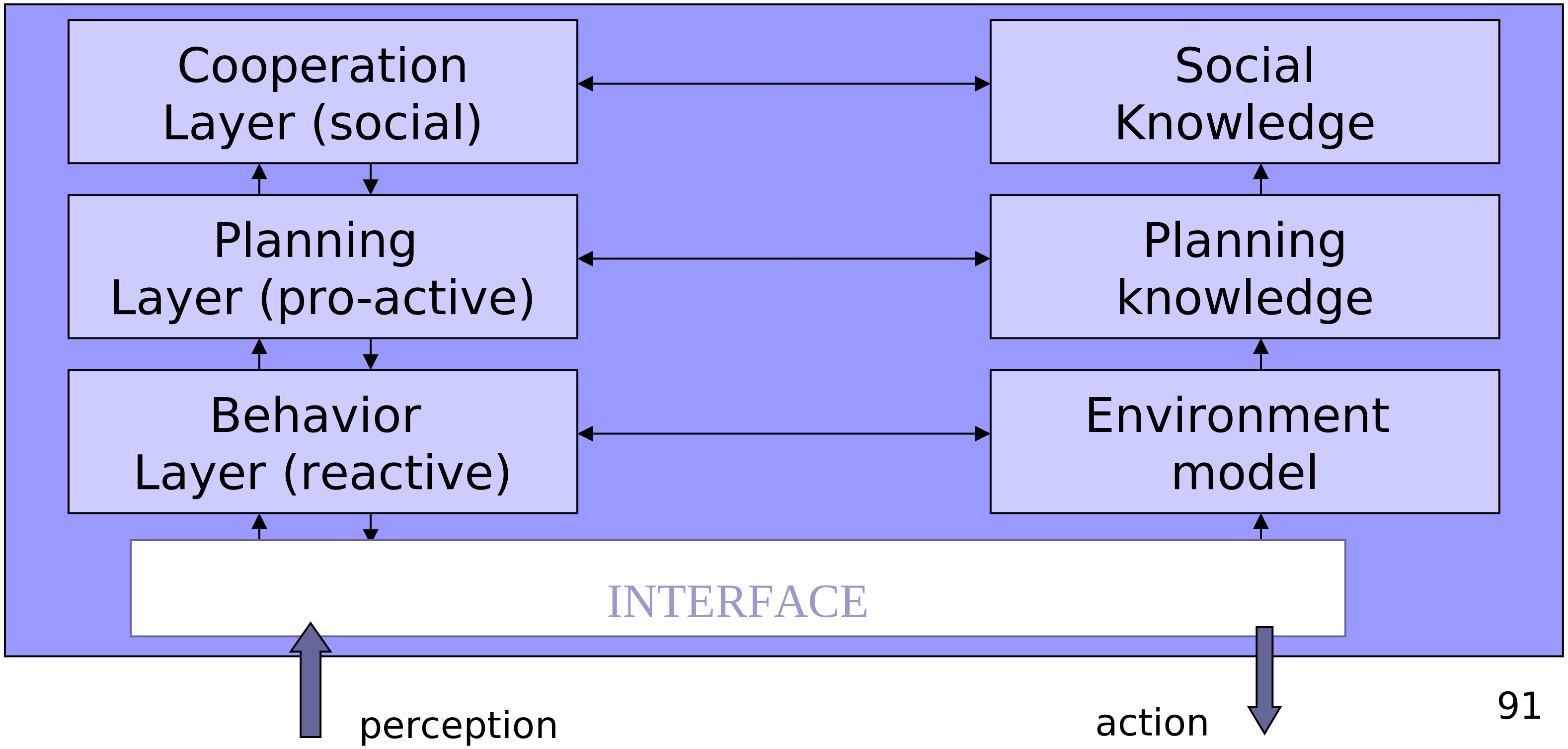
- Layered structure: Vertical

- Pros:
  - Good for balancing the different behaviors of the agent (reactivity, initiative)
- Cons:
  - Lack of clarity and flexibility
  - n-1 interfaces between layers with m possible actions

    m^2 * (n-1) interactions to consider

Hybrid Architectures

- Example of layered structure: Vertical
    - INTERRAP (Muller, 1997)
    - Three vertical layers, each layer has its knowledge base, two passes



Hybrid Architectures

- Example of layered structure: Vertical
    - INTERRAP (Muller, 1997)
    - Social knowledge:
        - represents the plans and actions of other agents in the environment
    - Planning knowledge:
        - represents the plans and actions of the agent himself
    - Environment Model:
        - information about the environment
    - Interaction between layers:
        - Bottom-up activation
        - Top-down execution

References

- TAIA 2012-2013. Maite Urretavizcaya. Grupo Galan. LSI. 2013.

- Brooks, R. A. (1991). Intelligence without representation. Artificial intelligence, 47(1), 139-159.
- Ferguson, I. A. (1992). TouringMachines: An architecture for dynamic, rational, mobile agents. Cambridge CB2 3QG, England: University of Cambridge, Computer Laboratory.
- Müller, J. P. (1997). A cooperation model for autonomous agents. In Intelligent Agents III Agent Theories, Architectures, and Languages (pp. 245-260). Springer Berlin Heidelberg.
- Russell S. and Norvig P. (2010). Artificial Intelligence: A Modern Approach, 3rd Edition. Pearson.
- Stone, P., & Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. Autonomous Robots, 8(3), 345-383.
- Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: Theory and practice. Knowledge engineering review, 10(2), 115-152.

Advanced Techniques in Artificial Intelligence

Curso 2021-2022

German Rigau
german.rigau@ehu.eus

Grado en Ingeniería en Informática