# Agents and Multiagent Systems

In this section, we give one of the definitions for term agent as it is understood by [1] and [2] in the field of Artificial Intelligence. Understanding what (intelligent) agent is in terms of AI, has particular importance for us because our goal is to create one. More precisely an agent – bot that would be able to play StarCraft and learn some of the decision making involved from professional players. Because it is intractable to learn whole decision-making trough IRL at once, and we need to use other techniques in our bot (actually many techniques are used together to do the job as is shown in section {link}) decomposition of some kind is inevitable. We decided to design our agent as Multiagent system (MAS) to help us decompose the problem of playing StarCraft as we think that it is the natural way how to approach decomposition. Therefore we also give a characterization of Multiagent Systems in this chapter to show that MAS is a logical way how to approach the problem. Using MAS of some sort for architecture is nothing new in StarCraft bot development as we show in {link chapter}. At the end of this chapter, we present reasoning behind the idea of using MAS as a tool for decomposition.

## Description of (intelligent) agent

We use the definition from [1] and [2] which defines an agent as an **entity** that perceives its **environment** through **sensors** and **acting** upon that environment through **actuators**. This idea is illustrated on figure {link}. Despite the vagueness of this definition, it can frame the thing to which we refer as game playing agent well. In the case of a human agent playing StarCraft, we can consider player’s eyes as sensors and his hands as actuators giving commands to the game. On the contrary, given this definition, one can consider even the game program as an agent as it receives commands as sensory inputs and acts on the environment by displaying the game.

{figure of simple agent architecture}

One of the differences between those two agents is **autonomy** – one agent decides on commands to send, and another one executes them. Another key difference is that the first agent mentioned can be considered as **rational**. One of the definitions for rational agents based on [1] can be stated in the following manner. For each possible sequence of sensory input, a rational agent should select an action that is expected to maximize desirability of the resulting situation of the environment for the agent (design objective), given the evidence provided by sensory input and built-in knowledge agent has. A rational agent autonomously acting according to its best interest in every situation can be to some extent considered **intelligent**. Environment properties play a major role in the design of intelligent agent. As was shown in section {link} designing a rational agent to play StarCraft still imposes significant challenge. Current agents are not flexible enough in every situation. They very often lack **reactivity** - the ability to respond in timely fashion to changes that occurred.

### Types of agents

According to [1], there are four basic types of agent models embody the principles of intelligent agents:

* **Simple reflex agents** are also the simple kind of agents. Any action is taken only based on current sensory inputs. The action picking is based on if-then rules. A good example of such agents in StarCraft is a unit controllers programmed only in reactive fashion.
* **Model-based agents** keep some internal state of affairs of the environment that depends on the history of their sensory inputs using a model. The model represents knowledge how the world works. The action is then selected not just according to current sensory inputs, but state plays an important role too. Many agents in StarCraft use this approach as they are employing variants of finite state machines.
* **Goal-based agents** are based on the idea that knowing about the present situation of the environment may not be enough. Agents need goals to describe situations which are desirable. They are combining way how model-based agents choose actions with emphasis on actions which may lead to a goal. An example of AI agent in StarCraft following this principles is in [3].
* **Utility-based agents** try to solve the problem of the goal-based agents that defining goals may still not be enough in complex environments like StarCraft. So, they work with utility as a measure of desirability (preferences) for particular states of the environment. The architecture of this agent is on figure {link}. This type is also in the vast interest of ours as Inverse Reinforcement Learning described in chapter {link} is a good example how to get a policy for such an agent. In this work, our main goal is to bring utility-based decision even to higher level of decision making (most utility-based decisions are currently restricted only to reactive control)

{figure of utility-based agent architecture}

## Description of Multiagent Systems

As was mentioned in previous section agent is in an environment. It is common that many agents share the same environment and there is a subset of agents where each agent must interact with agent different from itself. We refer to those kinds of environments with interconnected agents as Multiagent systems (MAS). Due to properties of agents, the system is highly distributed in nature, and one can perceive MAS as a form of distributed artificial intelligence (DAI). Solving problems using MAS is according to [2] best approach in situations where the multi-scale problem with following characteristics exists:

* The problem has many subproblems. Some of the subproblems can be (geographically) distributed and are heterogeneous.
* It has large content as it has a broad scope and covers a major part of a significant domain. Therefore there are many concepts to work with. Concepts in many cases work with huge amounts of data.
* The topology of the subproblems is dynamic, and content may change rapidly. Maintaining consistent information is hard.

The reasoning for using MAS in these situations is to let agents cooperate in solving the problems which are impossible to be solved by them individually using centralized approaches. Distribution of computation can allow solving situations which are otherwise impossible to solve or bring novel solutions. MAS forces system developers to implement the system in a modular fashion, to represent multiple viewpoints and knowledge of experts, therefore resulting system can be expected to be more fault tolerant and reusable. On top of that unpredictable interactions between agents make developers more likely to use declarative approach when defining agents.

From now on when the term MAS is referred to, it means a MAS where agents cooperate to reach common goals. For agents to be able to cooperate, they need to be able to communicate. Communication is necessary to achieve common goals as some form of coordination needs to take place between agents. Cooperation puts another requirement on most (rational) agents in those systems; they need to have the **social ability** – be able to communicate with others.

At the end of {chapter link} we mentioned two tools for decomposition – **Abstraction** and **Divide-and-conquer**, MAS enables both of them. Using different components in the form of agents can be seen as a natural way for Divide-and-conquer approach and structure of the system – relation of the agents is a kind of abstraction. For example in StarCraft domain, one can see a unit as a vivid candidate on being an agent. So the problem of unit behavior is decomposed to smaller subproblems which can be than influence by other agents in higher hierarchy – commander of some kind who issues commands to the unit. For example telling unit by agent higher in command is the abstraction. Another advantage of MAS is a loose definition of an agent which can be decomposed as well. All of the properties makes from MAS great tool for decomposition and way how to integrate many techniques to the system.

# Bibliografie

|  |  |
| --- | --- |
| [1] | RUSSELL, Stuart, Peter. NORVIG a Ernest. DAVIS. *Artificial intelligence: a modern approach*. 3rd ed. Upper Saddle River: Prentice Hall, 2010. ISBN 978-0136042594. |
| [2] | EDITED BY GERHARD WEISS., . *Multiagent systems: a modern approach to distributed artificial intelligence*. Reprint. Cambridge, Massachusetts: The MIT Press, 2001. ISBN 978-026-2731-317. |
| [3] | BEN G. WEBER., . *Integrating learning in a multi-scale agent* [online]. 2012. Santa Cruz: University of California, 2012 [cit. 2017-01-11]. ISBN 14-776-1473-7. Dostupné z: http://alumni.soe.ucsc.edu/~bweber/bweber-dissertation.pdf |