

2 Foundations

In this chapter, we briefly explain the main theoretical foundations of this work. Section 2.1 introduces the concept of agents, followed by a classification of different multi-agent systems in Section 2.2. Section 2.3 illustrates the main teamwork theories in the literature. Finally, Section 2.4 gives a short overview on constraint programming.

2.1 Agents

The concept of agents has been used in various settings. Agents were explored by researchers as a software technology entity (e.g., [115]), as means to simulate the behaviour of ecosystems (e.g., [61]), and as a concept of artificial intelligence, differentiating between an environment and an interior.

“An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.”
– Russell and Norvig [139, p. 32]

This is probably the most generic agent definition in the literature. An agent typically implements a control loop as depicted in Figure 2.1, although the names and numbers of internal steps vary between models.

In the first step, sensory input is processed, the result is used in a reasoning step, which yields a decision for actions. Executing this action will in turn modify the environment, which then leads to different input. There are various ways to classify agents, depending on how these steps are implemented, what kind of internal model is used and with what kind of environment the agent can deal with. An excellent overview is given by Russell and Norvig [139]. Here, we limit the discussion to three different agent models, namely rational agents, BDI-agents, and reasoning agents.

Rational agents try to maximise their expected performance measure. That is, they behave in a decision-theoretically optimal sense, provided appropriate information about possible rewards and probabilities is available.

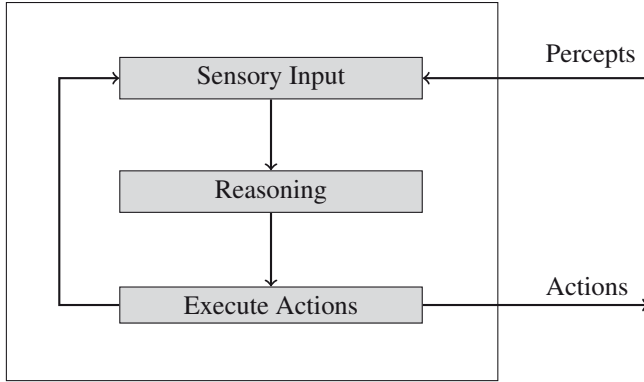


Figure 2.1: Agent Execution Loop

BDI-agents follow the BDI model by Bratman [12]. BDI concentrates on a practical way to perform reasoning, imbued with particular mental attitudes, namely: *Beliefs*, *Desires*, and *Intentions*. Beliefs represent the informational state of the agent about the world (including itself and other agents). The desires of an agent represent objectives or situations that the agent would like to accomplish or bring about. Intentions represent the deliberative state of the agent: what the agent has chosen to do. This practical approach was highly successful and led to a multitude of languages based on the BDI model.

Reasoning agents follow a very classical view of AI, where reasoning is mapped onto deductions in some formal logic. Research in this area focuses on how knowledge can be represented to allow for powerful reasoning techniques to be used. Foundational work in this area dates back to McCarthy and Hayes [100]. Two of the most influential calculi for reasoning agents are the situation calculus [134] and the fluent calculus [72, 166].

Our work is compatible with all of these agent models, although ALICA is strongly inspired by the BDI model, and can be seen as a BDI language. In the following, we use the words agent and robot interchangeably.

2.2 Multi-Agent Systems

Based on the notion of agents, systems comprising multiple agents were, and still are, subject to intensive research. Such systems can be classified in various ways, e.g., based on the abilities of the individuals, the organisational metaphors used, or whether the agents can be considered homogeneous or heterogeneous. An excellent overview over the field is given by Wooldridge [181]. One of the most important aspects for this thesis is the level of cooperation. A system can be classified as:

Cooperative Meaning that all agents try to achieve the same global goal. They will therefore cooperate in any way that is deemed beneficial to this goal.

Collaborative Agents do not necessarily try to achieve the same goal, but goals that can be compatible. They will therefore form coalitions whenever there individual goals align. Compared to cooperative systems, the organisational structure is much more dynamic, since teams can form and disband continuously within the system.

Neutral Agents have different goals, and ignore each other. Any form of cooperation happens purely by chance.

Antagonistic Agents have goals that are in direct competition with each other, that is one agent obtaining its goal entails that no other agent is able to do so.

Naturally, these categories are not strict and actual systems can feature elements of each category. In robotic soccer for instance, each team forms a purely cooperative system and each team player is completely altruistic. However, the multi-agent system that is formed by the two opposing teams during a match is, of course, antagonistic.

In our work, we only consider purely cooperative systems. While our main scenario contains an opponent team, we are not interested in analysing the overall behaviour of soccer playing robots, but instead want to model the behaviour of a single team. In other words, we treat the opponent team as part of the environment, since we cannot control their actions.

Multi-agent systems which feature a large number of typically homogeneous robots with fairly simple individual cognitive and communicative abilities are investigated by the fields of swarm intelligence and swarm robotics. Swarm-based approaches are often inspired by biological systems, e.g., [119]. Modelling a set of robots as a team on the other hand assumes that the individual is already capable

of relatively complex reasoning. In Section 2.3 we will discuss the main theories relevant to cooperative teams.

Besides swarms and teams, there are plenty of different possibilities to organise multi-agent systems. Horling and Lesser [73] give a thorough overview of the different paradigms investigated in the literature:

Hierarchy In a hierarchy, agents at higher levels have a more global view than agents at lower levels. Typically, agents only communicate with those directly connected to them through the tree-like structure. Data is communicated upwards, while commands are passed downwards. While the hierarchical structure is appealing in its simplicity and its relationship with hierarchical goal or task trees, it is rather rigid, cannot adapt well, and can feature single points of failure.

Holarchy Holarchies are nature-inspired structures, similar to hierarchies, where a group is formed at each level out of a certain kind of parts and these parts again are groups of parts at a lower level of abstraction. At the lowest level, a group consists of agents. Thus, holarchies can be seen as less strict hierarchies, allowing more communication between individuals and more autonomy.

Coalition Coalitions follow the idea of collaborative agents noted above. Here, agents form relatively short-lived, flat groups in order to achieve aligned goals.

Team Teams consist of cooperative agents that work together to achieve a common goal. In contrast to coalitions, they form a purely cooperative system.

Congregation Congregations are long-lived flat hierarchies of agents. In contrast to teams, they do not have a single goal, but form in order to combine complementing capabilities.

Society Agent societies are long-lived open systems, such as an electronic market. Agents pursue different goals, have heterogeneous capabilities, and interact with each other through various channels. The society imposes a set of constraints individuals must adhere to, commonly named social laws or norms.

Federation In agent federations, potentially complex groups of agents are each represented by a single distinguished member of the group, which is in charge of communicating and interacting with the representatives of other groups.

Market In contrast to societies, the whole interaction process in markets is modelled after commerce, i.e., they buying and selling of goods, placing bids, and offering services. The situation is typically competitive; that is, the individual goals conflict with each other.

Matrix Matrix-based agent organisations allow the definition of multiple dimensions of authority. The individual agent has to be equipped with a sufficient amount of autonomy in order to deal with the potential local conflicts than can arise from dealing with different authorities.

Compound The last paradigm basically combines different organisational structures for different purposes, such as data-flow, control, discovery, etc.

Since we are most interested in small scale groups of robots following a common goal, the approach presented in this work follows the characteristics of a team-based multi-agent system.

2.3 Teamwork

Teamwork among intelligent agents has been extensively analysed in the literature. Today, most approaches to modelling teamwork in agent languages are based on at least one of the two following prominent theories:

Joint Intentions Theory The Joint Intentions Framework [28, 97] is a theoretical framework founded on BDI logics. The framework focuses on a team's joint mental state, called a **joint intention**. A team jointly intends a team action if all team members are jointly committed to perform an action while in a specified mental state.

In order to enter a joint commitment, the team members have to establish appropriate mutual beliefs and individual commitments. Although the Joint Intentions Theory does not mandate communication and several techniques are available to establish mutual beliefs about actions from observations (see for example [75]), currently communication seems to be the only feasible way to attain joint commitments. A very interesting key aspect of the Joint Intention Theory is the commitment to attain mutual belief about the termination of a team action. This helps to ensure that the team stays updated about the status of the team actions. This behaviour is achieved by enforcing that agents committing to a joint intention also commit to inform their team about any relevant failures or premature terminations. Joint intentions and joint commitments provide a basic framework to reason about

coordination required for teamwork as well as guidance for monitoring and maintaining team activities. However, a single joint intention for a high-level team goal is not sufficient to model team behaviour in detail and to ensure coherent teamwork.

Shared Plans Theory In contrast to Joint Intentions, the Shared Plans Theory [63, 64] employs hierarchical structures over intentions, thus overcoming the shortcoming of a single Joint Intention for complex team tasks. The Shared Plans Theory is not based on a joint mental attitude but on an intentional attitude called **intending that**, which is very similar to an agent's normal intention to perform an action. However, an individual agent's 'intention that' is directed towards its collaborator's action or towards a group's joint action. 'Intention that' is defined via a set of axioms that guide an individual to take actions (including the communication), that enable or facilitate its team-mates, sub-team, or team to perform assigned tasks.

A SharedPlan for a group action specifies beliefs about how to do an action and sub-actions [63, 64]. The formal model captures intentions and commitments toward the performance of individual and group actions. A collaborative plan is composed of a **mutual belief**, of a (partial) recipe, individual **intentions to** perform the actions, individual **intentions that** collaborators succeed in their sub-actions and individual or collaborative plans for sub-actions. With the concept of actions and sub-actions the Shared Plans Theory describes a hierarchy of plans to reach a common goal. This is also the main difference between the Joint Intentions Theory and the Shared Plans Theory; the Shared Plans Theory describes the way to achieve a common goal whereas the Joint Intentions Theory describes only this common goal. However, the lack of principles like joint intentions and joint commitments results in limited possibilities to reason about team coordination and team activities.

The two theories Joint Intentions and SharedPlans have been extensively used to examine and describe teamwork. In our approach, we will draw from both, although the relationship to SharedPlans is most apparent due to the structural similarities between collaborative plans and ALICA plans in execution.

2.4 Constraint Programming

The paradigm of constraint programming advocates a declarative description of problems, which are then solved by appropriately chosen constraint solvers. A recent overview is given in Rossi et al. [138]. A very general framework has been

proposed by Frühwirth [49, 50], which defines *Constraint Handling Rules* (CHRs) as a unified way to express constraints. Slightly simplified, a constraint handling rule is of the form $\text{Head} \Leftarrow \text{Guard} \mid \text{Body}$, meaning, that if some of the constraints currently imposed unify with Head, and the Guard evaluates to true given the unification without modifying variables in the Head, the Head is replaced by Body. This system has been integrated into the constraint programming framework ECL^{PS^e} [2]. CHRs can be used to propagate and simplify constraints, however, they need to be combined with a search method in order to identify solutions to the constraint satisfaction problem encoded. Such a search typically makes assumptions about the variables by posting further constraints and backtracks once an assumption leads to a detectably unsatisfiable constraint. Backtracking can be extended to backjumping, where multiple steps are retracted at once in order to search a more promising region of the search space earlier [124, 26].

Constraint satisfaction in a Boolean domain has been shown to be NP-complete [29, 143]. Later, the same could be shown for general constraint satisfaction problems over finite domains [43]. While some tractable subclasses were identified [27], these hardly fit the requirements of robot control, especially since many different problems need to be formulated.

The class of problems spanning over continuous domains is even more interesting from the perspective of robotics research, as many values that appear in robotic domains are real-valued, such as positions or angular states of joints. In general, the problem of solving constraint-based mathematical models over the real numbers is undecidable [135]. However, solutions can be approximated. Finding intervals that approximate solutions of reasonable precision is NP-hard [10]. The most general case of a constraint satisfaction problem corresponds to a first order formula for whose free variables a solution is to be found, and this problem is undecidable, due to satisfiability being undecidable in first order logic in general.

In order to capture more complex problems, Jónsson and Frank [80] introduced dynamic constraint problems, where individual constraint problems are linked in a sequence. Neighbouring problems can be obtained from each other by restriction or relaxation. They propose a reasoning technique based on procedures to solve such systems. Thereby, the resulting system can solve problems efficiently, provided corresponding procedures are given. In this way, they relax the requirement that all variables need to be known before solving a system, and can consider problems where the number of variables are unknown in advance. This way, unbounded planning tasks become representable as constraint satisfaction problems. Similarly, Nareyek [105] represented planning as a constraint-based local search in a space of graphs, where each graph represents a possible plan.

More recently, distributed constraint optimisation problems have been used to control the behaviour of MAS systems. This field is described in detail by Petcu [121]. In this setting, each agent owns and controls a constraint optimisation problem, which relates to the problems possessed by other agents through some shared variables. The agents obtain a global solution by interleaved local solving and message exchange. Typically, no agent has a global view on the problem.

In summary, constraint programming constitutes a very concise and mathematically well-founded way to express problems. This is the main motivation to integrate constraint programming techniques into our solution. However, the problem classes that occur in the domains we consider together with the inherent dynamics of these domains have not yet been tackled under soft real-time considerations from a constraint programming point of view.

Modelling and Controlling of Behaviour for Autonomous
Mobile Robots

Skubch, H.

2013, XVII, 259 p. 46 illus., 16 illus. in color., Softcover

ISBN: 978-3-658-00810-9