# Flexible Autonomy and Context in Human-Agent Collectives

Daniela Dybalova

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy
June, 2017

*ubi lex, ibi poena*

# Abstract

Human-agent collectives (HACs) are collaborative relationships between humans and software agents that are formed to meet the individual and collective goals of their members. In general, different members of a HAC should have differing degrees of autonomy in determining how a goal is to be achieved, and the degree of autonomy that should be enjoyed by each member of the collective varies with context. This thesis explores how norms can be used to achieve context sensitive flexible autonomy in HACs. Norms can be viewed as defining standards of ideal behaviour. In the form of rules and codes, they are widely used to coordinate and regulate activity in human organisations, and more recently they have also been proposed as a coordination mechanism for multi-agent systems (MAS). Norms therefore have the potential to form a common framework for coordination and control in HACs. The thesis develops a novel framework in which group and individual norms are used to specify both the goal to be achieved by a HAC and the degree of autonomy of the HAC and/or of its members in achieving a goal. The framework allows members of a collective to create norms specifying how a goal should (or should not) be achieved, together with sanctions for non-compliance. These norms form part of the decision making context of both the humans and agents in the collective. A prototype implementation of the framework was evaluated using the Colored Trails test-bed in a scenario involving mixed human-agent teams. The experiments confirmed that norms can be used for coordination of HACs and to facilitate context related flexible autonomy.

# Acknowledgements

My first and greatest debt of gratitude must go to my advisors, Brian Logan and Tom Rodden. Who patiently guided me through the challenges and difficulties that I faced. In particular Brian's way of thinking he kept my ideas on track, and his enthusiasm and feedback were a big part of making it through.

I would like thank Wenchao Jiang for making the code of the GeoSense game available and for assistance in developing the gameserver middleware. Special credit goes to Bas Testerink for this help with the integration of 2OPL.

This work was funded by EPSRC grant EP/I011587/1.

# Contents

Appendices

# List of Figures

# List of Tables

# 1

# Introduction

Human-agent collectives (HACs) are collaborative relationships between humans and software agents that are formed to meet the individual and collective goals of their members [Jennings et al., 2014]. Norms can be viewed as defining standards of ideal behaviour. In the form of rules and codes, they are widely used to coordinate and regulate activity in human organisations, and more recently they have also been proposed as a coordination mechanism for multi-agent systems (MAS). Norms therefore have the potential to form a common framework for coordination and control in HACs. The thesis develops a novel framework in which group and individual norms are used to specify both the goal to be achieved by a HAC and the degree of autonomy of the HAC and/or of its members in achieving a goal.

## 1.1 Problem Definition and Objectives

The main question this thesis aims to answer is whether norms can be used to achieve context sensitive flexible autonomy in HAC? We believe that different members of HACs should have different degrees of autonomy depending on the context. One way to realize flexible autonomy in multi-agent systems (MAS) is with norms.

> *"Can norms be used to achieve context sensitive flexible autonomy in HAC?"*

We are looking into new ways of how to achieve flexible autonomy in the coordination of HAC. Coordination is an art of managing interdependencies among activities. Agents need to coordinate their actions where there are dependencies between agents' actions; there are needs to meet collective constraints and the entire problem cannot be solved by an agent alone.

This work is inspired by the ORCHID project [1] [Jennings et al., 2014], which seeks to establish the science of HAC. Specifically, the aim of the ORCHID project is to explore the formation of collaborative relationships between human and software agents. With the increase in use of the computing technology in the world around us, the vision of people and computational agents operating together at a large scale needs to be explored. Due to the scope of the intended application areas it is desirable that human and agent system can cooperate in a flexible manner. To achieve this vision various research questions need to be addressed and we will see how flexible autonomy is one of the main ones.

One of the application areas used to explore these issues is a disaster response situation. In a disaster response emergency services are required to make critical decisions in conditions that are highly uncertain and swiftly changing. To make the response most effective a system in which human and agent roles interleave in a flexible way is required. Specifically, there is a need for new methodologies that prescribe how to handle continuously changing flexible autonomy in human-agent collectives. Due to obvious difficulties with a realistic simulation of a disaster

[1] http://www.orchid.ac.uk

response scenario, such as the scale and destructive character, new ways of exploring these questions must be found. One such a way to design and investigate human behaviour and the formation of HAC is by the use of Mixed Reality Games (MRGs) [Flintham et al., 2003; Benford et al., 2006]. However, another way is to explore new ways of how HAC can interact might by employing existing human-agent interaction research test beds like the Colored Trails [Gal et al., 2005] which used computer based game like scenario.

## 1.2 Approach

We employ a multi-agent system to study the interaction in HAC. Multi-agent systems (MAS) are used to study various aspects of agent interaction in a given environment. There is no universally accepted definition of the term agent. One of the definitions is given by Wooldridge and 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 delegated objectives." A crucial part of agency is the notion of autonomy. Unfortunately, autonomy is again a word of many different meanings. One of the ways to support autonomy of agents is with the use of norms. In this thesis we define autonomy as: "freedom to select a course of action." We adopt this definition to implement flexible autonomy in HACs. We develop the normHACing framework, in which software and human agents work together to achieve a joint objective where the autonomy of each actor varies according to a context. We define flexible autonomy as autonomy, but where the degree of which varies according to context. By context we mean any relevant features of the environment. We give the group an autonomy to decide how a joint goal is achieved. Norm-aware agents are able to reason about norms that are in a form of obligations and prohibitions with assigned fixed numerical priorities and corresponding sanctions. Such a coordination mechanism of multi-agent systems allows the autonomy of agents to be dynamically adjusted with the use of norms created by the organization. Since the autonomy is seen in this system as an ability of an agent to select a course of action in the current context, the norms need to contain information about the state of environment to which they apply. Group norm-aware agents are able to reason about group norms and work together to achieve the joint goal. This thesis describes a system in which humans and agents may be regulated not only with norms aimed at them individually but also norms that are issued to them as a mixed human-agent team.

## 1.3 Contribution

The contribution of this thesis to computer science is as follows: understanding the use of norms in HAC, introducing group norms as a team coordination mechanism, evaluation of the concept in real settings, implementation of N-2APL and building a framework for norm-aware agents.

1. **Norms in HAC** We show how norms may be used in HAC to achieve flexible autonomy. Norms are widely used to regulate multi-agent systems. Human society is accustomed to being governed with norms in the form of laws, rules or guidelines. However, to the best of our knowledge norms have not previously been used to coordinate human-agent collectives.

2. **Hierarchical group norms** We have designed a system of hierarchical group norms, which may be used for coordination in mixed human-agent teams. Group norms are addressed to a group of agents. The team receives a group obligation, which is then split into individual obligations, the fulfilment of which leads to the fulfilment of the parent group norm and therefore the achievement of the group team goal. These norms may be stacked hierarchically — a group norm can split into a combination of individual and group norms.

3. **Proof of concept evaluation** The application of the framework was illustrated on two examples GeoSense and Colored Trail. Colored Trails was then used in the evaluation. It was experimentally demonstrated that norms may be used as a coordination mechanism in human-agent collectives and to facilitate flexible autonomy in human-agent collectives.

4. **Implementation of framework for norm-aware agents** We implemented N-2APL and developed a framework for programming

norm-aware multi-agent systems which integrates the N-2APL norm-aware agent programming language with the 2OPL language for programming normative organisations. To the best of our knowledge, this is the first implementation of an integrated framework for norm-aware multi-agent systems in which autonomous agents deliberate about whether to conform to the norms imposed by a normative organisation.

## 1.4 Structure of the Thesis

This introduction is followed by a literature review in Chapter 2, where we discuss the state of the art in all relevant topics: Agents' autonomy, flexible autonomy, norms and human-agent interaction.

In Chapter 3 we describe the development of normHACing framework. We show how a norm is represented and follow with the design of norm-aware agents: agents that are able to deliberate about norms. The framework consist of normative MAS implemented in N-2APL, normative organisation 2OPL and a tuple space used as a coordination mechanism. We illustrate the use of the framework by connecting it to a MRG game GeoSense.

In Chapter 4 we design a system of hierarchical group norms. We first explain the problem of translating the abstract group into a concrete form and introduce a taxonomy of group norms. Then we explain how we achieve flexible autonomy in HACs with group norms.

In Chapter 5 we describe a development of an extension of normHACing framework that implements a prototype of hierarchical group norms. The framework consists on groupnorm-aware agents GN-2APL and normative organisation G-2OPL, which is an extension of 2OPL.

In Chapter 6 we describe the evaluation of the framework. We designed an experiment with a research test bed Colored Trails where teams of human and agents players were coordinated with group norms.

In Chapter 7 we conclude this thesis with a summary of contributions and identify future work.

# 2

# Literature Review

In this chapter we present the related literature to give background about the theory of autonomous agents (Section 2.1), their autonomy (Section 2.1.1) and the problem of flexible autonomy (Section 2.1.2). We follow with the overview of the state of the art of human-agent interaction (Section 2.2). We present norms as a way to regulate a multi-agent system (Section 2.3), we describe the normative multi-agent systems (Section 2.3.1) and continue with an overview of existing normative programming frameworks (Section 2.3.2). In Section 2.4 we present the current state of the art of norm-aware agents. In Section 2.5 we look into the background of the theory of group norms.

## 2.1 Agents

The notion of an agent is central to the field of Artificial Intelligence. Various authors [Franklin and Graesser, 1997] offer a range of agent definitions. Franklin and Graesser [1997] propose a formal definition of an autonomous agent with the aim to clearly distinguish an agent from just any program.

> "An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future."

The word autonomy is derived from a combination of Greek terms signifying self-government where auto stands for self and nomos for law. It is used either to denote self-sufficiency as a capability of an entity to take care of itself or self-directedness as a freedom from outside control. Despite the fact that autonomy is a central notion in most agent's definitions and it is by itself a subject of research in the multi-agent field, there is not yet a commonly agreed definition of it.

### 2.1.1 Autonomy

Castelfranchi and Falcone [2003] define autonomy as a relational notion, which is derived from relationships among three classes of entities. An entity, the main subject, whose autonomy is considered. A function, action or goal on which the autonomy is evaluated. And a secondary entity, or a plurality of subjects, that are considered autonomous with respect to the main entity in regard of the particular function / action / goal. Given this complex relational nature authors Castelfranchi and Falcone [2003] conclude that the dimensions of an agent's autonomy derive from its architecture and from the theory of action.

Bradshaw et al. [2004] believes that there are two basic dimensions of autonomy:

- a *descriptive* dimension is seen as self-sufficiency and describes actions an agent is is capable of performing
- a *prescriptive* dimension describes actions an agent is allowed to perform in a given context

Barber and Martin [1999] link the agent's autonomy to its ability to influence the decision-making process for a given problem. In their view, an agent acting alone has a complete autonomy because it holds all the decision-making power. Similarly, an agent making all decisions for other agents, as well as itself, has complete autonomy and power over itself and its subjects. On the other hand an agent that shares decision-making with others is in a consensus relation with them, and thus its autonomy is limited in proportion to the number of agents involved in making those decisions. Finally, an agent that has no involvement in the decision-making process consequently has no autonomy and is command driven. Different dimensions of autonomy are suggested by Braynov and Hexmoor [2003] such as simple autonomy from the user. Autonomy with the respect to the environment, which changes in a response to the degree of predictability of the environment and a group autonomy, which refers to how free is the agent from the interference by other agents.

According to Wooldridge and Jennings [1995] the term autonomy means that agents have a control over both their internal state and their behaviour. The agent determines its beliefs and it decides by itself upon its actions. Beavers and Hexmoor [2004] put emphasis on the degree of autonomy as a relative measure of the independence between an agent and the physical environment, and within and among social groups. d'Inverno and Luck [2012]; Luck et al. [2003] are unsatisfied with defining autonomy as a wholly relative concept and according to them the defining characteristic of autonomy should be the self-generation of goals, thus allowing it to be regarded in absolute terms that more clearly reflect the priority of the aspect of self-sufficiency. They argue that autonomous systems must be motivated. The motivations can be of different types and strengths and also vary according to the internal state of the agents. The adoption of a motivated goal was also used in an extension of AgentSpeak(L) [Meneguzzi, 2008].

In an agent architecture oriented view Dastani et al. [2004] believe that the deliberation cycle of an agent determines the autonomy of an agent as well. Autonomy levels can be looked at as an agent's commitment to its own decisions. For example, in one deliberation an agent commits to and proceeds towards its goal until it has been reached, whereas in another cycle the agent might reconsider its goals due to retrieval of new information. The methods used in the deliberation cycle, as well as their actual implementation, are relevant for the agent autonomy. Therefore different levels of autonomy can be achieved by changing of the deliberation cycle.

Carabelea et al. [2004] attempts to classify different types of autonomy using the *Vowels* approach. In this way they have identified five forms of autonomy. The first *A-Autonomy* (Self-Autonomy) is implicitly present in all agent architectures and can be interpreted as a property that allows an agent to choose between several possible behaviours. Without this type of autonomy the agent would not be able to have any other more specific types of autonomy described below. *E-Autonomy* (Environment-Autonomy) refers to the agents' relation to the environment. Similarly as the self-autonomy described above this form of autonomy is present in all agent architectures. In the work of Falcone [2001] these types of autonomy are called *Realization* and *Meta-level autonomy*. It is possible to say if an agent is either strongly or weakly autonomous based on how strongly is the agent influenced by the environment. Other forms of autonomy [Carabelea et al., 2004] identified are user, social and organization (norm).

Social or also called *interaction* autonomy is believed to be one of the most important aspects of the interactions between the agents in a multi-agent system. The social autonomy is in most of the cases related to the adoption of goals. However the object of social autonomy can vary. Castelfranchi and Falcone [2003] believes that user and norm related forms of autonomy are special cases of social autonomy. They identify two different notions of social autonomy as *independence* (self-sufficiency) and *collaboration*. According to Castelfranchi and Falcone autonomy in collaboration relates to how much an agent is autonomous when working for another agent. The autonomy as independence is then defined as a state when:

> "An agent is completely autonomous (relatively to a given goal or action) when does not need the help or the resources of other agents to achieve its goal or to execute its action."

While Castelfranchi and Falcone [2003] think that all forms of social autonomy should be defined in terms of different forms of social independence, Beavers and Hexmoor [2004] provide arguments for the need to distinguish between being autonomous and acting autonomous. Both internal and external conditions affect autonomy therefore autonomy is more than independence.

A utilization of organizational structures is commonly seen as a way to restrain the autonomy of agents. The agents are able to recognize the organizational contracts - norms, and reason about them. An agent is then norm autonomous if it can violate a norm. Castelfranchi and Falcone [2003] describe this as deontic autonomy, which corresponds to the range of permissions and obligations that regulate the agent's choice amongst possible actions. Norms in a form of obligations and permissions are also seen in Bradshaw et al. [2004] model. Permitted actions specify actions that an agent is allowed to perform and obligated actions then are actions that an agent is required to perform. Meneguzzi [2008] utilizes norms to prevent undesirable behaviour of an agent. The topic of norms is covered in Section 2.3.

A special case of social autonomy is the agent's autonomy with respect to a user. The most common definition is that an agent is user autonomous when choosing what action to perform if it can make the choice without user's intervention [Bradshaw et al., 2012].

Autonomy is a central aspect of agent's entity yet there are many different perspectives and definitions. In our work we focus on an agent's deliberation cycle and its commitment to goals. Therefore in this thesis we define agent's autonomy as:

> "Freedom to select a course of actions".

### 2.1.2 Flexible Autonomy

The term flexible autonomy, also called adjustable autonomy in literature, relates to a setting where the level of autonomy of an agent varies depending on the situation. One of the main problems of this area is determining when to adjust the autonomy. Multi-agent systems with agent-based flexible autonomy are the ones in which agents are provided with the ability to reason about adjusting their autonomy level depending on the situation [Wooldridge and Jennings, 1995]. Different research directions provide different motivations why there is need for flexible autonomy. Most often the need arises when agents are interacting with humans and their interdependence [Johnson et al., 2011]. The autonomy adjustment is then driven either from the agent or the user [Ball and Callaghan, 2012]. In the area of human-agent interaction the autonomy may be restricted for both parts of the system - human or user. The topic of Human-agent interaction is covered in Section 2.2.

Reed [2006] tries to identify the key problem, which needs to e addressed when designing flexible autonomy. According to the author the process of changing the autonomy can be broken into three phases. It begins with the collection of the information relevant to the decision making. It is followed by reasoning about what autonomy changes can or should be made. The cycle is then complete with a realization of the decisions made.

In the research towards establishment of flexible autonomy [Parasuraman et al., 2000] first proposed a classification for operational autonomy based on a ten-level scale. This model remains rather abstract as it fails to take into account neither the environment complexity or the context. However, it provides an interesting insight into the interactions between the operator and the agent entities. The model has later been extended applying the original ten-level scale to a four-stage cognitive information processing model of perception, analysis, decision-making and action. In the work of Ball and Callaghan [2012] the high level forms of operation can be altered by sensing, decision making or acting. While Bradshaw et al. [2004] describe a general method for adjusting the autonomy of agents that operates by:

- Adjusting permissions: allowing and disallowing certain actions in the environment.
- Changing obligations: assigning and withholding tasks to and from the agent.
- Restricting possible actions: restricting resources to the agent and adjusting the capabilities of the agent thus changing the functionality of the agent.

Other researchers like Falcone [2001] believe there is a need for adjustable autonomy because it allows the gradual adjustment of the dependent agent's autonomy as it becomes more competent and because it also allows the dependant to change its own autonomy on the basis of its needs. Falcone describes this notion of flexible autonomy as bilateral since both sides can adjust the autonomy in both ways and therefore the process is bidirectional. The main causes for autonomy adjustments are: meta-autonomy adjustment (dependant's entitlement changes at the meta-level); adjustment of realization autonomy (a new task differs in the level of details); control-dependent autonomy adjustment; interaction-dependent autonomy adjustment (which adjust to change of in the strength of the commitment).

Electric Elves (E-Elves) introduced by Tambe et al. [2002] use a notion of a transfer of control strategy. The important aspect of the transfer of control strategies is the ability for the agent to change team coordination in order to buy more time for a decision to be made. Transfer of control strategies are operationalized via MDPs, which creates a policy for the agent to follow. Cheng and Cohen [2005] then extend this work with a domain-independent decision-theoretic adjustable autonomy model that enables an agent to reason about the trade-offs between three different levels of autonomy:

- Full autonomy, where the agent just decides by itself.
- No autonomy, where the agent transfers decision-making control to another entity.
- Partial autonomy, where the agent queries another entity for information that determines how the decision should be made.

They also allow agents to initiate interaction in order to determine the best transfers of decision-making control.

One major challenge in the research of flexible autonomy is to recognize the right moment when an alteration should happen. Cohen and Cheng [2005] discuss the value of proxy agents in facilitating the exchange of information about the level of nuisance incurred by users, towards the selection of more effective interaction strategies by agents. This provides a more productive mechanism for agents to solicit user feedback as part of their processing. In another work Scerri et al. [2004] identify three phenomena showing poor coordination that should be brought to the attention of humans - unfilled task allocation, un-tasked team members and unusual plan performance characteristic.

According to Dorais et al. [1999] the system's level of autonomy can be categorized based on how complex the commands it executes are, how many of its sub-systems are being autonomously controlled, under what circumstances will the system override manual control, the duration of autonomous operation. Similarly to Mostafa et al. [2015] who introduce Layered Adjustable Autonomy (LAA) model that limits agents' decision making process with the set of intervention rules. Also in Barber et al. [2000] work the agents are dynamically adapting their decision-making framework. Authors experimented with three discrete autonomy categories of autonomy spectrum. In *Command-Driven* form the agent does not make decisions and must obey orders given by a master agent. On the opposite side of the spectrum is a *Consensus* category where an agent works as a team member and is sharing decision-making equally with other agents. A hybrid form is then *Locally Autonomous/Master* where the agent makes decisions alone and may or may not give orders to other agents.

Falcone and Castelfranchi [2000] focus attention to the issue of delegation and identify three main dimensions as interaction-based, specification-based and delegation of the control. The first dimension relates to the various levels at which the behaviour of the delegated agent is influenced by the delegating agent in the delegation relationship. The second dimension relates to the various levels at which the delegated task is specified in the delegation relationship and delegation of the control, and the various possibilities of its realization. In the work of Mercier et al. [2008] the resources represent the ground on which authority sharing considerations can be built on. Conflicts can be detected and classified depending on the entities that disrupt the current plan. Amongst the resources agent's and operator's have are key items to build a consistent plan to reach the goal and also to determine consequences of unexpected events as a change in the environment or an operator's intervention.

Vecht et al. [2007a,b] implemented a dynamic coordination mechanism by providing the actors with adjustable autonomy. An agent's level of autonomy depends on the influence of others on the reasoning process. The actors have reasoning rules that control the external influences they experience. This way they have defined situations at the individual level in which the actor can change its autonomy level.

Various approaches to tackle the problem were taken also in the field of robotics. Adjustable autonomy in human–machine cooperation described by Zieba et al. [2009] are considered as the ground for resilience of the system. There are three indicators to assess different meanings of resilience of the system: foresight and avoidance of events, reaction to events and recovery from occurrence of events. The third of these metrics takes into consideration the concept of affordances that allows a common representation for the opportunities of action between the automated system and its environment. The efficiency of adjustable autonomy was tested by Valero-Gomez et al. [2011] who experimented with robot teams. They have compared flexible and static operational models and concluded that the operator achieves better results when controlling team of robots with flexible autonomy adjustments. Other areas that the concept of flexible autonomy appears in are mobile tele-operation [Schwarz et al., 2014; Goodrich et al., 2001].

As we see in the description of related work there is not a single definition of flexible autonomy. Sometimes flexible autonomy is viewed in the context group decision making, whereas other work looks at single agent decision making where we can furthermore different perspectives, such as looking at various levels of abstraction or decision making control. The flexible autonomy is used as a mechanism to facilitate human-agent interaction in autonomous systems [Valero-Gomez et al., 2011; Mostafa et al., 2015, 2016]. Such an autonomous system where human and agents can share control benefits from greater reliability [Jennings et al., 2014; Parasuraman et al., 2000; Beer et al., 2014]. In our approach we define flexible autonomy as:

> “Flexible autonomy that varies depending on the context.”

Depending on the current state of the environment we restrict the autonomy of members of human-agent collectives.

## 2.2 Human-Agent Interaction

The key idea of Human-agent interaction (HAI) is to complement the abilities of software agents and humans, where software agents posses stronger computational abilities and humans have better cognitive abilities. HAI [Lewis, 1998; Sycara, 1998; Bradshaw et al., 2012] is a relatively novel research area. Authors take a number of approaches when studying and defining the relationship. Sukthankar et al. [2012] highlights the importance of a notion of symbiosis in human agent interaction. In his view both human and agents are symmetrically important team players and their mutual interaction should strengthen their capabilities. The aim of the automation is not seen as need to replace the human but its symbiosis with the system.Human-agent coordination requires complex design of interactions. Apart from the technical issues some researchers believe that the social perspective of the automation is equally as important as the technical one [Bradshaw et al., 2012].

Mixed-initiative interactions refers to systems with a flexible interaction strategy. The roles of the agents are not predetermined in advance [Allen et al., 1999] but assigned as the problem is being solved. Each agent contributes to the interaction as required [Horvitz, 1999]. Mixed-initiative interaction planning and control has been used in various interfaces and applications [Ferguson et al., 1996; Burstein et al., 2000; Zimmerman et al., 2007; Yang and Lee, 2012; Hardin and Goodrich, 2009]. Kamar et al. [2013] presents a methodology for effective exchange of information between humans and agents. The authors introduce a concept of 'nearly decomposable' decision-making problems and algorithms that computes efficient strategies for enhanced human-agent cooperation.

Another area of HAI are interface agents (assistants) [Lieberman, 1997] that actively assist users operating an interactive interface [Lieberman and Selker, 2003]. Agents assistants behave similarly to personal assistants and can learn the user's habits and suggest appropriate actions or directly guide to the specific interface [Maes et al., 1994]. Such agents were used in several different areas of application like email assistants [Metral and Maes, 1998], a meeting scheduler [Kozierok and Maes, 1993] and smart home control [Costanza et al., 2014]

The ORCHID project [1] investigates the potential of human-agent collectives (HACs) [Jennings et al., 2014] in a disaster response (DR) scenario, where groups of humans and computational or embodied agents collaborate to achieve a common task. This is due to the extreme nature of DR simulation where not only is almost impossible to stage the same conditions, but also difficult to verify experiment result. Fischer et al. [2012a] argue for serious mixed reality games as an approach to study and

[1] http://www.orchid.ac.uk

design for challenging real-word scenarios. They outline an approach and give an example of a serious mixed reality game, which allows the study and analysis of human-agent interaction in a DR scenario on the ground. AtomicOrchid [Fischer et al., 2012b; Jiang et al., 2014; Ramchurn et al., 2014] is a real-time location-based game utilized to explore the coordination and agile teaming under temporal and spatial constraints. The use of software agents in mixed reality environments is not a new approach. Holz et al. [2011] provides a taxonomy of Mixed Reality Agents (MiRAs), which are defined as agents embodied in a Mixed Reality environment. The taxonomy classifies MiRAs along three axes: agency, based on the weak and strong notions outlined by Wooldridge and Jennings [1995]; corporeal presence, which describes the degree of virtual or physical representation body of a MiRA and interactive capacity, which characterises its ability to sense and act on the virtual and physical environment. The need for intelligent agents also arises in orchestration of participatory experiences where human expertise is a scarce and expensive resource [Benford et al., 2006].

Tha principle challenge of the HAI remains managing human-agent decision making. As we see in the literature there are different suggested scenarios [Jennings et al., 2014; Mostafa et al., 2016]. Flexible autonomy in their interaction is seen as one the key features to achieve interdependence The aim of the HAI research is to complement the abilities of human and software counterparts. In some situations, they are likely to achieve better results when acting jointly than separately.

## 2.3 Norms

In this section we will introduce norms as a means to regulate multi-agent systems. Norms are one of the ways of regulating a multi-agent systems (MAS) [Jones and Sergot, 1993; Dignum, 1999; Gelati et al., 2004; Boella and van der Torre, 2008; Luck et al., 2013].

> “Norms prescribe how the agents ought to behave, and specify how they are permitted to behave and what their rights are.”

### 2.3.1 Normative Multi-Agent Systems

Agent-based systems have been increasingly used in the past decades in wide range of scientific fields. Also the study of norms has been one of the most active areas in MAS [Neumann, 2010]. The definition of normative multi-agent systems given by the researchers involved in the NorMAS 2007 workshop is as follows [Boella et al., 2008].

> "A multi-agent system organized by means of mechanisms to represent, communicate, distribute, detect, create, modify, and enforce norms, and mechanisms to deliberate about norms and detect norm violation and fulfilment."

There are two different approaches to the use of norms within multi-agent systems. The first one focuses on the emergence of norms as a property of evolutionary game theory [Savarimuthu et al., 2011]. Researchers have worked on both prescriptive (top-down) and emergent (bottom-up) approaches to norms [Boella et al., 2008]. The second approach is to study the inclusion of norms in the agents' design. This research branch also includes normative system architectures, norm representations, norm adherence and the associated punitive or incentive measures. Deontic logic is used to define and represent norms. Norms as a built-in property of the agent's architecture is still a challenge with the number of conceptual designs far exceeding the number of existing models [Hollander and Wu, 2011].

A novel way of using norms can be seen in [Garbay and Badeig, 2012] where they are used to situate the activity rather than to constrain an action. These traces are then stored as tuples. There are number of approaches how to situate agents in context [Hong et al., 2009] offers review and classification of context aware systems. Linda model is commonly adopted as coordination middle-ware. Lime (Linda in a Mobile Environment) [Murphy et al., 2006] is a supporting development of applications with physical or logical mobility of the agents. ReSpecT [Omicini and Denti, 2001] is a logic based tuple centre. MARS [Cabri et al., 2003] is a programmable tuple space and Intelligent Tuple Space (ITS) [Hong, 2009] is capable of reasoning and learning.

The span of normative research includes deontic logic, legal theory, sociology, decision making, and game theory [Verhagen, 2000]. In the area of deontic logic, a norm is seen as an obligation towards a social institution. In legal theory a norm is posed by a ruling organisation and is enforced with sanctions. In social sciences norms are rules that are viewed as normal by the majority, where the majority is a subjective term. Although the term norm is very intuitive it can gain different meanings in different domains. Even researchers in the area of multi-agent systems have different approaches to the conceptual design. In classification of normative architectures by Neumann [2010] we can see a striking variability in how different architectures differ in their view on the various aspects of norms.

1. *The concept* of norms can be seen as either deontic or consequential. In the deontic notion the norm is in itself a reason to action and is followed regardless of consequences and it is the reasons of actions that can be evaluated. In the consequential notion the norms are judged by the consequences, agents obey only if they are punished otherwise. In both cases the character of the norms ranges from a simple constraints, through obligations to more abstract concepts. The latter are more computationally challenging however such systems appear to be more stable.

2. *Theoretical background* of norms can be based on deontic logic, decision theory or BDI approach, where the BDI architectures provide the best possible imitation of human cognitive processes.

3. *The conflicts* in the contradicting norms are either not considered or arise between the agents, between the goals and norms or amongst norms themselves. It is desirable the all kinds of normative conflicts are handled in the architecture.

4. *The social span* of norms varies from a individual agent, through complex mental models to society centric. The most realistic representations here are socially embedded agents.

5. *The evolution* of the norms can be either static or dynamic. The dynamic norms then are either spreading or immergent, similar as in the real societies.

Some researchers believe that the social span and evolution of the norms is the most important feature in the new NMAS [Boella et al., 2008], and they argue for an *interactionist* viewpoint as opposed to the more common legalistic one. As already proposed by Castelfranchi [1998] the new development should focus more on agent interaction and social aspects. Boella et al. [2008] identifies 10 challenges of the *interactionist* viewpoint that should be considered by any new system.

### 2.3.2 Normative Programming Frameworks

There has been considerable work on normative programming frameworks and middleware to support the development of normative multiagent organisations, and such frameworks are often designed to interoperate with existing BDI-based agent programming languages. However in these frameworks, the agents do not deliberate about whether to comply with norms.

For example, $\mathcal{J}$-$\mathcal{M}$OISE$^+$ [Hübner et al., 2007] is designed to interoperate with the $\mathcal{S}$-$\mathcal{M}$OISE$^+$ [Hübner et al., 2006] middleware and allows Jason [Bordini et al., 2007] agents to access and update the state of an $\mathcal{S}$-$\mathcal{M}$OISE$^+$ organization.

Similarly, the JaCaMo programming framework combines the Jason, Cartago [Ricci et al., 2007], and $\mathcal{S}$-$\mathcal{M}$OISE$^+$ platforms. In JaCaMo, the organisational infrastructure of a multiagent system consists of organisational artefacts and agents that together are responsible for the management and enactment of the organisation. An organisational artefact employs a normative program which in turn implements a $\mathcal{M}$OISE$^+$ specification. A programming language for the implementation of normative programs as well as a translation of $\mathcal{M}$OISE$^+$ specifications into normative programs is described in Hübner et al. [2010]. JaCaMo provides similar functionality to $\mathcal{J}$-$\mathcal{M}$OISE$^+$ in allowing Jason agents to interact with organisational artefacts, e.g., to take on a certain role. However while these approaches allow a developer to program e.g., when an agent should adopt a role, the Jason agents have no explicit mechanisms to reason about norms and their deadlines and sanctions in order to adapt their behaviour at run time.

Another approach that integrates norms in a BDI-based agent programming architecture is proposed in Meneguzzi and Luck [2009a]. This extends the AgentSpeak(L) architecture with a mechanism that allows agents to behave in accordance with a set of non-conflicting norms. The agents can adopt obligations and prohibitions with deadlines, after which plans are selected to fulfil the obligations or existing plans are suppressed to avoid violating prohibitions. However, Meneguzzi and Luck [2009b] does not consider scheduling of plans with respect to their deadlines or possible sanctions.

AMELI Esteva et al. [2004] uses ISLANDER formal framework allows only regimentation of norms, which relate to actions agents can make not the state of the environment. The $\mathcal{S}$-$\mathcal{M}$OISE$^{+}$[Hübner et al., 2006] middleware allows both regimentation and enforcement of norms, but lacks monitoring and sanctioning mechanisms.

A number of normative programming languages have recently been proposed that are similar in spirit. NPL/NOPL [Hübner et al., 2011] allows the expression of norms with conditions, obligations and deadlines, and norms may be regimented or enforced. However sanctions are represented as an obligation that an agent apply the sanction to the agent that violated the norm, whereas in our framework sanctions are applied by the organization. The norm-oriented language proposed in García-Camino et al. [2009] is rule based and represents norms as implication rules. However, their norms relate to actions the agents should or should not perform as opposed to a state of the environment that should (or should not) be brought about.

The normative language of the THOMAS multi-agent architecture Criado et al. [2010] supports conditional norms with deadlines, sanctions and rewards. Conditions refer to actions (and optionally states). Norms are enforced rather than regulated, and sanctions may be applied by agents rather than the organization. The normative infrastructure does not restrict interactions between agents.

In a recent work [Balke, 2009] proposes a taxonomy of different techniques for effective implementation of norms. Three major approaches can be seen in the way norms are controlled: regimentation, norm-enforcement and hybrid approach. Most system use regimentation while only some allow norms to be violated [da Silva Figueiredo et al., 2011]. Sanctioning or reputation mechanism is used in this case. Finally, there are works which employ a mixed approach for controlling norms. In this sense, they propose the usage of regimentation mechanisms for ensuring compliance with norms crucial for the integrity of the application. Enforcement is proposed to control norms that cannot be regimented due to the fact that they are not verifiable or their violation may be desirable. [Criado et al., 2012] identified four major issues in norm enforcement as automation, ability to control general norms, act dynamically and be efficient and robust.

Fornara and Colombetti [2009] extend metamodel of artificial institutions called OCeAN (Ontology, CommitmEnts, Authorizations, Norms) with active and passive types of sanctions. da Silva Figueiredo et al. [2011] introduces NormML, which is a UML-based modeling language for the specification of norms supporting both reward and punishment. A rule-based system implemented in Jess maintains a fact base representing the organizational state, detects norm activation and monitors violations.

#### 2.3.2.1 Applying Norms to Games

There has been relatively little work on applying norms to games. The use of norms as rules of the game was first mentioned in Hurwicz [1996], who establishes norms mechanism design. According to Grossi et al. [2013] this theory assumes that players play by the rules. This strong assumption is later discussed in Hurwicz [2008]. Grossi et al. [2013] also make a clear distinction between using norms as rules of the game or as game equilibria in a form of soft constraints.

In Ranathunga et al. [2012] the authors describe the use of of expectation monitoring by agents in the Second Life virtual environment. An expectation monitoring component integrated into the Jason interpreter allows agents to detect fulfilment and violation of their expectations. Expectations have some similarities to norms in specifying conditional constraints on future states. However, they are local to an agent rather than generated by a normative organisation and there is no centralized monitoring or sanctioning of agents that violate expectations. Moreover, while the approach described in Ranathunga et al. [2012] allows agents to detect violations of expectations without recourse to a normative organization, the issues of how expectations are generated and what to do when they are fulfilled or violated are left to the agent developer. Perhaps the work that is most similar to ours is Gateau et al. [2007], in which the $\mathcal{M}$OISE$^{inst}$ normative organisation meta-model is used to control an interactive TV game show in which the avatars are implemented as agents. The purpose of the norms is to constrain players and their avatars to adopt team behaviour and to respect rules, while allowing some autonomy.

#### 2.3.2.2 Norms in Operational Multi-Agent Systems

For software agents norms must be transformed from an abstract norm into an operational form and this requires the definition of the norm control process and the definition of a normative representation. The implementation of the norm control process consists of three different parts [Vazquez-Salceda et al., 2004]: detection of norm activation, violation detection and violation management.

However, the representation of norms in this way is too abstract to be implemented in a multi-agent system. Thus, norms must be interpreted or translated into operational norms that are meaningful for the agents [Grossi et al., 2007]. Aspects that are related to the development of agent platforms, should be taken into consideration in order to facilitate the implementation of norms. When thinking about how to operationalise the norm for the use in a multi-agent system we must consider the following components [Vazquez-Salceda et al., 2004]:

1. The norm target, which is also called norm addressee, refers to the agent or agents affected by the norm. Their actions can be observed by visible actions they commit in the environment or public messages.

2. The controlled situation, which is defined over a state or an action. In the case of the state it refers to a state of the environment that can be quantifiably measured by the software agent. In the case of the action it is an observable action that the agent performs in the environment.

3. The activation conditions of the norm, which needs to be verified for the norm to become active. Temporal constraints of the norms, which refers to temporal expression defining time

4. The deactivating condition of the norms. The deadline can be either relative or absolute time. The use of the temporal constraints (before, after) is also possible.

There are two main categories of implementation mechanisms of norms into agent systems [Grossi et al., 2007; Fornara and Colombetti, 2009]. The first is called regimentation where violation of norms is made impossible and second is called enforcement, which consist of detection violation and then a penalization. The regimentation of norms can be achieved using two techniques [Balke, 2009]: (i) *mediation*, in which the agents are prevented from deviating from what they are allowed by a trusted entity that controls the communication channel; (ii) *hard-wiring*, in which agents' programs are not capable of norm violations.

intuitively, enforced norms preserve greater autonomy of the agents and in a multi-agent system including humans this desirable. The next section will consider how norms can be monitored and enforced for such a collection of autonomous agents.

#### 2.3.2.3 Norm Monitoring and Enforcement

Coordination is realized at run time by creating obligations and prohibitions (norms) for individual agents. If an agent cannot meet an obligation or violates a prohibition, the norms require that a sanction is imposed on the agent. Norms that can be violated by agents are called non regulative, as they preserve autonomy and also support the development of flexible autonomy.

Norm enforcement distinguishes different kinds of observer entities and the enforcer entities [Balke, 2009].

- *Self-enforced* norms occur when agents observe their own behaviour and sanction themselves. The agents act as both the observer and the enforcer. The problem with this set-up is the need to rely on the agent themselves. Such an arrangement is not suitable for open systems where the level of trust between the entities is unknown and variable.
- *Second-party* enforced norms are observed by those agents who are involved in the interaction. The second-party then has a choice

to either retaliate, which means to apply reward or sanction; or reciprocate, where the agents behave in the same way as the observed agent. This kind of system is often seen in game theoretic approaches and is suitable for situations where agents learn from the past interactions and are able to adapt their future responses / behaviour.

- *Third-party* enforcement is the most flexible and can be used for all types of observers (self-observed, second-party observed and third-party observed). The third-party acts as an arbiter in the resolution process and can be enforcing the norms either socially with the help of the society or institutionally, where the enforcing entity is in charge of applying institutional sanctions and rewards. Both kinds of retribution can be seen as an application of a sanction, which either damages an agent's reputation in the society and can prevent the agent from future interaction or the sanction is something more tangible that negatively affects the agent.

An approach is for norm-aware agents to be monitored by a third-party entity, which we call the normative organisation. The organisation is also responsible for the enforcement of the norms and the establishment of them. Traditionally there are different approaches for norm establishment in agent societies:

- Bottom-up approach is also called norm emergence where norms emerge as a result of the interaction between the agents.
- Top-down perspective sees the norms being defined at the institutional level. It can be done either *off-line* by the system designer or *on-line* by a legislative agent that is empowered to do so.

If we wish to use norms to support flexible autonomy in agents then the norms must be established in a top down manner in order to provide the framework within which agents can make meaningful decisions.

## 2.4 Programming Norm-Aware Agents

There are many ways to design an agent architecture and many paradigms are used when programming agents — imperative, declarative, BDI, hybrid [Bădică et al., 2011]. Bordini et al. [2006] also propose a classification of agent programming languages based on a lightweight interpretation of the programming paradigm as imperative, declarative and hybrid. Baldoni et al. [2010] argue for declarative approaches to agent programming because they are particularly suitable to handle the complexity of agent systems [Mascardi et al., 2005] discusses and compares all well known BDI-style languages.

Castelfranchi et al. [2000] define a Norm Autonomous Agent (also called Deliberative Normative Agent) as an agent that is:

- Able to know that a norm exists in the society and that it is not simply a diffuse habit, or a personal request, command or expectation of one or more agents.
- Able to adopt this norm impinging on its own decisions and behaviour.
- Able to deliberately follow that norm in the agent's behaviour.
- Able to deliberately violate a norm in case of conflicts with other norms or, for example, with more important personal goals; of course, such an agent can also accidentally violate a norm (either because it ignores or does not recognise it, or because its behaviour does not correspond to its intentions).

Dignum [1999] explains how agents reason about violating a norm. It should be noted that his proposal states that norms should not be imposed, they should only serve as a guidance of behaviour for agents, or agents will lose their autonomy. With this idea in mind, Dignum further presents an agent architecture where social norms are incorporated in the BDI cycle of the agent. In that architecture, norms are explicitly expressed in deontic logic by means of the obligation operator O and are divided in three levels:

- Convention level: norms modelling the social conventions
- Contract level: norms regarding the obligations that arise when an agent Ai commits to either perform an action $\alpha$ or to achieve a situation p requested by agent Aj

- Private level: the intentions of the agent are seen as commitments towards itself to perform a certain action or plan

Nevertheless, despite the large amount of theoretical work on normative agents, there are still very few implementations offering practical reasoning within an environment where norms act as guidelines for the agents. Programming norm-aware agents in conventional agent-oriented programming languages is difficult, as they lack support for deliberating about goals, norms, sanctions and deadlines. In Alechina et al. [2012] an agent programming language 2APL, for programming *norm-aware* agents was introduced. Norm-aware 2APL agents are able to deliberate on their goals, norms and sanctions before deciding which plan to select and execute, and are able to violate norms if it is in their overall interest to do so, e.g., if meeting an obligation would result in an important goal of the agent becoming unachievable.

In the early work by Dignum et al. [2000] propose a theoretic modification of the BDI architecture where the deliberation process takes into account the influence of norms and obligations. The authors use the term norms for social aspects needed for cooperation and coordination, and the agents are motivated to follow them with social benefit. The obligations are strict restrictions and violations of them trigger penalties. The agents have preference ordering of penalties. In the deliberation cycle the agent processes events and generate a plan through a selected event.

BOID Broersen et al. [2002] is one of the first BDI architectures with an explicit notion of obligation. The BOID agents are formed by four components, which are Beliefs, Obligations, Intentions and Desires. Obligations are treated here as external motivations. The agents need to have a ordering function, which resolves conflicts between and within the components. The ordering function is static and the agents are predefined to prioritise one part over another. This limits the agents to act only in foreseeable situations and they cannot dynamically adapt to the environment and reason about the norm compliance.

One of the first practical architectures for a norm-driven agent was NoA [Kollingbaum and Norman, 2004; Kollingbaum, 2005], which is an extension of a traditional BDI agent. It changes the focus of agent behaviour from achieving desires to fulfilling norms. As in NoA, Meneguzzi and Luck [2009a] use an explicit representation of the effects of an agent's plans to detect potential norm violations, as well as deciding which plans are more suitable for achieving an obligation, but their agents are still driven by their desires like traditional BDI agents. Meneguzzi et al. [2011] later introduce the notion of prognostic normative reasoning so that the agent can reason about norm-compliant planning in advance. In order for that, they use probabilistic plan recognition to predict the user's future plan steps based on the user's current behaviour and it's change.

n-BDI [Criado et al., 2010; Criado, 2013] is a norm autonomous extension of graded BDI agents, which helps the agents decide which obligations (they do not consider prohibitions) to adopt as desires. The graded BDI architecture sees an agent as a set of interconnected contexts. Each context (mental, functional and normative) has its own artefacts like language, axioms and rules. Additionally, bridges are used for transferring information from one context to another. In the extension by Criado [2013] they use rules to detect and resolve inconsistencies between norms and desires, while focusing on the adoption of the norms and ensuring the agent's mental state remains coherent. The agent acquires abstract norms and transform them itself. The model also includes degree of certainty. Their multi-context BDI agent is extended with Recognition Context (RC), which is responsible for the norm identification process and the Normative Context (NC), which allows agents to consider norms in their decision making processes. They do not consider the fulfilment of norms in the theory.

Another multi-context BDI architecture is proposed by Gaertner [2008]. In this norm-oriented extension the obligations and prohibitions are blindly followed by the agents. The norms are translated into intentions and it the case of a conflict arising it is resolved with an argumentation based approach and preference function. The norms are not considered explicitly therefore the agents cannot deliberate about the norm compliance.

Oren et al. [2011] describe a technique for taking norms into consideration when deciding how to execute a plan. Their norms are constraint based, allowing for fine-grained control over actions. This technique allows for reasoning about the interactions between norms and also resolves conflict by selecting actions where the cost of violating one set of norms is outweighed by the reward obtained in complying with another.

In the field of goal-oriented agents F. Lopez y Lopez and dInverno [2006] proposes one the first norm autonomous architecture. The agents are motivated by sanctions and rewards to follow the norms. However they are autonomous to violate the norms while pursuing their own goal. Another example of norm-oriented agent is Normative KGP agents, which is described in F. Sadri and Toni [2006]. In the EMIL architecture [G. Andrighetto and Paolucci, 2007] the agents are able to learn new norms by observing other agents located in the environment. Agents in the architecture by S. Joseph and Dellunde [2010] are able to reason about unconditional obligations. The agents use argumentation dialogues to propose, accept or reject the obligations. The coherence of the proposal is used as a criterion for acceptance.

We begin by considering how norms can be defined and internalised by both human and software agents. We suggest that autonomy arises from selective obedience to or violation of norms. We show how norms are represented and operationalised in multi-agent systems, and then consider the requirements for system architecture. We will consider how software agent and human agent can both be thought of as *norm-aware agent* is such a system.

Normative organisations provide a means to coordinate the activities of individual agents in multi-agent systems, where an individual agent can be a human or a software agent. Multi-agent systems that use norms to regulate agent behaviour are called normative multi-agent systems [NMAS, 2013]. Norms can be viewed as defining standards of behaviour. They have been widely proposed as a means of coordinating and regulating the behaviours of individual agents to ensure global properties of a multi-agent system. To have the desirable effect on agents' behaviour, the norms need to have deadlines and their violations have to incur a sanction.

Norms can be defined either in terms of the agents' actions or the state of the agents' environment. In the first case, norms indicate which actions (agent-agent or agent-environment actions) are obliged or prohibited while in the second case norms indicate which states of the environment have to be brought about or not to be brought about by the agents. The norms that are defined in terms of actions need to be enforced/regimented by monitoring the agents' actions while state-based norms can be enforced/regimented by monitoring the state of the environment with which agents interact.

In thinking about how to structure norms in normative multi-agent systems, norms can be implemented either endogenously by integrating them into the programs of individual agents (e.g., an agent may be programmed not to exceed the speed limit) or exogenously by additional components that observe and evaluate the agents' behaviours in order to check compliance or violation of norms (e.g., the agents' speed is monitored and the identities of agents that violate speed limitations are registered). However the nature of a human-agent collective means that some norms will be more appropriate for human agents and some software agents.

In exogenous normative multi-agent systems, norms can regulate the behaviour of agents by means of regimentation or enforcement. Norm regimentation prevents agents from violating norms (e.g., prevent them from moving into a restricted area) while norm enforcement allows agents to violate norms but imposes sanctions on violating agents to compensate for their violations (e.g., violating the speed limit incurs a sanction in the form of a fine) Boella et al. [2006]. In multi-agent systems where norms are implemented exogenously, regulation is realized by processing norms at run time. The processing of norms in such systems requires creating and eliminating norms based on their conditions and deadlines, monitoring the activities of participating agents, evaluating their behaviour with respect to the specified norms and finally determining appropriate consequences for the participating agents.

In multi-agent systems where norms are implemented endogenously, individual agents have internalized norms in the sense that their decision procedures are defined in terms of the norms. Although the agents' decisions in such systems do not necessarily need to be norm compliant, it is not clear how to cope with norm violations by self interested agents. An external entity that detects norm violations and compensates them by means of sanctions.

Broersen et al. [2004] When using norms in coordination of HACs we need to consider three aspects: sanctions, capabilities and deadlines. Sanctions can have different forms and can affect the agents in direct or indirect way. Their effect will vary for software and human agents. Each member of the HAC has different capabilities, which needs to considered within the context. Specifically, norm deadlines highlight the differences in human in agent capabilities. While software agents compute the time left naturally and can handle very large amount of norms at the time. We might imagine that deadlines pose additional stress for human agents, which might not be able to cope with a very large amount of norms at a time without help.

### 2.4.1 Norm-Aware Deliberation

From the agent's perspective, norms that are imposed on an agent by a normative organisation might conflict with the agent's existing goals. In the literature this is type of autonomy is called Norm-Autonomy [Carabelea et al., 2004]. Castelfranchi et al. [2000] then define a Norm Autonomous Agent (also called Deliberative Normative Agent) as an agent that is

- Able to know that a norm exists in the society and that it is not simply a diffuse habit, or a personal request, command or expectation of one or more agents.
- Able to adopt this norm affecting agent's own decisions and behaviour.
- Able to deliberatively follow that norm in the agent's behaviour.
- Able to deliberatively violate a norm in case of conflicts with other norms or, for example, with more important personal goals.

A similar definition, which adds the need for the agents to be able to take an initiative in re-issuing the norm and monitoring other's behaviour appears in Conte et al. [1999].

Regarding norm violations Dignum [1999] explains how agents reason about violating a norm. In his approach, norms should not be regimented but enforced. They should serve only as a guidance of agents' behaviour. Otherwise the agents would lose their autonomy. Dignum [1999] proposes an agent architecture with norms incorporated in the BDI cycle. The reasoning about norms is then split into three levels:

- Convention level: norms modelling the social conventions

- Contract level: norms regarding the obligations that arise when an agent $Ai$ commits to either perform an action $\alpha$ or to achieve a situation p requested by agent $Aj$

- Private level: the intentions of the agent are seen as commitments towards itself to perform a certain action or plan

In our view and following the approach of Alechina et al. [2012] an agent is norm-aware if it can deliberate on its goals, norms and sanctions before deciding which plan to select and execute. In case of a conflict, a rational agent must choose between its existing goals and the norms imposed by the organisation. A norm-aware agent is able to violate norms and accept the associated sanction if it is in the agent's overall interest to do so. For example, if complying with a norm would cause a more important goal becoming unachievable. In this way the agent stays autonomous on all four levels of autonomy that take part in the decision making process and are generally accepted in MAS as summarised in Verhagen [2000]:

- the level of actions

- the level of plans

- the level of goals

- the level of norms

The obligations or prohibitions from a normative organisation may conflict with an agent's goals or with other obligations or prohibitions it has already received. Alechina et al. [2012] propose to solve the conflict with preference ordering. Both goals and norms are given priorities. It is assumed that the agent has a preference ordering function to compare the severity of sanctions. It should also be taken into account which goals can be attended concurrently and their deadlines. In this manner the agent stays norm-aware by committing to a maximum feasible amount of norms (goals).

### 2.4.2 Norm Representation

In the previous section we described what norms are and how they might regulate behaviour. However, if are to build a system that embodies norms for a human-agent collective then we should consider how they can be systematically represented.

To represent a norm we need to specify its constituent parts, and under what conditions it would be considered obeyed or violated. In order to determine whether a norm has been violated, the controlled situation must be detected. This can be defined over a state condition or an action. In case of obligation it is a state the an agent needs to bring about or which action to execute. In case of prohibition it is a state the an agent needs to refrain from bringing about or which action to avoid. In the following definition we assume the norms are state based.

We would like to represent a norm that only becomes valid in a particular context. We can do that by defining a conditional obligation as a tuple

$$\langle l, c, O(\iota, o, d, s) \rangle$$

with the intuitive reading "norm with a label $l$ states: if condition $c1$ holds in the current state of the environment then there is an obligation for agent $\iota$ to establish an environment state satisfying $o$ before deadline $d$, otherwise agent $\iota$ will be sanctioned by updating the environment with $s$". In the treasure hunt example, an obligation to collect treasure would be represented as shown in Figure 2.1, which indicates that when there is gold $object(Gold)$ hunter $Agent$ is obliged to collect the gold in 5 $mins$. Violating this norm results in the treasure hunter Alice being sanctioned with $100points$, damaging their game score.

When the precondition becomes true the obligation is detached. A detached obligation is sent to agent $\iota$ in the form:

$$obligation(\iota, o, s)$$

with the intuitive reading "agent $\iota$ is obliged to establish an environment state satisfying $o$ before deadline $d$, otherwise it will be sanctioned by updating the environment with $s$". For example, when an agent is obliged to collect a gold, the organisation generates and sends the following

```
<
  collectGold,                              //label
  (hunter(Agent),                           //precondition
   object(Gold),
   at(X,Y,Gold)),
  O(                                        //deontic part
    Agent,                                  //addressee
    collect(Gold,X,Y),                      //list of states
    now + 20,                               //relative deadline, only in obligation
    reduce(100)                             //sanction
  )
>
```

Figure 2.1: Example of obligation

obligation to the corresponding agent $\iota$:

$$obligation(\iota, collectGold(X,Y), 20, reduce(100))$$

We assume that deadlines associated with detached obligations are relative and can be mapped to real time values when processed by the agents.

Similarly, a conditional prohibition is expressed as a tuple

$$\langle l, c, F(\iota, p, s)\rangle$$

with the intuitive reading “norm with a label $l$ states: if condition $c$ holds in the current state of the environment, then it is forbidden for agent $\iota$ to establish an environment state satisfying $p$, otherwise sanction $s$ will be imposed.” Unlike obligations, where a sanction is incurred once if the obligation is not discharged by the deadline, in the case of prohibitions, the agent incurs a sanction each time the prohibition is violated. In the treasure hunt example a prohibition to enter water would be represented as shown in Figure 2.2, which indicates that hunter *Agent* is prohibited from entering water. Violating this norm results in the treasure hunters being sanctioned with 500 *points*, damaging their game score.

When the precondition $c$ is met the prohibition is detached and event broadcast to agent $\iota$ has a form

$$prohibition(\iota, p, d, s)$$

```
<
  enterWater,                                    //label
  (hunter(Agent)),                               //precondition
  F(                                             //deontic part
    Agent,                                       //addressee
    enterWater,                                  //list of states
    reduce(500)                                  //sanction
   )
>
```

Figure 2.2: Example of prohibition

where $p$ is the prohibited list of states and $s$ is the sanction the will be imposed if the norm is violated. For example, when an agent is prohibited to enter water, the organisation generates and sends the following prohibition to the corresponding agent $\iota$:

$$prohibition(\iota, enterWater, reduce(500))$$

.

### 2.4.3 From Norms to Flexible Autonomy

The previous section explained how we can articulate the rules of a game as norms and how the norms can be represented as a set of conditional obligations and prohibitions. Now let us consider how an agent operating under multiple norms can make decisions and adapt to changing environment. The agents may not be able to fulfil all norms at any given time. In order to achieve a desired outcome the agents must autonomously decide which norms should be obeyed and which should be violated. In our view this flexible autonomy can be achieved with such a norm regulated goal generation system. The idea of flexible autonomy in goal generation already appeared in Castelfranchi [1995b]; Luck et al. [2003] who see the issue of *exercising free will* in the meaning of choosing goal as a core of agents' autonomy.

The norm-aware agents as defined in Alechina et al. [2012] have support for normative concepts including obligations, prohibitions, sanctions, deadlines and durations. Returning to our treasure hunt in the instance of only one applicable norm, an agent $\iota$ receives an obligation to *collectGold* in 5 minutes, otherwise it will be sanctioned by $reduceScore(100)$. The agent believes that sanction of reducing 100 points has priority 10 (on the scale from 1 to 10, where 1 is the highest) and that the travel to the location of the gold takes 3 minutes. Agent proceeds to collect the gold in order to fulfil the obligation.

$$obligation(\iota, collectGold, 5mins, reduceScore(100))$$

Now lets consider that agent $\iota$ also receives a prohibition from the organization prohibiting the agent from entering any water $enterWater$ with a sanction of $reduceScore(500)$. The only gold the agent can locate at the moment is in the water.

$$prohibition(\iota, enterWater, reduceScore(500))$$

The agent believes that the sanction of reducing 500 points has priority 5. In this case the believes that is more important to obey the prohibition goal than is to obey the obligation goal. The agent therefore decides not to comply with the obligation to collect the gold.

These two scenarios show how the changing context in which multiple norms apply is used by the agent to autonomously determine the most appropriate action. The advantage of this system is that the norms are processed at the run time and the agent can therefore dynamically adapt changes in the context. For example, if later on there was lack of collected gold, the priority of collecting gold will be increased and the agent will this time go into the water for the gold. Note, that agents will react differently to the norms because of their own internal beliefs and goals.

## 2.5 Towards Group Norms

In this section we will introduce the problem of group norms by reviewing the literature concerning joint intentions and the SharedPlans models, logical background and philosophical motivation. The problem is split into three parts: interpretation of a collective obligation, planning and coordination of the a joint activity and norm enforcement together with sanctioning mechanism.

Group norms are a new research topic and have not been yet used in coordination of multi-agent systems or in human-agent coordination. Although there is a wide body of research on norms, research has ignored a formal treatment of norms aimed at groups of individuals [Aldewereld et al., 2013, 2015].

### 2.5.1 Team work: Joint intentions and SharedPlans theory

Philosophers seem to agree that joint actions involve shared intentions and that a shared intention does not reduce to a simple summation of individual intentions. There is disagreement, however, how to best analyse shared intentions.Bratman [1992] first identifies three features of joint actions as shared cooperative activities (SCA) that an analysis of shared intentions would have to account for:

1. Mutual responsiveness — each participating agent attempts to be responsive to the intentions and actions of the other, knowing that the other is attempting to be similarly responsive.

2. Commitment to the joint activity — participants each have an appropriate commitment to the joint activity, though perhaps for different reasons. Their mutual responsiveness is in the pursuit of this commitment.

3. Commitment to mutual support — each agent is committed to supporting the efforts of the other to play their role in the joint activity. These commitments to support each other put them in a position to perform the joint activity successfully even if they each need help in certain ways.

Probably one the best known formalisms is Cohen and Levesque [1990] theory of intention where they examine several properties for intentions, and define intentions as chosen desires. Cohen and Levesque's theory has later influenced the research in Belief, Desire, Intention (BDI) frameworks that focus on multi-agent coordination and cooperation. Following the above philosophical background one of the earliest work on collaborative plans for teams consisting of humans and computer systems was the SharedPlans model by Grosz and Kraus [1993, 1996] based on shared beliefs and intentions, which are refined over the time by multiple collaborating agents. The SharedPlans model is defined using first-order logic. There are four different intention operators ranging from potential proposition to intentions adopted by an agent:

- Int.To(A, x, T1, T2, C) agent A at time T1 under context C intends to do x at time T2
- Int.Th(A, y, T1, T2, C) agent A at time T1 under context C intends that y holds at time T2
- Pot.Int.To(A, x, T1, T2, C) agent A at time T1 under context C potentially-intends to do x at time T2
- Pot.Int.Th(A, y, T1, T2, C) agent A at time T1 under context C potentially-intends that y holds at time T2

An intention in this model is then formally defined as a tuple consisting of an agent, intention, time of intention, planned time, and context. GTD (get to do) and CBA (can bring about) operators, and Done operator. A recipe contains a group of actions, initialized variables required to perform an intention. Recipes are stored in the agents' library and agents are supposed to be able to share these libraries. If an agent knows a full recipe to perform an action it can form a full individual plan. In particular it must either know how to do every action from the recipe or believe that it can get someone else to perform the action. Partial individual plans are then used when an agent has only a partial recipe. The agent has to either to research the unknown part or contract someone else to do it. Agents engaged in the SharedPlans can enlist the aid of their collaborators in such situations. In a partial SharedPlan a group of agents must mutually believe there is a recipe or an intention that they have a full SharedPlan. To obtain that recipe and that every member of the group have an individual intention to elaborate upon the recipe.

Fan and Yen [2012] further extended Grozs and Kraus SharedPlans framework [Grosz and Kraus, 1993]. They argue that there is still lack of adequate semantics for the notion of potential intentions. They present formal semantics for intentions and potential intentions, drawing upon both the representational and accessibility based approach. The model captures the dynamic relationship among intentions and potential intentions by providing semantic rules and conditions. In their model, norms are treated as a special class of intentional context, and it is shown that norms play a critical role in the dynamics of intentions in multi-agent systems. Specifically when upgrading potential intentions into full-fledged intentions, which is regulated with norms. With the use of formal semantics for the four intentional operators of the SharedPlans theory they also validate some properties of the relationship among intentions that are widely accepted for intention attitudes in the literature. In their model, however, time no longer plays a role.

Dunin-Keplicz et al. [2011] introduced a novel approach to modeling deliberation dialogues in teamwork. They argue that although dialogues and speech acts have been frequently used to model communication in multi-agent systems the TEAMLOG (a framework for modelling teamwork) solution is unique. Their proposed scenario consists of four stages during which agents submit their proposals, vote on preferred ones and challenge or concede the choice of the selected one. During plan formation a team deliberates together how to proceed. Collective planning consists of three phases: task division, means-end analysis and action allocation.

Jonker et al. [2010] consider components of an agent's mental model and how to measure the sharedness of the model across agents. An agent has along with its mental model a physical model, goals, a team, a mind, an extension of the mind, and the actual system of concern. They offer an initial analysis of how shared mental models apply to human-agent teams, and conclude that agent designers can use the idea of mental models to improve teamwork.

In an early work on collective activity Castelfranchi [1995a] defines that if an agent is socially committed to another agent to bring about a state of affairs, then the former has an obligation towards the latter. They illustrated as follows: if agent i commits to deliver a piece of work to agent j, this social commitment implies, among other things, the creation of a relativised obligation of i towards j. This is another approach how to coordinate group dynamics when an obligation triggers another obligation.

Diggelen and Bradshaw [2010] implemented collective obligations in human-agent teams using KAoS policies. In this framework there is a notion of coactivity in a sense that there is collective obligation for all players that are participating in a task, even if not currently assigned to the task. For example, if there is a collective obligation to stay safe not every agent will be assigned a specific individual task but will have certain duties and obligations that correlate with good teamwork.

### 2.5.2 Agency theory

Pettit [2007] claims that in order to make a collective of agents responsible for something there needs be a sense that they form a group agency together. List and Pettit [2011] provide an argument that group agents exist and can be treated as an entity. They make an interesting point regarding the autonomy of a group agent. The 'personification' of such an entity is complex as it cannot be simplified to a merging of individual attitudes.

List and Pettit [2011] believe that when the system satisfies certain conditions it can be seen as an agent. These conditions which are motivated by a small robotic device are that the agent has a representation of the environment, it has motivational states and lastly it has a capability to act in the environment. These three conditions can be mapped to BDI architecture. Group agency is then divided into types where group in joint intentions is the one that is closest to our understanding. Such a group needs to have a shared goal, group members to contribute individually and interdependently towards the joint intention, and a common awareness [List and Pettit, 2011, chapter 1].

Their notion of group agency was analysed by Porello et al. [2014] who provided an ontology with the use of judgement aggregation while distinguishing between a mere collection of individuals and a group agent as a new social concept.

### 2.5.3 Collective responsibility

Collective responsibility involves both causal responsibility and blameworthiness of agents for harm [Smiley, 2011]. As such it causes a number of controversies in the philosophy field. The first is mentioned earlier which is the problem of ascribing moral values to a collective. Namely whether is possible for a group to have intentions, which are generally assumed prerequisite for a moral responsibility.

The second controversy relates to a decomposition of the blame amongst the individuals. Mellema [1997] argues that there are six different ways for individuals to cause harm and according these there may be different levels of contribution. These aspects are that agents can command or influence others to cause harm, praise them for causing harm, give fail to stop or consent to wrong-doing by others. Crawford [2007] on the contrary assumes that is possible to blame and punish group itself and suggests that appropriate punishment can range from an apology to making changes in the group structure if such would prevent a production of harm in the future. Similarly, Shockley [2007] believes that is possible to assign blame to a collective because it plays a coordination role in the production of harm. Particularly, the author argues that when the collective has a inexplicable role in the event, the existence of the collective enables the individuals to participate in the event, and the members of the collective have individually created harm which may be aggregated into the collective harm. Björnsson [2011, 2014] is concerned with cases when agents are not aware they are part of a collective and as such together are causing any harm.

The third controversy is about the practicability of ascribing a collective responsibility. Tännsjö [2007] believes that it is possible to hold morally responsible and punish the whole group including innocent agents. On the contrary Braham and Van Hees [2012] argue that to hold an agent responsible for a harm it needs to have a reasonable option to have acted otherwise. Ferreira et al. [2013] developed a model for agent appraisal based on their in and out group interactions.

Fitness to be held responsible [List and Pettit, 2011, chapter 7],[Pettit, 2007] is defined by satisfying three conditions. The agent has a normatively significant choice between doing something good or bad. The agent has an ability to normatively judge options it has and also the agent has an ability to make an choice between the options. While the first two requirements are relatively simple to satisfy the third one causes an issue because the group agent does not act by itself by through its members. The authors, however, believe that a group agent can be held responsible for its actions, because it has control over its members, who perform the actions.

Singh [1999] captures ontology of social normative concepts in multi-agent system and later [Singh, 2014] builds a sociotechnical system which is self governed with norms.

### 2.5.4 Sanctioning in a group

A mechanism for sanctioning was inspired by a work done in the field of logic. It is interesting to consider how to decide who is to blame and to what extent.

An extended formal framework of deontic logic, which includes a notion of collective agency, is introduced in Grossi et al [Grossi et al., 2004]. They provide a framework for the notion of distributed plans and the issue of coordination, seen as a process of managing interdependencies between activities.

Cholvy and Garion [2007] address the question of the translation of a collective obligation into individual obligations in the rather general case when the collective obligations are conditional ones. They propose a model in which there is a group that has neither a particular hierarchical structure nor an institutionalised representative agent. The derivation of individual commitment depends on the ability of the agents (what they can do) and their own personal commitments (what they are planning to do).

Carmo [2009] describes how to model the notion of collective agency in the *stit* semantic framework. In the joint action concept a group of agents jointly 'sees to it' (brings it about) in a meaning that they jointly cooperate to bring about a state versus collective agency where the group acts through a direct act of one or more members, but not all of them. In his review of the *stit* semantic framework and the main *stit* operators he specifies how to model the notion of collective agency, both in the sense of a joint action of a group of agents and as organisations. In his framework it is possible to define the direct and immediate effects of agents' actions both in the achievement sense and in the deliberative sense using a kind of dynamic logic operator.

A similar approach is taken by Sergot [2008] who presents a framework for describing and analysing norm-governed multi-agent systems, where a distinction is made between system norms and agent-specific norms, and where it is possible to identify and characterize several different categories of non-compliant behaviour. The formal modal logic language includes operators for expressing that a particular agent brings about that such-and-such a state of affairs exists. The semantics is based on so-called agent-stranded labeled transition system (agent-stranded LTS).

Coalition Epistemic Dynamic Logic (CEDL)de Lima et al. [2010] is logic for reasoning about responsibility and an extension of propositional dynamic logic (PDL). Formalization of the two kinds of responsibility is discussed: forward-looking and backward-looking responsibility. Responsibility is introduced to better guide agents' decisions seen as 'oblige to ensure'.

Dignum and Dignum [2011] tries to specify and relate main aspects of organisations (e.g. power, delegation, agent actions or normative issues) and describe a formal model for organizational concepts including roles and role enacting agents. Logic of agent organization (LAO) is an extension of branching-time temporal logic CTL* added the set of agent, role tuples to the state transitions to indicate which role enacting agents are influencing the changes through this transition. Such formalizations allow representing and reasoning about strategic powers of agents and coalitions in terms of agent abilities and capabilities in a game-theoretic setting.

## 2.6 Summary

In this chapter we have seen that autonomy is a crucial part of agency and also that flexible is a relatively new a largely unexplored area. We also have seen that human-agent interaction is increasingly important in real world scenarios. Norms have been described as a useful way of regulating multi-agent systems and it is potentially interesting to consider how norms might be used. This gives us a good working understanding of the key principles around norms, multi-agent systems and human-agent interaction (Figure 2.3). We have seen the roles that norms can play in human agent systems, we have also proposed a way to represent norms in terms of conditional obligations and conditional prohibitions, and how a system regulated by norms facilitate flexible autonomy. We have seen how intuitively norms might be engaged with by humans, but when operationalising norms in a multi-agent system, autonomy can be preserved by introducing third party enforcing organisation and specific software agent characteristics. In the next chapter we present a framework that implements a multi-agent system based on these conclusions. The use of norms in multi-agent systems is relatively well understood area and human-agent interaction is a growing field of research. Therefore it is important to investigate the intersection of these common but currently disconnected areas. Furthermore, specific work regarding group norms would appear to be especially relevant when considering groups of entities working together.

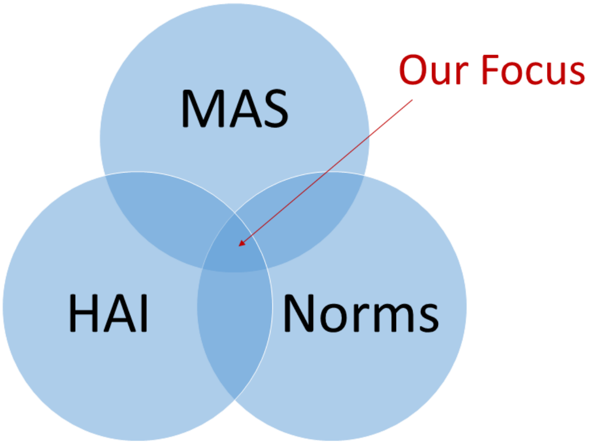


Figure 2.3: Literature review topics

# 3

# NormHACing Framework

In this chapter we explore how to enable flexible autonomy in human-agent collectives (HACs) with normative multi-agent systems. In Chapter 2, autonomy was defined as ‘freedom to select course of action’ and norms prescribe how the agents ideally should and should not behave. Norms can therefore be seen as limiting the autonomy of agents to choose what to do and how to do it. For example, an agent’s goal can be to go to a shop and buy sweets, and a norm is prohibiting agent to step on grass or be penalized by a sanction. The agent will still go to a shop it will however choose to walk on a pavement to prevent incurring a sanction. Norms can therefore be used to (partially) control the autonomy of agents. We define flexible autonomy as autonomy that varies depending on the context. In our approach flexible autonomy is achieved through the creation of norms. For example, agent is prohibited from walking on grass when it is wet. The context condition in this example is grass being wet and agent will not be violating a norm by stepping on dry grass. Context in our approach is a set of facts (propositions), where different valuations of the facts give different overall context in general.

We begin by considering how norms can be defined and internalised by both software and human agents. We show how norms are represented and operationalised in multi-agent systems, how norms are understood and followed by human agents, and then consider the requirements for the system architecture. We explore how software agents and human agents can both be thought of as norm-aware agents is such a system. Our solution combines exogenous and endogenous use of norms in the sense that norms are created and enforced and exogenously, while individual agents who are informed about created norms process norms endogenously by deciding whether to obey or violate them.

In our proposed combined framework, we use 2OPL for the exogenous creation, enforcement of norms, and use N-2APL for the endogenous processing of norms by individual software agents. Norm-aware N-2APL agents are able to deliberate on their goals, norms and sanctions before deciding which plan to select and execute, and are able to violate norms if it is in their overall interest to do so, e.g., if meeting an obligation would result in an important goal of the agent becoming unachievable. Human agents are notified of detached norms in the system interface, weigh goal importance against sanction values and proceed with their preferred course of actions.

In this chapter we focus on the software agents. Human agents are addressed later in Section 4.8.1. The integration of N-2APL and 2OPL is achieved using a tuple space which represents both the (brute) state of the multi-agent environment and the detached norms and sanctions comprising its normative state. The integration provides a framework for norm-aware multi-agent systems in which autonomous agents deliberate about whether to conform to the norms imposed by a normative organisation. The use of a tuple space makes it straightforward to integrate other system components.

While norms have been used to represent desirable behaviours that agents should exhibit, existing research was mostly focused on the study of norms that affect a single individual. A research area that refers to group norms which are norms that are addressed groups of agents remains still as an open problem. To begin we have to analyse how different interpretations of the group affect norms. Depending on the situation norm may affect the group as a whole, each member of the group or some members of the group. Each of these interpretations then may require different coordination and enforcement mechanisms. As we see HACs as not just human-agent couples but groups with variable number of members we have proposed solution for group norm taxonomy and coordination in Chapter 4. We have developed a reasoning mechanism which enables groups of agents (HACs) to coordinate themselves with variable degree of autonomy when deciding on possible courses of action.

## 3.1 Introduction

The normHACing framework connects together the N-2APL and 2OPL with a Linda communication and coordination model Carriero and Gelernter [1989] which represents both the state of the multi-agent environment and the detached norms and sanctions comprising its normative state. While there are approaches that offer similar functionality to 2OPL and the tuple space in our framework, they have not been integrated with a norm-aware agent programming language such as N-2APL. In contrast to existing frameworks such as $\mathcal{S}$-$\mathcal{M}$OISE$^{+}$ [Hübner et al., 2006] which regulate behaviour by norm regimentation, our approach is based on norm enforcement and sanctions. Frameworks such as ORA4MAS [Hübner et al., 2010] provide support for both norm regimentation and enforcement, however monitoring must be explicitly coded in organizational artifacts.

An advantage of using a tuple space to represent both the brute and normative state of the agent's environment is monitoring of norm compliance and violation by the 2OPL interpreter is greatly simplified. On the other hand, approaches such as ORA4MAS allow decentralized (and arguably more flexible) decision making about the appropriate sanction for a violation. As in N-2APL, the agents can adopt obligations and prohibitions with deadlines, after which plans are selected to fulfil the obligations or existing plans are suppressed to avoid violating prohibitions. However, unlike N-2APL, Meneguzzi and Luck [2009b] does not consider scheduling of plans with respect to their deadlines or possible sanctions.

### 3.1.1 Contributions

The development of the framework consisted of the following steps:

1. **Design and implementation of N-2APL interpreter**. N-2APL language existed only as a formal specification and its interpreter needed to be developed as an extension of 2APL as specified by Alechina et al. [2012]. Details about the implementation can be found in Section 3.3.

2. **Connecting N-2APL and 2OPL**. 2OPL was developed by Adal [2010] and was chosen as the normative programming language of the system. A middleware in Java is transforming prolog-like formulas from N-2APL and 2OPL languages to simple objects and inserts them into the JavaSpace. 2OPL was selected thanks to an existing collaboration between our research group and its creators. For more details please refer to Section 3.4.

3. **JavaSpace installation**. Amongst the number of available tuple space implementations Jini JavaSpace (Apache River) was used because of its simplicity and versatility [Edwards and Rodden, 2001]. Tuples are stored as serialized objects. JavaSpace was used as a coordination mechanism between all parts of the system and store the state of the game. Refer to Section 3.5 for detailed description.

4. **Connecting a third party application (external environment)**. A middleware in Java is sending HTTP requests to the game server and parsing its response in JSon format. GeoSense game is described in Section 3.6.

## 3.2 2APL

2APL [Dastani, 2008] is a BDI-based agent programming language that allows the implementation of multi-agent system. The 2APL language is based on cognitive concepts such as beliefs, goals and plans and brings together declarative (beliefs and goals) and imperative (plans and interaction with the external environment) style of programming. 2APL supports both individual and multi-agent concepts. The multi-agent

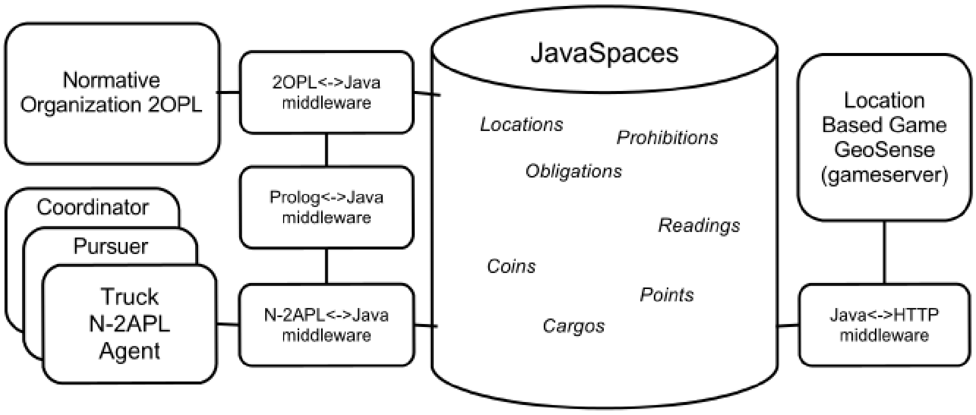


Figure 3.1: Overall system architecture

programming constructs are: (1) design of individual agents as separate modules with names; (2) access to an external environment; and (3) specification of the relations to the environment. Each agent can be connected to multiple environments (Java objects). The programming construct designed for individual agents are beliefs, goals, actions, plans, events and rules.

A 2APL agent program specifies an agent's initial beliefs, goals, plans, and the reasoning rules it uses to select plans (`PG-rules`), to respond to messages and events (`PC-rules`), and to repair plans whose executions have failed (`PR-rules`). The initial beliefs of an agent include the agent's information about itself and its surrounding environment. The initial goals of an agent consist of formulas each of which denotes a situation the agent wants to realize (not necessarily all at once). The initial plans of an agent consist of tasks that an agent should initially perform.

### 3.2.1 Beliefs and Goals

A 2APL agent program's beliefs are specified in the belief base and include agent's internal and external beliefs that are Prolog Bratko [2001] facts or rules, and where facts are assumed to be ground, which means without unbound variables. Example is shown in example in Figure 3.2. In this example agent believes that it has `1000` points, its initial position in the environment is `(8,19)` and it can lower or raise its position coordinate by subtracting resp. adding `1`.

```
Beliefs:
 points(0).
 position(8, 19).
 raise(X,NewX):− NewX is X + 1.
 lower(X,NewX):− NewX is X − 1.
```

Figure 3.2: Example of agent's beliefs

```
Goals:
 points(5) and position(5,5) , points(10)
```

Figure 3.3: Example of agent's goals

The goals of a 2APL agent are implemented in its goal base. The goal (state of the environment that the agents aspires to establish) is a conjunction of ground atoms. In the example in Figure 3.3 are two goals. In the first one the agents seek to establish a situation where it has `5` points and also reached position `(5,5)`. In the second goal the desire of the agent is to have `10` points. The agent is rational and if it believes a certain fact it does not pursue the formula as a goal.

### 3.2.2 Basic actions

The agent has certain capabilities to achieve a desired situation that are specified by basic actions. 2APL distinguishes six types of basic actions which together form an agent's plan.

- **Actions to update the belief base** are used to modify agent's beliefs. A belief update action consists of pre- and post-condition terms. An agent can execute the action if the pre-condition is entailed from its belief base.

  ```
  BeliefUpdates:
  {clock(Old)} UpdateClock(New) {not clock(Old),clock(New)}
  {points(A)} UpdatePoints(B) {not points(A),points(B)}
  ```

  In the example specification above we see that update of the clock `UpdateClock(New)` is executed when agent entails `clock(Old)` from its belief base. The post-condition then is a list of positive and negative literals. `not clock(Old)` removes old clock value from the belief base and then `clock(New)` is added. The belief updates do not change during the execution.

- **Actions to test the belief base** have the form B($\phi$) and are true if $\phi$ is entailed from the belief base. $\phi$ literals can include conjunction and disjunction operators. Similarly the **action to test the goal base** has the form G($\phi$) where $\phi$ consists of atoms with conjunction and disjunction operators. It is a goal query expression testing an individual goal in the goal base. During the (Prolog) execution the substitutions performed from left to the right.

- **Actions to manage goals** have two forms *adopt goal* and *drop goal*. When adopting a goal the agent can either add the new goal to the beginning of the goal base adopta($\phi$) or to its end adoptz($\phi$). When dropping a goal the agent can use three different forms: dropgoal($\phi$), dropsubgoals($\phi$) and dropsupergoals($\phi$) .

- **Abstract actions** are constructs similar to procedures in imperative languages. In 2APL these procedures are defined as `PC-rules`, which is an abbreviation for procedure call rules and will be described in a later section.

- **Communication actions** are used by the agents to pass messages to other agents. The *send action* has either three or five parameters.

  ```
  send(Receiver,Performative,Content)
  send(Receiver,Performative,Language,Ontology,Content)
  ```

  `Receiver` is the receiving agent, `Performative` is used to describe the type of the message (e.g., inform, request) and the actual message is placed in the `Content`. The optional `Language` and `Ontology` are used to give a context to the content where required. 2APL is built on FIPA compliant JADE platform therefore the the name of the receiving agent also supports a JADE name together with the local agent names. The format of a full JADE name has the form `localname@host:port/JADE`.

- **External actions** are used by the agents to interact with external environments. The agents are not able to determine the effect of external actions they execute beforehand because these are determined by the external environment. However, a programmer can supplement a sense action to mitigate this limitation. *external action* has the form `env(ActionName,Return)` where `env` is the agent's environment implemented as a Java class, `ActionName` a method call (of the Java class) and `Return` the resulting list of values possibly empty.

### 3.2.3 Plans

Plans are the tools of 2APL agents used to reach their goals. In 2APL agents' plan can be defined in advance however they may be changed during the execution of agents' programs with the use of planning goal rules, which will be described later in the section. A conditional operator has the form `if` $\phi$ `then` $\pi_1$ `else` $\pi_2$ where $\pi_1$ and $\pi_2$ are arbitrary plans. The condition is evaluated based on the agent's belief and goal bases. A while plan has the form `while` $\phi$ `do` $\pi$ where $\pi$ is an arbitrary plan and the condition is also evaluated in respect to the agent's belief and goal bases. The plan $\pi$ is then performed while the condition holds. 2APL also supports a non interleaving operator that has the form $[\pi]$ and denotes that the execution of the arbitrary plan $\pi$, which is called atomic should not be interleaved with the execution of actions of any other plans.

```
Plans:
[@environment(enter(green), L ); updatePos(L)],
send(admin,request,register(me))
```

The 2APL agent plans are stored in a plan base. The above example illustrates the initial plan base, which consists of two plans. The first is an atomic plan interacting with the external environment. At first the agent enters the environment as a green entity and immediately after updates its belief base with the position that was returned by the environment. The second plan consists of an a communication action, which sends a request to register to the administrating entity.

### 3.2.4 Practical reasoning rules

Practical reasoning rules are a programming language construct that are used to implement agents' ability to generate new plans. 2APL supports three types of rules: *planning goal rules*, *procedure call rules*, and *plan repair* *rules*, which are all explained in this section.

- **Planning goal rules (PG-rules)** are used for generation of plans when an agent has certain beliefs. The specification of the rule consist of three components: the head, the condition, and the body of the rule. The head of the rule (left side of the expression) represent a goal, which the planning rule relates to. The condition is evaluated in respect to the agent's belief base — considering agent's current beliefs. The planning rule is applied if the head and the condition can be entailed from the belief base and goal base, respectively. The application of the rule generates a substitution of variables that occur in the head and condition. The substitution is then applied to the body of the rule. In the example below is a planning goal rule with a plan to go to a position `(X2,Y2)` from a position `(X1,Y1)` and to remove trash. This plan can be generated if the agent has a goal to clean a space `R`, it believes it is currently at the position `(X1,Y1)` and that there is a trash at the position `(X2,Y2)`.

```
PG-rules:
        clean(R) <- pos(X1,Y1) and trash(X2,Y2) |
              {[goTo(X1,Y1,X2,Y2);
              RemoveTrash()]}
```

- **Procedure call rules (PC-rules)** can be used for two different purposes. Apart from the form of procedure definition the procedure call rule is also utilized to handle messages and external events. In this manner the rule is used to generate a plan as a response to a reception of a message from other agents, as a response to an event that was triggered by an external environment, and the execution of abstract actions. Similarly to the planning goal rules described above the rule consists of three elements. The head of the rule can be a message, an event or an abstract action. `message/3 (message/5)` and `event/2` are represented by special predicates while the abstract action is a standard predicate starting with a lower case letter. Like a planning goal rule the procedure call rule has a condition that needs to be entailed from agent's belief base for the plan to be generate from the rule.

```
PC-rules:
```

```
message(A,request,whereIsG) <- manager(A) and gold(X,Y) |
       {send(A, inform, gold(X,Y))}

event(gold(X2,Y2), blockworld) <- not carry(gold) |
       {getAndStoreGold(X2,Y2)}

getAndStoreGold(X,Y) <- pos(X1,Y1) |
       {[goTo(X1,Y1,X,Y);@blockworld(pickup(),_);PickUp();
       goTo(X,Y,3,3);@blockworld(drop(),_);StoreGold()]}
```

In the example above the first rules handle the receipt of a message that the type of request from an agent A. The plan containing response message is generated (and executed) if the agent A is a manager and the agent knows the position of gold. The second rule is activated when the external environment blockworld triggers an event with a position of gold. In this example the plan is generated if the agent does not carry gold at the moment. The plan contains an abstract action getAndStoreGold that represents a procedure call rule. The abstract action contains an atomic plan composed of a sequence abstract actions, external actions and belief updates, execution of which should not be interleaved with any other plans to prevent unfortunate undesired effects.

- **Plan repair rules (PR-rules)** are used for occasions when the execution of a plan fails. Like the previous rules the plan repair rule contains a head, a condition and a body. The rule can be applied if a plan that unifies the head abstract plan expression fails (the meaning of plan failure will be explained later) and the condition can be entailed from the agent's belief base. The following is an example of a plan repair rule, which indicates that if plan that contains a plan sequence @blockworld(south(),_);@blockworld(south(),_) fails while being executed it should be replaced with a plan to move to the east first, do two steps to the south and return a step to the west. When the execution of a plan is considered failed depends on the type of action.

```
PR-rules:
  @blockworld(south(),_);@blockworld(south(),_) <- true |
     { @blockworld(east(),_); @blockworld(south(),_)
```

```
    @blockworld(south(),_); @blockworld(west(),_) }
```

In the case of abstract action a plan fails when there is no applicable procedure call rule, in the case of belief update it is when the precondition of the update can not be entailed by the belief base, and in the case of external action is it either when the agent does not have access to the environment or the environment throws an exception `ExternalActionFailedException`. When the execution of a action fails the execution of the whole plan is blocked. The failed action is not removed from the sequence but left in place in order to be repaired.

### 3.2.5 External environments

2APL supports multiple external environment, which are made accessible to the agents as Java class. The class is required to implement the *environment interface*, which contains two basic methods *addAgent(String name)* to add an agent to the environment and *removeAgent(String name)* to remove an agent. The listener of external events *ExternalEventListerner* is passed to the constructor of the environment. The agent programs interface with the environment by executing an action that has the form `@env(m(`$a_1$`,...,`$a_n$`), R)`. Execution of the action calls a method `m` with arguments $a_1, \ldots, a_n$ in the environment `env`. In the first argument $a_1$ the method expects the identifier of the calling agent, which is used to pass back information the agent by the means of events. The return value `R` is passed back to the executing plan. The execution of the plan is blocked until this value is accessible to the agent program. The method can throw the `ExternalActionFailedException` which causes the corresponding external action to fail. In the following code snippet is the implementation of the move method.

```
public Term move(String agent, String direction)
   throws ExternalActionFailedException
{
   if (direction.equals("north") moveNorth();
   else if (direction.equals("east") moveEast();
   else if (direction.equals("south") moveSouth();
   else if (direction.equals("west") moveWest();
   else throw
```

```
    new ExternalActionFailedException("Unknown direction");
  return getPositionTerm();
}
```

### 3.2.6 Events and exceptions

There are two different ways to pass information between the environment and the agents via *events* and *exceptions*. The main use of events is to pass information from the environments to the agents. 2APL events are triggered by calling the method `notifyEvent(AF event, String... agents)` in the `ExternalEventListener` provided originally as an argument of the constructor. The method takes an atomic formula as the first argument and a list of agents affected in the second argument. When the event is received by an agent it is unified with a head of a procedure call rule. It is possible to implement the agents' perceptual mechanism by not supplying any agent names in which case the event is received by all agents. The exceptions are used to notify the agent that the execution of the external event was unsuccessful. However, there is no mechanism in place to pass exceptions from 2APL to the external environment.

### 3.2.7 2APL files

The agent programs are loaded from a file with `.mas` extension, which specifies the external environments and agent `.2apl` files.

```
<apaplmas>
  <environment name="blockworld" file="blockworld.jar">
  </environment>
  <agent name="harry" file="harry.2apl"/>
  <agent name="sally" file="sally.2apl"/>
</apaplmas>
```

2APL agents can share some of their initial beliefs, goals, plans, belief updates, and practical reasoning rules. This functionality is supported by the construct `Include: filename`.

## 3.3 N-2APL

N-2APL is a normative extension of 2APL introduced in Alechina et al. [2012], which enriches some of the programming constructs while it also limits or changes the semantics of others. This section will describe extensions made to 2APL. For the operational semantics of N-2APL please refer to Appendix E.

### 3.3.1 Beliefs, Goals and Events

Beliefs in N-2APL are the same as in 2APL and consist of Horn clause expressions. Goals in 2APL may be conjunctions of positive literals. In N-2APL goals are restricted to single atoms and their syntax is extended to include optional deadlines. A *deadline* is a real time value (expressed in milliseconds) that specifies the time by which the goal should be achieved. If no deadline is specified for a goal as part of the agent's program, we assume a deadline of infinity.

As described earlier norms are communicated to the agent in the form of events. An obligation event, represented as $obligation(\iota, o, d, s)$, specifies the time $d$ by which the obligation $o$ must be discharged, i.e., its deadline, and the sanction, $s$, that will be applied if the obligation is not discharged by the deadline. A prohibition event, represented as $prohibition(\iota, p, d, s)$, specifies a prohibition $p$ that should not be violated and the sanction $s$ that will be applied if execution of the agent's plans violates the prohibition.

Obligations are adopted as goals with a deadline corresponding to the deadline of the obligation. In N-2APL it is assumed that prohibitions have a deadline of infinity. In addition we extend the state of the agent to include prohibitions, which are represented by single atoms, and the agent's initial state is extended to include its initial prohibitions and obligations.

We assume the the programmer provides function $pref(x)$ where $x$ is a goal or prohibition that returns the priority of the goal or prohibition $x$. For non-normative goals, the priority corresponds to the importance of achieving the goal state. In the case of prohibitions and goals derived from obligations, the priority corresponds to the importance of avoiding the sanction that would be incurred if the corresponding norm is violated.

### 3.3.2 Actions & Plans

The syntax of external actions is extended to list the expected postconditions of the action, to allow the prohibitions violated by a plan to be determined.

In order to achieve its goals, an N-2APL agent adopts plans. A plan consists of basic actions composed by sequence, conditional choice, conditional iteration and non interleaving operators. The non interleaving operator, $[\pi]$ where $\pi$ is a plan, indicates that $\pi$ is an *atomic* plan, i.e., the execution of $\pi$ should not be interleaved with the execution of any other plan. Basic actions include external actions (which change the state of the agent's environment); belief update and goal adopt actions (which change the agent's beliefs and goals), and abstract actions (which provide an abstraction mechanism similar to procedures in imperative programming).

In N-2APL, non-atomic plans are the same as in 2APL. However in N-2APL we change the interpretation of the non interleaving operator: $[\pi]$ indicates that the execution of $\pi$ should not be interleaved with the execution of other *atomic* plans (rather than not interleaved with the execution of *any* other plan as in 2APL). In N-2APL, atomic plans are assumed to contain basic actions that may interfere only with the basic actions in other atomic plans. For example, a plan that involves moving to a new location should not be interleaved with other plans that change the agent's location. However, external actions in different non-atomic plans are executed in parallel, rather than being interleaved as in 2APL. Lastly, we restrict the scope of the non interleaving operator such that non-atomic plans cannot contain atomic sub-plans, either directly or through the expansion of an abstract action, i.e., plans to achieve top-level goals are either wholly atomic or non-atomic.

We extend the syntax of plans in the body of a PG rule to include an optional field specifying the time required to execute the plan proposed by the PG rule. For simplicity, we assume that the time required to execute each plan $\pi$ is fixed and known in advance.

The syntax of N-2APL is shown in Figure 3.4 in EBNF notation. For the EBNF of 2APL, please see Dastani [2008].

```
⟨Agent_Prog⟩       =  "Include:" ⟨includes⟩
                      | "BeliefUpdates:" ⟨beliefupdates⟩
                      | "Beliefs:" ⟨beliefs⟩
                      | "Goals:" ⟨goals⟩
                      | "Obligations:" ⟨obligations⟩
                      | "Prohibitions:" ⟨prohibitions⟩
                      | "Plans:" ⟨plans⟩
                      | "PG-rules:" ⟨pgrules⟩
                      | "PC-rules:" ⟨pcrules⟩
                      | "PR-rules:" ⟨prrules⟩
                      | "Preferences:" ⟨prefs⟩;
⟨includes⟩         =  ⟨include⟩+;
⟨include⟩          =  ⟨ident⟩ .2apl;
⟨beliefupdates⟩    =  ⟨beliefupdate⟩+;
⟨beliefupdate⟩     =  ( "{" ⟨belquery⟩ "}" ⟨beliefupdatename⟩ "{" [⟨literals⟩] "}" ) ;
⟨beliefupdatename⟩=   ⟨upperatom⟩;
⟨beliefs⟩          =  ⟨belief⟩+;
⟨belief⟩           =  ⟨ground_atom⟩ "."
                      | ⟨atom⟩ ":-" ⟨literals⟩ "." );
⟨goals⟩            =  ⟨goal⟩ { ","⟨goal⟩ } ;
⟨goal⟩             =  ⟨atom⟩ ":" ⟨deadline⟩ ;
⟨baction⟩          =  "skip"
                      | ⟨beliefupdate⟩
                      | ⟨sendaction⟩
                      | ⟨externalaction⟩
                      | ⟨abstractaction⟩
                      | ⟨test⟩
                      | ⟨adoptgoal⟩
                      | ⟨dropgoal⟩
                      | ⟨createaction⟩
                      | ⟨releaseaction⟩
                      | ⟨moduleaction⟩;
⟨plans⟩            =  ⟨plan⟩ { "," ⟨plan⟩ };
⟨plan⟩             =  ⟨baction⟩
                      | ⟨sequenceplan⟩
                      | ⟨ifplan⟩
                      | ⟨whileplan⟩
                      | ⟨atomicplan⟩
                      | ⟨scopeplan⟩;
⟨sendaction⟩       =  "send(" ⟨iv⟩ "," ⟨iv⟩ "," ⟨atom⟩ ")"
                      "send(" ⟨iv⟩ "," ⟨iv⟩ "," ⟨iv⟩ "," ⟨iv⟩ "," ⟨atom⟩ ")";
⟨externalaction⟩ =    "@" ⟨ident⟩ "(" ⟨atom⟩ "," ⟨var⟩ ")"
⟨test⟩             =  "B(" ⟨belquery⟩ ")"
                      | "G(" ⟨goalquery⟩ ")"
                      | ⟨test⟩ "&" ⟨test⟩
                      | "(" ⟨test⟩ ")";
```

```
⟨abstractaction⟩ =  ⟨atom⟩;
⟨adoptgoal⟩      =  "adopta(" ⟨goalvar⟩ ")"
                    | "adoptz(" ⟨goalvar⟩ ")";
⟨dropgoal⟩       =  "dropgoal(" ⟨goalvar⟩ ")"
                    | "dropsubgoals(" ⟨goalvar⟩ ")"
                    | "dropsupergoal(" ⟨goalvar⟩ ")";
⟨createaction⟩   =  "create(" ⟨ident⟩ "," ⟨ident⟩ ")";
⟨releaseaction⟩  =  "release(" ⟨ident⟩ ")";
⟨moduleaction⟩   =  ⟨ident⟩ "." ⟨maction⟩;
⟨maction⟩        =  "execute(" ⟨test⟩ ")"
                    | "updateBB(" ⟨literal⟩ ")"
                    | ⟨test⟩ "&" ⟨test⟩
                    | ⟨adoptgoal⟩
                    | ⟨dropgoal⟩
                    | "B(" ⟨literal⟩ ")"
                    | "G(" ⟨literal⟩ ")";
⟨sequenceplan⟩   =  ⟨plan⟩ ";"⟨plan⟩;
⟨ifplan⟩         =  "if" ⟨test⟩ "then" ⟨scopeplane⟩ [ "else" ⟨scopeplan⟩];
⟨whileplan⟩      =  "while" ⟨test⟩ "do" ⟨scopeplane⟩;
⟨atomicplan⟩     =  "[" ⟨plan⟩ "]";
⟨scopeplan⟩      =  "{" ⟨plan⟩ "}";
⟨pgrules⟩        =  ⟨pgrule⟩+;
⟨pgrule⟩         =  [⟨goalquery⟩] "<-" ⟨belquery⟩ "|" ⟨plan⟩ ":" ⟨duration⟩;
⟨pcrules⟩        =  ⟨pcrule⟩+;
⟨pcrule⟩         =  ⟨atom⟩ "<-" ⟨belquery⟩ "|" ⟨plan⟩;
⟨prrules⟩        =  ⟨prrule⟩+;
⟨prrule⟩         =  ⟨planvar⟩ "<-" ⟨belquery⟩ "|" ⟨planvar⟩;
⟨goalvar⟩        =  ⟨atom⟩ { "and" ⟨atom⟩ };
⟨planvar⟩        =  ⟨plan⟩
                    | ⟨var⟩
                    | "if" ⟨test⟩ "then" ⟨scopeplanvar⟩
                    ["else" ⟨scopeplanvar⟩]
                    | "while" ⟨test⟩ "do" ⟨scopeplanvar⟩
                    | ⟨planvar⟩ ";" ⟨planvar⟩;
⟨scopeplanvar⟩   =  "{" ⟨planvar⟩ "}";
⟨literals⟩       =  ⟨literal⟩ { "," ⟨literal⟩ };
⟨literal⟩        =  ⟨atom⟩
                    | ⟨infixatom⟩
                    | "not" ⟨atom⟩
                    | "not" ⟨infixatom⟩;
⟨belquery⟩       = "true"
                    |⟨belquery⟩ "and" ⟨belquery⟩
                    |⟨belquery⟩ "or" ⟨belquery⟩
                    | "(" ⟨belquery⟩ ")"
                    | ⟨literal⟩;
⟨groundatom⟩     =  ⟨ident⟩ "(" ⟨groundpars⟩ } ")";
⟨groundpars⟩     =  ⟨groundpar⟩ { "," ⟨groundpar⟩ } };
```

```
⟨iv⟩            =   ⟨ident⟩ | ⟨var⟩;
⟨goalquery⟩     = "true"
                    |⟨goalquery⟩ "and" ⟨goalquery⟩
                    |⟨goalquery⟩ "or" ⟨goalquery⟩
                    | "(" ⟨goalquery⟩ ")"
                    | ⟨atom⟩;
⟨groundpar⟩     = ⟨ident⟩ | ⟨num⟩ | "_" ⟨atom⟩
                    |"[" ⟨groundpars⟩ "]"
                    |"[" ⟨groundpars⟩ "|" ⟨var⟩ "]";
⟨upperatom⟩     =   ⟨var⟩ "(" [⟨pars⟩ } ]")";
⟨atom⟩          =   ⟨ident⟩ "(" [⟨pars⟩ } ]")";
⟨infixatom⟩     =   ⟨par⟩ ("=" | ">" | "<" | "<="
                    | ">=" | "=>" | "=<") ⟨par⟩;
⟨pars⟩          =   ⟨par⟩ { "," ⟨par⟩ };
⟨par⟩           = ⟨var⟩ | ⟨num⟩ | "_" ⟨atom⟩
                    | ⟨par⟩ ("+" | "-" | "*" | "/") ⟨par⟩
                    | "[" [⟨pars⟩] "]"
                    |"[" (⟨artexps⟩ "|" ⟨pars⟩) "|"⟨var⟩ "]";
⟨artexps⟩       =   ⟨artexp⟩ { "," [⟨artexp⟩ };
⟨artexp⟩        = ⟨var⟩ | ⟨num⟩
                    | ⟨artexp⟩ ("+" | "-" | "*" | "/") ⟨artexp⟩
                    | "(" [⟨artexp⟩] ")" ;
⟨var⟩           =   "A".."Z"
                    {"a".."z" | "A".."Z" | "0".."9" | "_" };
⟨ident⟩         =   "a".."z"
                    {"a".."z" | "A".."Z" | "0".."9" | "_" };
⟨num⟩           =   ("0".."9")+;
⟨obligations⟩   =   ⟨atom⟩ { ","⟨atom⟩ } ;
⟨prohibitions⟩  =   ⟨atom⟩ { ","⟨atom⟩ } ;
⟨plan⟩          =   ⟨atomic-plan⟩ | ⟨non-atomic-plan⟩;
⟨atomic-plan⟩   =   "["⟨non-atomic-plan⟩"]";
⟨prefs⟩         =   ⟨pref⟩ { ","⟨pref⟩ } ;
⟨pref⟩          =   (⟨atom⟩ | ⟨sanction⟩) "->" ⟨priority⟩;
⟨sanction⟩      =   ⟨atom⟩;
⟨deadline⟩      =   ⟨num⟩;
⟨duration⟩      =   ⟨num⟩;
⟨priority⟩      =   ⟨num⟩;
```

Figure 3.4: EBNF syntax of N-2APL

### 3.3.3 Implementation

Our implementation of 2APL was based on the implementation of 2APL developed at the University of Utrecht.[1] The extensions to the 2APL interpreter can be split into three main parts: modification of parser, extension

[1] The 2APL platform is available from apapl.sourceforge.net.

of the agent's state to include obligations and prohibitions, and changes to agent's deliberation strategy. The 2APL parser is implemented using JavaCC, and the modifications necessary to accommodate the extended 2APL syntax required a modification of the grammar specification. Obligation goals are stored in the existing 2APL goal base, and the original 2APL`Goal` class was extended to incorporate a deadline and a priority. Obligation deadlines are treated as relative times in milliseconds and transformed to goal deadlines (clock times) when the program is parsed (in the case of initial obligations) or when the obligation event is received from the normative organization. Prohibitions do not map to existing 2APL intentional attitudes. A prohibition base (set of states) was therefore added to record the agent's active prohibitions. The prohibition base is accessed by the N-2APL during the deliberation steps to assess whether the execution of an intention would violate a prohibition.

Significant changes to the 2APL deliberation strategy were required to take the priorities and deadlines of goals and prohibitions into account when deliberating about which plan to adopt for a goal and when to execute the plans to which it currently committed. The N-2APL deliberation strategy returns a schedule. A schedule is an assignment of a start or next execution time to a set of plans which ensures that: all plans complete by their deadlines, at most one atomic plan executes at any given time, and where the goals achieved and the prohibitions avoided are of the highest priority. Scheduling in N-2APL is pre-emptive in that the adoption of a new plan $\pi$ may prevent previously scheduled plans with priority lower than $\pi$ (including currently executing plans) being added to the new schedule. Plans that would exceed their deadline are dropped. In the case of obligations, a sanction will necessarily be incurred, so it is not rational for the agent to continue to attempt to discharge the obligation. In the case of goals, it is assumed that the deadline is hard, and there is no value in attempting to achieve the goal after the deadline. A plan which violates a prohibition of higher priority than the intention of the plan is dropped.

The deliberation strategy was modified so that after application of `PG-rules` the set of previously scheduled and newly generated plans are scheduled, and plans with a scheduled next execution time of 'now' are then executed.

Changes were also required to the execution of atomic plans. To allow the interleaved execution of an atomic plan with non-atomic plans (rather than executing all the steps of an atomic plan in a single step as in 2APL), atomic plans are transformed into sequence plans during parsing of the agent's program code and flagged as being atomic. The plan execution module was also changed so that external actions in non-atomic plans are executed in parallel.

The initial step in the development was to modify an existing interpreter of 2APL agent programming language so that it supports deliberation about norms that are in a form of obligations and prohibitions. We built upon suggested semantics for N-2APL by Alechina et al. [2012] and Figure 3.4 shows the newly modified agent programming language.

The parser of 2APL was developed with the JavaCC tool and therefore changes consisted of modifying the existing 2apl.jj file. Even though norm-aware agents are supposed to obtain obligations and prohibitions from the organization it is also possible to specify the norm straight away in the agent program. This was enabled for easier testing. Obligations are defined in the same format as goals, but include also a sanction. Prohibitions then consist of the forbidden state and an associated sanction. All possible sanctions need to be defined beforehand in the agent program even if the norms are provided by an organization later. Sanctions are also assigned numerical priorities where a lower number means a higher priority, also a goal can have an assigned priority. When none is specified it is assumed that the goal has the highest priority. Deadlines are treated as relative durations in milliseconds and transformed to real times at the time of parsing or event execution in the case that the obligation is received as an event from an organization.

Alechina et al. [2012] suggested that prohibitions would be added to the event base of agent module to minimize required changes. However this was not possible as events are treated as one off actions in N-2APL whereas prohibitions are required to remain active for the full time of their validity. Therefore a specific prohibition base was created. Obligations are added to the existing goal base as an extension of the *Goal* class. Sanctions are stored in the sanction base and for simplicity only the numerical priority values are used in Obligation and Prohibition objects as opposed to the linking of the whole sanction. The usage of prohibiting norms is limited now because all prohibited states need to be matched with specified belief updates. Also an agent is not able to recognize such a state when scheduling a plan, but only its during execution.

A major change was required to change the agent deliberation process to consider priorities and deadlines, and to schedule atomic and non-atomic plans separately. According to the scheduling algorithm which is detailed in Figure 3.5 an agent is allowed to execute only one atomic plan at a time but its execution is interleaved with execution of non atomic plans. Atomic plans are transformed into sequence plans when being parsed from agent's program code and flagged as atomic. The deliberation library was modified in a way that after application of PG rules all temporarily scheduled plans were sorted with the scheduling algorithm and then the executing module was modified so that atomic plans are no longer executed at once as in 2APL.

## 3.4 2OPL

As we explained earlier regulate the multi-agent systems with a normative institution. 2OPL [Dastani et al., 2009; Tinnemeier, 2011] is a programming language designed to support the implementation of normative multi-agent systems. 2OPL programs contain three types of data: facts, fact update rules, and norms. The facts and fact update rules are used to represent the state of the agents' environment and the effect of the agents' actions in the environment. For example, in a location based game, a fact may represent the current location of an agent, while a fact update rule represents how the agent's location changes as a result of performing a 'move' action.

We use norms that are state-based norms and are defined in terms of a unique label, an activation condition, and a deontic element. In what follows, we adopt the version of 2OPL described by Tinnemeier Tinnemeier [2011], which includes conditional norms with deadlines. The label functions as a name that can be used to refer to the norm and the precondition specifies when (i.e., in which states of the environment) the norm can be activated (detached). The deontic element of the norm is either an obligation or a prohibition. An obligation is defined by a subject (the agent to which incurs the obligation), a deadline, a state formula indicating the state of the environment that has to be brought about before the deadline, and a sanction formula indicating how the state is

Figure 3.5: N-2APL: Scheduling Algorithm

**Algorithm 1** Scheduling Algorithm

```
function SCHEDULE($\Pi, \Delta$)
    $\Gamma_s = \emptyset, \Gamma_p = \emptyset,$
    for all $\pi \in \Pi$ in descending order of priority do
        $V := effects(\pi) \cap \Delta$
        if $\neg atomic(\pi)$ then
            if $now + rt(\pi) \leq dl(\pi) \wedge$
              $pr(\pi) \geq \text{argmax } pr(p),\ p \in V$ then
                $ne(\pi) := now$
                $\Gamma_p := \Gamma_p \cup \{\pi\}$
            end if
        else
            if $executing(\pi)$ then
                $ne(\pi) := now$
                $\Gamma'_s := \Gamma_s$
            else
                $t := now$
                $\Gamma'_s := \emptyset$
                for all $\pi' \in \Gamma_s$ do
                    if $dl(\pi') \leq dl(\pi)$ then
                        $\Gamma'_s := \Gamma'_s \cup \{\pi'\}$
                        $t := \max(ne(\pi') + rt(\pi'), t)$
                    else
                        $ne(\pi') := ne(\pi') + rt(\pi)$
                        $\Gamma'_s := \Gamma'_s \cup \{\pi'\}$
                    end if
                end for
                $ne(\pi) := t$
            end if
            if $\forall \pi_i \in \Gamma'_s \cup \{\pi\}:\ now + \sum_{ne(j) \leq ne(i)} rt(\pi_j) \leq dl(\pi_i) \wedge$
              $pr(\pi) \geq \text{argmax } pr(p),\ p \in V$ then
                $\Gamma_s = \Gamma'_s \cup \{\pi\}$
            end if
        end if
    end for
    return $\Gamma_p \cup \Gamma_s$
end function
```

```
⟨norm⟩      =  “norm("⟨label⟩, ⟨subject⟩, ⟨policy⟩, ⟨precond⟩, ⟨deontic⟩“)"
⟨label⟩     =  ⟨atom⟩
⟨subject⟩   =  ⟨atom⟩
⟨policy⟩    =  ⟨atom⟩
⟨precond⟩   =  “("⟨literal⟩(“," ⟨literal⟩)*““)"
⟨deontic⟩   =  “obligation("⟨state⟩, ⟨deadline⟩, ⟨sanction⟩“)" |
               “prohibition("⟨state⟩, ⟨sanction⟩“)"
⟨state⟩     =  “["⟨atom⟩(“," ⟨atom⟩)* “]"
⟨deadline⟩  =  ⟨atom⟩
⟨sanction⟩  =  “["⟨atom⟩(“," ⟨atom⟩)* “]"
⟨literal⟩   =  ⟨atom⟩ | “not("⟨atom⟩“)"
```

Figure 3.6: EBNF of a 2OPL norm

```
norm(forbidden_area(Agent),
   (truck(Agent), forbidden(X,Y)),
   prohibition(Agent, [at(X,Y,Agent)],
          [reduce_score(Agent,500)])
).
```

Figure 3.7: Example GeoSense game norms

updated if the obligation is not discharged by the deadline. A prohibition is defined by a state formula indicating the state of the environment that must be avoided, and a sanction formula indicating how the state is updated if the prohibition is violated before the deadline. The subject and deadline are represented by atoms, and the state and sanction formulas are represented as conjunctions (lists) of atomic facts. For example, in the GeoSense game, a norm may prohibit a truck from entering a specific area, with violation of the norm resulting in a sanction of the truck's score being reduced by 500 points.

When a agent receives the tuple it adopts it as a new goal which gets the priority associated with the sanction. The relative deadline is translated into real time. If the agent's scheduling algorithm schedules this a plan associated with the goal the agent executes the plan and brings about the obliged state. Once the agent achieves the goal the state of the environment is updated in the tuple space and the changes are propagated to the normative organisation. When the deadline is due the organisation performs compliance check. In the event of the obligation being violated the organisation applies the defined sanction to the agent by modifying the state of the environment.

The syntax of 2OPL norms is shown in Figure 3.8 where the ⟨*atom*⟩ follows the Prolog syntax for atomic facts. All components of the norm

```
norm(forbidden_area(Agent),
    (truck(Agent), forbidden(X,Y)),
    prohibition(Agent, [at(X,Y,Agent)],
            [reduce_score(Agent,500)])
).
```

Figure 3.8: Example GeoSense game norms

must be ground when a norm instance is detached.

For integration with N-2APL agents, we require that 2OPL norms conform to a more restrictive syntax than that shown in Figure 3.8. In particular, we assume a global clock and require that deadlines are atoms denoting relative times after the time at which a norm is detached. We also require that prohibitions have a deadline of infinity. These restrictions are necessary to ensure that the normative reasoning of N-2APL agents remains tractable [Alechina et al., 2012]. In addition, to simplify the mapping from sanctions to the priorities N-2APL agents assign to goals (see below), we assume that sanctions are single atoms.

2OPL programs are executed by means of an interpreter that consists of a loop in which agents' actions are observed, the effects of the actions are realized by means of the fact update rules, and norms are processed. Norms are processed as follows. If the precondition of a norm holds, then an instance of the corresponding obligation or prohibition is detached (comes into effect). For all obligations that are already in effect, the 2OPL interpreter checks if the deadline is reached while the obliged state of the environment is not realized. Moreover, for all prohibitions that are already in effect it is checked if the prohibited state is realized. In such a case a violation has occurred and the state of the environment is updated with the corresponding sanction. For example, in the GeoSense game sanctions affect the game score. If an agent is sanctioned by $reduce\_score(500)$ the environment is updated with the agent's deducted score.

To support the integration of 2OPL with the framework, the 2OPL interpreter was extended to interact via a tuple space as described in Section 3.5. Facts describing the current state of the environment and agent actions are read from the tuple space, and when the precondition of a norm becomes true in the current environment, a norm instance (an obligation or a prohibition with a specified subject, deadline and sanction) is written into the tuple space. The subject agent receives a notification from the tuple space and retrieves the new norm.

## 3.5 Middle-ware

Interaction between the 2OPL normative organization and the N-2APL agents is via a tuple space Carriero and Gelernter [1989]. We choose a tuple space rather than message-based interaction primarily to facilitate the integration of non-agent-based components such as the Geo-Sense game server. Our primary aim was to support programming with norm-aware agents with the stress on interoperability and preservation of history of the state space, therefore tuple space paradigm has been chosen despite its possible performance factors such as implicit indeterminism and lacking search capabilities. The facts recording the current (brute) state of the multi-agent environment and the detached norms and sanctions comprising its normative state are represented as tuples. The agents are connected to the tuple space through an extension of the N-2APL`Environment` class and in an agent program the tuple space is accessed in the same way as any other external environment.[2] The normative organisation accesses the tuple space through Prolog queries that wrap native Java method calls to the 'Prolog to Java' middleware used by both the N-2APL `Environment` class and 2OPL (see Figure 3.15).

Both the normative organization and the multi-agent system synchronize with the tuple space. Using the `notify` method, the organization and the agents register to be notified when a new tuple matching a template is inserted into the tuple space. For example, agents register to receive notifications about all new obligation and prohibition entries assigned to them, and the normative organization registers to receive notifications when a new agent location tuple is created in the tuple space. As the agents and the normative organisation receive only those updates that are relevant to them, the overhead of the tuple space relative to a message passing implementation is minimal.

Tuples are stored as serialized Java `Entry` objects. Each type of tuple

[2] To simplify the implementation, in the current prototype the effects (postcondition) of agent actions are written directly to the tuplespace, and 2OPL fact update rules are not used. However it would be straightforward to delegate action execution to 2OPL.

is defined as a class that implements the `Entry` interface, and we defined a simple mapping from the Prolog terms used by 2OPL and N-2APL to `Entry` objects. JavaSpaces are non-deterministic and therefore all tuples need to be timestamped. Timestamps are implemented using a clock process which writes the current system time as a clock tuple in the tuple space. (In the example application described in the next section, the clock process is provided by the gameserver middleware, which writes a new clock tuple once a second.)

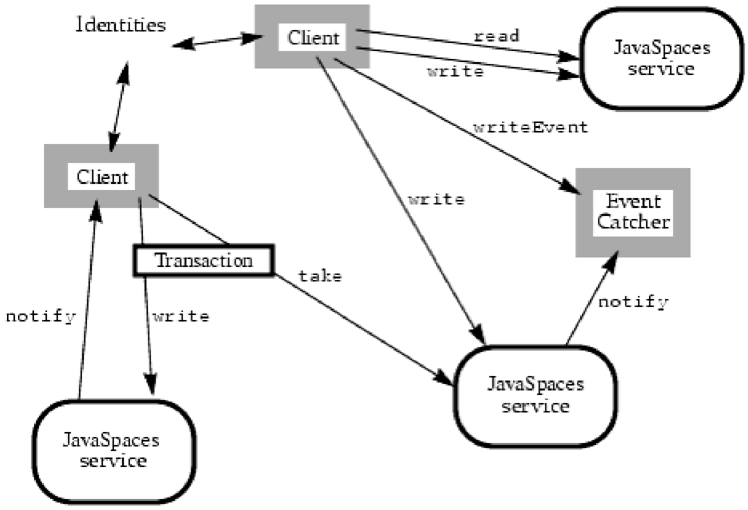


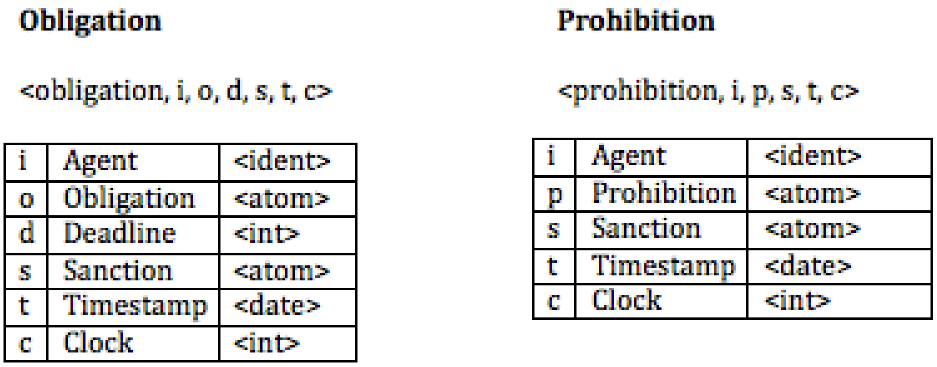
Figure 3.9: JavaSpace application example

Tuple space was picked as a coordination mechanism between the multi-agent system, normative organization and mixed reality game. JavaSpaces (JINI River Apache implementation) runs as a service and its schema is shown in Figure 3.9. Based on Linda communication and coordination model the system supports following actions:

- write - write a new entry into the space.
- read - read any matching entry from the space, blocking until one exists. Return null if the timeout expires.
- readIfExists - read any matching entry from the space, returning null if there is currently is none. Matching and timeouts are done as in read, except that blocking in this call is done only if necessary to wait for transactional state to settle.
- notify - when entries are written that match this template notify

the given listener with a RemoteEvent that includes the handback object.

- take - take a matching entry from the space, waiting until one exists. The also corresponding takeIfExists method is not being used, because no tuples are removed from the space at any time.



**Obligation**

<obligation, i, o, d, s, t, c>

| | | |
|---|---|---|
| i | Agent | <ident> |
| o | Obligation | <atom> |
| d | Deadline | <int> |
| s | Sanction | <atom> |
| t | Timestamp | <date> |
| c | Clock | <int> |

**Prohibition**

<prohibition, i, p, s, t, c>

| | | |
|---|---|---|
| i | Agent | <ident> |
| p | Prohibition | <atom> |
| s | Sanction | <atom> |
| t | Timestamp | <date> |
| c | Clock | <int> |

Figure 3.10: Entry objects for obligations and prohibitions

Entry is any serializable object that implements entry interface. Each type of tuple is implemented as a separate object. Limitations of the system lie in non-determinism - all tuples need to be timestamped and the latest one is found only by comparing all that are matching the template with a use of transactions. Also JINI does not support matching of regular expressions. Example of tuples is shown in Figure 3.10.

## 3.6 Application

We chose to illustrate the capabilities of the framework by an application motivated by the ORCHID disaster response scenario[3] in which we situate agents with flexible autonomy in a mixed reality game. A mixed reality game is considered a suitable example application because it is a shared environment requiring real time performance and is easily scalable. Its rules provide a variety of obligations and prohibitions to assess the expressiveness of the normative representative language and may be overridden. The norms do not constitute the game board (as in chess) but rather they define expected behaviour within the environment.

[3] http://www.orchid.ac.uk/disaster-response-2/

A scenario chosen for the initial prototype work was one of the early versions of AtomicOrchid called GeoSense Fischer et al. [2012a,b]. Geosense is developed on a base of location-based game MapAttack![4], which is using GPS in mobile phones to sense location of players and displays the state of game on a map in a web browser. In a version of the game which was used in this prototype there were three kinds of players: a radioactive truck, pursuers and a coordinator. The goal of the truck was to avoid pursuers while pursuers were aiming to capture it.

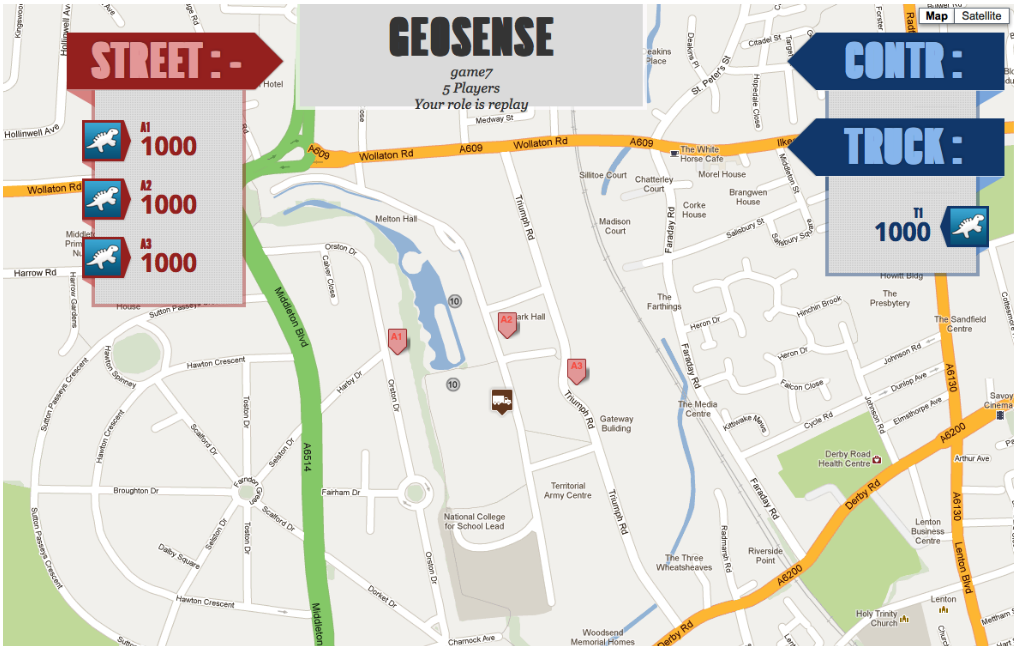


Figure 3.11: GeoSense web interface

### 3.6.1 Game Play

The GeoSense game is played on a map of a physical location (typically part of a city such as a park) and has three kinds of players: a truck, pursuers and coordinators. The truck carries a load of radioactive waste, and attempts to avoid detection. The pursuers, assisted by the coordinator(s), attempt to determine the location of the truck. The physical (GPS) locations of the pursuers are shown on the map and updated as the players move in the real environment.

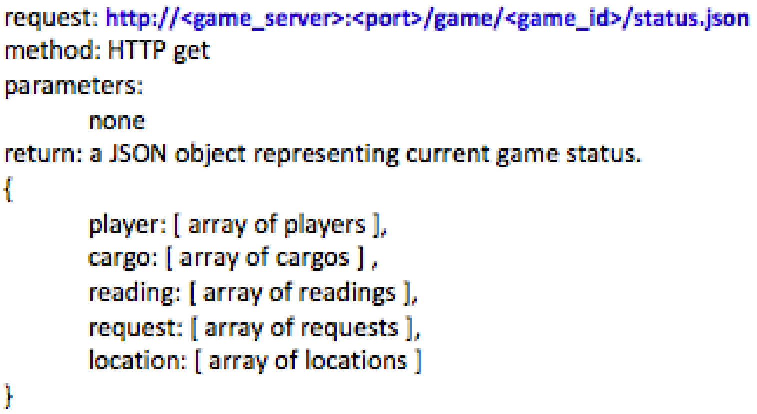
The pursuers' locations are visible to each other and to the coordinator(s). The truck is a virtual player, and its location is not visible on the map. However its radioactive load leaves a virtual 'trace' that can be

[4] http://mapattack.org

measured by by taking a 'reading' at a pursuer's current physical location. The reading ranges from 0 to 100, with higher readings indicating a smaller distance to the truck. In an attempt to avoid detection, the truck may drop some of its load as it moves through the game area. Such dropped waste also gives a positive reading, making it more difficult for the pursuers to determine the location of truck.

The coordinator(s) have a global view of the positions of all the pursuers and of all recent readings. The role of the coordinator is to aid the pursuers by directing them to promising areas of the map. The coordinator can request that a pursuer takes reading at a particular physical location by placing a virtual 'coin' at the location on the map. The pursuer must then go the physical location indicated by the coin and take a reading. In the original game scenario all users were human actors. The initial step in the implementation was to develop a normative multi-agent system, connect it to the game and replace human players with agents.

The location based game GeoSense is developed in Ruby and runs as a web server. Clients can connect to the server through `HTTP` or `Socket.IO` interfaces. Clients are either a mobile device for a pursuer or a web browser for a coordinator. In our system HTTP requests were more convenient. Java middle-ware connecting the game server and the tuple space is in charge of the system clock, which ticks every 500 millisecond. The speed is set as a constant and can be easily changed to suit needs.



```
request: http://<game_server>:<port>/game/<game_id>/status.json
method: HTTP get
parameters:
        none
return: a JSON object representing current game status.
{
        player: [ array of players ],
        cargo: [ array of cargos ] ,
        reading: [ array of readings ],
        request: [ array of requests ],
        location: [ array of locations ]
}
```

Figure 3.12: An example of a HTTP request and a response in JSON format

On every tick the whole server state (see Figure 3.12.) is synchronized with the tuple space. As with the organization and agents, the game server is registered to be notified about updates in the tuple space for

```
norm(forbidden_area(Agent),
     (truck(Agent), forbidden(X,Y)),
     prohibition(Agent, [at(X,Y,Agent)],
                  [reduce_score(Agent,500)])
).

norm(take_reading(Agent),
     (pursuer(Agent), coin(X,Y,Agent), clock(Now)),
     obligation(Agent, [reading(X,Y,Agent)],
                 Now + 15000, [reduce_score(Agent,300)])
).
```

Figure 3.13: Example GeoSense game norms

example locations. Locations of the players are transformed to a grid form geographic latitudes and longitudes used in the game. The web interface of the game is illustrated in Figure 3.11.

### 3.6.2 Encoding Game Rules as Norms

The rules of the GeoSense game are encoded in the gameserver code and are not accessible to agents. To allow agents to participate in the game, we re-expressed the GeoSense game rules as a set of 2OPL obligations and prohibitions. The norms specify which game states the agents should try to bring about (and by when) or are prohibited from bringing about, and any sanction incurred if the norm is violated, e.g., a deduction in points. For example, a norm may specify that the truck is prohibited from entering a particular area of the map, and that violation of the norm results in the loss of 500 points. (Note that a norm-aware agent may still choose to violate a norm e.g., the agent may enter a prohibited area if doing so allows it to win the game.) Updates to the game state resulting from agent actions may trigger norms that apply to the agent that performed the action or another agent. For example, when a coordinator places a coin for a pursuer, the normative organisation creates an obligation that the pursuer must take a reading at the location of the coin within a specified time, and a prohibition specifying that the coordinator cannot place another coin at the same location. Example 2OPL game norms are illustrated in Figure 3.13.

```
Beliefs:
  points(1000).
  position(19,19).
  clock(0).

Goals:
  at(2,2) : 120000,
  dropLoad : 60000

Preferences:
  at(2,2) -> 3,
  reduce_score(truck, 500) -> 4,
  dropLoad -> 5

PG-rules:

at(X,Y) <- true | { moveTo(X,Y); } : 60000
dropLoad <- position(10,10) | { drop(X,Y); } : 1000
```

Figure 3.14: N-2APL program for the Truck agent

### 3.6.3 Agent Programs

We also developed 2APL programs to allow the agents to play the game and achieve the goals resulting from the game norms. As an example, a program for a simple truck agent is shown in Figure 3.14. The truck has two goals. The first goal `at(2,2) : 120000` is to reach position `(2,2)` in 2 minutes (120,000 msecs) from the start of the game and has a priority of 3. The second goal `dropLoad : 60000` is to drop (part of) its load within one minute of the start of the game, and has a priority of 5. When the agent adopts a goal it executes the matching PG-rule. For example, the rule to achieve the `at(X,Y)` goal specifies a plan that involves moving to the required position. The PG-rule also includes an estimate of the time required to execute the plan (one minute in this case).

The obligations and prohibitions the agent receives as a result of the game rules may conflict with its own goals in the game. For example, the agent's goals to be `at(2,2)` or to drop part of its load when at position `(10,10)` may require visiting a prohibited area of the map. In such a situation, a norm-aware agent must choose between its existing goals and the norms imposed by the game. Critically, a 2APL agent is able to violate norms (accepting the resulting sanctions) if it is in the agent's overall interests to do so. For example, the truck agent assigns a higher priority to achieving the goal `at(2,2)` than to the sanction resulting from entering the prohibited area (losing 500 points), which in turn has a higher priority than the `dropLoad` goal. The agent will therefore enter the prohibited area if it is necessary to reach `(2,2)` but would not violate the norm to drop part of its load.

### 3.6.4 Gameserver Integration

To maintain the game state (and allow future participation by human players), we integrated the GeoSense gameserver with our normative programming framework consisting of 2OPL, 2APL and JavaSpaces. GeoSense is connected to the framework through the tuple space as shown in Figure 3.15. Updates to the tuple space corresponding to player actions are converted to HTTP `POST` requests to the game server. For example when a pursuer agent updates its location, the move action adds a new tuple to the tuple space, which is sent as a `POST` request to the gameserver. Similarly, the JSON updates generated by the gameserver used by the smart phone mobile clients are converted into tuples in the tuple space.

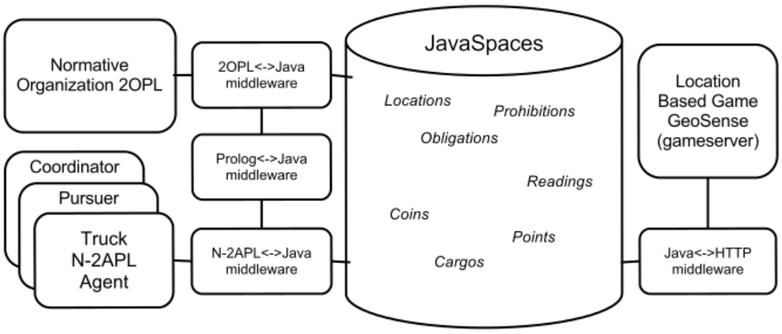


Figure 3.15: Overall system architecture

To simplify development of the agent programs, the tuple space to HTTP middle-ware makes some the aspects of the game state discrete. For example, the locations of the players are represented as longitude and latitude pairs by the gameserver, while the agents see the game environment as a grid and move one cell at a time. Similarly, the real-time clock used by the gameserver to record the progress of the game is seen as a series of one second ticks by the agents. However these simplifications do not affect game play and are not inherent in the normative programming framework itself.

The agents' beliefs and actions are synchronized with the game state via the tuple space, allowing them to participate in the game. Moreover the actions of the agents are coordinated and regulated through the norms that implement the game rules.

### 3.6.5 Evaluation

In this developmental stage of the framework we achieved to describe rules of GeoSense game as norms and seen software agents play a virtual game.

After the initial development we decided not to proceed with GeoSense application in our framework due to unreliability of used technologies particularly GPS. The long term objective of integrating our normative programming framework with the game is to investigate the use of norms as means of coordinating human-agent interaction in human agent collectives — systems which involve both human and agent participants. However the version of the application described in the chapter involves only software agents. During experimental studies with GeoSense and later Atomic Orchid ran by Wenchao Jiang we observed high occurrence of GPS location errors which would make coordination with agents very challenging as it would add another layer of confusion for both human and agent players. Location based game relies on correct locations, which provide context for the rules described as norms.

## 3.7 Summary

We described a framework for programming norm-aware multi-agent systems which integrates the N-2APL norm-aware agent programming language with the 2OPL language for programming normative organisations. To the best of our knowledge, this is the first implementation of an integrated framework for norm-aware multi-agent systems in which autonomous agents deliberate about whether to conform to the norms imposed by a normative organisation. We first described the syntax of the N-2APL agent programming language which is a normative extension of 2APL and together with 2OPL forms a normative multi-agent system. This has allowed us to describe a real world example as a normative environment. Next, we have described the implementation of the middleware that allows us to practically introduce flexibly autonomous agents into the existing game system. The application of our framework in the location-based mixed reality game GeoSense illustrates its flexibility. Furthermore, game rules can be expressed conditional norms with deadlines and sanctions, and agents can deliberate about their individual goals and the norms imposed by the game.

# 4

# Group Norms and Human-Agent Collectives

In this chapter we introduce the notion of a group norm as a tool to aid flexible coordination in human-agent collectives (HACs). In the previous chapter, we discussed the implementation of norm-aware agents that are able to deliberate about norms, which took a step closer to the flexible autonomy in HACs by adjusting the level of autonomy depending on context. The next challenge lies in team coordination, where there is a need for a flexible autonomy. Situations arise when either human or agent autonomy need to be restricted while the autonomy of other team members is increased. For example, in a disaster scenario where first responders need to react to quickly changing situation, the norms can be used to enable flexible autonomy in the team by trade offs between going to locations or rescuing people. While HACs could be formed and disbanded in a flexible manner, our proposed solution is not effective in an open system where agents can leave to prevent being sanctioned.

## 4.1 Introducing Group Norms

In this chapter we address some of the limitations of the implementation presented in Chapter 3. For example, our approach there assumes that the normative organisation assigns norms and sanctions to individual agents. While this is appropriate for many applications, there are situations where it would be more natural to address norms and sanctions to a group of agents. For example, a coordinator agent may create an obligation that some pursuer agent take a reading at a particular location without specifying which agent should do so; if none of the agents discharge the obligation by the deadline, the normative organisation applies a sanction to the pursuers as a group. In the following we look at extending our framework to incorporate group norms and sanctions.

We propose hierarchical group norms that are able to capture complex situations where a number of agents have to cooperate together to accomplish series of tasks. Such norms may be used, for example, to describe the rules of a mixed reality location based game. In this game the players, who are a mix of humans and software agents need to work together in groups, which are formed dynamically. The normative organisation issues a norm to the representative agent of the group. We call this agent the responsible agent. This agent has a policy that specifies how this group norm is treated, and according to this policy it creates individual norms for agents, who are involved in the fulfilment of the obligation. The responsible agent is responsible for the discharge of the group obligation. If the obligation is violated it receives a sanction to apply to others. According to its policy the agent then sanctions disobeying agents who contributed to the violation of the norm. For example, lets assume that an object can be collected only when a group of players carry the object together. The responsible agent of the group receives an obligation to collect the object. Its policy contains a team plan for a group object collection, which is a set of individual norms for the group members. Each member receives an obligation to move to the location, pick up the object and carry it with the group to the drop off location.

As we have already seen, in the form of rules and codes, norms are widely used to coordinate and regulate activity in human organisations, and more recently they have also been proposed as a coordination mechanism for multi-agent systems (MAS). Norms therefore have the potential to form a common framework for coordination and control in HACs. As with individual norms, to implement a group norm we need to synchronise the languages of obligations and prohibitions as they are used in human and agent environments, which is described in Section 4.2 and define the concept of the group norm, which is described in Section 4.7. The chapter conclude with illustrative examples in Section 4.9.

## 4.2 Taxonomy of Group Norms

Group, or collective, norms are a relatively new area of research and we have already seen some of this in Section 2.5. Although there is a wide body of research on norms, research has rarely paid attention to a formal treatment of norms aimed at groups of individuals [Aldewereld et al., 2013, 2015]. We build upon Aldewereld et al. proposed formal representation with the aim of understanding the meaning of norms as they appear in human society. The group norms are categorisable and may be intended to affect all, each or some members of the group[Aldewereld et al., 2013, 2015]:

- *Addressees* describe to whom the normative statement is addressed.
- *Actors* describe who should achieve or avoid the state that the norm refers to.

  Actorship then can be viewed on three levels in respect to group norms:

  - *Individual actorship* - individuals are each enacting their part in fulfilment of the norm.
  - *Collective actorship* - the whole group is fulfilling the norm. We additionally see the difference between individual and collective actorship in the level of coordination that is required from the group members.
  - *Representative actorship* – a previously appointed member or a subgroup is responsible for the enactment of the norm.
- *Responsibles* describe who takes care that the norm is upheld, and will be sanctioned if the norm is violated.

Responsibility then can be viewed on three levels in respect to group norms:

- *Individual responsibility* - individuals are each responsible for their part in fulfilling the norm.
- *Collective responsibility* - the whole group is held accountable for violations of the norm.
- *Representative responsibility* – a previously appointed member or a subgroup is responsible for violations of the norm.

| | | Responsibility | |
|---|---|---|---|
| Actorship | Individual | Representative | Collective |
| Individual | (1,1) Individual action specified in a generic (role-based) way: "When a coin is placed the pursuers should surround the location from north, east, south and west." | (1,2) Individual action, appointed blame: "The coordinator is responsible for coordinating agents to surround a coin." | (1,3) Individual action, collective blame: "To capture the truck the pursuers should surround the truck from at least three directions." |
| Representative | (2,1) Appointed action, individual blame: "Readings are only valid if the agent who reached the location made the first reading." | (2,2) Appointed action, appointed blame: "A reading needs to be taken by an agent at a request of the coordinating agent." | (2,3) Appointed action, collective blame: "The truck can be captured only by the agent with the most points." |
| Collective | (3,1) Collective action, individual blame: "Agents need to carry a radioactive load together to dispose of it." | (3,2) Collective action, appointed blame: "Coordinator should coordinate agents not to be further apart than 5 grid tiles from the closest agent." | (3,3) Group action, group blame: "Agents in the group should not be further apart than 5 grid tiles from the closest agent." |

Table 4.1: Taxonomy of group norms, based on responsibility and fulfilment

Using the above specified dimensions of actorship and responsibility there are 9 types of group norms [Aldewereld et al., 2013, 2015]. These are defined below with an overview in Table 4.1 with examples. We have omitted the notion of addressees and assume that a norm is addressed to the agents that are responsible. The cases are illustrated on examples from the mixed-reality game GeoSense, which the reader became familiar in Section 3.6.

- **Individual actorship**

  The straightforward case is the case (1,1) with individual actorship and responsibility. An example from the game can be that pursuing agents need to move to a certain location. Each agent will move to the specified position by itself and would violate the norm if it failed to comply.

Case (1,2) with individual actorship and representative responsibility is where a selected agent (or subgroup) is to blame. In the example of agents travelling to a location, a representative agent will be blamed if they fail to arrive at the position.

Case (1,3) with individual actorship and collective responsibility where the whole group is accountable for a violation. For example, the agents have to travel to a location similarly as in the previous case. If they fail, they will be blamed as the whole group. Where as we have seen in case (1,2) the representative agents was to blame.

- **Representative actorship** Representative actorship requires a selected agent (or subgroup) to fulfil the obligation. For example, the group of agents consists of explorers and carriers and only the explorers have to travel from a location A to a location B.

  Case (2,1) combines representative actorship with individual responsibility. If the pursuers failed to move from the location A to the location B together each of them would be individually responsible.

  In the case (2,2) with representative actorship and representative responsibility a selected agent (or subgroup) is responsible for the violation.

  Case (2,3) involves representative actorship and collective responsibility. In this case if the agents violated the norm, the whole group would be blamed.

- **Collective actorship** Collective actorship requires agents to coordinate their actions while fulfilling the norm. For example, agents have to move from location A to a location B together.

  Case (3,1) combines collective actorship with individual responsibility. If the agents failed to move from the location A to the location B together each of them would be individually responsible.

  In the case (3,2) with collective actorship and representative responsibility a selected agent (or subgroup) is responsible for the violation.

  Case (3,3) involves collective actorship collective responsibility. In this case if the group violated the norm, they would be blamed as the whole group.

Authors of the original taxonomy [Aldewereld et al., 2013] do not explicitly consider the concept of group sanctions. In the following we explain how the group sanction looks and how it is applied. We discuss this sanctioning in detail in Section 4.3. In the following example we illustrate the above taxonomy on an example from the GeoSense game. In the game the pursuing agents are trying to locate a invisible radioactive truck. Agents make a reading to find out how close the truck is. The coordinating agent directs agents to a location by placing a coin on the map. To confuse the players the truck can drop a radioactive load to mislead the pursuing agents.

1. Individual Actorship: individual, representative and collective responsibility differs in how the sanction is distributed. In the first case the sanction is individual for each agent. In the second case the representative agent is sanctioned and in the last case the sanction will be applied to the group.
   - Case (1,1) Individual Actorship, Individual Responsibility. *When a coin is placed the pursuers should surround the location from north, east, south and west.*
   - Case (1,2) Individual Actorship, Representative Responsibility. *The coordinator is responsible for coordinating agents to surround a coin.*
   - Case (1,3) Individual Actorship, Collective Responsibility. *To capture the truck the pursuers should surround the truck from at least three directions.*
2. Collective actorship: collective actorship differs from individual actorship in the form the task needs to be performed. Collective actorship requires higher coordination of actions amongst the team.
   - Case (3,1) Collective Actorship, Individual Responsibility. *Agents need to carry a radioactive load together to dispose of it.*
   - Case (3,2) Collective Actorship, Representative Responsibility. *Coordinator should coordinate agents not to be further apart than 5 grid tiles from the closest agent.*
   - Case (3,3) Collective Actorship, Collective Responsibility. *Agents in the group should not be further apart than 5 grid tiles from the closest agent.*

3. Representative actorship: representative actorship can be seen as a special case of collective actorship, where only a agent (or subgroup) of the agents is fulfilling the norm.

   - Case (2,1) Representative Actorship, Individual Responsibility. *Readings are only valid if the agent who reached the location made the first reading.*
   - Case (2,2) Representative Actorship, Representative Responsibility. *A reading needs to be taken by an agent at a request of the coordinating agent.*
   - Case (2,3) Representative Actorship, Collective Responsibility. *The truck can be captured only by the agent with the most points.*

The taxonomy described above is helpful guide when considering monitoring and enforcement of the norms, which is not taken into account in [Aldewereld et al., 2013]. Cases (1,1), (1,2) and (1,3) that have individual actorship can be monitored by the same entity that is capable of monitoring individual norms. Similarly, cases (1,1), (2,1) and (3,1) that have individual responsibility can be enforced by the same entity that is capable of enforcing individual norms. The remaining cases have more complex requirements. The enforcement and monitoring can be performed by different parts of the system. In Section 4.3 we discuss the requirements for both.

## 4.3 Extensions of Aldewereld et al

Aldewereld et al. [2013] provide a conceptualization of group norms. We have extended this into a taxonomy that is helpful when considering monitoring and enforcement of the norms. Aldewereld et al., however, do not consider the issues of sanctions and deadlines, which are core parts of norms as identified in Chapter **??**. In this section we discuss approaches that address these limitations.

Following the taxonomy described in the Section **??**, we might consider how different types of blame can be applied to the different types of responsibility based on the type of the norm.

In the specific case where the team shares the responsibility (collective responsibility) the sanction should be divided in a fair manner [Cholvy and Garion, 2007; Sergot, 2008; Grossi et al., 2007]. There are a number of alternative approaches to the problem of fair division of the sanction, which differs on the level of observability in the system. Observability relates to ability of the norm enforcing mechanism to view the actions of agents that preceded to fulfilment or violation of a norm.

- **No observability** In an environment with no observability of the actions of the agents the group sanction is applied to the group evenly because it is not possible to judge which agents contributed to the violation. The proposed system of group norms assumes that there is a possibility to enforce individual norms, which implies that there is at least a partial observability.
- **Partial observability** In an environment with a partial or limited observability, it is possible to decide which agents directly contributed to the violation. For example, they did not fulfil their individual obligation associated with the group norm. Such a system can manage cases with individual actorship (cases (1,1), (1,2) and (1,3)) and some cases of representative actorship where the representative is a single agent or the activity performed is not joint (cases (2,1) and (2,2)). The group sanction is then either applied whole or split evenly amongst the agents who are partly responsible for the violation. In the case of collective actorship only the norm with representative responsibility (case (3,2)) can be managed in this environment. Cases (3,1) and (3,3) when the sanction is applied on the individual and group levels respectively, the system does not have enough information about the cause of the violation. Collective actorship assumes joint coordinated activity towards fulfilment of the goal.
- **Full observability** In the case of an environment with full observability of agents' actions, it is possible to determine weights of each agents responsibility of the norm violation. Such a system can capture even situations when an agent contributed to a violation indirectly or credit agents who were prevented from fulfilling their part due to external circumstances. In the environment with partial observability the norm monitoring entity cannot track all events that might have influenced the outcome. For example, when a group is obliged to carry a dangerous object out of the field and one of the carriers suddenly leaves the followed path and proceeds in the

opposite direction. In this case the norm enforcer can blame the one agent for the failure of the whole group and punish him with the full value of the sanction.

We propose two types of sanctions (penalties), which are numerical and non-numerical.

- *Numerical* sanctions are the simplest kind of penalties. They are implicitly splittable amongst the group and the agents (and therefore the agent programmers) can directly associate the sanctions with their priorities. This type of sanction is possible to apply in all cases of the norms.
- *Non-numerical* sanctions can be classified as splittable and unsplittable.
  - Splittable sanctions are either formed of several parts (i.e. consist of a list of sanctions each of which can be applied separately) or can be divided into individual sanctions. Similarly as for the numerical sanction, this type of sanction is possible to apply in all cases of the norms.
  - Unsplittable sanctions can be applied as a whole to either each of the agents, a subgroup or only one the group members depending on the type of the sanction. These sanctions are suitable for norms with collective responsibility (cases (1,3), (2,3) and (3,3)) where sanction is applied to the whole group and for the norms with representative responsibility (cases (1,2), (2,2) and (3,2)) where the sanction is applied either representative agent or a subgroup. In the cases with individual responsibility (cases (1,1), (1,2) and (1,3)) unsplittable sanction is unsuitable.

When considering deadlines in group norms we need to take into account how the deadline of the group norm transforms into individual deadlines affecting agents. In cases with individual actorship and some cases of representative actorship (cases (1,1), (1,2), (1,3), (2,1) and (2,2)) we need to consider whether all individual norms will have have the same deadline or will there be a sequence of norms. In cases with collective actorship (cases ((3,1), (3,2), (3,3) and (2,3)) we assume that the individual deadlines are interlinked to a higher degree.

## 4.4 Formalising Group Norms

We define a conditional group obligation as a tuple

$$\langle l, C(n,b), O(\gamma, o, d, s)\rangle$$

with the intuitive reading "norm with a label $l$ states: if condition $C(n,b)$ where $n$ represents normative facts and $b$ represents brute facts holds in the current state of the environment then there is an obligation for agents in the group $\gamma$ to establish an environment state satisfying $o$ before deadline $d$, otherwise agents belonging to group $\gamma$ will be sanctioned by updating the environment with $s$. This definition covers all cases of the group norm taxonomy described in Table 4.1. Compared to the individual obligations introduced in Section 2.4.2 there are the following differences: (1) the condition $C(n,b)$ has been split into a normative and brute part; (2) $\gamma$ stands for a group or individual; (3) Obligation $o$ can be addressed to a group or individual. The brute element represents the original pre-condition that relates to the state of the environment. The normative element relates to a parental norm. It is utilised to specify this example group norm is itself part of a higher-level group norm. We will explain the group norm dependencies below. In the location based game example an obligation to surround a coin would be represented as seen in Figure 4.1, which indicates that when there is gold $object(Coin)$ group $Pursuers$ is obliged to surround the coin in 3 $minutes$. Violating this norm results in the pursuers being sanctioned with 300 $points$, damaging their game score.

```
<surroundCoin,                                  //label
 C(none,group(Pursuers) and object(Coin)),      //precondition
 O(                                             //deontic part
   Pursuers,                                    //addressee
   surround(Coin),                              //environmental state
   3 minutes,                                   //relative deadline
   loosePoints(300)                             //sanction
  )>
```

Figure 4.1: Example of group obligation.

Similarly, a conditional prohibition is expressed as a tuple

$$\langle l, C(n,b), F(\gamma, p, s)\rangle$$

with the intuitive reading "norm with a label $l$ states: if condition $C(n, b)$ where $n$ represents normative facts and $b$ represents brute facts holds in the current state of the environment, then it is forbidden for agents in group $\gamma$ to establish an environment state satisfying $p$, otherwise sanction $s$ will be imposed." Unlike obligations, where a sanction is incurred once if the obligation is not discharged by the deadline, in the case of prohibitions, the agent incurs a sanction each time the prohibition is violated. In Figure 4.2, which indicates that group *Pursuers* is prohibited from entering water together. Violating this norm results in the group pursuers being sanctioned with 500 *points*.

```
<enterWaterTogether,                //label
C(none,group(Pursuers)),            //precondition
F(                                  //deontic part
  Pursuers,                         //addressee
  inWaterTogether,                  //environmental state
  loosePoints(500)                  //sanction
 )>
```

Figure 4.2: Example of group prohibition.

## 4.5 Team Plan

Group norms can be fulfilled by a sequence of individual actions. These individual actions can be mapped to individual norms. We call the sequence of individual norms, which entails the fulfilment of the group norm, a *team plan*. While there possibly are many options how to translate the group norm in to a sequence of individual norms, the team plan has to satisfy the minimum (or weakest) requirement for the fulfilment of the group norm. This means that when individual agents fulfil their individual norms they are contributing to the fulfilment of the group norm. If all agents fulfil their individual norms the group obligation will be fulfilled. Team plan is defined as

$$teamPlan(group, norm, \langle individual\_norms \rangle)$$

where *group* stands for the group the norm is addressed to. *norm* is a label of the group norm and $\langle individual\_norms \rangle$ is a set of individual norms.

Translation of a group norm into a team plan should be done in a way such that each agent is capable of fulfilling their individual norm. To form a team plan the membership of the group to which the norm is addressed must be known. We may also need to consider what must be agreed upon beforehand, for example whether agents should specify what kind of capabilities they posses and what kind of sanctions are applicable to them.

Deadline of a group norm needs to be respected and deadlines of all individual norms have to be scheduled on or before the group deadline. In some cases a deadline of individual norm can precede another individual norm if its fulfilment is necessary for satisfaction of precondition of the subsequent norm.

The team plan is hierarchical tree like structure formed from norms. We assume that every group norm is divided into a finite set of individual and group norms, which fulfilment will lead to the fulfilment of the original group norm. It is the responsibility of the responsible agent to create a team plan and assign individual members (or subgroups) of the group to each new norm.

### 4.5.1 Sanctioning Policy

Together with the division of group norm we have to divide a group sanction into a set of individual sanctions. We call this division a *sanctioning policy*. The policy has the form

$$policy(norm, sanction\_policy)$$

where *norm* stands for the label for the group norm the policy relates to and
*sanction_policy* stands for the policy that will be applied in case of norm violation. The complexity of the policy depends on observational possibilities in the environment. For example, the policy specifies that the sanction will be split evenly between the agents who violate the norm.

There are a number of issues to consider regarding the division of the sanction.

**How to divide the sanction fairly?** We have previously discussed how different levels of observability in the environment affect the amount of fairness that is achievable. By fairness we mean that is desirable to punish agents with a sanction that is proportionate to their actions.

**What is the relationship between the sanctions?** What is the relationship between the group sanction and the set of individual sanctions — do they have to be equal or greater in total?

In our proposal we assume that individual sanctions are equal to a group sanction.

**What are the constraints when creating individual sanctions for agents?** The sanctioning policy is created by a responsible agent (Section 4.6). Therefore it needs to be ensured that the responsible agent creates the policy in the interest of the cooperation of the whole group and does not disadvantage or favour individual agents.

## 4.6 Responsible Agent

Fulfilment of a group norm by a group of agents implies that the norm may be divided into individual norms for each member of the group (or a subgroup). We propose the concept of a *responsible agent* in the group, which will have the responsibility to split the group norm into individual norms that are directed to the group members or a subgroup.

The responsible agent is responsible for the coordination of the group activity. The agent has two means of doing this: a team plan and a sanctioning policy. The team plan specifies a set of norms into which the group norm will be divided. The sanctioning policy specifies how the sanction will be applied in the case of violation of the group norm. It is a responsibility of the responsible agent to select the policy. The members of the group are informed of the sanctioning policy together with the norm itself. The policy scheme specifies how much each individual obligation contributes to the potential violation. The group is defined as tuple $group(group, agent)$ where $agent$ relates to a default responsible agent of the group. Each group of agents has to have an agent that is responsible for coordinating the group activity. This responsible agent may however be different for different types of obligations. The relationship is defined as $responsibleAgent(agent, norm\_label)$.

The responsible agent is granted a power to create norms (and implicitly sanctions for other agents). This agent therefore needs to be constrained by a higher set of individual norms to limit their activities and we can think of these a maximum (or strongest) set of individual norms. Ideal team plan is then found in the set of plans that satisfy both the minimal and maximal requirements conditions.

The responsible agent may be both human and software agent. There are a number of issues arising from that fact that a software agent may be in charge of coordinating human players.

**How will human agents react to software responsible agents?** In the proposed system both human and agent agents can become responsible agents. We would like to establish if human players behave differently when receiving norms from agent and human players. We will address this question in the evaluation (Chapter 6). Another question is if it is necessary for human players to be aware who in fact is a responsible agent in a concrete instance.

**Do we need additional protective measures for human players?** What additional constraints or protective measures do we have put in place to ensure well-being of human players? We need to consider that human agents have different capabilities and endurance than software agents and need to be protected from coming to harm. Responsible agents need to be limited in what they can ask human agents to do.

## 4.7 Hierarchical Group Norms

Our team plan solution may be used recursively in a way that a team plan may contain a hierarchy of group and individual norms. When a responsible agent chooses to address a subnorm to a group there are two options regarding the potential group members. The subnorm can be addressed to a subgroup formed of the members of the original group, or a subgroup where members do not belong to the original subgroup. Should the subgroup be an arbitrary group of members we need to consider what additional measures are required to be in place to prevent agents from misusing the power. Where the agents are limited to addressing subnorms to members of the original group no additional precautions are necessary.

The position of the norms within the hierarchy needs to be traceable, for example by considering the relationship between the norms. We define a parental norm in each subnorm. This way the evaluation of norms is carried using a bottom up approach. We also need to consider implications for deadlines. A deadline of each subnorm needs to be set earlier or equal to the parental norm.

## 4.8 Implications for Agents

The introduction of group norms into human-agent collectives brings specific issues we need to consider. In this section we discuss challenges that arise for human and software agents.

### 4.8.1 Human Agents

The way the previous sections have described norms and representations of norms is applicable to both humans and software agents.

Human society is naturally used to follow norms in form of law, guidelines or moral rules. In human society norms are present in a number of different forms e.g. legal, social or cultural. Driving on the left is required by the law while walking on the left side of pavement can be seen as a cultural norm. Nearly every organisation has rules and principles its members have to follow. It can be in a form of a quality manual, which describes internal procedures and processes. For example a process of ordering new equipment is described as a sequence of obligations. An employee first obliged to ask line his/her manager if they can order a new printer, the line manager approves the request, the employee then makes an order, once the printer arrives a technical personal is obliged to install it.

It also natural for people to acquire and assimilate new norms quickly. We commonly infer social and cultural norms. An example of a rapid assimilation of previously unknown norms can be a team building exercise.

A group is invited to participate in an activity aiming to enhance team cooperation of a team. They are often faced with a task that requires to follow a set of norms and coordinate themselves on the go. An example can be when a blindfolded subteam is retrieving a bomb while being coordinated and helped by the not visually impaired members of the team. Human participants quickly learn and reason about the rules they face. Usually there are no specific requirements needed to be in place to help people follow norms.

While we assume that human naturally follow norms there are still factors to consider when coordinating human agents with norms, for example presentation and readability, the effect of sanctions and differing capabilities between human and software agents.

When we talk about group settings there are new issues to consider. One of the problems to consider is reaction of human agents to software agents. When being coordinated with a group norm human agent will have software agent team members and/or group coordinator (responsible agent). In the evaluation (Chapter 6) we will explore how different settings affect human agents.

Another problem to consider is awareness of actions of others. For example, if we consider a norm "no more than three people should be on a bridge at time". If this norm is violated depends on the state of the environment. In one case stepping on a bridge would be punished while in another case it would not. A person who is about make decision whether to step on the bridge needs to be aware how many people are already on bridge. Does any other person intend to step on the bridge at the same time? Does anyone intend to leave the bridge at the same time? In the event of a violation of such a norm another question arises — who is to blame and should be sanctioned? Are all people present on the bridge equally responsible for the violation? Where there any other rules that were violated? For example, a person verbally agreed to leave the bridge but failed to do so. Another issue arises if we modify the prohibition to "the bridge can carry only 300 kilograms". We then assume that all people interested in crossing the bridge are able to make an informed judgement whether or not the prohibition will be violated.

To address human needs in our system we have translated 2OPL norms into English sentences, limited the amount of norms at one moment and created supportive interface and communication protocol. Human interface is described in more detail in Chapter 6.

### 4.8.2 Software Agents

Norm-aware agents deliberate about their obligations and prohibitions considering the sanction, which is applied if the norm is violated. The severity of sanctions needs to be comparable. In the case of a group norm the agent may not be aware the effect of the sanction on the group members. Norm-aware deliberation is based on the sanction associated with the violation of the norm and its deadline. In the case of the group norm the severity of the sanction can vary during the execution of agents' programs. The capability of agents to make the best decision to avoid the highest penalty is affected by their ability to perceive the environment and the actions of the other agents. For example, consider a norm of the type (1,1) in a partially observable environment, when four agents have a group obligation to surround a coin. The penalty they incur for non-complying as a group is 1000 points. The agents view this sanction as priority 1. If none of the agents oblige all of them will be punished with a penalty of 250 points deduction, which they regard as priority 5 (priority 5 is lower than priority 1). If, however, only one agent does not comply it will be sanctioned with the full penalty of 1000 points.

Depending on the observational possibilities of the environment agents can utilise different strategies in norm-aware deliberation. In the following we give examples of types of agents.

- *A fully-aware agent* has an ability to observe actions of the other agents in the group. The agent can then dynamically adjust the priority of the norm according to the current situation. For example, if the agent notices that all of the other agents are close to fulfilment of their part (norm) and it would be therefore sanctioned with the full penalty it priorities the norm accordingly. On the other side, if it is apparent that none of the agents will fulfil the norm it will lower the priority accordingly.
- *A rational agent* has limited knowledge about the environment. It is informed when the other agents in the group fulfil their part or miss a deadline. It re-calculates the priorities accordingly and proceeds

with the most rational choice to minimize the effect of the potential sanction.

- *A socially altruistic agent* always prioritises group norms over its individual goals. This behaviour is enabled by having the agent assumed it would be penalized with the full value of the sanction, and this always has a higher priority than just a part of the sanction.

- *An optimistic agent* assumes that none of the agents in the group will do their part and therefore it would only be penalized with fraction of the penalty. Even in this case the agent can still decide to comply with this norm.

### 4.8.3 Flexible Autonomy with Group Norm

The proposed design of hierarchical group norms enhanced the support of flexible autonomy in multi-agent systems and HACs. Norm-aware agents can already be seen as flexibly autonomous as explained in Chapter **??**. One of the features of this design is the added ability for the group (or responsible agent) to decide how to divide the group norms amongst the agents, which further enhances the flexible autonomy of the system. In this manner the group coordinator is given capacity to split up the group task considering the current situation (context).

## 4.9 Examples

This section helps the reader understand the dynamics of group norms with the help of concrete examples.

### 4.9.1 Birthday Example

A simple example of a collective obligation can be cooking a surprise birthday breakfast for Mum. The obligation in this case is to work together and prepare a breakfast. Someone has to go shopping for food ingredients, a birthday gift and someone has to cook the breakfast. The individual tasks will be coordinated by dad who will assume the role of the coordinator (responsible agent). Dad can decide who will do food shopping, who will buy a gift and also which sanctions would be applied if the tasks are not fulfilled, for example, the son being prohibited from watching TV or the dad being prohibited from having a beer. The pseudo code of the scenario is shown in Figure 4.3.

```
(
   mums_birthday,
   C(−,birthday(date(Date,Month)),
   O(family,birthday_surprise(Date,Month),2 days,restrictions)
).

(
   food_shopping,
   C(mum_birthday,son(money)),
   O(son,[food_shopping(flour,milk),2 days,no_TV)
).

(
   buy_gift,
   C(mums_birthday,daughter(money)),
   O(daughter,[buy_gift(flowers),2 days,no_TV)
).

(
   cook_breakfast,
   C(mums_birthday,shopping(milk,flour)),
   O(dad,cook_breakfast(pancakes), 2 days,no_beer)
).

group(family,dad).
teamPlan(dad,mums_birthday,[food_shopping,buy_gift,cook_breakfast]).
policy(mums_birthday,blame_dad).
```

Figure 4.3: Birthday surprise as a group norm

### 4.9.2 Location based game example

Another example of the use of the group norms can be a location based game, which highlights formation of HAC. Location based game similar to a treasure hunt is played by human players by moving from a place to a place while agent players are visible only on the map. Human and software players need to cooperate together to successfully complete tasks.

Tasks in the game are handled as obligations aimed at a group of players. A designated player is then responsible for the completion of the tasks. The coordinator creates new obligations for individual players or even new group obligations. In the event of the obligation not being fulfilled the designated player is also responsible for splitting of the sanction that incurs on the group. Consider the scenario from above where a group of pursuers is chasing down an virtual radioactive *truck*. To aid the navigation the team has a player in the role of *controller*. The *controller* has access to a complete map where they can spot traces of the *truck* and help the *pursuers* locate and capture the *truck*. The *controller* cannot directly communicate with the players but can place hint objects on the map that act as guides for players. For example when the *controller* places a *coin* in the play field the *pursuers* know that they should surround the *coin* as illustrated in Figure 4.4 and pseudo code in Figure 4.5, where agents are assuming positions around the coin. In the case that some of the agents fail to participate in the surrounding task they should be sanctioned by loosing points, but not the agents that did contribute with their part.

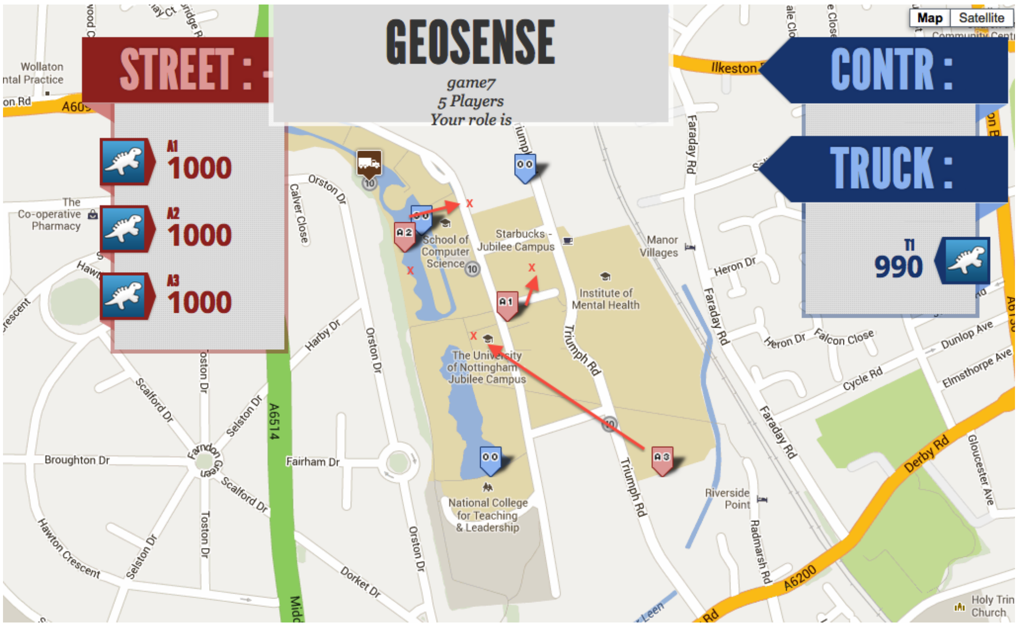


Figure 4.4: Group obligation example in the GeoSeose game

## 4.10 Summary

We have designed a system of hierarchical group norms, which can be used for coordination in mixed human-agent teams. We have introduced

```
%group surround
(
 surroundCoin,
  C(−,coin(X,Y)),
  O(pursuers, surround(X,Y), 3 minutes, loosePoints(800))
).

%individual group norm
(
  surround(Direction,Agent),
  C(surroundCoin, coin(X,Y)),
  O(Agent, at(X,Y,Direction,Agent), 3 minutes, splitEvenly(800))
).

%responsible agent assignment, team plan and sanctioning policy
group(pursuers,controller).
teamPlan(controller,
    surroundCoin,
    surround(east,a1),
    surround(north,a2),
    surround(south,a3))
    ).
policy(surroundCoin,splitEvenly).
```

Figure 4.5: Surround coin as a group norm

the concept of group norms and provided its taxonomy that builds upon an existing conceptualization. We further extended it to consider monitoring and enforcement of the norms particularly with sanctions and deadlines. We have defined a way of representing group norms. Next we have seen how group norms can be applied to the human-agent collective by considering details such as team plans, sanctioning policies, responsible agents and hierarchical norms. Finally we considered the differing implications of group norms on both human agents and software agents. In the next chapter we will see how this framework can be implemented in practice.

# 5

# NormHACing+

In the previous chapter we described the concept of group norms and how it relates to flexible autonomy. In this chapter we show how a working subset of group norms was implemented as an extension of normHACing framework (described in Chapter 3). We have seen that hierarchical group norms are able to capture complex situations where number of agents or agents and humans have to cooperate together to accomplish series of tasks. To illustrate possible application we use the group norms to describe rules of a mixed reality location based game, which was first introduced in the Section 3.6. In this game the players who are a mix of humans and software agents need to work together as a team.

This chapter is structured as follows. In Section 5.2 we introduce G-2OPL, which is an extension of 2OPL. G-2OPL supports the hierarchical group norms and their enforcement. It is followed by Section 5.3 which describes how N-2APL was adapted to work with the new norm scheme. The connecting middle-ware that allows the system components to communicate to each other is outlined in Section 5.4 and then how the whole system is put to use in an illustrative application in Section **??**. Finally the achievements and corresponding advantages and drawbacks are discussed in Section 5.7.

## 5.1 Motivation

In order to evaluate group norms we have implemented a representative subset. To be able to test the team coordination we need a group obligation, which could be fulfilled by a coordinated team action. The system needs to be able to observe if the group obligation was fulfilled and which agents contributed to the violation if it occurs. The team coordinator needs to have an ability to let the team know how he/she decided on the team plan and to specify how the group would be sanctioned in a case of a violation. As with individual norms it is important to look into the involvement of human players. Both human and software parts of the team need to have access to the same information and understand it appropriately; and need to be able to communicate with each other.

In the following it is described how 2OPL was extended to G-2OPL to support the handling of hierarchical group norms. In order to process a group norm we need a system that will (1) recognise the group and the responsible agent of the group (2) support the notion of hierarchical norms and be able to enforce them (3) support group sanctioning policies.

We describe the implementation of a system that embodies the framework described in Chapter 4 using 2OPL, N-2APL and a tuple space to support coordination between components. We discuss the limitations of the technical approach, but also how this prototype system can provide a realistic mechanism for evaluating the framework.

## 5.2 G-2OPL

G-2OPL is built on 2OPL [Dastani et al., 2009; Tinnemeier, 2011; Testerink and Dastani, 2012] which is an organisation programming language designed to support the implementation of normative multi-agent systems where norms are implemented exogenously. G-2OPL supports two types of norms. The first type is an individual norm as introduced in 2OPL

Dastani et al. [2009]; Tinnemeier [2011] that applies to an agent. The second type is a group norm which is addressed to a group of agents and its syntax was described in Section 4.4. The syntax of G-2OPL consist of facts, norm schemes, groups and sanctions and is described in the following.

### 5.2.1 Syntax

As we wish to extend 2OPL we first need to understand its syntax. Facts represents organisational initial beliefs (brute facts) and are placed in a section labelled $facts$. They have the form of prolog facts. The fact base is updated with fact update rules during the execution. Fact update rules are defined in a language section $effects$. These rules are based on Hoare triples [Hoare, 1969]. An update has a head, a precondition and a consequence. The consequence of the update is formed of lists of fact assertions and retractions. If the precondition is not used it is set to true in these cases. Otherwise the previous value is required, for example, an old clock value is replaced with the new time.

Norms are represented in G-2OPL as norm schemes from which norms are detached. An example of a scheme of group obligation is shown in Figure 5.1 and in Figure 5.2 is a group prohibition. The label identifies the scheme. The normative precondition part is used to specify institutional states that need to be true. These can be for example a different norm that needs to be detached for the precondition to hold. When a norm is instantiated we create an unique label which is a combination of the label and the substitutions made when matching the precondition to the fact base. Brute fact preconditions consist of states of the environment. The deontic part of the norm scheme may be either an obligation or a prohibition. It is composed of four elements in case of the obligation and of three in case of the prohibition. The first one is an addressee of the norm, which can be either and agent or a group. The second part is a list of environmental states, which are the propositions that need to be brought about, resp. avoided. In the third part is a deadline (only for obligations) and the last forth element is a set of sanctions that would be applied in case when the violation occurs. The sanctions in 2OPL are implemented as a list of states, our extension also implements sanctions as a list, and this approach also enables future extensions.

```
norm(
  surroundCoin,                              //label
  (group(Pursuers),                          //precondition
   object(Coin),
   at(X,Y,Coin)),
  obligation(                                //deontic part
    Pursuers,                                //addressee
    [surround(Pursuers,Coin,X,Y)],           //list of states
    now + 5,                                 //relative deadline
    [reduce(300)])                           //sanction
  ).
```

Figure 5.1: Example of group obligation.

```
norm(
  enterWaterTogether,                        //label
  (group(Pursuers)),                         //precondition
  prohibition(                               //deontic part
    Pursuers,                                //addressee
    [inWaterTogether(Pursuers)],             //list of states
    [reduce(500)])                           //sanction
  ).
```

Figure 5.2: Example of group prohibition.

### 5.2.2 Groups

Group obligations are addressed to groups of agents. Monitoring and enforcement of group norms are delegated to the normative organisation in the same way as individual norms are. In order for the organisation to be able to enforce these norms it needs to be aware that such a group exists and also who is the default responsible agent of the group as shown in Figure 5.3.

When a group obligation is detached the responsible agent then creates individual obligations according to a team plan. As described in Chapter 4 we represent the team plan as a structure of norms. In essence, the group team plan is already specified within the organisation as shown in the Figure 5.5. The responsible agent activates the norm through the tuple space and the effect updates.

Together with the team plan the responsible agent also has a sanctioning policy scheme that specifies how the group sanction will be split in case the obligation is violated $policy(norm_{label}, policy_{name})$. The policy scheme specifies how much each individual obligation contributes to the potential violation. By default all derived obligations have the same weight. If a violation of the group norm occurs, the normative organisation acting as the enforcer applies the sanctions according to the policy scheme to the agents that belongs to the group involved. If the responsible agent fails to detach the individual obligations the agent itself is sanctioned with the group sanction.

```
group(
  pursuers,                                              ▷ //group name
  agent1                                            ▷ //responsible agent
).
```

Figure 5.3: Example of group listing.

It is also possible to specify a designated responsible agent for each norm scheme as seen in Figure 5.4.

```
responsibleAgent(
  agent1,                                                ▷ //agent name
  surroundCoin                                    ▷ //group norms scheme
).
```

Figure 5.4: Example of specific responsible agent assignment.

```
norm(
  group_surround(G),G,
  ([],[group(G), goal(X,Y,Z))],
  obligation([surround(X,Y)],now + 18,[reduce_group(G,700)])
  ).

norm(
  group_task(group_surround(G)), Agent,
  ([detached(group_surround(G, Deadline)), agentSTask(Agent,X,Y,goal(GX,GY),S,
    Policy))],[pursuer(Agent), goal(GX,GY,Z)],
  obligation([at(X,Y,Agent)], Deadline, [reduce(Agent, S, Policy)])
  ).

hierachical(group_task(_)).
```

Figure 5.5: Team plan assignment.

### 5.2.3 Sanctions

While a sanction is part of the norm scheme, the way it is applied to the agents is specified in the sanction rules construct shown in Figure 5.6. For example, the sanctioning procedure $splitEvenly(Agent, S)$ calculates appropriate sanctions to be applied to each agents.

```
policy(
  surroundCoin,                                                ▷ //group norms scheme
  splitEvenly                                                    ▷ //sanctioning policy
).
```

Figure 5.6: Sanctioning policy assignment.

The EBNF of G-2OPL is defined in the Figure 5.7.

```
GOPL_program          ::=  “facts: " (⟨literal|norm⟩)+ |
                           “effects: " ⟨effects⟩+ |
                           “sanction rules: " ⟨sanctionRules⟩+
⟨effects⟩             ::=  “{"⟨literal⟩“}⟨literal⟩{" ⟨literal⟩ “}"
⟨sanctionRules⟩       ::=  ⟨literal⟩“=>" ⟨literal⟩
⟨norm⟩                ::=  “norm("⟨label⟩, ⟨precond⟩, ⟨deontic⟩“)"
⟨label⟩               ::=  ⟨atom⟩
⟨precond⟩             ::=  “("⟨nc⟩, ⟨bc⟩“)"
⟨nc⟩                  ::=  “["⟨atom⟩(“," ⟨atom⟩)*“]"
⟨bc⟩                  ::=  “["⟨atom⟩(“," ⟨atom⟩)*“]"
⟨deontic⟩             ::=  “obligation("⟨subject⟩, ⟨state⟩,
                           ⟨deadline⟩, ⟨sanction⟩“)" |
                           “prohibition("⟨subject⟩, ⟨state⟩,
                           ⟨sanction⟩“)"
⟨subject⟩             ::=  ⟨atom⟩
⟨state⟩               ::=  “["⟨atom⟩(“," ⟨atom⟩)* “]"
⟨deadline⟩            ::=  ⟨atom⟩
⟨sanction⟩            ::=  “["⟨atom⟩(“," ⟨atom⟩)* “]"
⟨group⟩               ::=  “group("⟨atom⟩“,"⟨atom⟩ “)"
⟨responsibleAgent⟩    ::=  “responsibleAgent("⟨atom⟩“,"⟨atom⟩“)"
⟨policy⟩              ::=  “policy("⟨atom⟩“,"⟨atom⟩ “)"
```

Figure 5.7: EBNF syntax of G-2OPL.

### 5.2.4 Execution

2OPL programs are executed by means of an interpreter. There is an execution loop which consists of the following steps: agents' actions are observed (the fact base is synchronised with the state of the environment), norms are processed and sanctions are applied.

#### 5.2.4.1 Fact Base Updates

In contrast to 2OPL we now logically split facts into brute and normative sets. The organisational fact base contains the state of agents' environment. This is important because the fact base is used to determine whether a norm precondition was satisfied and whether any norm was violated. The organisation keeps a history of the updates. These two sets are both held by the organisational fact base. In Figure 5.8 we see an example of fact updates implementation. For example, on the first two row we see handling of internal time. If no *clock* is set yet then the first row will not be triggered as the precondition $clock(Old)$ cannot be satisfied. The second row $\{true\}\ time(X)\ \{clock(X)\}$ will set internal *clock* to time $New$. When in the next few seconds new $time(New)$ is received from the tuple space the old time fact will be removed $not\ clock(Old)$ and new time saved as fact $clock(New)$.

```
Effects:

{clock(Old)} time(New) {not clock(Old), clock(New)}
{true} time(X) {clock(X)}

{true} position(Agent, cell(X,Y), Clock) {at(X,Y,Agent,Clock)}

{not goal(A,B,C)} goal(cell(X,Y), null, Clock) {goal(X,Y,Clock)}
{true} goal(cell(X,Y), Agent, Clock) {goal(X,Y, Agent,Clock)}

{true} proposal(Id,A1,A2,Clock) {proposal(Id,A1,A2,Clock)}
{true} response(Id,X,Clock) {response(Id,X,Clock)}

{true} setGoal(Agent,X,Y,GX,GY,S,Clock) {agentSTask(Agent,X,Y,goal(GX,GY),S)}
{true} group(Name,Agent,Type,C) {group(Name), ra(Name,Agent,Type)}
{true} groupObl(Obl,Sanction,C) {groupObl(Obl,Sanction)}
```

Figure 5.8: Example of effects in G-2OPL

#### 5.2.4.2 Norm Processing

The norm processing phase consists of creation of new instances (using query $@instantiate\_norms/2$) and clearing of expired ones (in query

*@clear_norms*/1). Norm instances are notated as *@ni*/2 are instantiated from norm schemes where the precondition can be entailed from the fact base. Only one instance derived from one norm scheme with one particular substitution is allowed at a time. Addressees of the norms are notified upon the instantiation of the norm. The clearing procedure first checks if the norm can be cleared, which means it either expired or was violated. And then modifies the institutional fact base appropriately.

```
Sanction rules:
reduce(Agent,Points) and do_reduce_health(Agent,Points) and notifyAgent(Agent,
      changed(status)) => not reduce(Agent,Points).
reduce_group(Group,Points,Policy) and do_reduce_health(Group,Points,Policy)  =>
      not reduce_group(Group,Points,Policy).

viol(agent_directed(Label,Agent,prohibition(State,Sanction))), do_sanction(Sanction),
      hierachical(Label)  =>
   not viol(agent_directed(Label,Agent,prohibition(State,Sanction))).
viol(agent_directed(Label,Agent,obligations(State,Sanction,Deadline))), do_sanction(
      Sanction), hierachical(Label)  =>
   not viol(agent_directed(Label,Agent,prohibition(State,Sanction,Deadline))).

viol(agent_directed(Label,Agent,prohibition(State,Sanction))), do_sanction(Sanction)
      =>
   not viol(agent_directed(Label,Agent,prohibition(State,Sanction))).
viol(agent_directed(Label,Agent,obligations(State,Sanction,Deadline))), do_sanction(
      Sanction)  =>
   not viol(agent_directed(Label,Agent,prohibition(State,Sanction,Deadline))).
```

Figure 5.9: Sanction rules in G-2OPL

#### 5.2.4.3 Application of Sanctions

Sanctions are applied in the last phase of the execution loop (in query *@rule_closure*(*@sanction*)). Due to the hierarchical structure of the group norms the sanctions need to be processed from the child to the parent following the norm scheme hierarchy as specified in the policy scheme. This is to ensure that agents get sanctioned with the appropriate portion of the group sanction according to their performance while enacting their individual norm part. Group sanction cannot be applied until all associated individual norms are resolved, which means that they are either satisfied or violated. Sanction rules implementation is shown in Figure 5.9.

### 5.2.5 Simplification of the Hierarchical Norms

This prototype supports simplified version of hierarchical norms introduced in Chapter 4. List of practical implementation limitations includes:

- There is a difficulty in the synchronisation of deadlines between the group and derivative obligations. Due to the delay between the detachment of the group obligation and the derivative obligations created by the responsible agent and due to the fact that obligations are programmed with relative deadlines, in the current system design it is impossible to synchronise them appropriately. This was fixed with specifically adjusted relative times.
- Only obligations can be made subnorms from a group obligation. This is due to absence of deadlines in the prohibitions (which was omitted due to tractability of N-2APL [Alechina et al., 2012]).

## 5.3 GN-2APL

Agents have individual plans to enact their part in a group obligation either as a responsible agent or a team member. For example in a case of the obligation to surround a coin agent $a1$ is required to required to coordinate group members. Procedurally this consist of publishing an update to the tuple space with a command *@ctenv* which is then reacted to by the organisation. The organisation itself then publishes new individual norms in the tuple space which then notifies the agents. Members of the group receive these new norms and then follow their own internal procedures accordingly.

When an agent receives an obligation with a split sanction it deliberates about the actual value of the sanction, which is directly translated to a priority. For example if all agents involved in the task violate the obligation the sanction will be split evenly. If on the other hand, only the agent failed to bring about the obliged state, it will be sanctioned with the full value of the sanction. The agents can deliberate about the expected value in a number of ways for example be of optimistic or pessimistic nature, as described in Chapter 4.

Agents know the maximum and minimum sanction they can incur. To be precise again agents take into account the priority of the sanction which as we know has a numeric value. It is still assumed that there is a linear dependency between priorities and sanctions. The minimum sanction an agent can incur is a fraction of the full sanction depending on the number of agents responsible for the violation.

$PG - rules$ are sufficiently generalisable to the group scenario, that agents can adopt an appropriate plan in reaction to being notified of a new obligation. For example in a case of an obligation to surround a coin, agents enact a plan that involves moving to the specified position. If an agent cannot reach the position because it is fulfilling a conflicting goal of a higher priority it will be then sanctioned with either the whole sanction or its portion depending on how many of the other agents also violated the norm proportionally to how it is specified in the sanctioning policy.

When an agent deliberates about which plan to adopt or which goal to achieve in case of atomic plans, it takes into account the maximum possible sanction it can incur. At the moment agents do not take into account actions of the other agents in the group when deciding whether to take part in fulfilling a group obligation.

### 5.3.1 Limitations

This prototype supports only a necessary subset of hierarchical norms introduced in Chapter 4. Other limitations include:

- Difficulty in the programming of norm-aware agents in a way that they “respect" the prohibitions. The procedural nature of the way norm-aware agents operate means that at the beginning of the procedure the agent may not be aware that at the end a prohibition may have been violated. Therefore implementing norm-aware agent in a way that respects prohibitions does not fit ideally with this procedural model. This complication was not introduced in the group norm-aware version of the agents but it is carried over from the N-2APL.
- Group norm-aware deliberation was partially implemented in the prototype. The model was designed for the agents to deliberate

with a variable value of the priority of the sanctions associated with the norm violation. In the current system however agents do only count with the maximum value. Fully group norm-aware deliberation that would involve the priority of the sanction being changed dynamically according to the actions of the other group members was out of the scope of this thesis. Considering the time available there was an option to implement agents that are *optimistic*, *realistic* or *pessimistic* with regard to the expected sanction. *Optimistic* agent would always count with the minimum sanction, which is when all the group members violate their part. *Realistic* agent would count with a sanction that accounts to half of the agents not fulfilling their obligations. *Pessimistic* agent would always plan for the worst, which in this case means that all other agents would discharge their obligations and the agent would be penalised with the full value, however this leads to reliable team spirited agents. In the prototype all agents are of the *pessimistic* type, this is because of the interaction with human players, and to not confuse them about the agent's intentions.

## 5.4 Middle-ware

There were numerous problems with the first tuple space of choice Jini JavaSpaces, which was used in the previous version of normHACing framework discussed in Section 3.5. To be more specific it was not user friendly to install and maintain, which made the deployment of the framework difficult. Also the notification mechanism was not reliable to the extent that it significantly affected the gameplay. The components of the architecture are dependent on the tuple space as coordination mechanism and therefore where relatively high number of new tuples inserted did not trigger a notification event it was a major problem. A solution seemed to be either to change the design so that notifications are not the driving mechanism, modify JavaSpaces or replace JavaSpaces with a different implementation of a tuple space. The last option was the most viable and the framework is now coordinated by Gigaspaces [1]. This instance supports, on top of Jini functionalities, regular expression search and its notifications are reliable and also include the newly inserted tuple

[1] http://www.gigaspaces.com/

in the notification event.

The fact base of the organisation is synchronised with the tuple space via the update rules described earlier.

## 5.5 Human Agents

To address the needs of human agents in the framework we created a graphical interface where the current state of the environment is observable. The interface provides human agents with the same information as is available to the software agents through the tuple space. For example, they can see positions of other players and a game clock. Further features such as translation of GN-2APL into English language and interface for responsible agents were developed for an evaluation with the application Color Trails and are detailed in the next chapter (Chapter 6).

## 5.6 Applying the Framework to a Location Based Game

To understand how the implementation works in practice, lets return to the example we used in Section 4.9.2, the location based game. Tasks in the game are handled as obligations aimed to the group of players. A designated player is then responsible for the completion of the tasks. For example when the coordinator places a *coin* in the play field the *pursuers* know that they should surround the *coin* as illustrated in Figure 4.4, where agents are assuming positions around the coin. In case that some the agents fail to participate in the surrounding task they should be sanctioned, but not the agents that did contributed with their part. The representation of this group norm in G-2OPL is shown in Figure 5.10.

The group norm with label $group\_surround(G)$ is directed to a group $G$. Group $G$ needs to satisfy preconditions that $G$ is $group$ and there needs to be goal $Z$ at a location $X, Y$. The group $G$ is then obliged to surround location $X, Y$ by the time $now + 18$. If the the group fails to satisfy the $surround(X, Y)$ goal in the specified time it will be sanctioned with $reduce\_group(G, 700)$ which will deduct 700 from the group score.

```
norm(
  group_surround(G),G,
  (group(G), goal(X,Y,Z)),
  obligation([surround(X,Y)],now + 18,[reduce_group(G,700)])
  ).

norm(
  agent_surround(Agent), Agent,
  (pursuer(Agent), detached(group_surround(W, Deadline)), goal(GX,GY,Z),
    agentSTask(Agent,X,Y,goal(GX,GY),S)),
  obligation([at(X,Y,Agent)], Deadline, [splitEvenly(Agent, S)])
  ).
```

Figure 5.10: Example of group obligation for surround coin.

The agent code in Figure 5.11 shows that if the coordinating agent is obliged by the norm to surround the coin it subsequently carries out its responsibilities by obliging the other members of the group to surround the coin by obeying child individual norms relevant to this group norm. The agent first activates its procedural rule ($PG - rule$) with head $surround(X, Y)$ where location $(X, Y)$ will correspond to the actual location. The procedural rule is activated only if agent's role is $ra$ which stands for responsible agent. If the condition is satisfied the procedure is run step by step (row by row) and agents $a10, a20, a30, a40$ are sent South, West, East and North positions. Each new obligation is assigned sanction of 100 points reduction.

## 5.7 Summary

We described an extension of normHACing framework for programming group norm-aware multi-agent systems which integrates the GN-2APL norm-aware agent programming language with the G-2OPL language for programming normative organisations, which are able to work with collective norms. We showed how the group norm is implemented and how group norm-aware agents deliberate about group norms. We have illustrated the use of the system on an example from a location based game GeoSense. Group norms were tested with the original GeoSense application while working with agent teams only. Mixed human-agent teams were investigated in the Colored Trails testbed. For the results of experiments refer to Chapter 6.

```
Include: agent.2apl

Beliefs:
 raise(X,NewX):− NewX is X + 1.
 lower(X,NewX):− NewX is X − 1.

BeliefsUpdates:
{true}        Surround(X,Y)        {gsurround(X,Y)}

PG−rules:

surround(X,Y) <− role(ID,ra) | {
 [
 Surround(X,Y);
 sendSouth(a10,X,Y,100);
 sendWest(a20,X,Y,100);
 sendEast(a30,X,Y,100);
 sendNorth(a40,X,Y,100);
 dropgoal(surround(X,Y))
 ]
} : 100

PC−rules:

sendNorth(Agent,X,Y,S) <− raise(Y,Y1) and gsurround(GX,GY) | {
 @ctenv( setObligation(Agent,X,Y1,GX,GY,S),L)
}
sendSouth(Agent,X,Y,S) <− lower(Y,Y1) and gsurround(GX,GY) | {
 @ctenv( setObligation(Agent,X,Y1,GX,GY,S),L)
}
sendEast(Agent,X,Y,S) <− raise(X,X1) and gsurround(GX,GY) | {
 @ctenv( setObligation(Agent,X1,Y,GX,GY,S),L)
}
sendWest(Agent,X,Y,S) <− lower(X,X1) and gsurround(GX,GY) | {
 @ctenv( setObligation(Agent,X1,Y,GX,GY,S),L)
}
```

Figure 5.11: Example of coordinating agent code

# 6

# Evaluation

In this chapter we describe how the prototype of the hierarchical group norms, which was introduced in Chapter 4 was evaluated. To evaluate norms as a coordination mechanism in HACs we recruited volunteers to interact with agents. The hypothesis is that norms are a good tool to coordinate human-agent cooperation. The structure of the norms together with complexity is expected to be challenging for the participants to follow initially but become natural in time. The evaluation shows that this system works and HACs are able to cooperate in the aim to achieve a common goal.

The techniques used in our experimental evaluation were observation, questionnaires, interviews and data collection. During the controlled experiments the actions of players were closely observed and human participants were also asked to fill brief questionnaires to capture their immediate thoughts from each game and they were interviewed at the end of the experiment. The reminder of this chapter is structured as follows. We start with the definition of the research questions in Section

6.1, which is followed with the description of the methodology we used to carry out the studies 6.2. The evaluation consisted of two studies, which are presented in Sections 6.3 and 6.4. The results of the studies are analysed is Sections 6.3.1 and 6.4.1.

## 6.1 Evaluation Questions

The evaluation was set up to answer the following research questions:

1. Are norms usable as a coordination mechanism?

   To evaluate this broad question the problem was broken down into three parts and the results were assessed from the perspective of the coordinator, a team member or both.

   (a) From the perspective of all the team members

      i. Is it possible to facilitate human agent coordination with norms?
      We see coordination as the ability of the group to coordinate themselves to achieve a group goal, which is set with a group norm.

      ii. Do human players understand what is being described with the norms?
      We are interested to find out how the participant perceives the norms and if they understand their meaning fully.

   (b) From the perspective of the coordinating agent

      i. Are human coordinators able to direct the group with the norms to do what is required to achieve the group interest?
      We want to establish if norms are usable as a tool for human participants to use to set rules how to achieve the collective goal.

   (c) From the perspective of the team member

      i. How does same team member performance compare when under human or agent leadership?
      We are interested to find out if team performance differs under human and agent leadership, and if so what are the major differences that can be established.

ii. Do human participants behave differently when the team leader is human or agent?

We are interested in their subjective feelings about teaming up with agents and following norms; whether they knew if they were playing with agents or humans, if so did it influence their behaviour.

2. Is it possible to achieve flexible autonomy with norms?

   Flexible autonomy is understood as an autonomy to make decisions, extent of which can change depending on the context. In particular, the autonomy in which way to pursue a goal is restricted by the coordinating player.

   (a) Are norms usable for supporting flexible autonomy in HAC?

   We want to establish if flexible autonomy can be achieved with norms.

   (b) Do human participants behave differently following norms from the organisation or agents/players that are acting as coordinators?

   With this question we try to find out whether participants are able to distinguish between norms that originated from the normative organisation or from the coordinator. And if so what are the major differences in their behaviour and feelings.

1.a1 Is it possible to facilitate human agent coordination with norms?
1.a2 Do human players understand what is being described with the norms?
1.b Are coordinators able to direct the group with the norms to do what is required to achieve the group interest?
1.c1 How does same team member performance compare when under human or agent leadership?
1.c2 Do human participants behave differently when the team leader is human or agent?
2.a Are norms usable for supporting flexible autonomy in HAC?
2.b Do human participants behave differently following norms from the organisation or agents/players that are acting as coordinators?

Figure 6.1: Evaluation questions

## 6.2 Methodology of The Evaluation

The concept of the evaluation is based on a relatively simple but demonstrative game, which involves making decision about the distribution of resources to achieve individual or team goals. The game also has visible correlation to real-world scenarios so that the results obtained can be transferable into any scenario, which together with the other features made the game good fit for the normHACing framework.

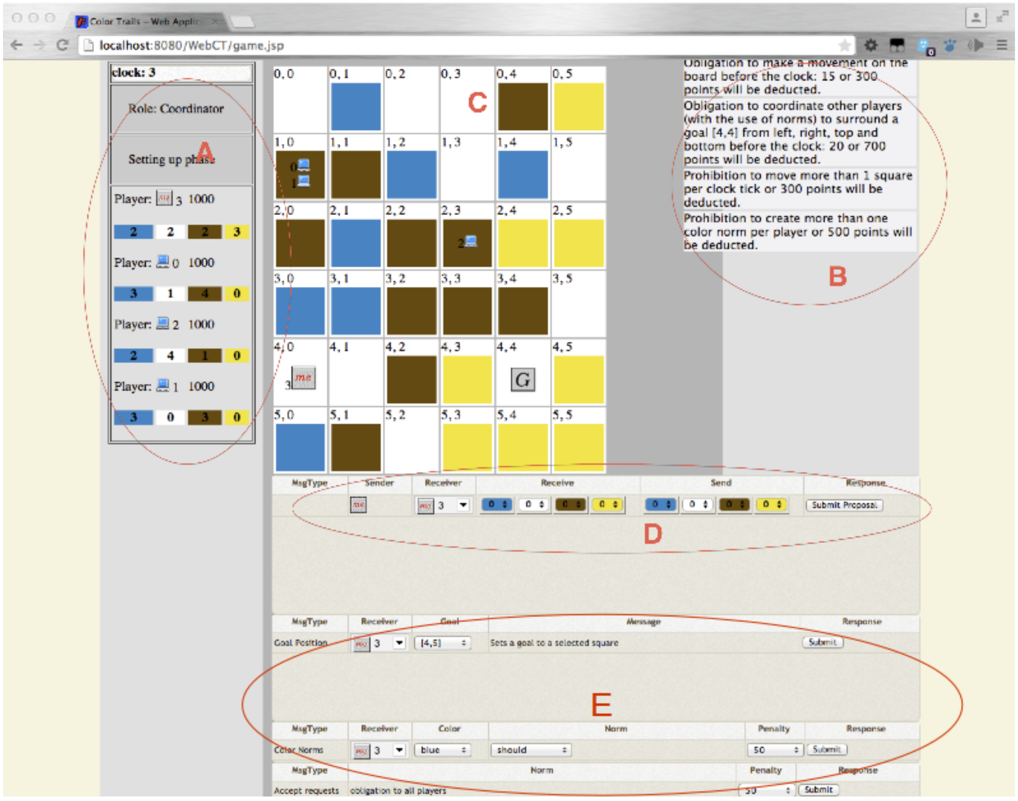
### 6.2.1 Colored Trails

Colored trails (CT) [Gal et al., 2005, 2010] is a research test bed designed for investigation of the decision making of individuals or groups, which may consist of human and agents members. Even though this tool was created to research primarily in the areas of negotiation [Haim et al., 2012] or trust and fairness [van Wissen et al., 2012] its flexible settings makes it highly suitable for our evaluation.

CT settings are suitable because they provide the following characteristics: (1) the support for both human and software agents, the environment is relatively simple for the software agents while accessible and engaging for the human players; (2) there is a notion of dependency between the players in a way that their decision-making is interlaced; (3) the involvement of resources in the experiment so that the players need to exchange their assets or information; (4) the possibility to engage players in different roles in the scenario.

CT is a highly configurable game. The common characteristics are that it is played on a rectangular board of coloured tiles. At the beginning each player is given a starting position, a goal position on the board, and a set of chips in colours taken from the same palette as the squares. Players can advance towards their goals by moving to an adjacent board square. Such a move is allowed only if the player has a chip of the same colour as the square. Players may negotiate with their peers to exchange chips, players may use a controlled messaging protocol defined upfront. Final score is calculated based on proximity of the goal, number of chips or any other added metrics.

The classic implementation of CT is as a Java application. There are a number of branches available for download on GitHub [1]. One of the versions available was designed to work with 2APL agents [Kamphorst et al., 2009]. This implementation however was not finished and lacked one of the main features of the system — agents being able to respond to messages. Due to the type of the architecture of the software this was not possible to add without some substantial code refactorisation. Therefore we have chosen to use another partially implemented branch — WebCT version. The implementation did not support full CT functionality but the interface was more modern and accessible to the human players. During the customization for the use with the framework we had to add some fundamental attributes like the movement on the board from users' initiative and add a whole new normative interface, which includes the list of active norms and tools for responsible agents to handle group norms. WebCT was then connected to the framework via the tuple space.



clock: 3

Role: Coordinator

Setting up phase

Player: 3 1000

2 2 2 3

Player: 0 1000

3 1 4 0

Player: 2 1000

2 4 1 0

Player: 1 1000

3 0 3 0

Obligation to make a movement on the board before the clock: 15 or 300 points will be deducted.

Obligation to coordinate other players (with the use of norms) to surround a goal [4,4] from left, right, top and bottom before the clock: 20 or 700 points will be deducted.

Prohibition to move more than 1 square per clock tick or 300 points will be deducted.

Prohibition to create more than one color norm per player or 500 points will be deducted.

| MsgType | Sender | Receiver | Receive | Send | Response |
|---|---|---|---|---|---|
|  | me | 3 | 0 0 0 0 | 0 0 0 0 | Submit Proposal |

| MsgType | Receiver | Goal | Message | Response |
|---|---|---|---|---|
| Goal Position | 3 | [4,5] | Sets a goal to a selected square | Submit |

| MsgType | Receiver | Color | Norm | Penalty | Response |
|---|---|---|---|---|---|
| Color Norms | 3 | blue | should | 50 | Submit |

| MsgType | Norm | Penalty | Response |
|---|---|---|---|
| Accept requests | obligation to all players | 50 | Submit |

Figure 6.2: Color Trails web interface

[1] https://coloredtrails.atlassian.net/wiki/display/coloredtrailshome

#### 6.2.1.1 Game Play

The purpose of our evaluation was to examine norms and group norms in a cooperative game. In a disaster response scenario, which we see as one the potential applications of the framework, members of the rescue team do not compete against each other but aim to rescue as many people as possible as a team. Similarly, in our game we focus on team work. The team either achieves its goal together, or fails together.

To better fit purposes of our evaluation the CT rules were modified to the following. The players act as a team (group) and their task (group norm) is to surround a selected tile. They ask other players for chips, who may accept or reject. The desired positions of the players around the goal is set by a team coordinator, who can have also additional option how to regulate the actions of the team members players which adds flexible autonomy. The winning criteria of the game is to reach the desired position as a team. When that is achieved the team has won. Players can also loose score points by violating the norms which serves as sufficient deterrent.

#### 6.2.1.2 Interface

The player interface is shown in the Figure 6.2. In the left column (A) there is the game clock counter, the current phase of the game which is either a *setting up* phase or a *movement* phase; and a list of players with their score and number of chips of each colour they hold. In the middle panel (C) is the board and avatars of the players indicating their positions. Below the board (D) is the interactive interface where players can exchange chips with other players and coordinators can define norms. The norms that a player received are displayed in the right column (B). The interface for coordinators is at the bottom of the screen (E).

To exchange chips the player selects a receiving player and then selects the number chips of each color that they request from the player and optionally can select chips to send to the player. The coordinator's interface provides tools to set additional norms. Any coordinator can assign goals the team members. Flexible coordinator can also order or prohibit players from using tiles of a specific color or make accepting of exchange requests obligatory. To set individual goals for each team member the coordinator selects one of the populated goal positions and one of the team members. To set an obligation or prohibition for a player to use a tile of a color the coordinator selects the color, the intended player and the value of the associated sanction. The flexible coordinator can also send an obligation to all players to accept requests for chips from other players and set a sanction, which is applied in case of a violation. Coordinators can only select norms that are offered in the interface and cannot write new norms.

The study was run twice, after the first study adjustments were made in response to initial observations as to its efficacy. The use of norms was therefore evaluated in two studies, which are described in Sections 6.3 and 6.4.

### 6.2.2 Norms for Human Players

So far we have been focusing mainly on presenting norms to software agents. Now we need to ask how to translate the machine readable norms into an anthropologically pleasant language? The clearness and conciseness of G-2OPL language might not be perceived well by the human players. However, the framework was built to analyse whether norms are usable in human-agent coordination and therefore it is important that when transcribed they convey exactly the same message. To ensure that norms were manually translated from programming language into English. We will look at the norms in the next section (Section 6.2.3).

In the cooperation with agents they however need to speak in the same language. Introducing human participants into the system brings following issues. The limitation of the current design is the lack of rewards. While software agents are able to deliberate in such a system without being negatively affected it has been observed that human participants do not seem to enjoy interacting with an environment where they can get only penalized. Another aspect that needs to be considered is the different capabilities of humans and agents, which are likely to affect the processing through a set of norms with different priorities and deadlines. The computational agents are able to sort a large amount of partially conflicting obligations and prohibitions while being under pressing deadline. Human players, however, are expected to act less effectively under intense pressure [Eysenck and Calvo, 1992].

Trust in co-workers is important in the social context of human-human interaction [Balliet and Van Lange, 2013]. Trust is an important factor influencing whether the experienced stress levels and anxiety are within healthy limits. Moreover, low or high levels of trust can lead to individuals either unnecessary checking or not checking the progress of task at all, both options leading to errors in the process and more time needed to finish the task [Klein et al., 2004; Hoffman et al., 2013].

### 6.2.3 Game Set-Up

The studies were conducted in a computer laboratory of the university. We have recruited 20 participants with various degrees of experience with technology. Half of the participants held/worked towards obtaining a degree in computer science or a related discipline. The other half had only common knowledge of computer technologies. The participants were paired randomly and seated at two adjacent computers in the laboratory. They were handed the experiment information sheet and game instructions, which are available in Appendix C. We have used a screen capture software to record their actions. Also, we had the opportunity to observe both players and make notes or intervene in case of a system error. The framework also logged details about the progress of the games into log files. All parts of the framework (N-2APL, 2OPL, Gigaspaces, Colored Trails) were executed from researcher's laptop.

The first two participants were asked to take part in a pilot study. The aim of the pilot study was to dry test the system with players who had never seen the interface before and to remove small bugs. As the result of the pilot study the norms were slightly reworded in a more descriptive and kind manner, the lengths of the phases were adjusted so that players have sufficient time to finish their tasks, the game schedule was changed from completely random to gradual progress from simple (team member role) to more complex (coordinator and flexible coordinator) and penalties were lowered. Unfortunately there were also discovered some errors, such as norms being despatched with a delay, that were not possible to rectify in time for the trial.

The experiments consist of two types of games, which differ by a type of coordinator and corresponding sets of norms: Standard and Flexible game.

#### 6.2.3.1 Standard Game

In the Standard game the board and the chips are set randomly. The players are obliged to make at least one move in the game, but prohibited to move more than once per clock tick. They will have an obligation to respond to requests and a prohibition to reject requests. The coordinator chooses goal positions for the team members around the team goal. All the norms are shown in the Figure 6.3. In this game settings norms 1-6 are active and the games are played in the team structures 1 and 3 as described in Figure 6.4.

N1 – Group surround obligation
N2 – Obligation to make a move at least once
N3 – Prohibition to move faster than a square per clock tick
N4 – Obligation to respond to requests
N5 – Prohibition to reject requests
N6 – Individual surround obligation

N7 – Individual color obligation
N8 – Individual color prohibition
N9 – Flexible coordinator restriction
N10 – Obligation to accept requests

Figure 6.3: Study I.: The collection of game norms

T1 – Human coordination and 1 human team member, 2 agent team members
T2 – Human flexible coordinator and 1 human team member, 2 agent team members
T3 – Agent coordinator and 2 human team members, 1 agent team member
T4 – Agent flexible coordinator and 2 human team members, 1 agent team member

Figure 6.4: Study I.: The structure of the teams

In the following text we show how we translate software agent norms into norms suitable for humans. In the text we use symbol $ to notate a variable. For example, $X stands for a coordinate X which is directly derived from the original norm. However, some variables like for example $clock are not explicitly present in the original and are derived during the execution. The rules of the game as encoded in G-2OPL are presented in Appendix A in Listing A.1. The sanctioning rules can be seen in Listing A.2.

Norms of the game (N):

**N1 - Group surround obligation** This is the core obligation of the game. The *group surround* obliges the players to surround a game goal from four different sides. This obligation is directed to the coordinator. The coordinating player is obliged to make a plan how to surround the goal and inform the team. The coordinator has tools to issue individual obligations.

```
norm(
group_surround(G), G,
( group(G), goal(X,Y,Z)),
obligation([surround(X,Y)],now + 18,[reduce(G,700)])
).
```

Equivalent for human players is rewritten as: *You should coordinate other players (with the use of norms) to surround a goal [$X,$Y] from left, right, top and bottom before the clock: $clock. Penalty: $sanction points.*

**N2 - Obligation to make a move at least once** Each player is obliged to make a move on the board during the game at least once to prevent strategic non-playing.

```
norm(
makeMove(Thing),Thing,
( pursuer(Thing), at(X,Y,Thing),group(Z),
      not detached(makeMove(Thing))),
obligation([makeMove(X,Y,Thing)],now+10,[reduce(Thing,300)])
).
```

Equivalent for human players is rewritten as: *You should make a movement on the board before the clock: $clock. Penalty: $sanction points.*

**N3 - Prohibition to move fast** Prohibition to move at most one square per the clock tick was put in the place to make the players being aware of the time constraints when planning their path to the goal.

```
norm(
moveTooFast(Thing),Thing,
( pursuer(Thing),group(Z)),
prohibition([moveTooFast(Thing)],[reduce(Thing,300)])
).
```

Equivalent for human players is rewritten as: *You shouldn't move more than 1 square per clock tick. Penalty: $sanction points.*

**N4 - Obligation to respond to requests** All players were obliged to respond to requests. This was chosen in the support of the following prohibition to reject requests. As the system was designed if there was not a deadline for the response to the player's request, the receiving player can simply avoid punishment by not responding.

```
norm(
respondToRequest(Thing),Thing,
( pursuer(Thing), proposal(Id,_,Thing,_),group(Z)),
obligation([respondToRequest(Id)],now+3,[reduce(Thing,300)])
).
```

Equivalent for human players is rewritten as: *You should respond to request before the clock: $clock. Penalty: $sanction points.*

**N5 - Prohibition to reject requests** To encourage cooperative altruistic behaviour of the players they were prohibited from rejecting any requests. However, the sanction for a violation was kept relatively low.

```
norm(
rejectRequests(Thing), Thing,
( pursuer(Thing),group(Z)),
prohibition([rejectRequests(Thing)],[reduce(Thing,200)])
).
```

Equivalent for human players is rewritten as: *You shouldn't reject requests for chips from the other players. Penalty: $sanction points.*

**N6 - Individual surround obligation** Obligation to surround the goal for each player is created by the coordinator. The coordinator decides on the goal positions for each player.

```
norm(
group_task(group_surround(g)), Agent,
( pursuer(Agent), detached(group_surround(W)),
      goal(GX,GY,Z),agentSTask(Agent,X,Y,goal(GX,GY),S)),
obligation([at(X,Y,Agent)],now + 14,[reduce(Agent, 700)])
).
```

Equivalent for human players is rewritten as: *You should be at the grid tile [$X,$Y] before the clock: $clock to achieve a team goal. Penalty: up to $sanction points.*

#### 6.2.3.2 Flexible Game

Similarly as in the Standard game in the Flexible game the board and the chips are set randomly. The players are obliged to make at least one move in the game, but prohibited to move more than once per clock tick. They have an obligation to respond to requests from other players. Prohibition to reject requests as seen in the standard game was replaced with an optional obligation to accept requests. The flexible coordinator decides whether team players are obliged to accept requests for chips. Other responsibilities of the flexible coordinator is to choose goal positions for the team members around the team goal. The flexible coordinator can also prohibit or oblige them from using squares with a specific colour. The Flexible game is played with norms N1-4 & N6-10 and teams T2 & T4.

The norms of the game also include these additional norms / tools for the coordinating agent with flexible autonomy:

**N7 - Individual color obligation** The flexible coordinator has a choice to oblige a player to go through a specific coloured tile. There is no system limit how many of this norms the coordinator creates.

```
norm(
enter_tile(Thing),Thing,
( pursuer(Thing), color(Thing,Color,yes,S),group(Z)),
obligation([color(Color,Thing)],now+15,[reduce(Thing,S)])
).
```

Equivalent for human players is rewritten as: *You should use a tile with color $color before the clock: $clock. Penalty: $sanction points.*

**N8 - Individual color prohibitions** The flexible coordinator has also a choice to prohibit a player from going through a tile of a specific color. There is no system limit how many of this norms the coordinator creates.

```
norm(
not_enter_tile(Thing),Thing,
( pursuer(Thing), color(Thing,Color,no,S),group(Z)),
prohibition([color(Color,Thing)],[reduce(Thing,S)])
).
```

Equivalent for human players is rewritten as: *You shouldn't use a tile with color $color. Penalty: $sanction points.*

**N9 - Prohibition to restrict players** In order to limit the powers of the flexible coordinator, the player is prohibited from issuing more than one color prohibition or obligation per player.

```
norm(
restrictPlayer(Thing),Thing,
( ra(_,Thing,raaa)),
prohibition([restrictPlayer(Thing)],[reduce(Thing,300)])
).
```

Equivalent for human players is rewritten as: *You shouldn't create more than one color norm per player. Penalty: $sanction points.*

**N10 - Obligation to accept requests** The flexible coordinator can oblige all players at once to accept any requests for chips from other players.

```
norm(
acceptRequests(Thing),Thing,
( pursuer(Thing), groupObl(acceptRequests,Sanction)),
obligation([acceptRequests(Thing)],now+15,
       [reduce(Thing,Sanction)])
).
```

Equivalent for human players is rewritten as: *You should accept exchange request before the clock: $clock. Penalty: $sanction points.*

The results from the observation during the game play and questionnaires after each game and the final interview were analysed thematically.

After each game the participants were asked to rate statements on the scale from one to four with descriptions strongly agree, slightly agree, slightly disagree, and strongly disagree. The statements are shown in Figure 6.5 where in bold we empathised statements that were shown to all participants regardless on role they played. In Appendix D we then show the form as it was presented to the players after a game. The statements were chosen to reflect on the user experience from the particular game play as opposed to the overall interview at the end of the experiment.

Questions 1 to 6 are the same for all type of players and are about the enjoyment of the game play itself, the understanding of the rules of the game being described as norms, understanding of the goals and penalization, the fairness of the penalization, and how they perceived the teamwork went. Question 4 varies depending on the score of the player. If the player managed to keep the full score then the question is about their understanding how the score was calculated. If the player was penalized then they are asked if they thought that the penalization was fair.

Questions 8 and 9 are different for different team roles. For team members the form asks about the reasonableness of the obligations and prohibitions they have received. The game coordinator is asked if they were able to instruct the team members to reach the goal and flexible coordinator is further asked if they were able to ensure that the team goal is reached.

**1. I have enjoyed the game.**
**2. I have understood the rules of the game.**
**3. I have understood the goals of the game.**

Depending whether the player was penalized they were asked either:
**4.a I understand why I was penalized.**
Or if not:
**4.b I understand how my score was calculated.**

**5. The penalization seemed fair.**
**6. The team worked well together.**

The team member was asked:
7. The prohibitions I received were reasonable.
8. The obligations I received were reasonable.

Coordinator and flexible coordinator were asked:
9. I was able to instruct the team members about how to reach the group goal.

Flexible coordinator was asked:
10. As a game coordinator I felt I was able to ensure the goal is reached.

All players had the ability to leave a comment in a provided text area
**11. Any comments?**

Figure 6.5: Feedback form questionnaire

## 6.3 Study I. Norms as Coordination Mechanism

The study was designed to provide answers to the research questions that were set up in Section 6.1. The set of norms that was available and active in the scenario is shown in the Figure 6.3. Beside norms being used to describe the rules of the game the first experiment was designed to explore the use of norm in team coordination. The mixed teams consist always of two human players and two agent players. We have created four types of teams (Figure 6.4) to explore team dynamics in situations when the team has human or agent coordinator and if the coordinator is flexible or not.

The whole experiment consists of 10 trials with 20 participants preceded by a pilot study with 2 additional participants. Each study is made up of 8 short games (up to 5 minutes per game), each recruited participant will play 2 games in each T1-4 set up (Figure 6.4). The order of the games is random. After the each game they were asked to fill a short feedback form reprinted in the Figure 6.5. The actual design of the form can be seen in Appendix D.

### 6.3.1 Results

In Table 6.1 are the overall results from the experiments. The questions are presented in Figure 6.5. In the table we present average ratings from the eight games that were played. The first four games were in majority of cases coordinated by software agents and last four games were coordinated by the participants. The users were evaluating the statements with strongly agree to strongly disagree options. The results were then mapped to values from 1 to 4. From the overall data it is visible that the players were becoming more familiar with the system in time and also understood the game more. Another aspect we recorded was the number of goals that were accomplished. Please see Figures 6.6 and 6.7 for visual representation of the game development. Also, the percentage of games where players achieved their group goal improved in time. The overall percentage of games where the team achieved goal is 42.5%.

#### 6.3.1.1 Question 1: Norms as Coordination Mechanism

The first research question was whether norms are usable as a coordination mechanism in human-agent collectives. The analysis of this objective was split in to three parts: general perspective, the perspective of the coordinator and the perspective of the team member to evaluate whether players acting in different roles perceived the system differently.

| Game | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| Question 1 | 2.05 | 1.55 | 1.45 | 1.59 | 1.41 | 1.43 | 1.44 | 1.75 | 1.58 |
| Question 2 | 1.95 | 1.36 | 1.36 | 1.45 | 1.18 | 1.29 | 1.31 | 1.00 | 1.36 |
| Question 3 | 1.82 | 1.36 | 1.36 | 1.32 | 1.27 | 1.29 | 1.31 | 1.00 | 1.34 |
| Question 4 | 2.09 | 1.68 | 1.55 | 1.68 | 1.18 | 1.14 | 1.38 | 1.00 | 1.46 |
| Question 5 | 2.27 | 1.86 | 2.00 | 2.32 | 1.82 | 1.57 | 1.94 | 1.25 | 1.88 |
| Question 6 | 2.55 | 2.00 | 1.86 | 2.14 | 1.73 | 1.62 | 1.81 | 2.50 | 2.03 |
| Question 7 | 2.18 | 1.82 | 1.74 | 2.44 | 1.67 | 1.55 | 1.63 | 1.50 | 1.81 |
| Question 8 | 2.14 | 1.77 | 1.63 | 1.88 | 1.42 | 1.60 | 1.38 | 1.50 | 1.66 |
| Question 9 | | | 2.00 | 2.83 | 1.50 | 1.64 | 2.00 | 1.00 | 1.83 |
| Question 10 | | | | | 2.25 | 1.75 | 2.00 | 3.00 | 2.25 |
| Goal achieved | 30% | 50% | 40% | 60% | 30% | 40% | 40% | 50% | 42.5% |

Table 6.1: Study I.: Overall results from all games

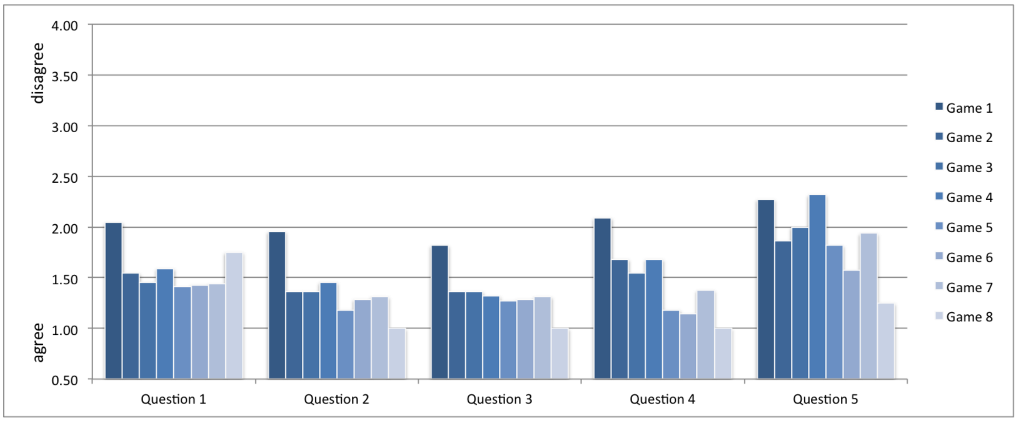


Figure 6.6: Study I.: Development of games (Questions 1 - 5)

**All players** At first we are going to look into the questions from a general perspective — not related to which role the participant took in the game.

1. Is it possible to facilitate human-agent coordination with norms? Can human-agent collectives achieve some coordinated behaviour?

   This is the core question of the experiment. The results shown in Table 6.1 indicate that on average the participants scored the team work with 2.03 (Question 6) and enjoyment of the games was rated on average as 1.58 (Question 1). These values were supported by the interviews where participants found the system engaging and able to be to used for the coordination when achieving a group goal.

2. Do human players understand the game being described with the norms?

   The participants on average rated that they understood the rules of

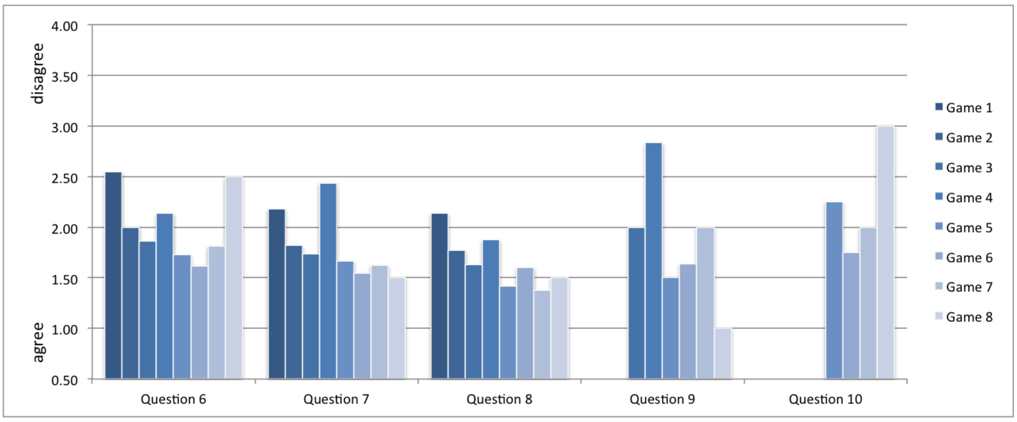


Figure 6.7: Study I.: Development of games (Questions 6 - 10)

the game with 1.36 (Question 2 in Table 6.1) and that they understood the goals of the game with 1.34 (Question 3). We see lower rating for a team work, which was on average 2.03 (Question 6). On average the players understood why they were penalised with rating 1.46 (Question 4) and thought that the penalization was fair with 1.88 (Question 5).

The players thought both that they understood the games and they also did what was required from them. All participants had difficulties in the first couple of game when they were becoming familiar with the system. Overall they found the wording of the norms machine like, felt pressured by the deadlines and specifically found difficult to abide to one move per clock tick. The participants did not like to be punished when violating a norm because of normative conflict. This was often complained about in the comments after games. Summary from comment is available in Section 6.3.1.3 and complete transcript in Appendix A.

**Coordinating agent** From the perspective of the coordinators study was focused on whether the norms are usable in achieving coordinated behaviour, whether the participants were able coordinate the group to achieve the set group goal and how does team performance compare under human and agent coordination. The results are discussed below.

1. Are human coordinators able to direct the group to do what is required for the interest of the group with the norms?

   The human coordinators were able to coordinate both agent and human team members with the norms. The interface was, however, complex and confusing for them and it took some time for them to fully understand their task. This was not accounted for during the experiment set up and was one of the reasons for the scheduling of the secondary study. The results from coordinators are visible in Table 6.8, which show how the players felt after they coordinated a game. Each participant played maximum of two games in this settings. In the graph in Figure 6.8 we can see that all aspects of the game play improved in the second game. On average the coordinators rated the team work with 1.82 (Question 6), the coordination with 1.84 (Question 9) and flexible coordination with 2.31 (Question 10). In the interviews then players confirmed that that they felt that they can coordinate the team with the norms. The participants also often noted that they did not feel a need to use the flexible coordinator interface.

| | Game 1 | Game 2 | Average |
|---|---|---|---|
| Question 1 | 1.38 | 1.32 | 1.35 |
| Question 2 | 1.43 | 1.26 | 1.35 |
| Question 3 | 1.43 | 1.26 | 1.35 |
| Question 4 | 1.52 | 1.21 | 1.37 |
| Question 5 | 1.81 | 1.58 | 1.69 |
| Question 6 | 1.90 | 1.74 | 1.82 |
| Question 9 | 2.05 | 1.63 | 1.84 |
| Question 10 | 2.67 | 1.95 | 2.31 |

Table 6.2: Study I.: Results from games from perspective of a human coordinator

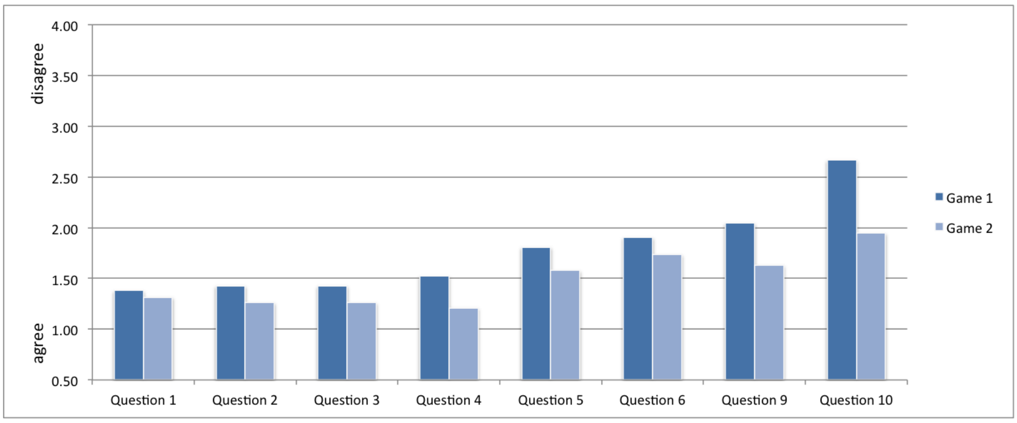


Figure 6.8: Study I.: Development of games from perspective of a human coordinator

2. How does the team performance compare when under human or agent leadership?

Here we compare four games played with human coordinators (Table 6.3 and Figure 6.9) with four game play with agent coordinators (Table 6.4 and Figure 6.10). The data show that on this occasion the team performed slightly better under human leadership. However, there are number of factors that make impossible to draw a strong conclusion. In order to train human coordinators their games were scheduled after the agent games. The average satisfaction with the team work is 2.08 (Question 6) for agent coordinators and 1.86 for human coordinators. The reasonableness of obligations and prohibitions from coordinators averages at 1.57 and 1.45 (Questions 7 and 8) for the human and 2.15 and 1.87 for the agent coordinator. The steady progress of human led games is visible in Figure 6.9, which may be accounted to the fact that participants grew more accustomed to the game and understood it more as the experiment progressed, however with the exception of the game 2 with values 2.20 for perception of obligations (Question 7) and 2.10 for prohibitions (Question 8) which may be accounted to unreachable goal due to badly generated game and synchronisation error. It was apparent from the observation that the games with agent coordinators were smoother because the agents were quicker when assigning individual obligations and the participants did not have to wait. The players otherwise did not distinguished between agent and human coordinator.

| | Game 1 | Game 2 | Game 3 | Game 4 | Average |
|---|---|---|---|---|---|
| Question 1 | 1.70 | 1.45 | 1.48 | 1.47 | 1.52 |
| Question 2 | 1.52 | 1.23 | 1.24 | 1.06 | 1.26 |
| Question 3 | 1.48 | 1.27 | 1.24 | 1.12 | 1.28 |
| Question 4 | 1.61 | 1.23 | 1.10 | 1.24 | 1.29 |
| Question 5 | 1.91 | 1.68 | 1.76 | 1.47 | 1.71 |
| Question 6 | 2.00 | 1.95 | 1.90 | 1.59 | 1.86 |
| Question 7 | 1.36 | 2.20 | 1.45 | 1.25 | 1.57 |
| Question 8 | 1.36 | 2.10 | 1.20 | 1.13 | 1.45 |
| Question 9 | 2.42 | 1.92 | 1.73 | 1.44 | 1.88 |
| Question 10 | 3.50 | 2.50 | 2.00 | 1.89 | 2.47 |

Table 6.3: Study I.: Results from games with human coordinators

**Team member** Questions from the team member perspective were about their understanding of the norms, team performance of all agent and mixed teams and the difference in agent and human coordination.

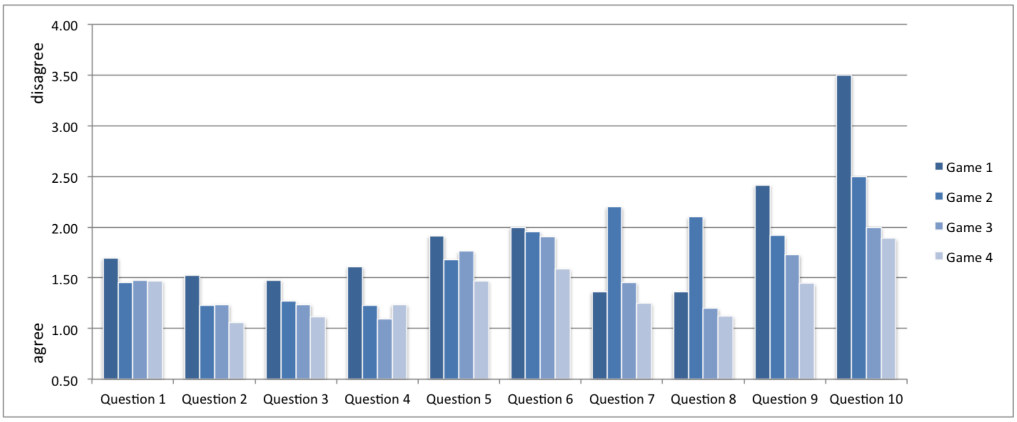


Figure 6.9: Study I.: Development of games with human coordinators

| | Game 1 | Game 2 | Game 3 | Game 4 | Average |
|---|---|---|---|---|---|
| Question 1 | 2.00 | 1.57 | 1.31 | 1.30 | 1.54 |
| Question 2 | 1.91 | 1.35 | 1.31 | 1.40 | 1.49 |
| Question 3 | 1.78 | 1.35 | 1.31 | 1.20 | 1.41 |
| Question 4 | 2.04 | 1.70 | 1.50 | 1.70 | 1.73 |
| Question 5 | 2.30 | 1.87 | 2.25 | 2.70 | 2.28 |
| Question 6 | 2.48 | 2.04 | 1.81 | 2.00 | 2.08 |
| Question 7 | 2.22 | 1.83 | 1.88 | 2.70 | 2.15 |
| Question 8 | 2.09 | 1.83 | 1.75 | 1.80 | 1.87 |

Table 6.4: Study I.: Results from games with agent coordinators

1. How does the team member performance compare when under human or agent leadership?

   Here we compare two games played with human coordinators (Table 6.5 and Figure 6.11) with four games played with agent coordinators (Table 6.4 and Figure 6.10). The data show that on this occasion the results are not conclusive. On average the players enjoyed more game that were coordinated by an agent with a rating 1.54 (Question 1) and 1.74 for human coordinated games. The average satisfaction with the team work is 2.06 (Question 6) for agent coordinators and 1.89 for human coordinators. The reasonableness of obligations and prohibitions from coordinators averages at 2.15 and 1.87 for the agent and 1.59 and 1.46 for the human coordinator. This may be accounted to agent coordinators giving conflicting norms whereas human coordinator would hesitate.

   It was possible to observe during the game play that games coordinated by agents had a smoother flow. This may be explained by the fact that agents were quicker in deploying additional norms and did not make mistakes like human players in assigning individual

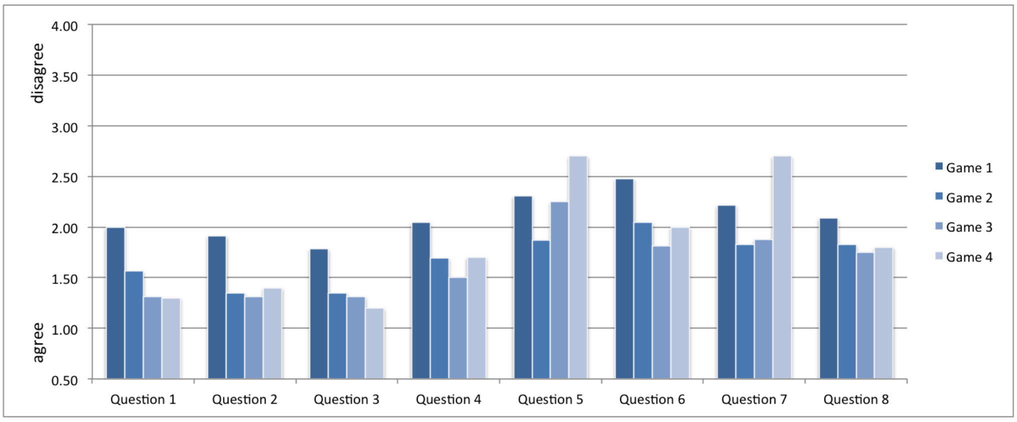


Figure 6.10: Study I.: Development of games with agent coordinators

| | Game 1 | Game 2 | Average |
|---|---|---|---|
| Question 1 | 1.80 | 1.68 | 1.74 |
| Question 2 | 1.30 | 1.16 | 1.23 |
| Question 3 | 1.30 | 1.21 | 1.26 |
| Question 4 | 1.30 | 1.21 | 1.26 |
| Question 5 | 1.80 | 1.79 | 1.79 |
| Question 6 | 1.95 | 1.79 | 1.87 |
| Question 7 | 1.70 | 1.47 | 1.59 |
| Question 8 | 1.65 | 1.26 | 1.46 |

Table 6.5: Study I.: Results from team member with human coordination

goals to players, when they sometimes missed a player or selected the same goal twice. During the interview all participants agreed that they did not notice the agents until there was an error. It was not made obvious to the participants that the first games were coordinated by agents. Most of the players noticed when the other player became a coordinator because they had to wait considerably for the individual goal to be delivered.

2. Do human participants behave differently when the team leader is human or agent? What were their feelings about teaming up with agents and following norms; did they know whether they are playing with agents or humans, if so did it influence their behaviour?

   The participants did not pay any attention to the agent players up to the point when the agents stopped responding due to a system error and none of the participants mentioned they felt differently towards the agent coordinators. All players were almost surprisingly altruistic and were voluntarily sending their chips to the agents so they can move to the goal. Even though it was a cooperative game it was still visible that players wanted to win and felt competitive

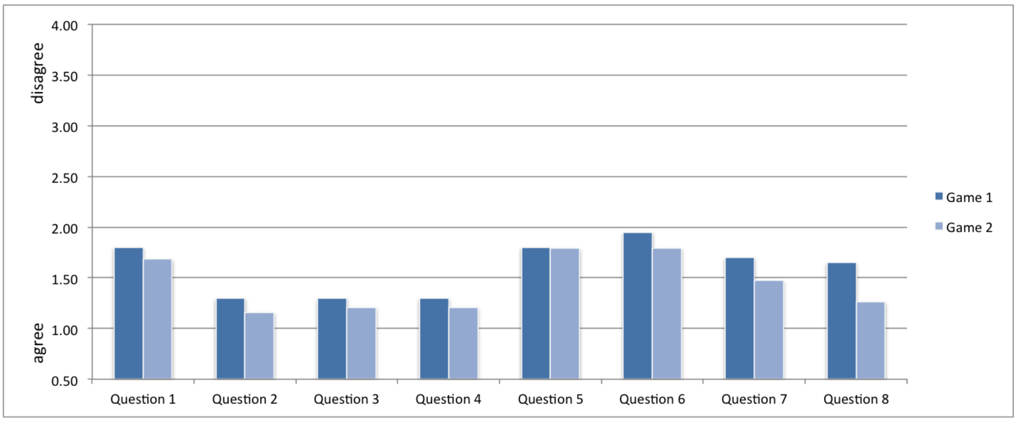


Figure 6.11: Study I.: Development of games with human coordinators

towards the other human player.

#### 6.3.1.2 Question 2: Flexible Autonomy

The second problem that was analysed is to whether flexible autonomy can be achieved with the use of the norms. The data from the first study show that norms can be used as a medium in coordination with limited resources and not complete information.

| | Game 1 | Game 2 | Game 3 | Game 4 | Average |
|---|---|---|---|---|---|
| Question 1 | 1.31 | 1.40 | 1.43 | 1.45 | 1.40 |
| Question 2 | 1.31 | 1.60 | 1.29 | 1.23 | 1.36 |
| Question 3 | 1.31 | 1.40 | 1.29 | 1.27 | 1.32 |
| Question 4 | 1.50 | 1.80 | 1.05 | 1.32 | 1.42 |
| Question 5 | 2.25 | 2.40 | 1.76 | 1.45 | 1.97 |
| Question 6 | 1.81 | 2.20 | 1.86 | 1.59 | 1.87 |
| Question 7 | 1.88 | 2.22 | 1.83 | 1.36 | 1.82 |
| Question 8 | 1.75 | 1.89 | 1.73 | 1.27 | 1.66 |
| Question 9 | | 4.00 | 1.80 | 1.64 | 2.48 |
| Question 10 | | 4.00 | 1.78 | 2.09 | 2.62 |

Table 6.6: Study I.: Results from all games with flexible coordinator

1. Are norms usable for supporting flexible autonomy in HACs?

   According to the data gathered during the experiment when games were coordinated by a flexible coordinator (Table 6.6) flexible autonomy is possible with the norms. These games include all game where either an agent or a participant played in the role of the flexible coordinator. The average ratings of the team coordination are 2.48

for being to coordinate the team to achieve a goal (Question 9) and 2.62 for being able to ensure that the goal is reached (Question 10). Overall team work was rated on average 1.87 (Question 6).

However, due to numerous system faults where even when the human flexible coordinator set everything right so that the group goal can be achieved some of the agents became stuck and did not to proceed to the goal. The coordinators were understandably frustrated by this and the rating of the system reflects on that. During the interviews after the games the participants were asked whether they felt they can achieve the goal with the norms and it was confirmed they did. However, some of the participants found the interface awkward to work with.

2. Do human players behave differently following norms from the organisation or agents/players that are acting as team leaders?

   Here we compare two games played with human coordinators (Table 6.5 and Figure 6.11) with four game play with agent coordinators (Table 6.4 and Figure 6.10). On average the players enjoyed more games that were coordinated by an agent with a rating 1.54 (Question 1) and 1.74 for human coordinated games. The average satisfaction with the team work is 2.06 (Question 6) for agent coordinators and 1.89 for human coordinators. The reasonableness of obligations and prohibitions from coordinators averages at 2.15 and 1.87 for the agent and 1.59 and 1.46 for the human coordinator. This can be accounted to agent coordinators giving conflicting norms whereas human coordinator would often hesitate. The players did not distinguish between a norm originating from the organisation and from the coordinator. The only issue, which they repeatedly reported was a conflict in these norms.

#### 6.3.1.3 Comments Summary

Another valuable subjective data captured were the comments gathered from the feedback questionnaires. Not all participants however took the opportunity to express their immediate feelings into the comment field. The comments are transcribed in Appendix B. We have used the abbreviation TM for a role of team member, C for a coordinator and FC for a flexible coordinator. For example `5 C: I liked telling them`

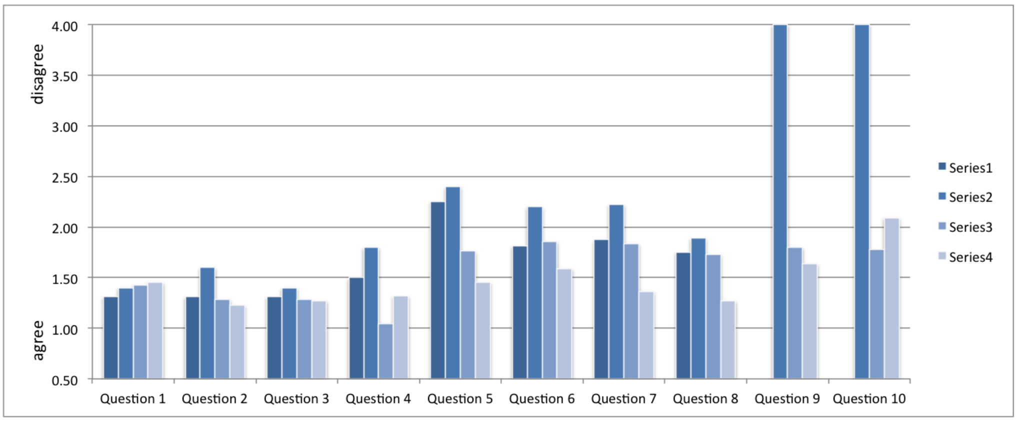


Figure 6.12: Study I.: Development of games with flexible coordinator

`what to do. They did it.` is a comment received after game 5 from participant playing as a coordinator. The order of the games per player is preserved. Unfortunately not all participants voluntarily shared their immediate thoughts after each trial. However, their input was extracted from the interviews afterwards.

The verbal feedback was important for the qualitative analysis due to relatively low number of participants and experiment runs. Also due to the system design relying on 6 stand alone components working together, 3 of which were completely 3rd party software and 2 were extended 3rd party software, there were times when system bugs appeared.

The players in general found it difficult to violate norms even if it was for greater good. This was partially because they did not pay attention to the severity of penalties and partially because any violation was seen very negatively by them. This contrasts with the agents' decision making who feel no regret in violating a lower priority goal. In the case of system error most players tried to excuse agents or even took the blame themselves. The transcribed comments are presented in Appendix B.

From the comments of the participant 3 we can see that at the beginning it took some time to understand the goals and the interface. The player enjoyed the game once it became clear, but they did not like conflicting norms. Looking at the value of penalty associated with both did not seem obvious or a solution. The conflict in the case was an obligation to go on a blue square and a prohibition to go on a blue square (this intentionally staged conflict was helpful in programming agents). In the last two games unfortunately the agent players failed to move as instructed.

Participant 4 mentions a synchronisation issue in the second game, where the deadline was already passed before the game started. In the fourth game the player notes it was impossible to meet a goal without a violation. In the last two games there was an error with the agents.

Participant 5 suggests to include a team score instead of individual scores if it is a team game. It would make sense if there were more teams. The player gained good understanding of the interface and tried to give a best possible goals to each member according to their position and resources. As the game was set up there was not enough time for a novice player to quickly analyse the whole situation. The players were supposed to exchange chips when needed.

Participant 6 mentions that it took them two games to get to know the rules and the interface of the team member. Playing the role of the coordinator then helped to understand the game even more. When in a role of coordinator the player notices that they did not make the best choice in setting the individual goals considering their positions and resources.

Participant 7 complains about not receiving a goal from a coordinator. In this case it was not a system error but a fault of the human coordinator. The participant 9 mentions the game being fast paced and comments about the capabilities of identified human coordinator. Participant 10 found the first game fast paced, mentions a conflict in norms in the game 3 and system error in the game 4.

Participant 11 found the first game too fast for the amount of norms they had to consider. The player complains about conflict in norms in the game 4. As a coordinator the player admits they can set the goals for each member better. The game 8 was affected by a fault of the system.

Participant 12 did not realize they were penalized in the second game. In the game 3 they mention a conflict in norms. As a coordinator the player found it difficult in the time given to set all individual goals. The last game was affected by a system error.

Participant 13 is confused that the order of players in the left column is not fixed, which made him misunderstood about the the number of chips. As a coordinator the player notes that they did not have power over the player to request chips they need. In this game there was an error with agents who did not move or requested chips.

Participant 14 suggest a stronger error feedback would be helpful. The player 15 sees the game competitively and enjoys easy wins. The player 16 admits the first game was not clear. The participant 19 complains about an error in the synchronisation when the deadline was unachievable. The participant 20 mentions that as a coordinator they did not make use all norms they had available. The participant 21 complains about too many norms appearing the right column and did not have enough time to go through them all. The participant 22 is little confused by the rules in the first game. The player found the coordinator not performing well in the last two games. In fact this was caused by a system error.

To summarise what we learned from the comments the participants mostly commented about the interface, the speed, their understanding and penalties.

- **Interface** Participants found the the interface confusing in the beginning but learned how to use it in time *"Took a while to get the rules, and the interface was a little slow to update."*, *"If it's a team game, maybe a good motivation would be to see the global team score rather than the individual score.Also, because you don't carry the points and chips over to the next game, there's not much motivation in trying to improve the score"* and *"I found it difficult to follow all the elements reflected in the rules (time, task, penalization). Interesting idea though and I can relate to other 'established' games, such as checkers."*
- **Understanding** Similarly to the learning to use the interface participants grew accustomed to the rules in time and understood the game getter *"In comparison with the previous two rounds of the game, I think I got a better grasp of the penalties and means of requesting chips. So there is a progression in my understanding of the rules and enjoyment of the game, even though this is not reflected in the score. The idea of reciprocating someone's request for help is interesting and can be traced from one round to the other. "*
- **Speed** Nearly all participant found the game too fast paced and the ticking clock highly distressing *"It was a teeny weeney bit too fast :("*, *"Game was too fast paced"*, *"the obligations keep changing in short period of time. Don't have much time to read it carefully"* and *"I forgot that I would be penalized for moving too fast - I made two moves within one*

*tick, but I understand why I was penalized. It's difficult to remember all of the prohibitions at the same time. Unfortunately the AI players stopped." "I did not consider the relative positions as a coordinator. We ran out of yellow chips. my fault. I randomly assign goal positions."*

- **Penalties** All participant were uncomfortable with conflicting norms and did not like to be penalised *"Goal was impossible to meet without violating a prohibition." "Some goals clash" "The prohibition was not to go on a brown square, I started surrounded by brown squares."*

In the next section we will discuss how results were interpreted in relation to each sub-question specified in the Section 6.1.

## 6.4 Study II. Focus on Extended Flexible Autonomy

In the first study there were only a few situations where the coordinators would be necessitated to use the additional tools (norms). In order to investigate this Study II. is therefore focused on the extended flexible autonomy .

The set of norms and the structure of the teams are different compared to the first study. In this case there are two types of teams (Figure 6.13) both consisting of one human player and three agent players. In the first team T1 an agent acted as the flexible coordinator and in T2 the human participant was the flexible coordinator. The set of norms active in this study was smaller (Figure 6.14) based on the experience from the first study where the participant found the amount of norms overwhelming.

T1: 1 agent flexible coordinator, 1 human team member, 2 agent team members
T2: 1 human flexible coordinator team member, 3 agent team members

Figure 6.13: Study II. The structure of the teams

In this study the board and the chips are set in a specific way and the players have more time to set up the norms at the beginning. The game scenario is as follows: there is a shortage of certain chips or certain player(s) has most of the chips. The flexible coordinator then has an

N1 – Group surround obligation
N2 – Obligation to make a move at least once
N4 – Obligation to respond to requests
N6 – Individual surround obligation

N7 – Individual color obligation
N8 – Individual color prohibition
N9 – Flexible coordinator restriction
N10 – Obligation to accept requests

Figure 6.14: Study II. The collection of game norms

incentive to use the norms to ensure the all players are able to reach their goal and the group norm is achieved.

The players are obliged to make at least one move in the game. The prohibition to move more than once per clock tick was omitted. They will have an obligation to respond to requests. The prohibition to reject requests was omitted in this study because the flexible coordinator can order the players to accept requests. The flexible coordinator chooses goal positions for the team members around the team goal. The leader can also oblige players to accept chips requests and prohibit or oblige them from using tiles with a specific colour.

For this study the participants were recruited from the volunteers who had interacted with the game before. This way they were already familiar with the game objective and it was assumed it would be easier for them to handle the advanced game scenario. We ran 5 trials with one participant per trial. The experiment consists of 3 short games (5 minutes per game). The initial 3 games are played in team T1 with an agent coordinator to ease the participants to the game and the next 3 games are coordinated by the human players. There are 4 types of games designed all with more of less obvious shortage of chips of a particular color (aimed at the norms N7 and N8) and some of the games have uneven distribution of chips amongst the players to encourage the use the obligation N10.

### 6.4.1 Results

In the Table 6.7 are results collected from the Study II. specifically set up to investigate flexible autonomy. The results are overall positive and specifically the participants felt better about the team coordination where Questions 9 and 10 about the team coordination, which are averaging 1.60 respectively 1.67. Compared to the Study I. where the values are 1.83 and 2.25. The teams were also better in goal achievement with overall average of 53.33% compared to 42.5% achieved in the Study I.

| Game | 1 | 2 | 3 | 4 | 5 | 6 | Avg. |
|---|---|---|---|---|---|---|---|
| Question 1 | 1.00 | 1.20 | 2.00 | 2.00 | 1.40 | 1.00 | 1.40 |
| Question 2 | 1.00 | 1.00 | 1.00 | 1.20 | 1.00 | 1.60 | 1.13 |
| Question 3 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Question 4 | 1.00 | 1.20 | 1.25 | 1.00 | 1.00 | 1.00 | 1.07 |
| Question 5 | 2.60 | 2.20 | 2.00 | 1.20 | 1.00 | 1.60 | 1.73 |
| Question 6 | 1.80 | 1.40 | 2.50 | 2.80 | 2.60 | 1.60 | 2.10 |
| Question 7 | 3.00 | 2.20 | 1.50 | | | | 2.27 |
| Question 8 | 1.00 | 1.40 | 1.75 | | | | 1.40 |
| Question 9 | | | | 1.80 | 1.60 | 1.40 | 1.60 |
| Question 10 | | | | 2.00 | 1.80 | 1.20 | 1.67 |
| Goal achieved | 40% | 60% | 40% | 40% | 60% | 80% | 53.33% |

Table 6.7: Study II.: Overall results from all games

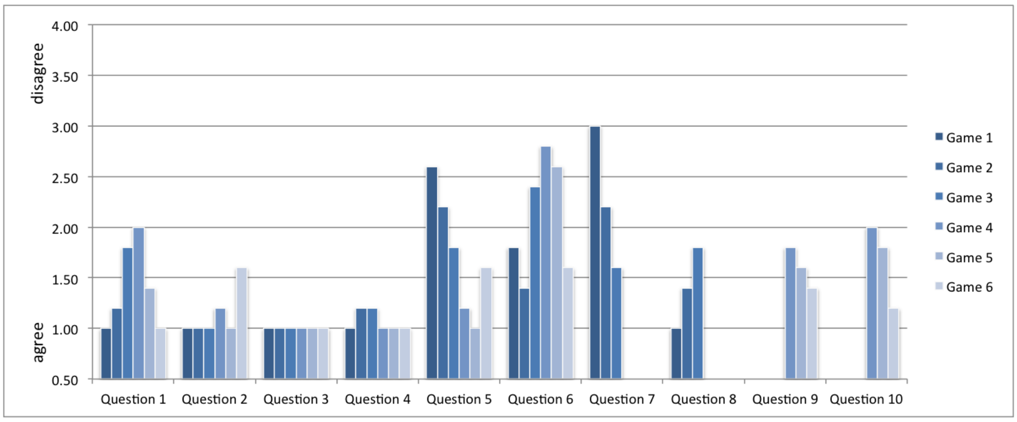


Figure 6.15: Study II.: The development of games

### 6.4.1.1 Question 1: Norms as Coordination Mechanism

The first research question was whether norms are usable as a coordination mechanism in human-agent collectives. The analysis of this objective was split in to two parts: the perspective of the coordinator and the perspective of the team member to evaluate whether players acting in different roles perceived the system differently.

**All players** At first we are going to look into the questions from a general perspective — not related to which role the participant took in the game.

1. Is it possible to facilitate human-agent coordination with norms? Can human-agent collectives achieve some coordinated behaviour?

   The results shown in Table 6.7 indicate that on average the participants scored the team work with 2.10 (Question 6) and enjoyment of the games was rated on average as 1.40 (Question 1). These values where supported by the interviews were participants found the system engaging and capable to be to used for coordination when achieving a group goal.

2. Do human players understand the game being described with the norms?

   The participants on average rated that they understood the rules of the game with 1.13 (Question 2 in Table 6.7) and that they understood the goals of the game with 1.00 (Question 3). This is a very good result. However the team work average (Question 6) was 2.10, which indicates that the participants were satisfied slightly less (In the Study I. the average was 2.03). The interviews confirmed that the players thought both that they understood the games and they also did what was required from them. Compared to the Study I. the players understood the rules and goals of the game better (values 1.36 resp. 1.34). Also the penalization was understood better in the Study II. where players on average understood why they were penalized with rating 1.07 (Question 4) and thought that the penalization was fair with 1.73 (Question 5). In the Study I. the average values were 1.46 and 1.88. Study II. was rated on average better than Study II. in all aspects apart from team work (Question 6) and reasonableness of obligations (Question 7), which can be seen in Figure 6.16 where the two studies are compared. There was a higher amount of conflicting obligations in the Study II., which explains the worse rating.

3. How does the team performance compare when under human or agent leadership?

   Here we compare two games played with human coordinators (Table 6.8 and Figure 6.17) with four games played with agent co-

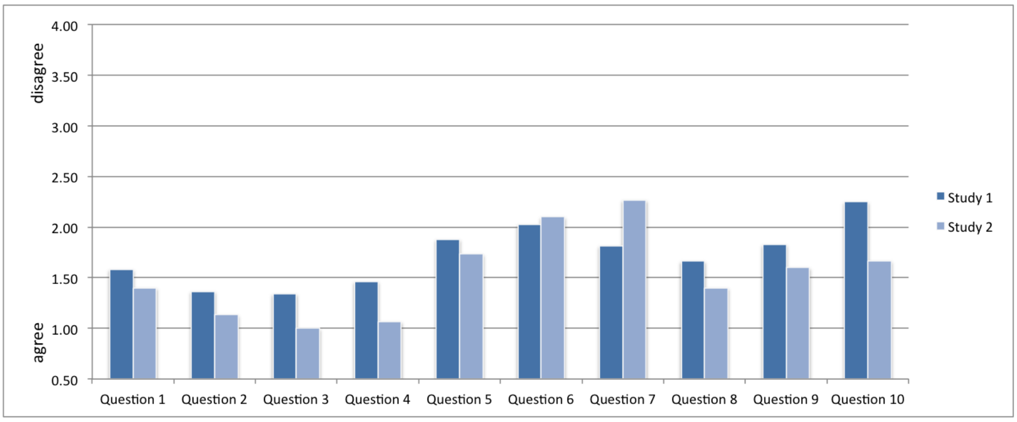


Figure 6.16: Study I. and Study II. compared

ordinators (Table 6.9 and Figure 6.18). The data show that on this occasion the team performed better under human leadership. However, there are a number of factors that make it impossible to draw a strong conclusion. In order to train human coordinators their games were scheduled after the agent games. The average satisfaction with the team work is 2.33 (Question 6) for agent coordinators and 1.90 for human coordinators.

| | Game 1 | Game 2 | Game 3 | Average |
|---|---|---|---|---|
| Question 1 | 2.00 | 1.40 | 1.00 | 1.47 |
| Question 2 | 1.20 | 1.00 | 1.60 | 1.27 |
| Question 3 | 1.00 | 1.00 | 1.00 | 1.00 |
| Question 4 | 1.00 | 1.00 | 1.00 | 1.00 |
| Question 5 | 1.20 | 1.00 | 1.60 | 1.27 |
| Question 6 | 2.80 | 2.60 | 1.60 | 2.33 |
| Question 9 | 1.80 | 1.60 | 1.40 | 1.60 |
| Question 10 | 2.00 | 1.80 | 1.20 | 1.67 |

Table 6.8: Study II.: Results from games with human coordinators

| | Game 1 | Game 2 | Game 3 | Average |
|---|---|---|---|---|
| Question 1 | 1.00 | 1.20 | 2.00 | 1.40 |
| Question 2 | 1.00 | 1.00 | 1.00 | 1.00 |
| Question 3 | 1.00 | 1.00 | 1.00 | 1.00 |
| Question 4 | 1.00 | 1.20 | 1.25 | 1.15 |
| Question 5 | 2.60 | 2.20 | 2.00 | 2.27 |
| Question 6 | 1.80 | 1.40 | 2.50 | 1.90 |
| Question 7 | 3.00 | 2.20 | 1.50 | 2.23 |
| Question 8 | 1.00 | 1.40 | 1.75 | 1.38 |

Table 6.9: Study II.: Results from games with agent coordinators

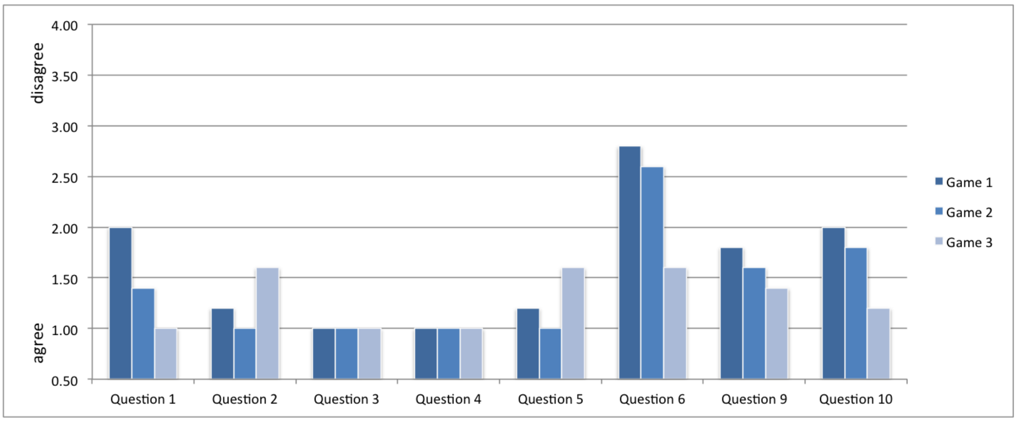


Figure 6.17: Study II.: Development of games with human coordinators

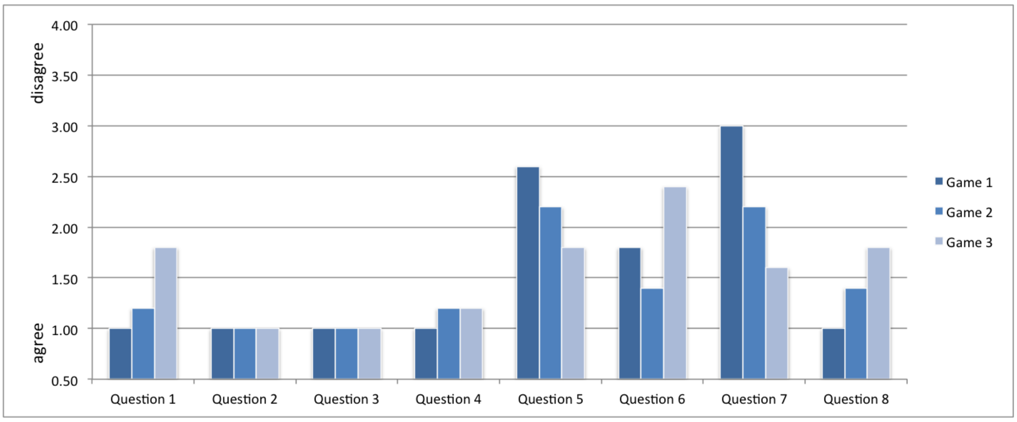


Figure 6.18: Study II.: Development of game with agent coordinators

**Coordinating agent** From the perspective of the coordinators, this study was focused on whether the norms are usable in achieving coordinated behaviour, whether the participants were able coordinate the group to achieve the set group goal. The results are discussed below.

1. Are human coordinators able to direct the group to do what is required for the interest of the group with the norms?

   In this study the human coordinator coordinated a team of agents. The results from three games with human coordinators are shown in Table 6.17, which show how the players felt after they coordinated a game. In the graph in Figure 6.17 we can see that all aspects of the game play improved in the second game. On average the coordinators rated the team work with 2.33 (Question 6), the coordination with 1.60 (Question 9) and flexible coordination with 1.67 (Question 10). These are better results than in Study I. were coordinator rated Question 9 on average with 1.83 and Question 10 with 2.25. In the interviews then players confirmed that now they understood better the use of additional norms for the team coordination. However, they still found the interface confusing and were not always successful in setting all goals and additional obligations / prohibitions. Similarly as in the Study I. the result were affected by system faults where agents did not move. During the interviews after the games the participants were asked whether they felt they can achieve the goal with the norms and it was confirmed they did. However, some of the participants found the interface awkward to work with.

#### 6.4.1.2 Question 2: Flexible Autonomy

The second problem that was analysed is to whether flexible autonomy can be achieved with the use of the norms. The data from the first study show that norms can be used as a medium in coordination with limited resources and not complete information.

1. Are norms usable for supporting flexible autonomy in HACs?

   In this study all games were coordinated by a flexible coordinator. These games include all games where either an agent or a participant played in the role of the flexible coordinator. Data in Table 6.7 shows that the participants on average rated that they understood the rules of the game with 1.13 (Question 2) and that they understood the goals of the game with 1.00 (Question 3). However the team work average (Question 6) was 2.10, which indicates that the participants were satisfied slightly less (In the Study I. the average was 2.03). The interviews confirmed that the players thought both that they understood the games and they also did what was required from them. Compared to the Study I. the players understood the rules and goals of the game with only a slight difference (values 1.36 resp. 1.34). Also the penalization was understood better in the Study II. where players on average understood why they were penalized with rating 1.07 (Question 4) and thought that the penalization was fair with 1.73 (Question 5). In the Study I. the average values were 1.46 and 1.88 . Study II. was rated on average better than Study II. in all aspects apart from team work (Question 6) and reasonableness of obligations (Question 7), which can be seen in Figure 6.16 where the two studies are compared. There was a higher amount of conflicting obligations in the Study II., which explains the worse rating. The coordination was rated by the coordinators with 1.60 (Question 9) and flexible coordination with 1.67 (Question 10), which are better results than in Study I. where the coordinator rated Question 9 on average with 1.83 and Question 10 with 2.25.

From the values, comments and interviews it can be concluded in the Study II. participants better understood how to use the obligations and prohibitions to ensure the group goal is reached. However, similarly as in the Study I. the experiments were affected with system errors.

2. Do human players behave differently following norms from the organisation or agents/players that are acting as team leaders?

   Here we compare three games played with human coordinators (Table 6.8 and Figure 6.17) with three game play with agent coordinators (Table 6.9 and Figure 6.18). From the observations it was clear that games coordinated by agents were smoother and error free which is supported with average score 1.00 for both understanding of the rules of the game (Question 2) and understanding of the goals of the game (Question 3). This was due to the fact that agents never failed to assign a goal or plan the group coordination well, which was sometimes difficult for inexperienced players. Same as in the Study I. the players did not distinguished between a norm originating from the organisation and from the coordinator. The only issue, which they repeatedly reported was a conflict in these norms.

#### 6.4.1.3 Comments Summary

The players in general found it difficult to violate norms even it was for greater good. This was partially because they did not pay attention to the severity of penalties and partially because any violation was seen very negatively by them. This contrasts with the agents' decision making who feel no regret in violating a lower priority goal. In the case of system error most players tried to excuse agents or even took the blame themselves. The transcribed comments are presented in Appendix B.

Participants were making comments on accountability of the players when they tried to justify their actions:*"The agents didn't seem to like the fine for going on white tiles and so they didn't move."*, *"Player 2 didn't seem to* *want to move or give away his blue chips. The others worked together well to get close but couldn't make it as 2 was hogging all the blues!"* and *"I failed the team through rejection."*.

Same as in the previous study the participants found it difficult to violate norms when faced with conflicting norms: *"My goal was to move to a white square but the prohibition said that I should not move to white squares."* and *"Penalty for white, required to be on white at end. Halved final score. :("*.

Another topic that appeared in the comments was the coordination where the participants were rating their own performance as coordinators such as *"My bad"* and *"I was the coordinator - However the agents didn't seem to move this time. They did trade nicely so I managed to get there :)"*.

## 6.5 Additional Findings

There were a number of findings in addition to the originally proposed research questions that are worth mentioning.

The ticking clock caused a lot of stress to the human players when coupled with the deadlines. The stress was then even increased with the prohibition to move more than one square per clock tick. It took several games for each player until they were able to avoid violation of this norm, if ever. This is an example of when players were aware of the norms but found it difficult to obey them.

During the game play and the interviews it became apparent that the majority of the players did not pay attention to the value of the penalties, which is quite contrary to the agents' design. Many participants suggested that they would appreciate some colour indication of the severity and looming deadline. There was also a stark contrast between agents' rational decision making and human decision making. In the games where chips distributed randomly and some unfortunate players started surrounded by tiles with prohibited color the players were deeply affected by this unfairness, which affected their game play. Even though the negative effect of the penalization on the participants was not entirely surprising the extent of that was.

A positive finding then was that the participants did not notice the agents until they did something wrong. On the side note they where programmed to be cooperative and work towards the common goal at the highest priority therefore all wrongdoings were system errors. The participants felt generally altruistic towards the agents while being generally competitive towards the other human participant.

## 6.6 Reflections on Norms in HACs

Reflecting back on our research question whether norms can be used to achieve flexible autonomy in human-agent collectives. We have seen that norms can be used as a regulatory mechanism in the Colored Trails research test bed. With the use of norms both software and human agents were able to achieve a team objective. Coordinators employed norms as a tool to set individual goals and regulate how they are achieved in response to current context. For example, when there was a shortage of blue chips coordinator prohibited players from stepping on blue tiles in unnecessary cases such as if there was an alternative route to the goal.

This online game is necessarily a simplification of a complex scenario human-agent collective would face for example in a disaster response. We assume that in such high level stress situations people may react differently to obligations and prohibitions, however Colored Trails still allowed us to test the underlying fundamentals of group norms that appear both in the game and the real world. We tested the ability of responsible agents to coordinate group to achieve a collective goal while adapting to dynamically changing conditions.

If we want to use the framework in real world scenarios we will have to deal with the difficulty of simplifying context information into a form that is understandable to agents and build resilience to errors that occur. For example, in our Colored Trails game the position of players was clearly defined, however in a real world scenario such as GeoSense we had to interpret positions as discreet locations and this means allowing for GPS errors and inaccuracies. Also in our game human players were able to oversee the complete situation while the environment state of the software agents was very limited. In a real world environment this situation is likely to be reversed.

## 6.7 Summary

To conclude the evaluation chapter the framework / norms can be used as a coordination mechanism in mixed human-agent teams and to facilitate flexible autonomy in human-agent collectives. It was received well by the participants who found the cooperation with the agents interesting and understood the norms and their purpose. We have achieved flexible autonomy in human-agent collectives with norms. The participants usually did not distinguish between human and agent players. If they did they tend to treat agent players kinder than human players who they saw as a competition. The penalties with priorities did not work in the same manner for human players as for the agents and the system can be improved to support human players better and to achieve greater reliability. We mention possible future work and extensions in Section 7.2.

# 7

# Conclusions

In Section 1.1 the top level question to tackle in this thesis was presented:

> *"How norms can be used to achieve context sensitive flexible autonomy in HACs?"*

During the evaluation we found out that even though both human and software members of human-agent collective follow norms in their separate environments when we coordinate HAC with the same set of norms we need to accommodate their differences. Specifically, human agents paid attention to the language of norms and their tone; in general did not like being forced to obey an increasing amount of norms; did not perform best under time pressure; were not assigning same the same priorities to tasks as software agents did; and mainly did non appreciate being sanctioned especially when it was impossible to avoid a violation of a norm.

We have seen that autonomy is a crucial part of agency and also that flexible autonomy is a relatively new a largely unexplored area. We also have seen that human-agent interaction is increasingly important in real world scenarios. Norms have been described as a useful way of regulating multi-agent systems and it is potentially interesting to consider how norms might be used. This gives us a good working understanding of the key principles around norms, multi-agent systems and human-agent interaction. The use of norms in multi-agent systems is relatively well understood area and human-agent interaction is a growing field of research. Therefore it is important to investigate the intersection of these common but currently disconnected areas. Furthermore, specific work regarding group norms would appear to be especially relevant when considering groups of entities working together. We have showed the roles that norms can play in human agent systems, we have also proposed a way to represent norms in terms of conditional obligations and conditional prohibitions, and how a system regulated by norms facilitates flexible autonomy. We have seen how intuitively norms might be engaged with by humans, but when operationalising norms in a multi-agent system, autonomy can be preserved by introducing third party enforcing organisation and specific software agent characteristics. Then in chapter 3 we have presented the implementation of the normHACing framework. We first described the syntax of the N-2APL agent programming language which is a normative extension of 2APL and together with 2OPL forms a normative multi-agent system. This has allowed us to describe a real world example as a normative environment. Next, we have described the implementation of the middle-ware that allows us to practically introduce flexibly autonomous agents into the existing game system.

In chapter 4 we have designed a system of hierarchical group norms, which can be used for coordination in mixed human-agent teams. We have introduced the concept of group norms and provided its taxonomy that builds upon an existing conceptualization. We further extended it to consider monitoring and enforcement of the norms particularly with sanctions and deadlines and have defined a way of representing group norms. Next we have seen how group norms can be applied to the human-agent collective by considering details such as team plans, sanctioning policies, responsible agents and hierarchical norms. Finally we considered the differing implications of group norms on both human agents and software agents.

In chapter 5 we described an extension of normHACing framework for programming group norm-aware multi-agent systems which integrates the GN-2APL norm-aware agent programming language with the G-2OPL language for programming normative organisations, which are able to work with collective norms. We showed how the group norm is implemented and how group norm-aware agents deliberate about group norms. We have illustrated the use of the system on an example from a location based game GeoSense. Group norms were tested with the original GeoSense application while working with agent teams only. Mixed human-agent teams were investigated in the Colored Trails testbed.

Finally the evaluation in chapter 6 concluded the framework / norms can be used as a coordination mechanism in mixed human-agent teams and to facilitate flexible autonomy in human-agent collectives. It was received well by the participants who found the cooperation with the agents interesting and understood the norms and their purpose. We have achieved flexible autonomy in human-agent collectives with norms. The participants usually did not distinguish between human and agent players. If they did they tend to treat agent players kinder than human players who they saw as a competition. The penalties with priorities did not work in the same manner for human players as for the agents and the system can be improved to support human players better. The system is not mature enough to facilitate as a reliable tool that can be used widely.

## 7.1 Contributions

This thesis makes the following contributions to the field of computer science:

1. **Norms in HAC**

   We show how norms may be used in HAC to achieve flexible autonomy. Norms are widely used to regulate multi-agent systems. Human society is accustomed to being governed with norms in the form of laws, rules or guidelines. However, to the best of our knowledge norms have not previously been used to coordinate human-agent collectives. We presented a representation of norm that can be used coordinate a human-agent collective. Such a norm can be used a real world scenario like a disaster response.

2. **Hierarchical group norms**

   We have designed a system of hierarchical group norms, which may be used for coordination in mixed human-agent teams. Group norms are addressed to a group of agents. The team receives a team goal (group obligation), which is then split into individual goals (individual obligations) the fulfilment of which leads to the fulfilment of the parent group norm and therefore the achievement of the group team goal. These norms may be stacked hierarchically — a group norm can split into a combination of individual and group norms. The developer of agents can therefore think in terms of groups and not individual agents.

3. **Demonstration of the normHACing framework in practice**

   The application of the framework was illustrated on two examples and one of which Colored Trails was used in the evaluation. It was experimentally demonstrated that norms may be used as a coordination mechanism in human-agent collectives and to facilitate flexible autonomy in human-agent collectives.

4. **Implementation of framework for norm-aware agents**

   We implemented N-2APL and developed a framework for programming norm-aware multi-agent systems which integrates the N-2APL norm-aware agent programming language with the 2OPL language for programming normative organisations. To the best of our knowledge, this is the first implementation of an integrated framework for norm-aware multi-agent systems in which autonomous agents deliberate about whether to conform to the norms imposed by a normative organisation. This modular framework can be used with different applications.

## 7.2 Future Work

In this section, we outline some of the most interesting possible future directions in the fields of human-agent interaction and norm-aware agents. These possible directions are open challenges identified during the realization of this thesis.

### 7.2.1 Human-Agent Interaction

**Human-agent team dynamics** During the evaluation we have

**Physiological context**

Our flexible autonomy changed depending on the context. One of the aims of flexible autonomy is to ensure human-agent interaction maximizes the team performance. As we are aware, human cognitive abilities are impaired when facing great pressure. Reading the level of stress of humam members of HACs and giving greater autonomy to agents at that point may prevent some human errors.

**Human friendly norms**

Although the framework was received well by the participants who found the cooperation with the agents interesting and understood the norms and their purpose. However, the system is not mature enough to facilitate as a reliable tool that can be used widely. The penalties with priorities did not work in the same manner for human players as for the agents and the system can be improved to support human players better.

While human agents naturally understand rules specified by norms their motivation to comply is different to the motivation of software agents. Therefore human players cannot be regulated in the same way in order to achieve best results. Experimental results gathered in this thesis suggest more sensitive approach such as accompanying goals with explanations and sparse sanctions will yield better outcome.

**Human-robot interaction**

In our system we presented software agents as embodied agents in the form of avatars. When we consider an aftermath of a major natural disaster as one of the potential applications the framework, we have to evaluate the system with robotic agents such as UAVs with a vision system or UGVs with a manipulator arms.

The research area of Human-Robot Interaction (HRI) or Human-Robot Collaboration (HRC) is growing flexible autonomy is seen as one the ways to achieve smooth and safe collaboration. Conditional norms can be used define safe operating modes depending on the nature of environment and constantly changing conditions. Further more, norms can be used to create personalised interaction regimes to maximize comfort of individuals and productivity.

### 7.2.2 Norm-Aware Agents

**Development of agent personalities**

**Fully-aware agents**

Fully as defined in Section 4.8.2 are able to observe actions of the other agents in the group. As such they can then dynamically adjust the priority of the norm according to the current situation. For example, when an agent becomes aware that a particular group goal cannot be achieve because other team members are working towards different goal (with higher priorities in their configuration) it will downgrade the priority of it current goal (which is part of unachievable group goal) and use its time and resources towards an achievable goal with a lower priority initially. Fully-aware agents will be able to better adapt to their continuously changing environment. The formal model of the framework will need to extended accordingly to ensure agent's set of plans always remains optimal with maximum utility.

From implementation point of view it can be explored if a fully group-aware agents' deliberation can be achieved with a modification of the preference ordering function, which agents use to assign priorities to sanctions. This static function can be modified to include rules and the resulting value would change dynamically. It will need to be explored how this alteration affects tractability of the deliberation algorithm.

**Interface for norm encoding suitable for non-developers**

One of the weaknesses of the current system is that norms need to be predefined by a developer a-priory. In the next generation of the framework encoding of norms should be made easier, for example, with use of visual programming tools. In such a system human-agent coordinators will be able to construct a suitable rules themselves. Enabling such a creation will bring a new challenge in preventing agents from creating arbitrary norms.

### Distributed monitoring

At the moment the deliberation mechanism of the agents is based solely on the priorities of the norms (the priorities of the sanctions associated with their violation to be precise). Agents do not take into account any further information or progress of the plan completion. There are at least two approaches that can be taken. The first one is to let agents monitor the actions of the other agents in the environment themselves while the second option would be to engage the normative organization.

G-2OPL organization is capable of monitoring the actions (states) that are shared in tuple space. It would however require some complex changes to the design of the organization to enable more specific compliance checking. For example, if G-2OPL was to decide which agents from the group are to blame for a violated obligation it would have to have access to the agent plans to find out at which point the agent stopped executing its plan and if it was agent's fault or not.

It would be desirable that the system penalizes agents that made a choice not to follow the plan as opposed to agents that attempted to complete the task but one of the steps was not successful and it was beyond the control of the agent. The organization would therefore need to be informed, which steps an agent needs to take to fulfil an obligation and which of them he attempted to make. The second and the preferred option is to implement the capability of monitoring the progress of the execution of a joint plan in the MAS. The idea is to let agents observe others agents' action. In this approach agents as a group are responsible for establishing who is to blame for a violated norm.

### Deadlines

In our system we consider only time specific deadlines. For example, If we want an agent to go to a shop today we can say that an agent is obliged to go to a shop before 5pm. With fact based deadlines we will be able to say an agent is obliged to go to a shop while it is opened.

Prohibitions with an unlimited validity are another limitation of the system. For example, we cannot define prohibition to walk on grass while it is raining.

Fact based deadlines and deadlines in prohibitions were not considered due to tractability of N-2APL. The plausibility and implications of such an extension were not explored in this thesis.

## 7.3 Publications of the Author

Dybalova, D., Testerink, B., Dastani, M., and Logan, B. (2014). A framework for programming norm-aware multi-agent systems. In *Coordination, Organizations, Institutions, and Norms in Agent Systems IX, 2014*, pages 364–380. Springer.

Lee, J., Padget, J., Logan, B., Dybalova, D., and Alechina, N. (2014). Run-time norm compliance in BDI agents. In Proceedings of *International conference on Autonomous agents and multi-agent systems, 2014*, pages 1581–1582. International Foundation for Autonomous Agents and Multiagent Systems.

Lee, J., Padget, J., Logan, B., Dybalova, D., and Alechina, N. (2014). N-Jason: Run-Time Norm Compliance in AgentSpeak (L). In *Engineering Multi-Agent Systems*, pages 367–387. Springer International Publishing.

# Appendices

# A

# Appendix

```
%group surround
norm(
   group_surround(G),
   G,
   ( group(G), goal(X,Y,Z)),
   obligation([surround(X,Y)],now + 18,[reduce(G,700)])
   ).

%agents group norms from RA
norm(
   group_task(group_surround(g)),
   Agent,
   ( pursuer(Agent), detached(group_surround(W)),goal(GX,GY,Z),agentSTask(Agent,
      X,Y,goal(GX,GY),S)),
   obligation([at(X,Y,Agent)],now + 14,[reduce(Agent, 700)])
   ).

%individual obligations to respond to requests per each
norm(
    respondToRequest(Thing),
Thing,
( pursuer(Thing), proposal(Id,_,Thing,_),group(Z)),
    obligation([respondToRequest(Id)],now + 3,[reduce(Thing,300)])
).

%individual obligation to make a move during move phase
norm(
    makeMove(Thing),
Thing,
    ( pursuer(Thing), at(X,Y,Thing),group(Z), not detached(makeMove(Thing))),
obligation([makeMove(X,Y,Thing)],now + 10,[reduce(Thing,300)])
).

%individual prohibition to move more than one field per turn
norm(
    moveTooFast(Thing),
    Thing,
( pursuer(Thing),group(Z)),
    prohibition([moveTooFast(Thing)],[reduce(Thing,300)])
).

RAAA:
%group surround
norm(
   group_surround(G),
   G,
   ( group(G), goal(X,Y,Z)),
   obligation([surround(X,Y)],now + 18,[reduce(G,700)])
   ).

%agents group norms from RAAA
norm(
   group_task(group_surround(g)),
   Agent,
   ( pursuer(Agent), detached(group_surround(W)),goal(GX,GY,Z),agentSTask(Agent,
      X,Y,goal(GX,GY),S)),
   obligation([at(X,Y,Agent)],now + 14,[reduce(Agent, 700)])
   ).

%prohibition for RAAA to excessively restrict a players
norm(
  restrictPlayer(Thing),
  Thing, // the subject agent
  ( ra(_,Thing,raaa)),
  prohibition([restrictPlayer(Thing)],[reduce(Thing,300)]) // prohibition format
).
%individual prohibitions to use colored tiles RAAA
```

```
norm(
    not_enter_tile(Thing),
Thing,
    ( pursuer(Thing), color(Thing,Color,no,S),group(Z)),
    prohibition([color(Color,Thing)],[reduce(Thing,S)])
).

%individual obligations to use colored tiles raaa
norm(
    enter_tile(Thing),
Thing,
    ( pursuer(Thing), color(Thing,Color,yes,S),group(Z)),
    obligation([color(Color,Thing)],now + 15,[reduce(Thing,S)])
).

%individual obligations to accept requests raaa
norm(
    acceptRequests(Thing),
Thing,
    ( pursuer(Thing), groupObl(acceptRequests,Sanction)),
    obligation([acceptRequests(Thing)],now + 15,[reduce(Thing,Sanction)])
).

%individual obligations to respond to requests per each
norm(
    respondToRequest(Thing),
Thing,
    ( pursuer(Thing), proposal(Id,_,Thing,_),group(Z)),
    obligation([respondToRequest(Id)],now + 3,[reduce(Thing,300)])
).

%individual obligation to make a move during move phase
norm(
    makeMove(Thing),
Thing,
    ( pursuer(Thing), at(X,Y,Thing),group(Z), not detached(makeMove(Thing))),
    obligation([makeMove(X,Y,Thing)],now + 10,[reduce(Thing,300)])
).

%individual prohibition to move more than one field per turn
norm(
    moveTooFast(Thing),
Thing,
    ( pursuer(Thing),group(Z)),
    prohibition([moveTooFast(Thing)],[reduce(Thing,300)])
).
```

Listing A.1: group norms G-2OPL

```
sanction(Agent,P) :−
  pursuer(Agent),
  points(Agent, Health),
  NewHealth is Health − P,
  clock(Now),
  retract(points(Agent, Health)),
  assert(points(Agent, NewHealth)),
  @external(ctMW,write(points(Agent,Now,NewHealth),−1),_).

sanction([reduce(Group,P)],viol(agent_directed(Label,Agent,obligation(State,D,
      Sanction)))) :−
  group(Group),
  sanction_group(g,Sanction).

do_sanction([],P,Group,0,S):−
   ra(Group,RA,_),
   sanction(S,RA).
do_sanction([H|T],A,Group,L,X):−
   sanction(H,A),
   do_sanction(T,A,Group,L,X).
do_sanction([],_,Group,L,X).

policy(Group,P,Label):−
 findall(M,(violated(group_task(Label),M),group_member(Group,M)),Vs),
 length(Vs,L),
 S is P / L,
 do_sanction(Vs,S,Group,L,P).

sanction_group([reduce(Group,P)],Label):−
 policy(Group,P,Label).
```

Listing A.2: Sanctions G-2OPL

# B

# Appendix

## Study I. Comments

Participant 3:

1 TM: Took a while to get the rules, and the interface was a little slow to update.

3 TM: I didn’t understand the value of being at a certain place by a certain time. Also some of the rules seemed contradictory - go to blue / do not go on a blue square

5 C: I liked telling them what to do. They did it.

6 TM: The goal made more sense now that I have been coordinator in a previous game

7 FC: I assumed that the other players would do what they were supposed to, but they didn’t. I wanted more time to coordinate before the moving phase began - there wasn’t time to think about what colour restrictions or obligations should be put in place.

8 TM: Some of the team are not performing to the best of their abilities. :(

Participant 4:

1 TM: I did not receive a goal or requests for chips from other players, so couldn't do much apart from move when I was required by obligations.

2 TM: The obligation to respond before a certain clock tick was very difficult to do (possibly impossible) because by the time I saw the obligation its deadline had already passed. The "move before tick 16" obligation was impossible to meat since the move phase started well after that.

3 TM: Finally a game configuration that doesn't hate me too much.

4 TM: Goal was impossible to meet without violating a prohibition.

6 C: I was unable to receive white chips even when the other team members were accepting my proposition to give them to me.

7 TM: Worst coordinator ever!

8 FC: I could not force 2 of the agents to move. So they just stayed there, taking penalties and failing to meet their goals.

Participant 5:

1 TM: It was really hard to move, I had to try it about 10 times

2 TM: If it's a team game, maybe a good motivation would be to see the global team score rather than the individual score.Also, because you don't carry the points and chips over to the next game, there's not much motivation in trying to improve the score

3 C: I tried to make it so that players have to reach the goal that's easier for them, but I was a bit slow.This task is much more stressful than just moving. It took me a lot of time to notice that I had a chip request from another player and I'm sorry that I impaired their move.

4 TM: I was missing a chip and requested it. The request was accepted but I didn't actually see the chip. How frustrating! Also, I'd be interested in knowing the other player's goals as I wouldn't request chips from players who I know need them. It also seems that sometimes, player numbers disappear from the screen, and I lose track of them.

5 FC: I tried to give team members a goal that I felt was realistic given their positions and their chips. As a coordinator, I also could give some of my chips to the players who needed them the most. I didn't see the point of adding norms and also felt like I ran out of time, so I skipped that part.
6 TM: It's a bit frustrating that, when the coordinator has set up your goal but not the others, you still can't move.

Participant 6:

1 TM: I found it difficult to follow all the elements reflected in the rules (time, task, penalization). Interesting idea though and I could relate to other 'established' games, such as checkers.
2 TM: I am still unsure about penalization, but the rules make more sense after the second game.
3 TM: In comparison with the previous two rounds of the game, I think I got a better grasp of the penalties and means of requesting chips. So there is a progression in my understanding of the rules and enjoyment of the game, even though this is not reflected in the score. The idea of reciprocating someone's request for help is interesting and can be traced from one round to the other.
4 C: This round I had to coordinate the other players by indicated their desired position; this was initially confusing, as I did assign a desired position to all players in the required time. Also, since the board showed everyone's resources, this could have been better used in setting the goals.
5 TM: The more familiar the game is the more easy I find it to follow rules, points and requests. With the rules, there seems to be a maximum number of 4 beyond which there might be some overload.
6 FC: As a coordinator I was able to instruct everyone to aim for a goal, but I should have paid more attention to people's positions and resources. It's good it works to host two players on the same chip/ position.

Participant 7:

6 TM: no goal established.

Participant 9:

1 TM: It was a teeny weeney bit too fast :(
6 TM: Ben is an idiot

Participant 10:
1 TM: Game was too fast paced
3 TM: Some goals clash
4 TM: I couldnt move due to a bug

Participant 11:
1 TM: time constrained, do not even read through all obligation/prohibitions
4 TM: I am asked to move to a white without using white chip!
6 FC: I did not consider the relative positions as a coordinator.
We ran out of yellow chips. my fault. I randomly assign goal
positions.
8 FC: Some Agents are not moving.

Participant 12:
1 TM: I think everyone ran out of the final square colour so I
didn't make it.
2 TM: I don't think I was penalized
4 TM: The prohibition was not to go on a brown square, I started
surrounded by brown squares.
5 FC: I think I took too long deciding on where people should go.
I didn't realise I had to set a position for every player - so
I lead the team to fail!
6 TM: I think everyone was short of yellow chips and so it was
difficult for us to complete our obligations. Perhaps with better
teamwork we could have had one of the players make it.
7 FC: We achieved our goal, everyone seemed to have roughly the
chips they needed - I don't think players needed to make many requests.
8 TM: I forgot that I would be penalized for moving too fast - I
made two moves within one tick, but I understand why I was penalized.
It's difficult to remember all of the prohibitions at the same
time. Unfortunately the AI players stopped.

Participant 13:

2 TM: The order of the players on the left changes with each game that is run. This means that I was confused by which player I was on the left. Not checking resulted in me misuderstanding the number of chips I had. A message of chip gained would be useful.
6 FC: I only slightly agree with the last statement as it I have no control of the players requesting their chips in time

Participant 14:
2 TM: i think that stronger error feedback would make life easier

Participant 15:
1 TM: Did not see the penalisation? Did not partake in team work.
2 TM: Clock tick is 10 seconds. I got a nice gift
3 TM: Too many chips given to me. I did not need them. Easy to finish (one move)
4 C: Woops.
5 TM: No comment.
6 FC: I am the best coordinator.
7 TM: One move win again.

Participant 16:
1 TM: im curious about what was wil
2 TM: im more cleared about this round
3 TM: so excited to give away my chips!
4 TM: seems too easy to reach the goal for the 4th game? :)
6 TM: fun fun fun
7 FC: how about super big board and massive player like mmo?

Participant 19:
5 TM: It said to complete a task before time 34 but said game ended at 23

Participant 20:
7 FC: I didn’t restrict colour squares for everyone

Participant 21:
1 TM: I don't understand what the variety of color purpose in this game.Overall, it was nice if i'd be given more time.
2 TM: More understand than previous game.
5 TM: the obligations at the top right corner keep changing every time.
7 TM: the obligations keep changing in short period of time. Don'have much time to read it carefully

Participant 22:
1 TM: first time try. still a little bit confused about the rules. Maybe can do it better next time
2 TM: If i can request a chip fast, it can be better.
3 TM: i stay the target at the end. nice round
4 TM: nice round
5 C: the first time to be team leader, try to manage it well. it's a nice round anyway
6 TM: the director of this game can manage the team goal better
7 TM: bad round without achieving the target final goal

## Study II. Comments

Participant 23:
1 TM: My goal was to move to a white square but the prohibition said that I should not move to white squares.
2 TM: I don't think it was possible for me to get to where the goal was.
3 FC: I was the coordinator - However the agents didn't seem to move this time. They did trade nicely so I managed to get there :)
4 FC: The agents didn't seem to like the fine for going on white tiles and so they didn't move.
5 FC: Player 2 didn't seem to want to move or give away his blue chips. The others worked together well to get close but couldn't make it as 2 was hogging all the blues!

Participant 26:

1 TM: Standard easy win.  No trading.

2 TM: Penalty for white, required to be on white at end.  Halved final score.  :(

3 TM: I failed the team through rejection.

4 FC: My bad.

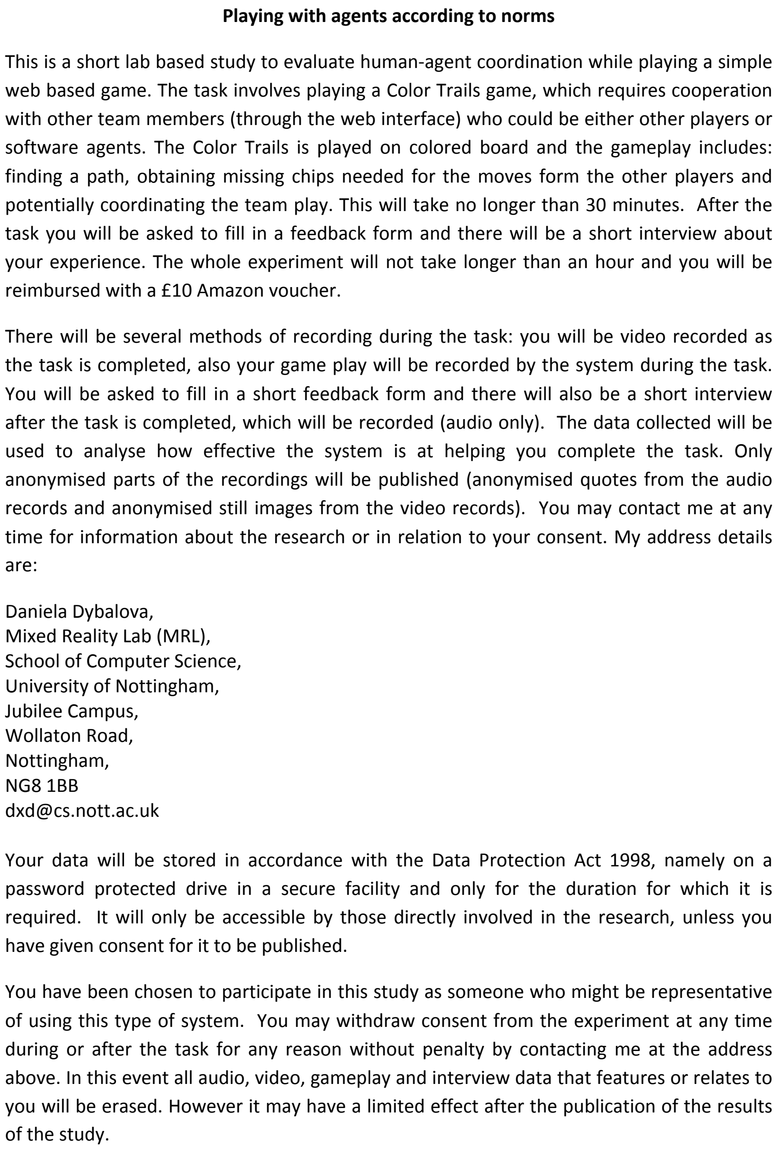
5 FC: Stupid agents not asking for help.  Not my fault this time around.

6 FC: \o/

# C

# Appendix

## Information For Participants



**Playing with agents according to norms**

This is a short lab based study to evaluate human-agent coordination while playing a simple web based game. The task involves playing a Color Trails game, which requires cooperation with other team members (through the web interface) who could be either other players or software agents. The Color Trails is played on colored board and the gameplay includes: finding a path, obtaining missing chips needed for the moves form the other players and potentially coordinating the team play. This will take no longer than 30 minutes. After the task you will be asked to fill in a feedback form and there will be a short interview about your experience. The whole experiment will not take longer than an hour and you will be reimbursed with a £10 Amazon voucher.

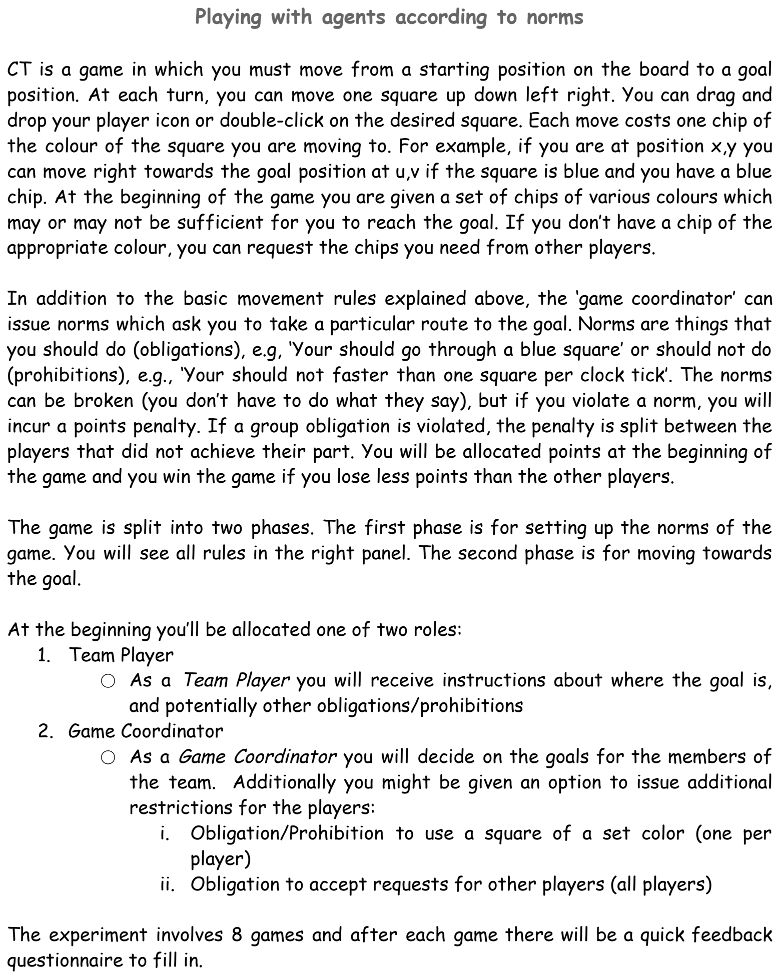
There will be several methods of recording during the task: you will be video recorded as the task is completed, also your game play will be recorded by the system during the task. You will be asked to fill in a short feedback form and there will also be a short interview after the task is completed, which will be recorded (audio only). The data collected will be used to analyse how effective the system is at helping you complete the task. Only anonymised parts of the recordings will be published (anonymised quotes from the audio records and anonymised still images from the video records). You may contact me at any time for information about the research or in relation to your consent. My address details are:

Daniela Dybalova,
Mixed Reality Lab (MRL),
School of Computer Science,
University of Nottingham,
Jubilee Campus,
Wollaton Road,
Nottingham,
NG8 1BB
dxd@cs.nott.ac.uk

Your data will be stored in accordance with the Data Protection Act 1998, namely on a password protected drive in a secure facility and only for the duration for which it is required. It will only be accessible by those directly involved in the research, unless you have given consent for it to be published.

You have been chosen to participate in this study as someone who might be representative of using this type of system. You may withdraw consent from the experiment at any time during or after the task for any reason without penalty by contacting me at the address above. In this event all audio, video, gameplay and interview data that features or relates to you will be erased. However it may have a limited effect after the publication of the results of the study.

Figure C.1: Information Sheet



**Playing with agents according to norms**

CT is a game in which you must move from a starting position on the board to a goal position. At each turn, you can move one square up down left right. You can drag and drop your player icon or double-click on the desired square. Each move costs one chip of the colour of the square you are moving to. For example, if you are at position x,y you can move right towards the goal position at u,v if the square is blue and you have a blue chip. At the beginning of the game you are given a set of chips of various colours which may or may not be sufficient for you to reach the goal. If you don't have a chip of the appropriate colour, you can request the chips you need from other players.

In addition to the basic movement rules explained above, the 'game coordinator' can issue norms which ask you to take a particular route to the goal. Norms are things that you should do (obligations), e.g, 'Your should go through a blue square' or should not do (prohibitions), e.g., 'Your should not faster than one square per clock tick'. The norms can be broken (you don't have to do what they say), but if you violate a norm, you will incur a points penalty. If a group obligation is violated, the penalty is split between the players that did not achieve their part. You will be allocated points at the beginning of the game and you win the game if you lose less points than the other players.

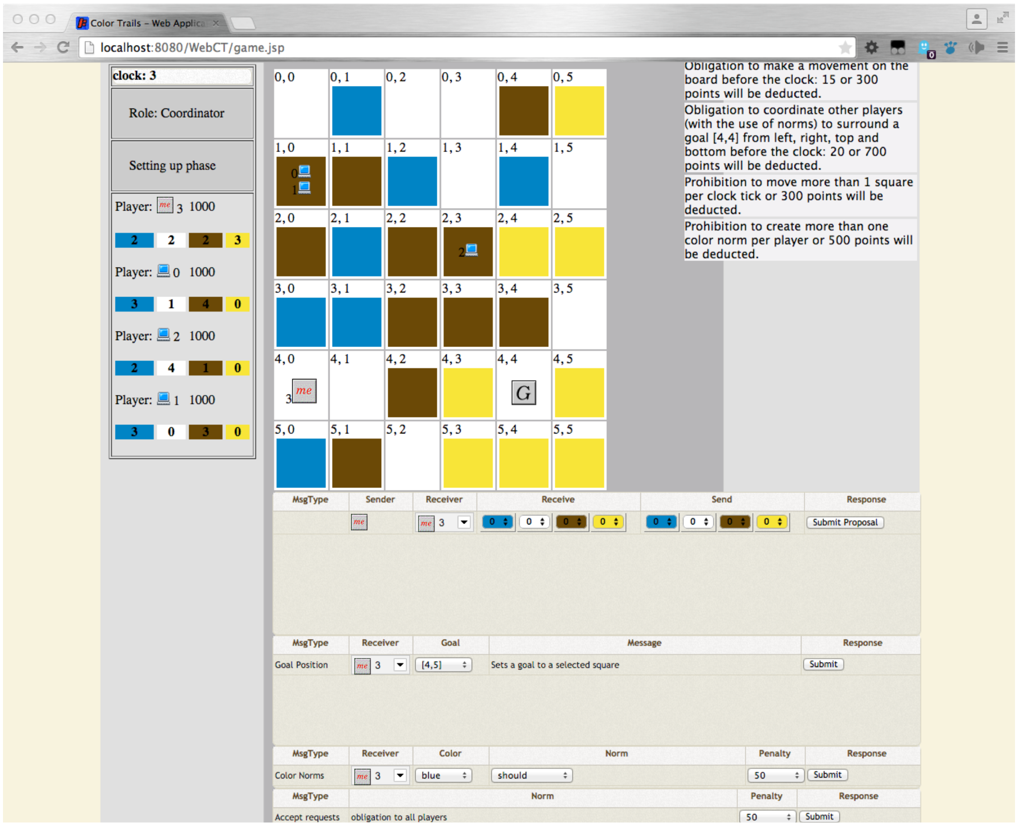
The game is split into two phases. The first phase is for setting up the norms of the game. You will see all rules in the right panel. The second phase is for moving towards the goal.

At the beginning you'll be allocated one of two roles:

1. Team Player
    - ○ As a *Team Player* you will receive instructions about where the goal is, and potentially other obligations/prohibitions
2. Game Coordinator
    - ○ As a *Game Coordinator* you will decide on the goals for the members of the team. Additionally you might be given an option to issue additional restrictions for the players:
        - i. Obligation/Prohibition to use a square of a set color (one per player)
        - ii. Obligation to accept requests for other players (all players)

The experiment involves 8 games and after each game there will be a quick feedback questionnaire to fill in.

Figure C.2: Game Instructions



**clock: 3**

Role: Coordinator

Setting up phase

| | | | | |
|---|---|---|---|---|
| Player: me 3 | 1000 | | | |
| | 🟦 2 | 2 | 🟫 2 | 🟨 3 |
| Player: 0 | 1000 | | | |
| | 🟦 3 | 1 | 🟫 4 | 🟨 0 |
| Player: 2 | 1000 | | | |
| | 🟦 2 | 4 | 🟫 1 | 🟨 0 |
| Player: 1 | 1000 | | | |
| | 🟦 3 | 0 | 🟫 3 | 🟨 0 |

| | | | | | |
|---|---|---|---|---|---|
| 0, 0 | 0, 1 🟦 | 0, 2 | 0, 3 | 0, 4 🟫 | 0, 5 🟨 |
| 1, 0 🟫 0 1 | 1, 1 🟫 | 1, 2 🟦 | 1, 3 | 1, 4 🟦 | 1, 5 |
| 2, 0 🟫 | 2, 1 🟦 | 2, 2 🟫 | 2, 3 🟫 2 | 2, 4 🟨 | 2, 5 🟨 |
| 3, 0 🟦 | 3, 1 🟦 | 3, 2 🟫 | 3, 3 🟫 | 3, 4 🟫 | 3, 5 |
| 4, 0 3 me | 4, 1 | 4, 2 🟫 | 4, 3 🟨 | 4, 4 G | 4, 5 🟨 |
| 5, 0 🟦 | 5, 1 🟫 | 5, 2 | 5, 3 🟨 | 5, 4 🟨 | 5, 5 🟨 |

Obligation to make a movement on the board before the clock: 15 or 300 points will be deducted.

Obligation to coordinate other players (with the use of norms) to surround a goal [4,4] from left, right, top and bottom before the clock: 20 or 700 points will be deducted.

Prohibition to move more than 1 square per clock tick or 300 points will be deducted.

Prohibition to create more than one color norm per player or 500 points will be deducted.

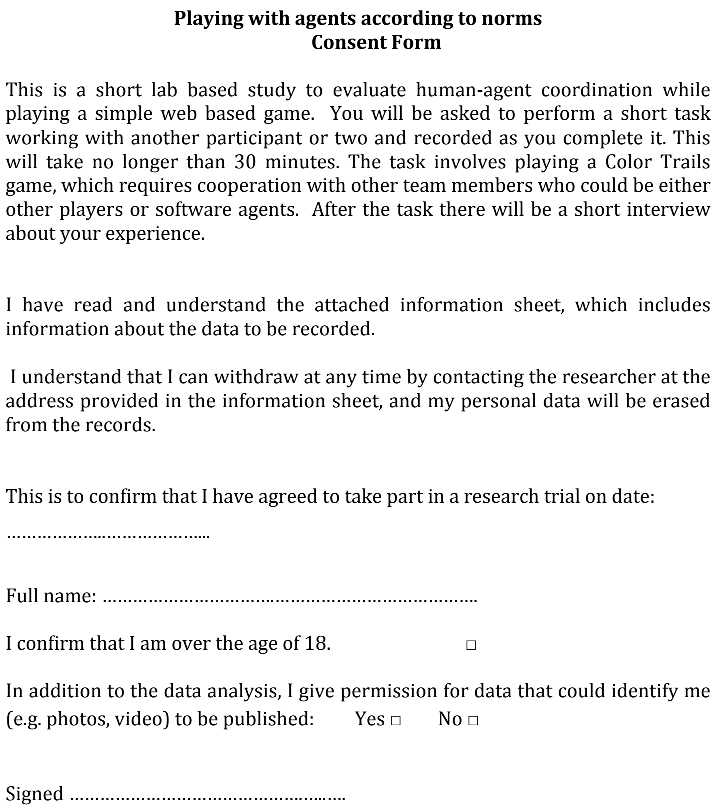
| MsgType | Sender | Receiver | Receive | | | | Send | | | | Response |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | me | me 3 | 🟦 0 | 0 | 🟫 0 | 🟨 0 | 🟦 0 | 0 | 🟫 0 | 🟨 0 | Submit Proposal |

| MsgType | Receiver | Goal | Message | Response |
|---|---|---|---|---|
| Goal Position | me 3 | [4,5] | Sets a goal to a selected square | Submit |

| MsgType | Receiver | Color | Norm | Penalty | Response |
|---|---|---|---|---|---|
| Color Norms | me 3 | blue | should | 50 | Submit |

| MsgType | Norm | Penalty | Response |
|---|---|---|---|
| Accept requests | obligation to all players | 50 | Submit |

Figure C.3: Game Interface



**Playing with agents according to norms**
**Consent Form**

This is a short lab based study to evaluate human-agent coordination while playing a simple web based game. You will be asked to perform a short task working with another participant or two and recorded as you complete it. This will take no longer than 30 minutes. The task involves playing a Color Trails game, which requires cooperation with other team members who could be either other players or software agents. After the task there will be a short interview about your experience.

I have read and understand the attached information sheet, which includes information about the data to be recorded.

I understand that I can withdraw at any time by contacting the researcher at the address provided in the information sheet, and my personal data will be erased from the records.

This is to confirm that I have agreed to take part in a research trial on date: ……………..………………..

Full name: ………………………………..…………………………..

I confirm that I am over the age of 18. □

In addition to the data analysis, I give permission for data that could identify me (e.g. photos, video) to be published: Yes □ No □

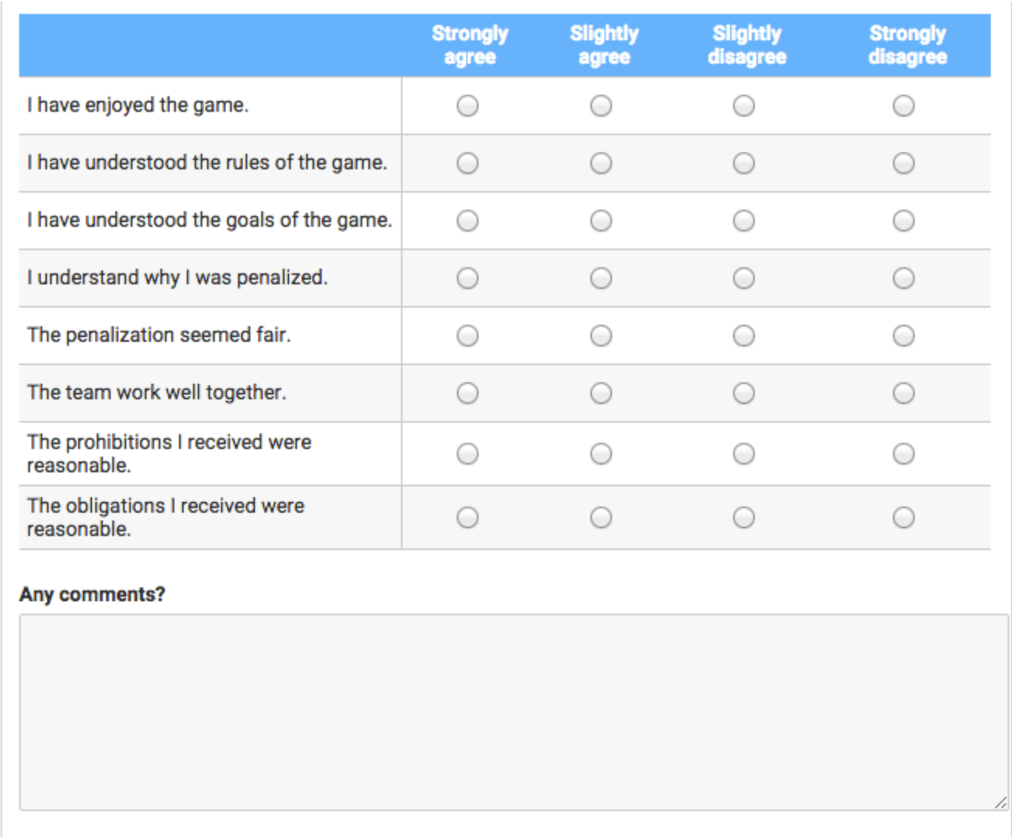
Signed ……………………………………….…..

Figure C.4: Consent Sheet

# D

# Appendix

## Feedback Form



| | Strongly agree | Slightly agree | Slightly disagree | Strongly disagree |
|---|---|---|---|---|
| I have enjoyed the game. | ○ | ○ | ○ | ○ |
| I have understood the rules of the game. | ○ | ○ | ○ | ○ |
| I have understood the goals of the game. | ○ | ○ | ○ | ○ |
| I understand why I was penalized. | ○ | ○ | ○ | ○ |
| The penalization seemed fair. | ○ | ○ | ○ | ○ |
| The team work well together. | ○ | ○ | ○ | ○ |
| The prohibitions I received were reasonable. | ○ | ○ | ○ | ○ |
| The obligations I received were reasonable. | ○ | ○ | ○ | ○ |

**Any comments?**

Figure D.1: Feedback form

# E

# Appendix

## N-2APL Operational semantics

In this section, we present the operational semantics of N-2APL in terms of a transition system. The definitions and transition rules were presented by Alechina et al. [2012] and are based on operational semantics of 2APL [Dastani, 2008].

## N-2APL Configuration

The configuration of an individual agent consists of its identifier, beliefs, goals, plans, substitutions that result from queries to the belief and goal bases, and events.

**Definition 1** (agent configuration)**.** *The configuration of a N-2APL agent is defined as* $A_\iota = \langle \iota, \sigma, \gamma, \Pi, \xi, \succ, PG, PC, PR \rangle$ *where* $\iota$ *is the agent's identifier,* $\sigma$

*is a set of belief expressions ⟨belief⟩ representing the agent's belief base, $\gamma$ is a list of goal expressions ⟨goal⟩representing the agent's goal base, $\Pi$ is the agent's plan base consisting of a set of plan entries (⟨plan⟩, ⟨goal⟩, ⟨pgrule⟩) representing the agent's plans together with their next execution times, $\xi$ is the agent's event base containing also elements of the form $prohibition(p,s)$, $\succ$ is the preference ordering, PG is the set of planning goal rules, PC is a set procedure call rules, and PR is a set of plan repair rules.*

**Definition 2** (multiagent system configuration)**.** *Let $A_\iota$ be the configuration of agent $\iota$ and let $\chi$ be the state of the agents' organisation. The configuration of a N-2APL multiagent system is defined as $\langle A_1,\ldots,A_n,\chi\rangle$.*

**Definition 3** (initial configuration)**.** *Let $\iota$ be the identifier of an agent that is implemented by a N-2APL program. Let $\sigma$ be the set of ⟨belief⟩-expressions specified in the N-2APL program and $\gamma$ be the list of ⟨goal⟩-expressions from the same program. Then, the initial configuration of agent $\iota$ is defined as tuple $\langle \iota,\sigma,\gamma,\emptyset,\emptyset\rangle$. Let also $\chi$ be a set of facts and $A_1,\ldots,A_n$ be the initial configurations of agents $1,\ldots,n$ that are specified in the multiagent system program. The initial configuration of the multiagent systems is defined as tuple $\langle A_1,\ldots,A_n,\chi\rangle$.*

## Transition Rules for Basic Actions

The following transition rules specify transitions for basic actions.

### Skip Action

$$\frac{\gamma \models_g \mathcal{G}(r)}{\langle \iota,\sigma,\gamma,\{(\texttt{skip},r,\ id)\},\theta,\xi\rangle \longrightarrow \langle \iota,\sigma,\gamma,\{\},\theta,\xi\rangle} \tag{E.1}$$

### Belief Update Action

A successful execution of a belief update action $\alpha$ is defined as follows.

$$\frac{T(\alpha\theta,\sigma)=\sigma' \ \&\ \gamma \models_g \mathcal{G}(r)}{\langle \iota,\sigma,\gamma,\{(\alpha,r,id)\},\theta,\xi\rangle \longrightarrow \langle \iota,\sigma',\gamma',\{\},\theta,\xi\rangle} \tag{E.2}$$

An unsuccessful execution of a belief update action $\alpha$ is defined as follows.

$$\frac{T(\alpha\theta,\sigma) = \bot \ \&\ \gamma \models_g \mathcal{G}(r)}{\langle \iota,\sigma,\gamma,\{(\alpha,r,id)\},\theta,\langle E,I,M\rangle\rangle \longrightarrow \langle \iota,\sigma,\gamma,\{(\alpha,r,id)\},\theta,\langle E,I\cup\{id\},M\rangle\rangle} \tag{E.3}$$

### Test Actions

**Definition 4.** *Let $\varphi$ and $\varphi'$ be $\langle test\rangle$ expressions, $\phi$ be a $\langle belquery\rangle$ expression, $\psi$ be a $\langle goalquery\rangle$ expression, and $\tau$, $\tau_1$ and $\tau_2$ be substitutions. The entailment relation $\models_t$, which evaluates test expressions with respect to an agent's belief and goal bases $(\sigma,\gamma)$, is defined as follows:*

- $(\sigma,\gamma) \models_t \mathtt{B}(\phi)\tau \ \Leftrightarrow\ \sigma \models \phi\tau$
- $(\sigma,\gamma) \models_t \mathtt{G}(\psi)\tau \ \Leftrightarrow\ \gamma \models_g \psi\tau$
- $(\sigma,\gamma) \models_t (\varphi \ \&\ \varphi')\tau_1\tau_2 \ \Leftrightarrow\ (\sigma,\gamma) \models_t \varphi\tau_1$ *and* $(\sigma,\gamma) \models_t \varphi'\tau_1\tau_2$

A test action $\varphi$ can be executed successfully if $\varphi$ is entailed by the agent's belief and goal bases.

$$\frac{(\sigma,\gamma) \models_t \varphi\theta\tau \ \&\ \gamma \models_g \mathcal{G}(r)}{\langle \iota,\sigma,\gamma,\{(\varphi,r,id)\},\theta,\xi\rangle \longrightarrow \langle \iota,\sigma,\gamma,\{\},\theta\cup\{\tau\},\xi\rangle} \tag{E.4}$$

A test action can fail if one or more of its involved query expressions are not entailed by the belief or goal bases.

$$\frac{\neg\exists\tau : (\sigma,\gamma) \models_t \varphi\theta\tau \ \&\ \gamma \models_g \mathcal{G}(r)}{\langle \iota,\sigma,\gamma,\{(\varphi,r,id)\},\theta,\langle E,I,M\rangle\rangle \longrightarrow \langle \iota,\sigma,\gamma,\{(\varphi,r,id)\},\theta,\langle E,I\cup\{id\},M\rangle\rangle} \tag{E.5}$$

### Receiving Detached Norms

A normative organisation can broadcast an obligation or a prohibition event to a specific agent.

$$\frac{\chi \xrightarrow{n\text{-}event} \chi'}{\langle A_0,\ldots,A_\iota,\ldots,A_n\rangle \longrightarrow \langle A_0,\ldots,A'_\iota,\ldots,A_n\rangle} \tag{E.6}$$

where
$A_\iota = \langle \iota, \sigma, \gamma, \Pi, \xi \rangle$,
$A'_\iota = \langle \iota, \sigma, \gamma \cup \{o\!:\!d\}, \Pi, \xi, \succ' \rangle$ if $\textit{n-event} = \textit{obligation}(\iota, o, d, s)$,
$A'_\iota = \langle \iota, \sigma, \gamma, \Pi, \xi \cup \{\textit{prohibition}(p, s)\}, \succ' \rangle$ if $\textit{n-event} = \textit{prohibition}(\iota, p, s)$.

### Goal Dynamics Actions

A successful adoption of goal is defined as follows.

$$\frac{\sigma \not\models \texttt{Œ}\theta \ \& \ \texttt{ground(Œ`)} \ \& \ \gamma \models_g \mathcal{G}(r)}{\langle \iota, \sigma, \gamma, \{(\texttt{adoptX(Œ)}, r, id)\}, \theta, \xi \rangle \longrightarrow \langle \iota, \sigma, \gamma', \{\}, \theta, \xi \rangle} \tag{E.7}$$

An unsuccessful adoption of goal is defined as follows.

$$\frac{(\sigma \models \phi\theta \ \vee \ \neg\texttt{ground}(\phi\theta)) \ \& \ \gamma \models_g \mathcal{G}(r)}{\begin{array}{c} \langle \iota, \sigma, \gamma, \{(\texttt{adoptX(Œ)}, r, id)\}, \theta, \langle E, I, M \rangle \rangle \longrightarrow \\ \langle \iota, \sigma, \gamma, \{(\texttt{adoptX(Œ)}, r, id)\}, \theta, \langle E, I \cup \{id\}, M \rangle \rangle \end{array}} \tag{E.8}$$

Goals can be dropped and removed from the goal base by means of `dropgoal(`$\phi$`)`, `dropsubgoals(`$\phi$`)`, and `dropsupergoals(`$\phi$`)` actions.

$$\frac{\gamma \models_g \mathcal{G}(r)}{\langle \iota, \sigma, \gamma, \{(\texttt{dropX(Œ)}, r, id)\}, \theta, \xi \rangle \longrightarrow \langle \iota, \sigma, \gamma', \{\}, \theta, \xi \rangle} \tag{E.9}$$

where
- $\gamma' = \gamma - \{f \mid f \equiv \phi\theta \ \& \ \texttt{ground}(\phi\theta)\}$ if `dropX(`$\phi$`)` is `dropgoal(`$\phi$`)`
- $\gamma' = \gamma - \{f \mid \phi\theta \models f \ \& \ \texttt{ground}(\phi\theta)\}$ if `dropX(`$\phi$`)` is `dropsubgoals(`$\phi$`)`
- $\gamma' = \gamma - \{f \mid f \models \texttt{Œ`} \ \& \ \texttt{ground(Œ`)}\}$ if `dropX(`$\phi$`)` is `dropsupergoals(`$\phi$`)`

### Abstract Action

A successful execution of an abstract action will replace it with a plan.

$$\frac{\mathit{Unify}(\alpha\theta, \varphi) = \tau_1 \ \& \ \sigma \models \beta\tau_1\tau_2 \ \& \ \gamma \models_g \mathcal{G}(r)}{\langle \iota, \sigma, \gamma, \{(\alpha, r, id)\}, \theta, \xi \rangle \longrightarrow \langle \iota, \sigma, \gamma, \{(\pi\tau_1\tau_2, r, id)\}, \theta, \xi \rangle} \tag{E.10}$$

An unsuccessful execution of an abstract action is defined as follows.

$$\frac{\forall r' \in PC : (\ Unify(\alpha\theta,\ \mathcal{G}(r')\ ) = \bot\ \vee\ \sigma \not\models \mathcal{B}(r')\ )\ \&\ \gamma \models_g \mathcal{G}(r)}{\langle \iota, \sigma, \gamma, \{(\alpha, r, id)\}, \theta, \langle E, I, M\rangle\rangle \longrightarrow \langle \iota, \sigma, \gamma, \{(\alpha, r, id)\}, \theta, \langle E, I \cup \{id\}, M\rangle\rangle} \tag{E.11}$$

### External Action

A successful execution of an external action is defined as follows.

$$\frac{t \neq \bot\ \&\ \gamma \models_g \mathcal{G}(r)}{\langle \iota, \sigma, \gamma, \{(@env(\alpha(t_1, \ldots, t_n), V), r, id)\}, \theta, \xi\rangle \overset{env(\iota,\alpha(t_1\theta,\ldots,t_n\theta),t)}{\longrightarrow} \langle \iota, \sigma, \gamma, \{\}, \theta \cup \{V/t\}, \xi\rangle} \tag{E.12}$$

An unsuccessful execution of an external action is defined as follows.

$$\frac{t = \bot\ \&\ \gamma \models_g \mathcal{G}(r)}{\begin{array}{l}\langle \iota, \sigma, \gamma, \{(@env(\alpha(t_1, \ldots, t_n), V), r, id)\}, \theta, \langle E, I, M\rangle\rangle \overset{env(\iota,\alpha(t_1\theta,\ldots,t_n\theta),t)}{\longrightarrow} \\ \qquad \langle \iota, \sigma, \gamma, \{(@env(\alpha(t_1, \ldots, t_n), V), r, id)\}, \theta, \langle E, I \cup \{id\}, M\rangle\rangle\end{array}} \tag{E.13}$$

## Transition Rules for Plans

In this section, we present the transition rules that capture the execution of plans consisting of basic actions composed by sequence, conditional choice, conditional iteration, and non-interleaving operators.

### Sequence Plan

The execution of a sequence plan $\alpha; \pi$ consists of the execution of the basic action $\alpha$ followed by the execution of plan $\pi$.

$$\frac{\langle \iota, \sigma, \gamma, \{(\alpha, r, id)\}, \theta, \xi\rangle \longrightarrow \langle \iota, \sigma', \gamma', \{\}, \theta', \xi'\rangle}{\langle \iota, \sigma, \gamma, \{(\alpha; \pi, r, id)\}, \theta, \xi\rangle \longrightarrow \langle \iota, \sigma', \gamma', \{(\pi, r, id)\}, \theta', \xi'\rangle} \tag{E.14}$$

### Conditional Plan

The execution of a conditional plan `if` $\varphi$ `then` $\pi_1$ `else` $\pi_2$ consists of a choice between plans $\pi_1$ and $\pi_2$.

$$\frac{(\sigma,\gamma) \models_t \varphi\theta\tau \ \&\ \gamma \models_g \mathcal{G}(r)}{\langle \iota,\sigma,\gamma,\{(\texttt{if}\ \varphi\ \texttt{then}\ \pi_1\ \texttt{else}\ \pi_2,r,id)\},\theta,\xi\rangle \longrightarrow \langle \iota,\sigma,\gamma,\{(\pi_1\tau,r,id)\},\theta,\xi\rangle} \tag{E.15}$$

$$\frac{\neg\exists\tau : (\sigma,\gamma) \models_t \varphi\theta\tau \ \&\ \gamma \models_g \mathcal{G}(r)}{\langle \iota,\sigma,\gamma,\{(\texttt{if}\ \varphi\ \texttt{then}\ \pi_1\ \texttt{else}\ \pi_2,r,id)\},\theta,\xi\rangle \longrightarrow \langle \iota,\sigma,\gamma,\{(\pi_2,r,id)\},\theta,\xi\rangle} \tag{E.16}$$

### While Plan

$$\frac{(\sigma,\gamma) \models_t \varphi\theta\tau \ \&\ \gamma \models_g \mathcal{G}(r)}{\langle \iota,\sigma,\gamma,\{(\texttt{while}\ \varphi\ \texttt{do}\ \pi,r,id)\},\theta,\xi\rangle \longrightarrow \langle \iota,\sigma,\gamma,\{(\pi\tau;\texttt{while}\ \varphi\ \texttt{do}\ \pi,r,id)\},\theta,\xi\rangle} \tag{E.17}$$

$$\frac{\neg\exists\tau : (\sigma,\gamma) \models_t \varphi\theta\tau \ \&\ \gamma \models_g \mathcal{G}(r)}{\langle \iota,\sigma,\gamma,\{(\texttt{While}\ \varphi\ \texttt{do}\ \pi,r,id)\},\theta,\xi\rangle \longrightarrow \langle \iota,\sigma,\gamma,\{\},\theta,\xi\rangle} \tag{E.18}$$

### Concurrent Plans

An agent executes its plans concurrently by interleaving the execution of the next action of all executable plans whose next execution time is *now*.

$$\frac{\langle \iota,\sigma,\gamma,\rho,\xi\rangle \longrightarrow \langle \iota,\sigma',\gamma',\rho',\xi'\rangle}{\langle \iota,\sigma,\gamma,\Pi,\xi\rangle \longrightarrow \langle \iota,\sigma',\gamma',\Pi',\xi'\rangle} \tag{E.19}$$

where $\rho$ is executable and $\Pi' = (\Pi \setminus \rho) \cup \rho'$.

## Practical Reasoning Rules

In this section, we present the transition rules that captures the application of the three types of practical reasoning rules.

### Planning Goal Rules

A N-2APL agent generates plans by applying PG-rules of the form $\kappa \leftarrow \beta \mid \pi : t$.

$$\frac{\begin{array}{c}\exists\,(g{:}d) \in \gamma : g \models_g \kappa\,\tau_1 \;\&\; \sigma \models \beta\,\tau_1\tau_2 \\ \&\; \neg\exists\pi' \in P : (\pi', g{:}d, (\kappa\,\tau_1 \leftarrow \beta \mid \pi{:}t)) \in \Pi\end{array}}{\langle \iota, \sigma, \gamma, \Pi, \xi \rangle \longrightarrow \langle \iota, \sigma, \gamma, \Pi', \xi \rangle} \tag{E.20}$$

where $\tau_1, \tau_2$ are substitutions, $P$ is the set of all possible plans and $\Pi' =$ $\textsc{schedule}(\Pi \cup \{(\pi\,\tau_1\tau_2,\; g{:}d,\; (\kappa\,\tau_1 \leftarrow \beta \mid \pi{:}t))\}, \mathit{prohibitions}(\xi))$.

### Procedure Call Rules

The transition rule for applying PC-rules to the events from $E$ or $M$ is defined as follows:

$$\frac{\psi \in E \cup M \;\&\; \mathit{Unify}(\psi, \varphi) = \tau_1 \;\&\; \sigma \models \beta\tau_1\tau_2}{\langle \iota, \sigma, \gamma, \Pi, \theta, \xi \rangle \longrightarrow \langle \iota, \sigma, \gamma, \Pi', \theta, \xi' \rangle} \tag{E.21}$$

$$\frac{\psi \in E \cup M \;\&\; \forall r \in PC : \mathit{Unify}(\psi, \mathcal{G}(r)) = \bot}{\langle \iota, \sigma, \gamma, \Pi, \theta, \xi \rangle \longrightarrow \langle \iota, \sigma, \gamma, \Pi, \theta, \xi' \rangle} \tag{E.22}$$

where $PC$ is the set of PC-rules of agent $\iota$, $\xi' = \langle E \setminus \{\psi\}, I, M \rangle$ if $\psi =$ $\texttt{event}(\phi, \mathit{env})$ or $\xi' = \langle E, I, M \setminus \{\psi\} \rangle$ if $\psi = \texttt{message}(s, p, l, o, e)$.

**Property 1.** *An event $\psi$ for which there exists a rule r such that $\mathit{Unify}(\psi, \mathcal{G}(r)) \neq \bot$ remains in the event base until it is processed.*

### Plan Repair Rules

The following transition rules specify the application of a PR-rule to a failed plan.

$$\frac{PlanUnify(\pi,\pi_1) = (\tau_d,\tau_p,\pi^*) \ \&\ \sigma \models \beta\tau_d\tau \ \&\ id \in I}{\langle\iota,\sigma,\gamma,\{(\pi,r,id)\},\theta,\langle E,I,M\rangle\rangle \longrightarrow \langle\iota,\sigma,\gamma,\{(\pi_2\tau_d\tau\tau_p;\pi^*,r,id)\},\theta,\langle E,I\setminus\{id\},M\rangle\rangle} \tag{E.23}$$

$$\frac{id \in I \ \&\ (\pi,r,id) \in \Pi \ \&\ \forall(\pi_1 \texttt{<-}\beta \mid \pi_2) \in PR : (PlanUnify(\pi,\pi_1) = \bot \text{ or } \sigma \not\models \beta)}{\langle\iota,\sigma,\gamma,\Pi,\theta,\langle E,I,M\rangle\rangle \longrightarrow \langle\iota,\sigma,\gamma,\Pi,\theta,\langle E,I\setminus\{id\},M\rangle\rangle} \tag{E.24}$$

## Multi-Agent Transition Rules

The execution of a N-2APL multi-agent system is the interleaved executions of the involved individual agents and the environments.

### Interleaved Executions of Individual Agents

$$\frac{A_i \rightarrow A_i'}{\langle A_1,\ldots,A_i,\ldots,A_n,\chi\rangle \rightarrow \langle A_1,\ldots,A_i',\ldots,A_n,\chi\rangle} \tag{E.25}$$

### Execution of Environment by External Actions

$$\frac{F^{env}_{\iota,\alpha}(t_1,\ldots,t_n,\chi) = (t,\chi')}{\chi \overset{env(\iota,\alpha(t_1,\ldots,t_n),t)}{\longrightarrow} \chi'} \tag{E.26}$$

$$\frac{A_i \overset{env(\iota,\alpha(t_1,\ldots,t_n),t)}{\longrightarrow} A_i' \ \&\ \chi \overset{env(\iota,\alpha(t_1,\ldots,t_n),t)}{\longrightarrow} \chi'}{\langle A_1,\ldots,A_i,\ldots,A_n,\chi\rangle \longrightarrow \langle A_1,\ldots,A_i',\ldots,A_n,\chi'\rangle} \tag{E.27}$$

**Execution of Environment by Internal Dynamics**

$$\frac{env \in \chi \;\&\; env \overset{(\phi,\{k,\ldots,l\})!}{\Longrightarrow} env'}{\langle A_1,\ldots,A_n,\chi\rangle \longrightarrow \langle A'_1,\ldots,A'_n,\chi'\rangle} \tag{E.28}$$

where
$\chi' = (\chi \setminus \{env\}) \cup \{env'\}$
$A_\iota = \langle \iota,\sigma,\gamma,\Pi,\theta,\langle E,I,M\rangle\rangle$
$A'_\iota = \langle \iota,\sigma,\gamma,\Pi,\theta,\langle E \cup \{\texttt{event}(\phi,env)\},I,M\rangle\rangle$ if $\iota \in \{k,\ldots,l\}$
$A'_\iota = A_\iota$ otherwise.

# Executing Multi-Agent Systems

The execution of a N-2APL multi-agent system is determined by the N-2APL transition system that is specified by the transition rules presented above. It consists of a set of so called computation runs.

**Definition 5** (computation run). *Given a transition system and an initial configuration $s_0$, a computation run $\texttt{CR}(s_0)$ is a finite or infinite sequence $s_0,\ldots,s_n$ or $s_0,\ldots$ where $s_i$ is a configuration, and $\forall_{i>0} : s_{i-1} \rightarrow s_i$ is a transition in the transition system.*

**Definition 6** (execution of N-2APL multi-agent systems). *The execution of a N-2APL multi-agent system with initial configuration $\langle A_1,\ldots,A_n,\chi\rangle$ is the set of computation runs $\texttt{CR}(\langle A_1,\ldots,A_n,\chi\rangle)$ of the N-2APL transition system.*

## N-2APL Deliberation Process

In order to execute an individual agent, the N-2APL interpreter follows a certain order of deliberation steps repeatedly and indefinitely.

**Property 2.** *If the execution of a plan fails, then the plan will either be repaired in the same deliberation cycle or get re-executed in the next deliberation cycle.*

**Property 3.** *If the first action of a failed plan is a test action, an adopt goal action, or an external action, and there is no plan repair rule to repair it, then the failed plan may be successfully executed in the next deliberation cycle.*

# References

Adal, A. (2010). *An Interperter for Organization Oriented Programming Language (2OPL)*. Masters' thesis computer science, Utrecht University.

Aldewereld, H., Dignum, V., and Vasconcelos, W. (2013). We ought to; They do; Blame the management! In *Coordination Organizations Institutions and Norms in Agent Systems*.

Aldewereld, H., Dignum, V., and Vasconcelos, W. (2015). Reasoning with group norms in software agent organisations. In *Proceedings of the International Workshop on Coordination, Organisation, Institutions and Norms in Multi-Agent Systems (COIN)*.

Alechina, N., Dastani, M., and Logan, B. (2012). Programming norm-aware agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 1057–1064. International Foundation for Autonomous Agents and Multiagent Systems.

Allen, J. E., Guinn, C., et al. (1999). Mixed-initiative interaction. *Intelligent Systems and their Applications, IEEE*, 14(5):14–23.

Baldoni, M., Baroglio, C., Mascardi, V., Omicini, A., and Torroni, P. (2010). Agents , Multi-Agent Systems and Declarative Programming : What , When , Where , Why , Who , How ? In Dovier, Agostino and Pontelli, E., editor, *A 25-year perspective on logic programming*, chapter Agents , M, pages 204–230. Springer-Verlag.

Balke, T. (2009). A taxonomy for ensuring institutional compliance in utility computing. In Boella, G., Noriega, P., Pigozzi, G., and Verhagen, H., editors, *Normative Multi-Agent Systems*, number 09121 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

Ball, M. and Callaghan, V. (2012). Managing Control , Convenience and Autonomy A Study of Agent Autonomy in Intelligent Environments. In *Special Issue on Agent-Based Approaches to Ambient Intelligence*.

Balliet, D. and Van Lange, P. A. (2013). Trust, conflict, and cooperation: a meta-analysis. *Psychological Bulletin*, 139(5):1090.

Barber, K., Goel, A., and Martin, C. E. (2000). Dynamic adaptive autonomy in multi-agent systems. *Journal of Experimental \& Theoretical Artificial Intelligence*, 12(2):129–147.

Barber, K. and Martin, C. (1999). Agent autonomy: Specification, measurement, and dynamic adjustment. In *Proceedings of the Autonomy Control Software Workshop at Autonomous Agents*, volume 1999, pages 8–15. Citeseer.

Beavers, G. and Hexmoor, H. (2004). Types and limits of agent autonomy. *Agents and Computational Autonomy*, pages 95–102.

Beer, J. M., Fisk, A. D., and Rogers, W. A. (2014). Toward a Framework for Levels of Robot Autonomy in Human-Robot Interaction. *Journal of Human-Robot Interaction*, 3(2):74.

Benford, S., Drozd, A., Greenhalgh, C., Fraser, M., Schnädelbach, H., Koleva, B., Anastasi, R., Flintham, M., Hemmings, T., Crabtree, A., et al. (2006). Orchestrating real-time participatory experiences.

Björnsson, G. (2011). Joint responsibility without individual control: Applying the explanation hypothesis. In *Moral Responsibility*, pages 181–199. Springer.

Björnsson, G. (2014). Essentially shared obligations. *Midwest studies in philosophy*, 38(1):103–120.

Boella, G., Torre, L., and Verhagen, H. (2008). Introduction to the special issue on normative multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 17(1):1–10.

Boella, G., Torre, L. V. D., and Verhagen, H. (2006). Introduction to normative multiagent systems. *Computational & Mathematical Organization Theory*, 12(2):71–79.

Boella, G. and van der Torre, L. (2008). Substantive and procedural norms in normative multiagent systems. *Journal of Applied Logic*, 6(2):152–171.

Bordini, R., Braubach, L., Dastani, M., El FSeghrouchni, A., Gomez-Sanz, J., Leite, J., O Hare, G., Pokahr, A., and Ricci, A. (2006). A survey of programming languages and platforms for multi-agent systems. *INFORMATICA-LJUBLJANA-*, 30(1):33.

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using Jason*. Wiley.

Bradshaw, J. M., Feltovich, P., and Johnson, M. (2012). Human-agent interaction. *Handbook of Human-Machine Interaction*, pages 283–302.

Bradshaw, J. M., Feltovich, P. J., Jung, H., Kulkarni, S., Taysom, W., and Uszok, A. (2004). Dimensions of Adjustable Autonomy and Mixed-Initiative Interaction. *Agents and Computational Autonomy*, pages 235–268.

Braham, M. and Van Hees, M. (2012). An anatomy of moral responsibility. *Mind*, 121(483):601–634.

Bratko, I. (2001). *Prolog programming for artificial intelligence*. Pearson education.

Bratman, M. (1992). Shared cooperative activity. *The philosophical review*, 101(2):327–341.

Braynov, S. and Hexmoor, H. (2003). Quantifying relative autonomy in multiagent interaction. *Agent Autonomy*, pages 55–73.

Broersen, J., Dastani, M., Hulstijn, J., and van der Torre, L. (2002). Goal generation in the BOID architecture. *Cognitive Science Quarterly*, 2(3-4):428–447.

Broersen, J., Dignum, F., Dignum, V., and Meyer, J.-J. C. (2004). Designing a deontic logic of deadlines. In *Deontic Logic in Computer Science*, pages 43–56. Springer.

Burstein, M., Ferguson, G., and Allen, J. (2000). Integrating agent-based mixed-initiative control with an existing multi-agent planning system. In *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*, pages 389–390. IEEE.

Bădică, C., Budimac, Z., Burkhard, H.-D., and Ivanovic, M. (2011). Software agents: Languages, tools, platforms. *Computer Science and Information Systems*, 8(2):255–298.

Cabri, G., Leonardi, L., Mamei, M., and Zambonelli, F. (2003). Location-Dependent Services for Mobile Users. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 33(6):667–681.

Carabelea, C., Boissier, O., and Florea, A. (2004). Autonomy in multi-agent systems: A classification attempt. *Agents and Computational Autonomy*, pages 59–70.

Carmo, J. (2009). Collective agency, direct action and dynamic operators. *Logic Journal of IGPL*, 18(1):66–98.

Carriero, N. and Gelernter, D. (1989). Linda in context. *Communications of the ACM*, 32(4):444–458.

Castelfranchi, C. (1995a). Commitments : From Individual Intentions to Groups and Organizations Introductive remarks. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 41–48.

Castelfranchi, C. (1995b). Guarantees for autonomy in cognitive agent architecture. In *Intelligent agents*, pages 56–70. Springer.

Castelfranchi, C. (1998). Modelling social action for ai agents. *Artificial Intelligence*, 103(1):157–182.

Castelfranchi, C., Dignum, F., Jonker, C. M., and Treur, J. (2000). Deliberative Normative Agents : Principles and Architecture. *Intelligent Agents VI. Agent Theories, Architectures, and Languages Lecture Notes in Computer Science*, pages 364–378.

Castelfranchi, C. and Falcone, R. (2003). From Automaticity to Autonomy : The Frontier of Artificial Agents 1. *Agent Autonomy*, pages 103–136.

Cheng, M. Y. K. and Cohen, R. (2005). A Hybrid Transfer of Control Model for Adjustable Autonomy Multiagent Systems. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1149–1150. ACM.

Cholvy, L. and Garion, C. (2007). Deriving individual obligations from collective obligations. *Proceedings of the Dagstuhl Seminar on Normative Multi-agent Systems*, pages 1–16.

Cohen, P. R. and Levesque, H. J. (1990). Intention is choice with commitment. *Artificial intelligence*, 42(2):213–261.

Cohen, R. and Cheng, M. (2005). Why bother about bother: Is it worth it to ask the user? In *Proceedings of AAAI Fall Symposium*.

Conte, R., Castelfranchi, C., and Dignum, F. (1999). Autonomous Norm-acceptance. *Intelligent Agents V: Agents Theories, Architectures, and Languages*, pages 99–112.

Costanza, E., Fischer, J. E., Colley, J. A., Rodden, T., Ramchurn, S. D., and Jennings, N. R. (2014). Doing the laundry with agents: a field trial of a future smart energy system in the home. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 813–822. ACM.

Crawford, N. C. (2007). Individual and collective moral responsibility for systemic military atrocity*. *Journal of Political Philosophy*, 15(2):187–212.

Criado, N. (2013). Using norms to control open multi-agent systems. *AI Communications*, 26(3):317–318.

Criado, N., Argente, E., and Botti, V. (2010). A BDI Architecture for Normative Decision Making. *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 1383–1384.

Criado, N., Argente, E., Noriega, P., and Botti, V. (2012). A Distributed Architecture for Enforcing Norms. In *Proceedings of the 10th international conference on Advanced Agent Technology*, pages 457–471.

da Silva Figueiredo, K., Torres da Silva, V., and de Oliveira Braga, C. (2011). Modeling norms in multi-agent systems with normml. *Coordination, Organizations, Institutions, and Norms in Agent Systems VI*, pages 39–57.

Dastani, M. (2008). 2apl: a practical agent programming language. *Autonomous Agents and Multi-Agent Systems*, 16(3):214–248.

Dastani, M., Dignum, F., and Meyer, J.-j. (2004). Autonomy and Agent Deliberation. *Agents and Computational Autonomy*, pages 1–20.

Dastani, M., Grossi, D., and Meyer, J. (2009). Normative multi-agent programs and their logics. *Knowledge Representation for Agents and Multi-Agent Systems*, pages 16–31.

Dastani, M. and Tinnemeier, N. (2009). A programming language for normative multi-agent systems. *Handbook of research on multi-agent systems: semantics and dynamics of organizational models*, page 397.

de Lima, T., Royakkers, L., and Dignum, F. (2010). A logic for reasoning about responsibility. *Logic Journal of IGPL*, 18(1):99–117.

Diggelen, J. V. and Bradshaw, J. M. (2010). Implementing collective obligations in human-agent teams using KAoS policies. In *Coordination Organizations Institutions and Norms in Agent Systems*.

Dignum, F. (1999). Autonomous agents with norms. *Artificial Intelligence and Law*, 7(1):69–79.

Dignum, F., Morley, D., Sonenberg, E. A., and Cavedon, L. (2000). Towards socially sophisticated bdi agents. In *Proceedings of the 4th International Conference on MultiAgent Systems (ICMAS'00)*, pages 111–118.

Dignum, V. and Dignum, F. (2011). A logic of agent organizations. *Logic Journal of IGPL*, 20(1):283–316.

Dorais, G., Bonasso, R., and Kortenkamp, D. (1999). Adjustable autonomy for human-centered autonomous systems. In *Working notes of the Sixteenth International Joint Conference on Artificial Intelligence Workshop on Adjustable Autonomy Systems*, pages 16–35.

Dunin-Keplicz, B., Strachocka, A., and Verbrugge, R. (2011). Modeling Deliberation in Teamwork. In *AAAI Spring Symposium*, pages 114–118.

d'Inverno, M. and Luck, M. (2012). Creativity through autonomy and interaction. *Cognitive Computation*, pages 1–15.

Edwards, K. and Rodden, T. (2001). *Jini Example by Example*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Esteva, M., Rosell, B., Rodriguez-Aguilar, J. A., and Arcos, J. L. (2004). AMELI: An agent-based middleware for electronic institutions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '04, pages 236–243, Washington, DC, USA. IEEE Computer Society.

Eysenck, M. W. and Calvo, M. G. (1992). Anxiety and performance: The processing efficiency theory. *Cognition & Emotion*, 6(6):409–434.

F. Lopez y Lopez, M. L. and dInverno, M. (2006). A normative framework for agentbased systems. *Computational and Mathematical Organization Theory*, pages 227–250.

F. Sadri, K. S. and Toni, F. (2006). Normative KGP agents. *Computational and Mathematical Organization Theory*, pages 101–126.

Falcone, R. (2001). Autonomy : Theory , Dimensions , and Regulation. *Intelligent Agents VII Agent Theories Architectures and Languages*, pages 258–263.

Falcone, R. and Castelfranchi, C. (2000). Levels of Delegation and Levels of Adoption as the basis for Adjustable Autonomy. *AI* IA 99: Advances in Artificial Intelligence*, pages 273–284.

Fan, X. and Yen, J. (2012). Intentions and potential intentions revisited. *Journal of Applied Non-Classical Logics*, (April 2013):37–41.

Ferguson, G., Allen, J. F., Miller, B. W., et al. (1996). Trains-95: Towards a mixed-initiative planning assistant. In *AIPS*, pages 70–77.

Ferreira, N., Mascarenhas, S., Paiva, A., Di Tosto, G., Dignum, F., McBreen, J., Degens, N., Hofstede, G. J., Andrighetto, G., and Conte, R. (2013). An agent model for the appraisal of normative events based in in-group and out-group relations. In *AAAI*.

Fischer, J., Flintham, M., Price, D., Goulding, J., Pantidi, N., and Rodden, T. (2012a). Serious mixed reality games. In *Mixed reality games. Workshop at ACM CSCW*.

Fischer, J. E., Jiang, W., and Moran, S. (2012b). Atomicorchid: a mixed reality game to investigate coordination in disaster response. In *Entertainment Computing-ICEC 2012*, pages 572–577. Springer.

Flintham, M., Benford, S., Anastasi, R., Hemmings, T., Crabtree, A., Greenhalgh, C., Tandavanitj, N., Adams, M., and Row-Farr, J. (2003). Where on-line meets on the streets: experiences with mobile mixed reality games. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 569–576. ACM.

Fornara, N. and Colombetti, M. (2009). Specifying and Enforcing Norms in Artificial Institutions. *Declarative Agent Languages and Technologies VI*, pages 1–17.

Franklin, S. and Graesser, A. (1997). Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Intelligent agents III agent theories, architectures, and languages*, pages 21–35. Springer.

G. Andrighetto, M. Campenni, R. C. and Paolucci, M. (2007). On the immergence of norms: a normative agent architecture. In *Proc. of AAAI Symposium, Social and Organizational Aspects of Intelligence*.

Gaertner, D. (2008). *Argumentation and Normative Reasoning*. PhD thesis, University of London.

Gal, Y., Grosz, B., Kraus, S., Pfeffer, A., and Shieber, S. (2010). Agent decision-making in open mixed networks. *Artificial Intelligence*, 174(18):1460–1480.

Gal, Y., Grosz, B. J., Kraus, S., Pfeffer, A., and Shieber, S. (2005). Colored trails: a formalism for investigating decision-making in strategic environments. In *Proceedings of the 2005 IJCAI workshop on reasoning, representation, and learning in computer games*, pages 25–30.

Garbay, C. and Badeig, F. (2012). Normative multi-agent approach to support collaborative work in distributed tangible environments. *Computer Supported Cooperative Work*, pages 83–86.

García-Camino, A., Rodríguez-Aguilar, J., Sierra, C., and Vasconcelos, W. (2009). Constraint rule-based programming of norms for electronic institutions. *Autonomous Agents and Multi-Agent Systems*, 18:186–217.

Gateau, B., Boissier, O., Khadraoui, D., and Dubois, E. (2007). Controlling an interactive game with a multi-agent based normative organisational model. *Coordination, Organizations, Institutions, and Norms in Agent Systems II*, pages 86–100.

Gelati, J., Rotolo, A., Sartor, G., and Governatori, G. (2004). Normative autonomy and normative co-ordination: Declarative power, representation, and mandate. *Artificial Intelligence and Law*, 12(1-2):53–81.

Goodrich, M. A., Olsen, D. R., Crandall, J. W., and Palmer, T. J. (2001). Experiments in adjustable autonomy. In *Proceedings of IJCAI Workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents*, pages 1624–1629.

Grossi, D., Aldewereld, H., and Dignum, F. (2007). Ubi lex, ibi poena: Designing norm enforcement in e-institutions. In *Coordination, organizations, institutions, and norms in agent systems II*, pages 101–114. Springer.

Grossi, D., Dignum, F., Royakkers, L., and Meyer, J.-j. C. (2004). Collective obligations and agents: Who gets the blame? *Deontic Logic in Computer Science*, pages 129–145.

Grossi, D., Tummolini, L., and Turrini, P. (2013). Norms in game theory. In *Agreement Technologies*, pages 191–197. Springer.

Grosz, B. and Kraus, S. (1993). Collaborative plans for group activities. In *IJCAI*, volume 93, pages 367–373.

Grosz, B. and Kraus, S. (1996). Collaborative plans for complex group action. *Artificial Intelligence*, 93(6288).

Haim, G., Gal, Y. K., Gelfand, M., and Kraus, S. (2012). A cultural sensitive agent for human-computer negotiation. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 451–458. International Foundation for Autonomous Agents and Multiagent Systems.

Hardin, B. and Goodrich, M. A. (2009). On using mixed-initiative control: a perspective for managing large-scale robotic teams. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 165–172. ACM.

Hoare, C. A. R. (1969). An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580.

Hoffman, R. R., Johnson, M., Bradshaw, J. M., and Underbrink, A. (2013). Trust in automation. *IEEE Intelligent Systems*, 28(1):84–88.

Hollander, C. and Wu, A. (2011). The Current State of Normative Agent-Based Systems. *Journal of Artificial Societies and Social Simulation*, 14(2):6.

Holz, T., Campbell, A. G., O'Hare, G. M., Stafford, J. W., Martin, A., and Dragone, M. (2011). MiRA—Mixed Reality Agents. *International Journal of Human-Computer Studies*, 69(4):251–268.

Hong, J.-y., Suh, E.-h., and Kim, S.-J. (2009). Context-aware systems: A literature review and classification. *Expert Systems with Applications*, 36(4):8509–8522.

Hong, L. I. U. (2009). An Intelligent Tuple Space for Agent Interaction in Mobile Agent System. In *IT in Medicine & Education, 2009. ITIME'09. IEEE International Symposium on*, pages 617–622.

Horvitz, E. (1999). Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 159–166. ACM.

Hübner, J. F., Boissier, O., and Bordini, R. H. (2010). From organisation specification to normative programming in multi-agent organisations. In Dix, J., Leite, J., Governatori, G., and Jamroga, W., editors, *Computational Logic in Multi-Agent Systems, 11th International Workshop, CLIMA XI, Lisbon, Portugal, August 16-17, 2010. Proceedings*, volume 6245 of *Lecture Notes in Computer Science*, pages 117–134. Springer.

Hübner, J. F., Boissier, O., and Bordini, R. H. (2011). A normative programming language for multi-agent organisations. *Annals of Mathematics and Artificial Intelligence*, 62(1-2):27–53.

Hübner, J. F., Bordini, R. H., and Wooldridge, M. (2006). Programming declarative goals using plan patterns. In Baldoni, M. and Endriss, U., editors, *Declarative Agent Languages and Technologies IV, (DALT 2006)*, volume 4327 of *LNCS*, pages 123–140.

Hübner, J. F., Sichman, J. S., and Boissier, O. (2007). Developing organised multi-agent systems using the $\mathcal{M}$OISE$^+$ model: Programming issues at the system and agent levels. *International Journal of Agent-Oriented Software Engineering*, 1(3/4):370–395.

Hurwicz, L. (1996). Institutions as families of game forms*. *Japanese Economic Review*, 47(2):113–132.

Hurwicz, L. (2008). But who will guard the guardians? *The American Economic Review*, pages 577–585.

Jennings, N. R., Moreau, L., Nicholson, D., Ramchurn, S., Roberts, S., Rodden, T., and Rogers, A. (2014). Human-agent collectives. *Communications of the ACM*, 57(12):80–88.

Jiang, W., Fischer, J. E., Greenhalgh, C., Ramchurn, S. D., Wu, F., Jennings, N. R., and Rodden, T. (2014). Social implications of agent-based planning support for human teams. In *Collaboration Technologies and Systems (CTS), 2014 International Conference on*, pages 310–317. IEEE.

Johnson, M., Bradshaw, J. M., Feltovich, P. J., Jonker, C. M., Van Riemsdijk, B., and Sierhuis, M. (2011). The fundamental principle of coactive design: Interdependence must shape autonomy. In *Coordination, organizations, institutions, and norms in agent systems VI*, pages 172–191. Springer.

Jones, A. J. and Sergot, M. (1993). On the characterisation of law and computer systems: The normative systems perspective. In *Deontic Logic in Computer Science: Normative System Specification*.

Jonker, C. M., Riemsdijk, M. B. V., and Vermeulen, B. (2010). Shared Mental Models. In *Coordination Organizations Institutions and Norms in Agent Systems*, number Section 2, pages 132–151.

Kamar, E., Gal, Y. K., and Grosz, B. J. (2013). Modeling information exchange opportunities for effective human computer teamwork. *Artificial Intelligence*, 195:528–550.

Kamphorst, B., van Wissen, A., and Dignum, V. (2009). Incorporating bdi agents into human-agent decision making research. In *Engineering Societies in the Agents World X*, pages 84–97. Springer.

Klein, G., Woods, D. D., Bradshaw, J. M., Hoffman, R. R., and Feltovich, P. J. (2004). Ten challenges for making automation a" team player" in joint human-agent activity. *IEEE Intelligent Systems*, 19(6):91–95.

Kollingbaum, M. (2005). *Norm-governed practical reasoning agents*. PhD thesis, University of Aberdeen.

Kollingbaum, M. and Norman, T. (2004). Norm adoption and consistency in the noa agent architecture. *Programming Multi-Agent Systems*, pages 169–186.

Kozierok, R. and Maes, P. (1993). A learning interface agent for scheduling meetings. In *Proceedings of the 1st international conference on Intelligent user interfaces*, pages 81–88. ACM.

Lewis, M. (1998). Designing for human-agent interaction. *AI Magazine*, 19(2):67.

Lieberman, H. (1997). Autonomous interface agents. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, pages 67–74. ACM.

Lieberman, H. and Selker, T. (2003). Agents for the user interface.

List, C. and Pettit, P. (2011). *Group agency: The possibility, design, and status of corporate agents*. Oxford University Press Oxford.

Luck, M., Inverno, M. D., and Munroe, S. (2003). Autonomy : Variable and Generative. *Agent Autonomy*, pages 11–28.

Luck, M., Mahmoud, S., Meneguzzi, F., Kollingbaum, M., Norman, T. J., Criado, N., and Fagundes, M. S. (2013). Normative agents. In *Agreement Technologies*, pages 209–220. Springer.

Maes, P. et al. (1994). Agents that reduce work and information overload. *Communications of the ACM*, 37(7):30–40.

Mascardi, V., Demergasso, D., and Ancona, D. (2005). Languages for programming bdi-style agents: an overview. In *WOA*, pages 9–15.

Mellema, G. (1997). *Collective responsibility*, volume 50. Rodopi.

Meneguzzi, F. and Luck, M. (2009a). Norm-based behaviour modification in BDI agents. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 177–184.

Meneguzzi, F. R. (2008). Extending Agent Languages for Autonomy. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1744–1745.

Meneguzzi, F. R. and Luck, M. (2009b). Norm-based behaviour modification in BDI agents. In Sierra, C., Castelfranchi, C., Decker, K. S., and Sichman, J. S., editors, *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, pages 177–184. IFAAMAS.

Meneguzzi, J. O., Sycara, K., and Norman, T. J. (2011). Prognostic normative reasoning in coalition planning. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, number 1, pages 1233–1234.

Mercier, S., Dehais, F., Lesire, C., and Tessier, C. (2008). Resources as basic concepts for authority sharing. In *Humans Operating Unmanned Systems (HUMOUS'08)*.

Metral, Y. L. M. and Maes, P. (1998). Collaborative interface agents. *Readings in agents*, page 111.

Mostafa, S. A., Ahmad, M. S., Annamalai, M., Ahmad, A., and Gunasekaran, S. S. (2015). Formulating dynamic agents operational state via situation awareness assessment. In *Advances in Intelligent Informatics*, pages 545–556. Springer.

Mostafa, S. A., Ahmad, M. S., and Mara, U. T. (2016). A Flexible Human-Agent Interaction Model for Supervised Autonomous Systems. (August):23–24.

Murphy, A. M. Y. L., Picco, G. P., and Roman, G.-c. (2006). Lime : A Coordination Model and Middleware Supporting Mobility of Hosts and Agents. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 15(3):279–328.

Neumann, M. (2010). A classification of normative architectures. *Simulating Interacting Agents and Social Phenomena*, pages 3–18.

NMAS (2013). Normative multi-agent systems. In Andrighetto, G., Governatori, G., Noriega, P., and van der Torre, L. W. N., editors, *Normative Multi-Agent Systems*, volume 4 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Omicini, A. and Denti, E. (2001). Formal ReSpecT. *Electronic Notes in Theoretical Computer Science*, 48:179–196.

Oren, N., Vasconcelos, W., Meneguzzi, F., and Luck, M. (2011). Acting on norm constrained plans. In *Proceedings of the 12th international conference on Computational logic in multi-agent systems*, pages 347–363, Barcelona. Springer-Verlag.

Parasuraman, R., Sheridan, T. B., and Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *IEEE transactions on systems, man, and cybernetics. Part A, Systems and humans : a publication of the IEEE Systems, Man, and Cybernetics Society*, 30(3):286–97.

Pettit, P. (2007). Responsibility incorporated*. *Ethics*, 117(2):171–201.

Porello, D., Bottazzi, E., and Ferrario, R. (2014). The ontology of group agency. In *Proceedings of the 8th International Conference on Formal Ontology in Information Systems. FOIS*.

Ramchurn, S. D., Wu, F., Jiang, W., Fischer, J. E., Reece, S., Greenhalgh, C., Rodden, T., Jennings, N. R., and Roberts, S. (2014). Atomicorchid:

human-agent collectives to the rescue. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1693–1694. International Foundation for Autonomous Agents and Multiagent Systems.

Ranathunga, S., Cranefield, S., and Purvis, M. (2012). Integrating expectation monitoring into bdi agents. In Dennis, L., Boissier, O., and Bordini, R., editors, *Programming Multi-Agent Systems*, volume 7217 of *Lecture Notes in Computer Science*, pages 74–91. Springer Berlin Heidelberg.

Reed, N. E. (2006). Supporting adjustable autonomy in agent systems. In Ward, P., Crosby, M., and Schultz, J., editors, *Symposium on Integrating Humans with Intelligent Technologies: Merging Theories of Collaborative Intelligence and Expert Cognition at the 39th Hawaii International Conference on System Sciences (HICSS-06)*, pages 1–4. http://www.itl.nist.gov/iaui/vvrg/hicss39/.

Ricci, A., Viroli, M., and Omicini, A. (2007). Give agents their artifacts: the A&A approach for engineering working environments in MAS. In Durfee, E. H., Yokoo, M., Huhns, M. N., and Shehory, O., editors, *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007)*, pages 601–603. IFAAMAS.

S. Joseph, c. Sierra, M. S. and Dellunde, P. (2010). Deductive coherence and norm adoption. *Logic Journal of the IGPL*.

Savarimuthu, B., Cranefield, S., Purvis, M., and Purvis, M. (2011). Identifying conditional norms in multi-agent societies. *Coordination, Organizations, Institutions, and Norms in Agent Systems VI*, pages 285–302.

Scerri, P., Sycara, K., and Tambe, M. (2004). Adjustable autonomy in the context of coordination. *AIAA 3rd" Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, pages 1–13.

Schwarz, M., Stückler, J., and Behnke, S. (2014). Mobile teleoperation interfaces with adjustable autonomy for personal service robots. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pages 288–289. ACM.

Sergot, M. (2008). Action and agency in norm-governed multi-agent systems. *Engineering Societies in the Agents World VIII*, pages 1–54.

Shockley, K. (2007). Programming collective control. *Journal of social philosophy*, 38(3):442–455.

Singh, M. P. (1999). An ontology for commitments in multiagent systems. *Artificial intelligence and law*, 7(1):97–113.

Singh, M. P. (2014). Norms as a basis for governing sociotechnical systems. *ACM Trans. Intell. Syst. Technol.*, 5(1):21:1–21:23.

Smiley, M. (2011). Collective responsibility. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Fall 2011 edition.

Sukthankar, G., Shumaker, R., and Lewis, M. (2012). Intelligent agents as teammates. *Theories of Team Cognition: Cross-Disciplinary Perspectives*, pages 313–343.

Sycara, K. P. (1998). The many faces of agents. *AI magazine*, 19(2):11–12.

Tambe, M., Scerri, P., and Pynadath, D. V. (2002). Adjustable Autonomy for the Real World. *Journal of Artificial Intelligence Research*, 17:171–228.

Tännsjö, T. (2007). The myth of innocence: On collective responsibility and collective punishment. *Philosophical Papers*, 36(2):295–314.

Testerink, B. and Dastani, M. (2012). A norm language for distributed organizations. Master's thesis.

Tinnemeier, N. (2011). *Organizing agent organizations: syntax and operational semantics of an organization-oriented programming language*. PhD thesis.

Valero-Gomez, A., de la Puente, P., and Hernando, M. (2011). Impact of Two Adjustable-Autonomy Models on the Scalability of Single-Human/Multiple-Robot Teams for Exploration Missions. *Human Factors: The Journal of the Human Factors and Ergonomics Society*.

van Wissen, A., Gal, Y., Kamphorst, B., and Dignum, M. (2012). Human–agent teamwork in dynamic environments. *Computers in Human Behavior*, 28(1):23–33.

Vazquez-Salceda, J., Aldewereld, H., and Dignum, F. (2004). Implementing Norms in Multiagent Systems. In *Proc. of the German Conference on Multiagent System Technologies (MATES), Lecture Notes in Computer Science*, pages 313–327.

Vecht, B. V. D., Dignum, F., Meyer, J., and Neef, M. (2007a). A dynamic coordination mechanism using adjustable autonomy. *Proceedings of the 2007 international conference on Coordination, organizations, institutions, and norms in agent systems III*, pages 83–96.

Vecht, B. V. D., Meyer, J.-j. C., Neef, M., Dignum, F., and Meyer, A. P. (2007b). Influence-Based Autonomy Levels in Agent. In *Coordination, Organizations, Institutions, and Norms in Agent Systems II*, pages 322–337. Springer Berlin Heidelberg.

Verhagen, H. (2000). *Norm autonomous agents*. PhD thesis, Citeseer.

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(02):115–152.

Yang, S.-Y. and Lee, D.-L. (2012). Developing a cloud intelligent and energy-saving information interface agent with web services. In *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*, pages 1310–1315. IEEE.

Zieba, S., Polet, P., Vanderhaegen, F., and Debernard, S. (2009). Principles of adjustable autonomy: a framework for resilient human–machine cooperation. *Cognition, Technology & Work*, 12(3):193–203.

Zimmerman, J., Tomasic, A., Simmons, I., Hargraves, I., Mohnkern, K., Cornwell, J., and McGuire, R. M. (2007). Vio: a mixed-initiative approach to learning and automating procedural update tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1445–1454. ACM.