A typology of relationships and goals for coordination and regulation

Ronald Ashri[1] and **Michael Luck**[2] and **Mark d'Inverno**[3]

Abstract.
The types of relationships that arise between interacting agents in a multi-agent system can greatly influence the effectiveness of the entire system. However, the behaviour of agents cannot always be anticipated, especially when dealing with open and complex systems. Such systems must incorporate relationship management mechanisms that attempt to guide the behaviour of agents towards desired outcomes. To effectively design such mechanisms, we must first be able to identify the types of relationships that may emerge and how they can be understood, when faced with the constraints and opportunities presented by inter-agent relationships. We have previously addressed this in the limited context of restricting malicious behaviour through the application of regulations. In this paper we generalise that model to identify and characterise a much broader range of relationships. This results in a generic analysis tool, which can be used to achieve two crucial tasks: to identify opportunities for more effective coordination between agents; and to enable the analysis of the control that agents can exert over their own goals in the context of such relationships. Combining these types of analysis allows us to identify conflicts and opportunities for cooperation in multi-agent systems both at design time and at run-time, allowing for finer-grained system control.

1 Introduction

The ability of agents to interact in order to collectively tackle tasks is one of the central arguments for the utility of multi-agent systems [12]. Such interactions take place whenever one agent performs an action which, intentionally or not, affects one or more other agents. Thus, when agents interact we can say that they are *related* by virtue of the fact that they are affecting each other. Identifying, analysing and understanding the implications of the various types of such relationships is of critical importance, since they can have both beneficial and adverse effects.

In this respect, system designers have two overarching challenges. On the one hand, the system design must ensure that the interactions that are necessary for achieving system-wide goals take place. For example, if agents require assistance to achieve their goals, they must be provided with mechanisms for discovering other agents able to assist them. On the other hand, the design must also ensure that undesirable interactions do not take place. For example, if a number of agents depend on a limited resource, there must be appropriate mechanisms in place to control access to that resource. These challenges are compounded by the fact that in open or simply large agent systems the possible interactions between agents cannot all be explicitly specified at design time. This is especially true when dealing with *autonomous* agents operating in heterogeneous environments in which agents may join or leave the system at any time, and no assumptions are made about agent behaviour.

In response to these challenges, the need to provide some *external* form of control over the behaviour of agents was identified long ago [5]. Currently, this is largely achieved through the use of regulatory frameworks, stemming from work on policies (e.g. [10, 18, 13]), institutions (e.g. [11, 20]) and norms (e.g. [21, 3, 6]). In addition, there is significant work on coordinating middleware to enhance agent infrastructure [4, 15].

However, in order for such mechanisms to be applied an implicit assumption is made that the system designer is already aware of the relationships that may arise in a multi-agent system or of the general regulations required to lead only to appropriate relationships. This assumption no longer holds in dynamic systems, so there must be some method for *systematically identifying* what relationships can arise and only then addressing the problem of identifying appropriate regulation or coordination mechanisms.

Therefore, if coordination and regulation of agents is to be achieved as an agent society evolves, either by external intervention or through interventions by the agents themselves, we require some means of identifying at run-time how agents are related to other agents. Of course, this information is only useful if we are also able to determine how the identified relationships may impact on individual agent operation and the system as a whole. Thus, we also require a principled and comprehensive typology for *characterising agent relationships.* The ability to identify and characterise agent relationships can be beneficial for the following reasons.

- It can guide the choice and design of appropriate regulatory frameworks to prevent malicious behaviour or interference between agents.
- Potentially missed opportunities for cooperation between agents can be identified at run-time.
- It can provide a template of generic relationship types for agents to use at run-time, allowing them to identify how

[1] University of Southampton, Southampton, SO17 1BJ, United Kingdom, email: {ra}@ecs.soton.ac.uk
[2] University of Southampton, Southampton, SO17 1BJ, United Kingdom, email: {mml}@ecs.soton.ac.uk
[3] University of Westminister, London, W1M 6UW, United Kingdom,email:dinverm@westminster.ac.uk

other agents may affect their own operation.

In order to develop the types of tools discussed here, we begin this paper with the introduction of a model of agent interaction, which acts as the conceptual basis on which the rest is built. Subsequently, we introduce a typology of agent relationships, which builds on and refines previous work in this area [1]. Then we discuss how knowledge of the goals agents are pursuing can be considered alongside this typology to allow us to better model the possible relationships between agents. Finally, we conclude with some examples and a discussion on what other work can be motivated from the analytical tools presented here.

2 Model of Agent Interaction

A number of different and interrelated issues determine why, how and when two agents enter into a relationship. For example, some relationships are *built-in* at design time. Alternatively, relationships may develop opportunistically as agents seek assistance in order to achieve goals, or unintentionally as agents perform actions that affect the environment. It would thus be counter-intuitive to attempt to identify relationships by looking at the internal operation of individual agents. In this section, therefore, we introduce a model of agent interaction derived from a focus on the *interface* between individual agents and their environment, through the capabilities of agents. The notions that underpin this model are based on the SMART framework [9], and are discussed in more detail in [1], so they are only briefly described below.

2.1 Agents

An agent in our model is considered as an entity that is described by a set of *attributes*. Attributes are simply features of the *environment*, and are the only characteristics that are manifest. Agents are able to perform *actions*, which can change the environment by adding or removing attributes. Agents also pursue *goals*, which are desirable environment states described by non-empty sets of attributes.

2.2 Agent Perception and Action

Agent actions are divided into those that retrieve the value of specific attributes of the environment, representing the agent's *sensor capabilities*, and those that attempt to *change* attribute values of the environment, representing the agent's *actuator capabilities.*

These are the only aspects of the agent's architecture that concern us at this point, as they form the the *interface* between the agent and its external environment. We are concerned neither with the internal state nor the decision-making capabilities because it means that interactions can be analysed without reference to specific agent architectures, and without knowledge of the agent's decision-making mechanisms or internal state.

2.3 Viewable Environment and Region of Influence

Given that agents interact with the environment through actuators and sensors, and that the environment as a whole is defined through a set of attributes, we can intuitively think of actuators and sensors as defining *regions of the environment*, or subsets of the entire set of attributes that make up the environment. The attributes that an agent's actuators could *possibly* manipulate define a *Region of Influence* (RoI), while the attributes that an agent's sensors could *possibly* view define a *Viewable Environment* (VE).

The *VE* and the *RoI* of an agent provide us with a model that relates an agent and its individual capabilities to the environment. In order to model how two agents may interact, we need to look at how their *VEs* and *RoIs* overlap. The different ways in which these overlaps occur plays a role in determining the possible relationships between them. [4] In Figure 1, these concepts are illustrated by using an ellipse to represent the *VE* and a pentagon shape for the *RoI.* We use this notation throughout when illustrating different situations.

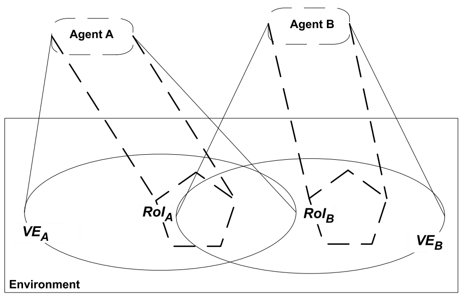

Figure 1. Region of influence affects viewable environment

2.4 Discussion

Figure 1 shows the situation in which Agent A's *RoI* overlaps with Agent B's *VE*, and both agents' *VEs* overlap. Given this information, we can infer that Agent A and Agent B could be related, with A able to directly affect the *VE* of B, since it partly falls under A's *RoI.* In other words, B can be *influenced* by the actions of A. Crucially, A cannot *directly* affect the results of an action of B because it has no influence in the *RoI* of B. For example, such a situation could occur if the overlap between the *VEs* and A's *RoI* represented research papers that Agent A made available to other agents. With the goal of reporting to other agents on all documents of a specific type (for example, research papers on multi-agent systems), B could periodically view the documents stored by A (i.e. sample the environment) while waiting for a relevant document to appear before informing other agents about its existence. Thus, whenever A performs an action that adds a relevant document to its public document store, it will eventually *influence* B's actions, since B must now inform interested parties about this addition. The illustrated situation can also be interpreted as the ability of B to *observe* the results of actions performed by A. If the document store was not public, then appropriate steps should be taken to prevent B from observing what documents were placed within it.

[4] In defining a "canonical" view of agents, Jennings [12] also briefly mentions the concept of an agent's sphere of visibility and influence. However, those notions are neither developed nor formalised as we do here.

2.5 Underlying assumptions

Before presenting the relationship typology, we clarify a number of issues relating to the diagrams and some assumptions we make about the agents themselves that hold throughout our analysis. Firstly, the *VE* and *RoI* are *not* necessarily well-defined continuous areas of the environment as the diagrams may suggest. In fact, there is no requirement for the *VE* and the *RoI* of an agent to overlap at all. If the *RoI* of an agent does not fall under its *VE*, then it is not able to view the results of its actions, a situation that is not improbable. For example, if an agent can save information to an external data store but has no means to check what information is in the data store, this could be modelled as the data store being within the *RoI* but not within the *VE*. Secondly, the model does not deal with the possibility that the *VE* of an agent may be variable because the sensors are not always reliable. Thirdly, we do not assume that the *VE* is the *only* kind of information an agent can model. An agent could also have built-in knowledge as well as communicate with other agents that provide information about the state of the environment. Finally, we do not assume that when an agent acts within its *RoI* it can be certain that those actions take place. The only way to verify this is by sensing the affected environment, either through its own sensing capabilities or through communication with other agents.

3 Relationship Analysis

Previously, we have presented an initial analysis in which more details of the underlying theory described above are given, and some exemplar types of relationships that are useful in the context of developing regulations for preventing malicious behaviour are described [1]. In this paper, we develop a comprehensive typology of interactions that can provide the building blocks for defining a wide range of different relationships. To do this we systematically examine all the salient possibilities for interactions between just two agents. First we consider the possible types of interaction when actions of other agents can be observed, and then consider the possibilities when actions can be directly influenced by other agents due to overlapping *RoIs*. To aid clarity and precision in the definitions, we use the Z language [17] used in SMART and related work [1].

3.1 Mutually Viewable Environment

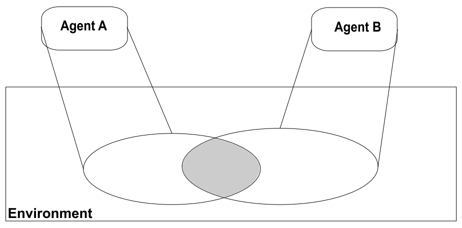

Figure 2. Mutually Viewable Environment

We begin by examing the *VEs* of agents, irrespective of the *RoIs*. With just *VEs* we can simply determine whether there is some region where they overlap or not. The schema *MutViewEnv*, which represents the mutually viewable environment, captures this situation and Figure 2 illustrates it.

Previously [1], we defined schemas for the *VE* and the *RoI*, named *ViewEnv* and *RegOfInf*. These schemas define a *viewable* function and an *roi* function, which both accept as an argument the agent state and return a set of attributes defining the *VE* and *RoI*. Agent state is defined by the *AgState* schema as the state of the agent and includes the agent's *possible* perceptions and actions in the current environment. All these definitions are used in the schemas introduced here.

In the *MutViewEnv* schema, the *mve* function takes the relationship between the two agent states and returns a set of attributes (*Att*), as defined in the predicate part. The predicates state that the mutually viewable environment of A and B is the intersection of A and B's *VE*. We should clarify that the *mve* function could return an empty set, indicating that there is no mutually viewable environment and thus describing the case where the viewable environments do not overlap.

```
MutViewEnv
  ViewEnv
  mve : (AgState × AgState) → ℙ Att
  ─────────────
  ∀ a, b : AgState; e : Environment •
      mve(a, b) = viewable a ∩ viewable b
```

Although such a situation cannot directly identify any interactions between the agents in the sense that none of the two agents are able to affect the actions of each other, it may be particularly important in certain environments. For example, the knowledge that two buyers are operating in a common online auction market and, as a result, have an overlapping *VE* since they get the same information from that market, may prove useful when attempting to explain their behaviour.

3.2 Influenced Viewable Environment

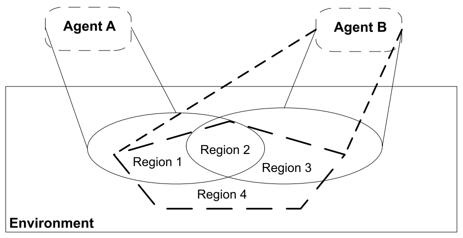

Figure 3. Observable and Invisible actions

The next step is to introduce the *RoI* for just one agent. This is done for Agent B in Figure 3 in which, from A's point of view, there are two clear distinct possibilities, and two further refinements for each. Firstly, in Regions 1 and 2, the results of the actions of B are visible to A since they fall within A's *VE*. Of course, at the same time we can say that B is able to influence the *VE* of A. Secondly, in Regions 3 and 4, the results of B's actions are not visible to A. We define these general cases before going on to specialise them further.

The schema *ObsActs* defines the appropriate function for the first case. It states that the *observable actions* of B are those actions of B whose *RoI* is within A's *VE*.

$$
\begin{array}{|l}
\textit{ObsActs} \\ \hline
\textit{ViewEnv};\ \textit{RegOfInf} \\
oa : (\textit{AgState} \times \textit{AgState}) \rightarrow \mathbb{P}\,\textit{Att} \\ \hline
\forall\, a, b : \textit{AgState} \bullet \\
\quad oa(a, b) = \textit{viewable}\ a \cap \textit{roi}\ b \\ \hline
\end{array}
$$

Similarly, the *InvActs* schema states that *invisible actions* of B with reference to A are those that are not within A's *VE*.

$$
\begin{array}{|l}
\textit{InvActs} \\ \hline
\textit{ViewEnv};\ \textit{RegOfInf} \\
ia : (\textit{AgState} \times \textit{AgState}) \rightarrow \mathbb{P}\,\textit{Att} \\ \hline
\forall\, a, b : \textit{AgState} \bullet \textit{invacts}(a, b) = \textit{roi}\ b \setminus \textit{viewable}\ a \\ \hline
\end{array}
$$

Based on these definitions, we can now describe more restricted cases. We begin with the situation in which *both* agents can observe some actions of B, which would lie in Region 2 in Figure 3. The *BiObsActs* schema defines this situation in which actions are bilaterally observable, and are given by the intersection of the *observable actions* for A on B and for B on itself.

$$
\begin{array}{|l}
\textit{BiObsActs} \\ \hline
\textit{ObsActs} \\
boa : (\textit{AgState} \times \textit{AgState}) \rightarrow \mathbb{P}\,\textit{Att} \\ \hline
\forall\, a, b : \textit{AgState} \bullet \\
\quad boa(a, b) = oa(a, b) \cap oa(b, b) \\ \hline
\end{array}
$$

Knowledge of the possibility of bilaterally observable actions can be relevant for those agents that require confirmation of their actions by another party, or for those agents that are concerned about the observability of their actions and would perhaps prefer to avoid it.

Now, unilaterally observable actions are those actions of B that A can observe but B cannot (Region 1). In this case, there is perhaps a stronger incentive for B to exploit the situation by cooperating with A so as to gain confirmation of the results of actions. The schema *UnObsActs* describes this.

$$
\begin{array}{|l}
\textit{UnObsActs} \\ \hline
\textit{ObsActs};\ \textit{InvActs} \\
uoa : (\textit{AgState} \times \textit{AgState}) \rightarrow \mathbb{P}\,\textit{Att} \\ \hline
\forall\, a, b : \textit{AgState} \bullet \\
\quad uoa(a, b) = oa(a, b) \cap ia(b, b) \\ \hline
\end{array}
$$

Bilaterally invisible actions, represented by the *BiInvActs* schema are those actions of B that both A and B cannot observe (Region 4).

$$
\begin{array}{|l}
\textit{BiInvActs} \\ \hline
\textit{InvActs} \\
bia : (\textit{AgState} \times \textit{AgState}) \rightarrow \mathbb{P}\,\textit{Att} \\ \hline
\forall\, a, b : \textit{AgState} \bullet \\
\quad bia(a, b) = ia(a, b) \cap ia(b, b) \\ \hline
\end{array}
$$

Finally, we can also define unilaterally invisible actions (Region 3), as those actions of B that A cannot see but B can. However, for the sake of space we avoid presenting the schema here.

3.3 Mutual Influence

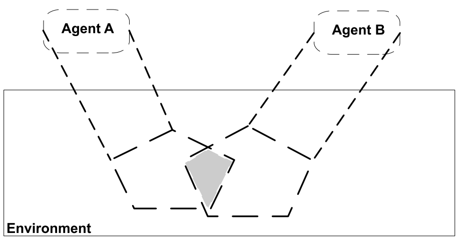

Figure 4. Mutually Influenced Actions

Up to this point, we have only dealt with the issue of observability of actions. We now move on to examine the situations in which agents can influence actions of each other, by introducing *RoIs* for both agents. In the first instance, as illustrated in Figure 4, we can say that two agents are able to directly influence each other if their *RoIs* overlap (the grey shaded area). The function for determining this is defined below, in the *MutInf* schema.

$$
\begin{array}{|l}
\textit{MutInf} \\ \hline
\textit{RegOfInf} \\
mi : (\textit{AgState} \times \textit{AgState}) \rightarrow (\mathbb{P}\,\textit{Att}) \\ \hline
\forall\, a, b : \textit{AgState} \bullet mi(a, b) = \textit{roi}\ a \cap \textit{roi}\ b \\ \hline
\end{array}
$$

Now, when a mutual influence relationship may occur (i.e. a non-empty set is returned) it is important to be able to model whether the two agents can observe the results of actions taking place in this region of the environment. We can use the previous definitions on observability of actions to model this.

First, we define the relationship by which Agent A can observe the region of mutual influence in the *ObsMutInf* schema, which includes the *MutInf* schema and states that this area is the intersection of the *VE* of A and the area of mutual influence between A and B.

$$
\begin{array}{|l}
\textit{ObsMutInf} \\ \hline
\textit{MutInf} \\
omi : (\textit{AgState} \times \textit{AgState}) \rightarrow (\mathbb{P}\,\textit{Att}) \\ \hline
\forall\, a, b : \textit{AgState} \bullet \\
\quad omi(a, b) = \textit{viewable}\ a \cap (mi\ (a, b)) \\ \hline
\end{array}
$$

Similarly, Agent A may not be able to observe this region of mutual influence. We define the case of invisible mutual influence in the schema *InvMutInf*.

$$
\begin{array}{|l}
\textit{InvMutInf} \\ \hline
\textit{MutInf} \\
imi : (\textit{AgState} \times \textit{AgState}) \rightarrow (\mathbb{P}\,\textit{Att}) \\ \hline
\forall\, a, b : \textit{AgState} \bullet \\
\quad imi(a, b) = (mi\ (a, b)) \setminus \textit{viewable}\ a \\ \hline
\end{array}
$$

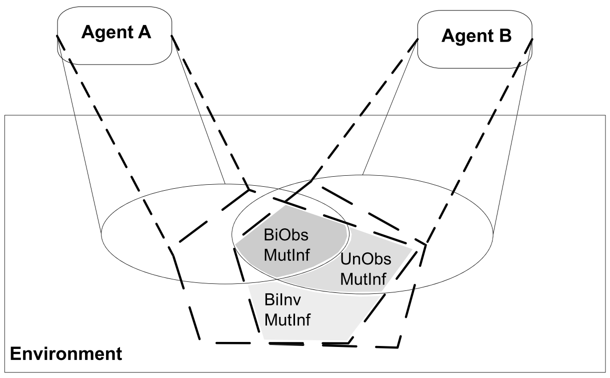

Figure 5. Bilateral Mutual Influence

Having provided definitions from one agent's perspective, we consider the situation in which both agents' *VEs* are examined. The first case is bilaterally observable mutual influence, in which both agents can observe the mutual influence area, as illustrated in Figure 5. The region under question is where both agents' *RoIs* overlap as well as their *VEs*. The *BiObsMutInf* schema formalises this.

```
BiObsMutInf
  ObsMutInf
  $bomi : (AgState \times AgState) \rightarrow \mathbb{P}\, Att$
  ----------------
  $\forall\, a, b : AgState \bullet$
      $bomi(a, b) = omi(a, b) \cap omi(b, a)$
```

The *BiInvMutIf* schema provides the necessary functions for the mutual influence being bilaterally invisible.

```
BiInvMutInf
  InvMutInf
  $bimi : (AgState \times AgState) \rightarrow \mathbb{P}\, Att$
  ----------------
  $\forall\, a, b : AgState \bullet$
      $bimi(a, b) = imi(a, b) \cap imi(b, a)$
```

Finally, we define the situation in which one agent unilaterally observes the region of mutual influence in the *UnObsMutInf* schema.

```
UnObsMutInf
  ObsMutInf
  $uomi : (AgState \times AgState) \rightarrow \mathbb{P}\, Att$
  ----------------
  $\forall\, a, b : AgState \bullet$
      $uomi(a, b) = (omi\ (a, b) \setminus omi\ (b, a))$
```

These types of possible relationships are particularly relevant for agents that wish to better coordinate their actions. For example, knowledge of a possible bilaterally invisible mutual influence can indicate that agents should be particularly careful when performing actions in that region since not only are they unable to observe the results of those actions, but they can also upset actions of another agent that is also unable to observe the results. A situation of unilaterally observable mutual influence could give one agent the upper hand, since only one of them is able to observe the results of its and the other's actions in that region. We illustrate through an example how these relationship types can be used later on.

3.4 Generic Relationship Types

In this section we illustrate how the definitions provided above can be used to define other types of relationships. We begin by dealing with a relationship type presented previously [1]. There we described a *Weak Influence* relationship as occurring when one agent could directly influence the *VE* of another agent outside of that agent's *RoI*. We can now define this relationship more concisely by using the "building blocks" provided above. In the *WeakInfluence* schema, we include the *ObsActs* schema and state that A can weakly influence B where B can observe A's actions and there is no overlap with B's *RoI*.

```
WeakInfluence
  ObsActs
  $wi : (AgState \times AgState) \rightarrow \mathbb{P}\, Att$
  ----------------
  $\forall\, a, b : AgState \bullet wi(a, b) = (oa\ (b, a)) \setminus roi(b)$
```

Another example of an interesting type of relationship is when one agent can be said to subsume all the sensory and effectory capabilities of another agent. In a design situation, identifying this could raise the question of whether both agents are required. The *SubsumingInfluence* schema inludes the *ViewEnv* and RegOfInf schemas. It defines a subsuming influence as occuring where the *VE* of B is a subset of the *VE* of A and the same happens for the *RoIs* of A and B.

```
SubsumingInfluence
  $ViewEnv \wedge RegOfInf$
  $si : (AgState \times AgState) \rightarrow \mathbb{P}\, Att$
  ----------------
  $\forall\, a, b : AgState \bullet$
      $si(a, b) = viewable\ b \subset viewable\ a\ \wedge$
      $si\ (a, b) = roi\ b \subset roi\ a$
```

4 Goals Region Analysis

Knowledge of the sensor and actuator capabilities of agents can provide us with enough information to identify a significant number of possible relationships and categorise them along the lines of the typology introduced above. Nevertheless, not *all* the possible relationships identified will actually be instantiated, and there may still be instantiated relationships that have not been identified. The reason for this is that the goals agents decide to pursue play an important role in determining which of all the possible relationships agents choose to instantiate. With the additional knowledge of what goals an agent may actually pursue, we can narrow or expand the space of possible relationships by identifying interactions that an agent may pursue that are beyond its range in terms of its *RoI* or its *VE* , or by excluding those within its *VE* and *RoI* that it will not pursue. Therefore, a more focused analysis of relationships between agents could take place if we can incorporate knowledge of which regions of the environment an agent's goals refer to into the model of agent interaction with the environment. In order to achieve this, we provide a typology of agent goals with reference to an agent's *VE* and *RoI*. However, before we do that we need to differentiate between different types of goals according to whether the goal is to retrieve information from the environment or change it.

4.1 Query and Achievement Goals

In the broadest sense agents can have only two types of goals. On the one hand, they may want to *effect some change* in the environment, which implies changing attributes of the environment, while on the other hand, they may just want *some information about the environment*, which does not lead to any direct changes in the environment. Distinguishing between these two types of goals is important since the latter can only be achieved directly by an agent if that goal is in the *RoI* of the agent, while the former can only be achieved if the goal is in the *VE*of the agent.

We distinguish between these two types of goals by using the same terminology as the dMARS agent system, which is formalised in [7] using the SMART framework. Essentially, a *query* goal is one for which an agent tries to elicit some information, either from its internal beliefs or from the environment. As such, it can be satisfied if it falls within an agent's *VE*. Conversely, an *achievement* goal may require that the agent performs certain actions in order to change the environment, if the environment is not already in the desired goal state. Thus, an *achievement* goal can be satisfied if it lies within an agent's *RoI*.

4.2 Goal Regions

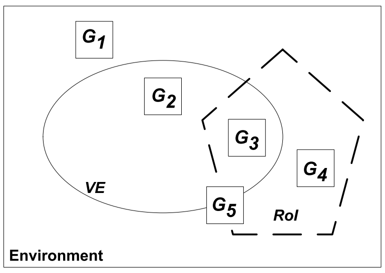

Figure 6. Types of goals

We categorise goals according to where they occur within the *VE* and *RoI* of the agent pursuing the goal. In Figure 6 we represent goals by a square including a capital G. The different situations are described below.

No control — G_1 The agent has a goal that describes an environmental state falling outside of both the agent's *VE* and its *RoI*. As a result, this agent has no *control* over satisfying that goal, irrespective of whether it is a *query* or *achievement* goal. Some form of cooperation with another agent would be essential in this case.

View control — G_2 In this case, the agent can satisfy a *query* goal but not an *achievement* goal, since the goal is within the agent's *VE*.

Total control — G_3 A *total control* goal is one that lies both within the agent's *VE* and *RoI*. As a result, regardless of whether it is a *query* or *achievement* goal, the agent can satisfy it.

Blind Control — G_4 In this case, the goal falls within the agent's *RoI* but not within its *VE*. As a result, the agent is able to satisfy it if it is an *achievement* goal but not if it is a *query* goal. However, the agent is not able to *verify* the results of its actions.

Partial Control — G_5 Finally, a goal may fall in a region that is partially under the agent's *VE* or the agent's *RoI*. In this case, the agent will have some combination of control based on the four types described above.

The *AgentGoals* schema formalises the four main cases above. It includes the *AgState*, *ViewEnv* and *RoI* schemas and uses the *viewable* and *roi* functions from them.

```
AgentGoals
  AgState; ViewEnv; RegOfInf
  canview, caninf : ℙ Att
  none, blind, total, view : ℙ Goal
  ────────────
  canview = viewable (θAgState)
  caninf = roi (θAgState)
  none =
      {g : goals | ¬ g ⊆ (canview ∪ caninf)}
  blind =
      {g : goals | g ⊆ (caninf \ canview)}
  view =
      {g : goals | g ⊆ (canview \ caninf)}
  total =
      {g : goals | g ⊆ (canview ∩ caninf)}
```

With the goal typology in place, as well as the interaction typology, we have two significant analytical tools for identifying and characterising possible relationships between agents.

5 Example Analysis

Consider a situation in which we wish to develop an agent-based infrastructure to support the collaboration and sharing of information between researchers in a computer science research lab. Each researcher is to be represented by a personal agent that will make public their personal profile (interests, publications, and availability) as well as research material (links to online material, presentations, software, etc) that they have stored locally.

An initial analysis of the domain reveals that researchers typically use at least two devices in the lab to achieve their day-to-day tasks, including one relatively powerful desktop or laptop computer, as well as a more limited mobile device, such as PDA. In order to effectively support the users, the application should allow use of the agent-based system through all user devices, requiring agents to be installed on each device. Furthermore, agents serving the same user through different devices should cooperate. Each agent is to take advantage of the connectivity, storage and computing capabilities of their device so as to more effectively support the user.

Given this, there are two main problems to solve from the perspective of enabling the cooperation between a user's devices. Firstly, how can coordination and cooperation be effectively supported if there is no clear knowledge, at design time, of the exact capabilities of each of the devices or the exact tasks that they may attempt to carry out, since this depends on the equipment of each user and their individual choices as to how they want to use the system. Secondly, how can the infrastructure deal with changes in devices and possible changes in the application requirements as the system develops. These two problems make it practically impossible to define coordination using any application-specific knowledge, such as the connectivity capabilities of a device. This information will be discovered at run-time and the type of coordination based on the discovered information must be decided at run-time.

In this case, the ability to define coordination based on generic relationship types is valuable. The agents belonging to a single user are instructed to communicate, whenever possible, so as to share information on their capabilities and the current user goals. This allows the modelling both of the relationship between them and of the goals that must be satisfied. With such knowledge in place, we can attempt to guide coordination, at run-time, by defining generic rules such as the ones presented below.

- If there exists a goal which is of type *total control* for only one device then that device should attempt to achieve the goal, since it is the only one that can both attempt the actions and verify the results.
- If for a common goal two devices (or more) are in a relationship of *Bilaterally Observable Mutual Influence* (i.e. where their *RoIs* and *VEs* overlap) it means that they could both attempt to achieve it. In this case, we could use other information, such as identifying the workstation and assigning the goal to that since we can make the assumption that its resources will be more readily available and not limited by battery concerns or unreliable connections due to solely wireless access.
- If a goal is of type *view control* for one device and *blind control* for the other, which implies that there is some region of the environment in which they are in a relationship of *Unilaterally Observable Actions*, the agents should cooperate, with one performing the actions and the other verifying the results.

For example, while attending a presentation, a user requests that the personal agent on a Bluetooth-enabled device collects all information on the topic of the meeting that is available through other researchers in the lab and downloads any relevant publications. The user then switches off the device, because the battery is running out. Once back at the desk and at the workstation, the mobile device is switched on and communicates wirelessly or through the usual synchronisation mechanisms with the workstation. The information on goals and capabilities is exchanged and the two agents identify that while they can both access information on other users, for the PDA-based agent the *VE* is limited to just those users whose information is accesible via Bluetooth-enabled devices that are in range. The workstation on the other hand is able to access to all relevant users through their workstation agents, so it adopts the goal.

Through this basic example, we see how access to a relationship analysis tool that can identify generic relationship types can play a valuable role in facilitating coordination between agents at *run-time*. Given the emerging landscape of computing environments, in which constant change and heterogeneity become permanent features, such tools will become increasingly important.

6 Conclusions and Further Work

In this paper we introduced a typology of relationships in support of coordination and regulation, building on a basic model of interaction between agents and the environment. Furthemore, we associated these relationship types to the goals of an agent, by defining goal regions. The combination of relationship analysis with information on goals can provide a useful tool for identifying possibilities for coordination between agents, especially in situations in which we cannot predefine coordination because of incomplete information about an agent's capabilities and goals. The same tools can also be used to identify how a multi-agent system should be regulated to avoid conflicts, as we discuss in [1].

The issue of relationship analysis has not in general been sufficiently addressed by existing research. Initial attempts such as [14] take a different approach, since the analysis tools are geared towards learning about agent behaviour and are focused on analysing teams of agents. Although there is also a wider body of work on conflict management (a representative collection can be found in [19]), once more the agent identification issue is not addressed. The work presented here also has some similarities to social dependency networks [2, 16], which where also modeled using SMART [8]. However, our approach differs, since we make minimal assumptions about other agents, basing our models solely on agent interaction with the environment and the observability of those actions.

The key contribution of our approach is the introduction of a typology of relationships that can be applied to a wide range of systems due to the minimal assumptions it makes about agents themselves. The typology can both aid during design, as well as form the core of a run-time agent management tool. The system ties into a wider theory on agent systems in general [9] and as such benefits from the clarity of exposition on the underlying concepts. Further work will include the development of appropriate tools to allow the automated analysis of agent relationships, so that they can be incorporated within the toolkit of agent application developers as well as the integration of such techniques within existing agent methodologies. We aim to investigate the *scalability* of the approach as well as precise mechanisms that will enable agents to form societies based on reasoning about other agents using the models presented here. Furthermore, we aim to experiment with the typology as a a means to provide a general analysis of agent systems aiming towards the definition of metrics revealing issues such as the level of *potential interference* between agents.

REFERENCES

[1] R. Ashri, M. Luck, and M. d'Inverno, 'On Identifying and Managing Relationships in Multi-Agent Systems', in *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, eds., George Gottlob and Toby Walsh, pp. 743–748. Morgan Kaufmann Publishers, (2003).

[2] C. Castelfranchi, M. Miceli, and A. Cesta, 'Dependece relations among autonomous agents', in *Decentralised Artificial Intelligence*, eds., E. Werner and Y. Demazeau, 215–231, Elsevier, (1992).

[3] Cristiano Castelfranchi, Frank Dignum, Catholijn M. Jonker, and Jan Treur, 'Deliberative Normative Agents. Principles and Architecture', in *Intelligent Agents IV (ATAL99)*, eds., N. Jennings and Y. Lesperance, volume 1757 of *LNCS*, pp. 364–378. Springer, (2000).

[4] P. Ciancarini, R. Tolksdorf, and F. Zambonelli, 'A Survey of Coordination-Middleware for XML-Centric Applications', *The Knowledge Engineering Review*, **17**(4), (2003).

[5] Daniel David Corkill and Victor Lesser, 'The use of meta-level control for coordination in a distributed problem solving network', in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, eds., Alan H. Bond and Les Gasser, pp. 748–756. Morgan kaufmann Publishers, (1983).

[6] Frank Dignum, 'Autonomous Agents with Norms', *Artificial Intelligence and Law*, (7), 69–79, (1999).

[7] Mark d'Inverno, David Kinny, Michael Luck, and Michael Wooldridge, 'A Formal Specification of dMARS', in *Intelligent Agenrs IV: Proceedings of the Fourth International Workshop on Agent Theories, Architectures and Languages*, eds., Munindar P. Singh, Anand S. Rao, and Michael Wooldridge, volume 1365 of *LNCS*, pp. 155–176. Springer, (1996).

[8] Mark d'Inverno and Michael Luck, 'A Formal View of Dependece Networks', in *Distributed Artificial Intelligence Architecture and Modelling: Proceedings of the First Australian Workshop on Distributed Artificial Intelligence*, eds., C. Zhang and D. Lukose, volume 1087 of *LNCS*, pp. 155–129. Springer, (1996).

[9] Mark d'Inverno and Michael Luck, *Understanding Agent Systems*, Springer, 2nd edn., 2004.

[10] Naranker Dulay, Nicodemos Damianou, Emil Lupu, and Morris Sloman, 'A policy language for the management of distributed agents', in *Agent-Oriented Software Engineering II*, eds., M. J. Wooldridge, G. Weiss, and P. Ciancarini, volume 2222, pp. 84–100. Springer-Verlag, (2001).

[11] Marc Esteva, David de la Cruz, and Carles Sierra, 'ISLANDER: an electronic institutions editor', in *The First International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1045–1052. ACM Press, (2002).

[12] Nicholas R. Jennings, 'On agent-based softare engineering', *Artificial Intelligence*, **117**(2), 277–296, (2000).

[13] Lalana Kagal, Tim Finin, and Anupam Joshi, 'A Policy Based Approach to Security for the Semantic Web', in *Proceedings of the 2nd International Semantic Web Conference*, eds., Dieter Fensel, Katia Sycara, and John Mylopoulos, volume 2870 of *LNCS*, pp. 402–418. Springer, (2003).

[14] Ranjit Nair, Milind Tambe, Stacy Marsella, and Taylor Raines, 'Automated assistants for analyzing team behaviours', *Journal of Autonomous Agents and Multi-Agent Systems*, **8**(1), (2004).

[15] Andrea Omicini, Alessandro Ricci, Mirko Viroli, and Giovanni Rimassa, 'Integrating Objective & Subjective Coordination in MultiAgent Systems', in *19th ACM Symposium on Applied Computing (SAC 2004)*, pp. 485–491. ACM Press, (2004).

[16] J.S. Sichman, Y. Demazeau, R. Conte, and C. Castelfranchi, 'A social reasoning mechanism based on dependence networks', in *11th European Conference on Artificial Intelligence*, pp. 188–192. John Wiley and Sons, (1994).

[17] J.M. Spivey, *The Z Notation*, Prentice Hall, 2nd edn., 1992.

[18] Niranjan Suri, Marco Carvalho, Jeffrey Bradshaw, Maggie R. Breedy, Thomas B. Cowin, Paul T. Groth, Raul Saavedra, and Andrzej Uszok, 'Enforcement of Communications Policies in Software Agent Systems through Mobile Code', in *4th IEEE International Workshop on Policies for Distributed Systems and Networks*, pp. 247–250. IEEE Computer Society, (2003).

[19] *Conflicting Agents: Conflict Management in Multi-Agent Systems*, eds., Catherìne Tessier, Laurent Chaudron, and Heinz-Jurgen Muller, Kluwer Publishers, 2000.

[20] Javier Vzquez-Salceda and Frank Dignum, 'Modelling electronic organizations', in *Multi-Agent Systems and Applications III*, eds., Vladimír Marík and Jörg Müller, volume 2691, pp. 584–593. Springer, (2003).

[21] Fabiola Lopez y Lopez, Michael Luck, and Mark d'Inverno, 'Constraining autonomy through norms', in *The First International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 674–681. ACM Press, (2002).