In Distributed Artificial Intelligence Architecture and Modelling: Proceedings of the First Australian Workshop on Distributed Artificial Intelligence, C. Zhang and D. Lukose, (eds.), Lecture Notes in Artificial Intelligence, 1087, 70-84, Springer-Verlag, 1996

Engagement and Cooperation in Motivated Agent Modelling

Michael Luck[1] and Mark d'Inverno[2]

[1] Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK.
Email: mikeluck@dcs.warwick.ac.uk
[2] School of Computer Science, University of Westminster, London, W1M 8JS, UK.
Email: dinverm@westminster.ac.uk

Abstract. The title of this paper suggests two distinct aspects of the models that we propose and consider. The first of these is the modelling of other agents by *motivated* agents. That is to say that the act of modelling is itself motivated and constrained by the agent doing that modelling. The second aspect is that all such models will also be of motivated agents. It is not sufficient merely to know what other agents are like, but also to know why they are like that. This *why* aspect is what provides the extra information that allows a greater understanding of the interactions between entities in the world, and consequently provides for more resilient agents capable of effectively dealing with new and unforeseen circumstances in an uncertain world. Previous work has described a formal framework for agency and autonomy in which agents are viewed as objects with *goals*, and autonomous agents are agents with *motivations*. This paper considers the nature of cooperation within that framework. We identify distinct kinds of interaction, depending on the nature of the entities involved. In particular, we describe and specify the differences that arise in these interactions which we characterise as *engagements* of non-autonomous agents, and *cooperation* between autonomous agents.

1 Introduction

Recent work in artificial intelligence has begun to investigate many aspects of single-agent and multi-agent systems. One reason for the concentration of effort into agent-oriented work is that much previous research was concerned with toy problems unrelated to the embodiment of the problem-solver, or whether it was situated in a real external environment. Thus, though significant advances have been made through such work, these have been limited since many of the solutions developed will not adapt well to more realistic scenarios. The relatively recent recognition of these limitations has been a key motivating factor in the research and development of *agents* as systems capable of intelligent behaviour in a resilient and flexible way.

The rapid growth of the field, however, has led to diverse notions of agents and autonomous agents, which are only now, at least partly, becoming reconciled[14]. Our work seeks to contribute to that reconciliation by providing formal but accessible models of agents and autonomous agents, and their interactions.

In previous work, we have proposed definitions of *agency* and *autonomy*, and described how autonomy is distinct but is achieved by *motivating* agency [6] and generating goals [7]. This paper considers aspects of agent modelling within that framework, and describes how the framework provides useful structure that can be exploited by intelligent agents for more effective operation. We begin by outlining previous work on the agent hierarchy. Then we consider the types of interactions that occur in the world, distinguishing in particular between *engagements* of non-autonomous agents and *cooperation* between autonomous agents. We end by illustrating how these relationships structure the information that is available to an agent enabling a more appropriate use of resources.

2 A Framework for Agency and Autonomy

In essence, our framework is a three-tiered hierarchy of entities comprising *objects*, *agents* and *autonomous agents*. In this hierarchy, all known entities are objects of which some are agents, and of these agents, some are autonomous agents. This is shown as a Venn diagram in Figure 1. Note the addition of some extra categories to be used later in this paper. The central set covers autonomous agents, the ring enclosing it covers non-autonomous or *server-agents* (SAgents in the diagram), and the outer ring covers non-agent objects or *neutral-objects* (NObjects). This section briefly outlines the agent hierarchy. Many details are omitted, and a more complete treatment can be found in [6].

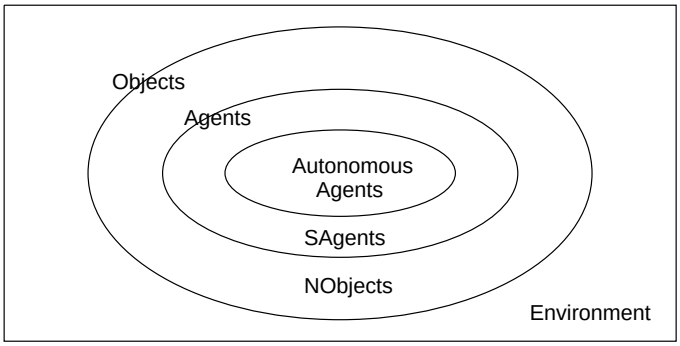

Fig. 1. The Entity Hierarchy

An object is just something with abilities and attributes. For example, the attributes of a table specify that it is stable, wooden, brown and has a flat surface, while its capabilities specify that it can support things. Another well-used example is a robot without a power supply whose capabilities are limited and include just those which rely on its physical presence, such as supporting things, weighing things down, and so on. Its attributes specify that it is blue, that it is upright, that it is large, and so on.

An agent is an object that is (typically) useful to another agent where this usefulness is defined in terms of satisfying that agent's goals. In other words, an agent is an object with an associated set of goals. For example, if I support my computer on a table, then the table is my agent for supporting the computer. The table may not actually *possess* the goal, but is certainly satisfying it. A particular object may produce different instantiations of agents which are created in response to another agent. Agency is thus *transient*, and objects may become agents at some times, while those agents may revert to objects at other times.

For example, if the robot of the earlier example now has a power supply, there are two possibilities. In the first case, the robot does not have a goal and can therefore use its actuators only in a random way, and must be considered an object. In the second case, the robot does have a goal, and this allows it to use its actuators in a *directed* way, such as riveting a panel onto a hull. This means that it is an agent. The goal need not be explicit, but can instead be implicit in the robot's design.

In this sense, an agent can be benevolent. If a robot is designed simply to accept requests to pick up panels and rivet them onto a hull without any evaluation of the request, then it is benevolent. Moreover it is an agent for the *requesting* agent. This is important, for by modelling the relationship of one agent satisfying the goal of another, we can infer much about the situation. For example, if I want the robot to be my agent and satisfy my goals by riveting particular panels for me, then I may have to negotiate with the agent whose goals are currently being satisfied by the robot. Thus agency indicates a relationship between two or more parties, providing useful information about the possibility of interacting with them. (The special case of autonomous agency is considered next.)

This notion of agency relies upon the existence of other agents which provide goals that are adopted in order to instantiate an agent. We must therefore have some agents which can generate their own goals so that there is not an infinite regress in adopting goals. It is these goal-generating agents which are *autonomous*, since they do not depend on the goals of others, but instead possess goals which are *generated* from internal *motivations* which characterise the agent. Motivations are related to goals, but are qualitatively different in that they are not describable states of affairs in the environment. The motivation *greed*, for example, does not specify a state of affairs to be achieved, but may give rise to the generation of a goal to rob a bank. The motivation provides a *reason* for achieving the goal which specifies *what* to do.

In psychology, Kunda [5] informally defines motivation to be, "Any wish, desire, or preference that concerns the outcome of a given reasoning task," and suggests that motivation affects reasoning in a variety of ways including the accessing, constructing and evaluating of beliefs and evidence, and decision making. An autonomous agent is an agent with motivations and some means of evaluating behaviour in terms of the environment and those motivations, so that its behaviour is determined by both external and internal factors. An autonomous agent must be a motivated agent.

Our example robot will be autonomous if it has an internal mechanism for goal generation. For example, with motivations of achievement, hunger and self-preservation, where achievement is defined in terms of riveting panels, hunger in terms of maintaining power levels, and self-preservation in terms of avoiding system breakdowns, the robot will normally generate goals to rivet panels. With low power levels, however, it may instead generate a new goal to recharge its batteries. Alternatively, if it works for too long and is in danger of overheating, it may generate a goal of pausing to avoid damage to its components. In the view described above, the robot is autonomous because its goals are internally generated through motivations in response to its environment.

To present a mathematical description of our definitions, we use the specification language, Z [13], which is increasingly being used in the AI community [4]. However, we are sometimes loose with the syntax when detailing sequences and sets, for example, for the purposes of concision and explication.

An *entity* comprises a set of *motivations*, a set of *goals*, a set of *actions*, and a set of *features* or *attributes* of the entity.

$$
\begin{array}{|l}
\hline
Entity \\
\hline
attributes : \mathbb{P}\ Attribute \\
capableof : \mathbb{P}\ Action \\
goals : \mathbb{P}\ Goal \\
motivations : \mathbb{P}\ Motivation \\
\hline
attributes \neq \{\} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
Object \\
\hline
Entity \\
\hline
capableof \neq \{\ \} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
Agent \\
\hline
Object \\
\hline
goals \neq \{\} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
AutonomousAgent \\
\hline
Agent \\
\hline
motivations \neq \{\} \\
\hline
\end{array}
$$

An object, described by its features and capabilities, is just an automaton and cannot engage other entities to perform tasks, nor is it itself used in this way. However, it serves as a useful abstraction mechanism by which it is regarded as distinct from the remainder of the environment, and can subsequently be transformed into an agent, an augmented object, engaged to perform some task or satisfy some goal. Viewing something as an agent means that we regard it as satisfying or pursuing some goal. Furthermore, it means that it must be doing so either for another agent, or for itself, in which case it is autonomous. If it is for itself, it must have generated the goal through its motivations.

For the purposes of the following sections in which we discuss the kinds of relationship that can be modelled, we further refine the agent hierarchy through

the introduction of two new terms. A *neutral-object* is an object that is *not* an agent, and a *server-agent* is an agent that is *not* an autonomous agent.

$$
\begin{array}{|l}
\hline
NeutralObject \\
Object \\
\hline
goals = \{\} \land motivations = \{\} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
ServerAgent \\
Agent \\
\hline
motivations = \{\} \\
\hline
\end{array}
$$

We must consider the multi-agent world as a whole rather than just individual agents. This world is defined below, where all autonomous agents are agents, all agents are objects and all objects are entities. An agent is either a server-agent or an autonomous agent, and an object is either a neutral-object or an agent.

$$
\begin{array}{|l}
\hline
World \\
entities : \mathbb{P}\ Entity \\
objects : \mathbb{P}\ Object \\
agents : \mathbb{P}\ Agent \\
autonomousagents : \mathbb{P}\ AutonomousAgent \\
neutralobjects : \mathbb{P}\ NeutralObject \\
serveragents : \mathbb{P}\ ServerAgent \\
\hline
autonomousagents \subseteq agents \subseteq objects \subseteq entities \\
agents = autonomousagents \cup serveragents \\
objects = neutralobjects \cup agents \\
\hline
\end{array}
$$

3 Engagement and Cooperation

Much existing work has defined cooperation only in terms of helpful agents that are predisposed to adopt the goals of another (e.g.[12, 2]). This assumes that agents are already designed with common or non-conflicting goals that facilitate the possibility of helping each other satisfy additional goals. Our view differs in that autonomous agents will only adopt a goal if it is to their advantage to do so, while non-autonomous agents may benevolently adopt goals. This leads to the distinction between *cooperation* and *engagement* as discussed below.

3.1 Engagement

A direct engagement occurs when a neutral-object or a server-agent adopts some goals. In a direct engagement, an agent with some goals, which we call the *client*, uses another agent, which we call the *server*, to assist them in the achievement of those goals. Remember that a server-agent is non-autonomous, and either exists already as a result of some other engagement, or is instantiated from a neutral-object for the current engagement. No restriction is placed on a client-agent.

We define a *direct engagement* to consist of a client agent, *client*, a server agent, *server*, and the goal that *server* is satisfying for *client*. An agent cannot engage itself, and both agents must have the goal of the engagement.

DirectEngagement

$$client : Agent$$
$$server : ServerAgent$$
$$goal : Goal$$

$$client \neq server$$
$$goal \in (client.goals \cap server.goals)$$

The set of all *direct* engagements in the world is given by *dengagement*. For any direct engagement in *dengagement*, there can be no intermediate *direct* engagements of the goal, so there is no other agent, y, where *client* engages y for *goal*, and y engages *server* for *goal*. An agent, c, *directly engages* another server-agent, s, if, and only if, there is a direct engagement between c and s. All of these relationships are given as a set denoted by *dengages*. Finally, the server-agents comprise all agents which are the server agent for some direct engagement and the agents are a superset of those agents which are part of some engagement.

WorldEngagements

$$World$$
$$dengagement : \mathbb{P}\ DirectEngagement$$
$$dengages : Agent \leftrightarrow ServerAgent$$

$$\forall\, eng : dengagement \bullet \neg\, (\exists\, y : Agent; e_1, e_2 : dengagement \mid e_1.goal = e_2.goal = eng.goal \bullet e_1.server = eng.server \wedge e_2.client = eng.client \wedge e_1.client = e_2.server = y)$$
$$dengages = \{e : dengagement \bullet (e.client, e.server)\}$$
$$serveragents = \{d : dengagement \bullet d.server\}$$
$$\{d : dengagement \bullet d.server\} \cup \{d : dengagement \bullet d.client\} \subseteq agents$$

An *engagement chain* represents a sequence of *direct engagements*. For example, if I use a computer terminal to run a program to access a database in order to locate a library book, then there is a direct engagement of myself and the terminal, of the terminal and the program, and of the program and the database, all with the goal of locating the book. An engagement chain thus represents the goal and all the agents involved in the sequence of direct engagements. In the above example, the agents are: Me, $Terminal$, $Program$, $Database$

Specifically, an *engagement chain* comprises some goal, *goal*, the autonomous client-agent that generated the goal, *autoagent*, and a sequence of server-agents, *chain*, where each agent in the sequence directly engages the next. For any engagement chain, there must be at least one server-agent, all the agents involved must share *goal*, and each agent can only be involved once.

EngagementChain

$$goal : Goal$$
$$autoagent : AutonomousAgent$$
$$chain : \mathsf{seq}_1\ Agent$$

$$goal \in autoagent.goals$$
$$goal \in \bigcup \{s : Agent \mid s \in \mathsf{ran}\ chain \bullet s.goals\}$$
$$\#(\mathsf{ran}\ chain) = \#chain$$

The set of all engagement chains in the world is given in the schema below

by *engchain*. For every engagement chain, *ec*, there must be a direct engagement between the autonomous agent, $ec.autoagent$, and the first client, of *ec*, $head\ ec.chain$, with respect to the goal of *ec*, $ec.goal$. Further, there must be a direct engagement between any two agents which follow each other in $ec.chain$ with respect to $ec.goal$. In addition, all the autonomous agents involved in an engagement chain are a subset of all the autonomous agents.

WorldEngagementChains

$$
\begin{array}{l}
WorldEngagements \\
engchain : \mathbb{P}\ EngagementChain \\
\hline
\forall\, ec : engchain;\ s_1, s_2 : Agent \bullet \\
\qquad (\exists\, d : dengagement \bullet\ d.goal = ec.goal \land d.client = ec.autoagent \\
\qquad\qquad \land\ d.server = head\ ec.chain) \land \\
\qquad \langle s_1, s_2 \rangle \ \mathsf{in}\ ec.chain \Rightarrow (\exists\, d : dengagement \bullet \\
\qquad\qquad d.client = s_1 \land d.server = s_2 \land d.goal = ec.goal) \\
\{ec : engchain \bullet ec.autoagent\} \subseteq autonomousagents
\end{array}
$$

Now, in order to additionally specify some other relations between agents, we will make use of the generic relation *follows*, defined below. This holds between a pair of elements and a sequence of elements if the first element of the pair precedes the second element in the sequence.

$[X]$

$$
\begin{array}{l}
follows : (X \times X) \leftrightarrow \mathsf{seq}\ X \\
\hline
\forall\, a, b : X;\ s : \mathsf{seq}\ X \bullet ((a, b), s) \in follows \Leftrightarrow \\
\qquad (\exists\, t, u, v : \mathsf{seq}\ X \bullet s = t \frown \langle a \rangle \frown u \frown \langle b \rangle \frown v)
\end{array}
$$

In general, an agent *engages* another agent if there is some engagement chain in which it precedes the server agent. An agent *owns* another agent, if there is no other agent using it for a different purpose. In other words, *c* owns *s* if, for every sequence of server-agents in an engagement chain in which *s* appears, *c* precedes it or is the autonomous client-agent that initiates the chain. Lastly, an agent *c directly owns* another agent *s*, if it owns it, and is directly engaging it.

AgentRelations

$$
\begin{array}{l}
WorldEngagementChains \\
engages, owns, downs : Agent \leftrightarrow ServerAgent \\
owns : Agent \leftrightarrow ServerAgent \\
downs : Agent \leftrightarrow ServerAgent \\
\hline
engages = \{ec : engchain \bullet (ec.autoagent, head\ ec.chain)\} \cup \\
\qquad \{ec : engchain;\ c, s : Agent \mid ((c, s), ec.chain) \in follows \bullet (c, s)\} \\
\forall\, c : Agent;\ s : Agent \bullet (c, s) \in owns \Leftrightarrow (\forall\, ec : engchain \mid s \in \mathsf{ran}\ ec.chain \bullet \\
\qquad\qquad ec.autoagent = c \lor ((c, s), ec.chain) \in follows) \\
\forall\, c : Agent;\ s : Agent \bullet (c, s) \in downs \Leftrightarrow (c, s) \in owns \cap dengages
\end{array}
$$

3.2 Cooperation

Two autonomous agents are said to be *cooperating* with respect to some goal if one of the agents has adopted goals of the other. This notion of autonomous

goal acquisition applies both to the *origination* of goals by an autonomous agent for its own purposes, and the *adoption* of goals from others, since in each case the goal must have a positive motivational effect [7]. For autonomous agents, the goal of another can only be adopted if it has such an effect, and this is also exactly why and how goals are originated. Thus goal adoption and origination are related forms of goal generation.

That is to say that the term *cooperation* can be used only when those involved are autonomous and, at least potentially, capable of resisting. If they are not autonomous, nor capable of resisting, then one simply *engages* the other. The difference between engagement and cooperation is in the autonomy or non-autonomy of the entities involved. It is senseless, for example, to consider a terminal cooperating with its user, but meaningful to consider the user engaging the terminal. Similarly, while it is not inconceivable for a user to engage a secretary, it makes better sense to say that the user and the secretary are cooperating, since the secretary can withdraw assistance at any point. This applies at the level of the definitions in the model, and in real situations. Cooperation is thus a *symmetric* relation between two autonomous agents. Engagement, by contrast, is asymmetric between a server-agent and another client agent.

A *cooperation* describes a goal, the autonomous agent that generated the goal, and those autonomous agents who have adopted that goal from the generating agent. Thus in this view, cooperation cannot occur unwittingly between agents, but must arise as a result of the motivations of both of the individuals involved.

Cooperation

$$
\begin{array}{l}
goal : Goal \\
generatingagent : AutonomousAgent \\
cooperatingagents : \mathbb{P}\, AutonomousAgent \\
\hline
\#cooperatingagents \geq 1 \\
\forall\, aa : cooperatingagents \bullet goal \in aa.goals \\
goal \in generatingagent.goals
\end{array}
$$

The set of all cooperations in the world is given by *cooperations*. Further, we say that agent x_1 *cooperates* with agent x_2 if and only if x_1 and x_2 are autonomous, there is some cooperation in which either x_1 or x_2 is the agent that generated the goal, and x_2 or x_1 respectively is one of the cooperating agents. The set of all such relationships is given below by *cooperates*. We assert also that the relationship is symmetric — it is equal to its own inverse — since if agent x_1 is cooperating with x_2 then, necessarily, x_2 is cooperating with x_1. Thus order in cooperation is not relevant.

WorldCooperations

$$
\begin{array}{l}
World \\
cooperations : \mathbb{P}\, Cooperation \\
cooperates : AutonomousAgent \leftrightarrow AutonomousAgent \\
\hline
cooperates = \bigcup \{ a1, a2 : AutonomousAgent \mid \\
\qquad (\exists\, c : cooperations \bullet a1 = c.generatingagent \wedge \\
\qquad\quad a2 \in c.cooperatingagents) \bullet \{(a1, a2), (a2, a1)\}\} \\
cooperates^{\sim} = cooperates
\end{array}
$$

4 Agent Models

The framework described above provides useful structure that can be exploited by intelligent agents for more effective operation. However, this is only possible if each agent maintains a model of their view of the world. Specifically, each agent must maintain information about the different entities in the environment, so that both existing and potential relationships between those entities may be understood and consequently manipulated as appropriate.

For example, objects are not involved in any relationship with agents. If they were serving some useful purpose at a particular point in time, then they would be viewed as agents. The view of an entity as an agent thus indicates that it is *engaged* by another agent, either directly or back through a chain of intermediate agents, grounded with an autonomous agent at the head of the chain. Knowledge of the agency of an entity allows us to reason about its function and the agent or agents that are engaging it. Thus, I understand that my colleague's pencil is her agent for writing, and that if I want to use it I must negotiate with her as the engaging agent. Alternatively, if one robot is assisting another to move a crate of books, my models of the two robots may provide information as to the direction of the relationship. That is to say that if I have a model of one as an agent and another as an autonomous agent then, subject to circumstances, it may be sensible for me to infer that the non-autonomous agent is engaged by the autonomous agent. In this case, I must negotiate with the autonomous robot if I want either of the two agents to help me in my efforts to achieve my own goals.

In this example, this agency information provides useful structure which allows us to consider issues such as whether it is possible to negotiate with another, if such negotiation would be likely to achieve the desired result, how the negotiation might be approached, and so on.

Returning to the example of the pencil above, we can see how the richness of the situation can be captured through agent models. If my colleague is using her pencil, then I can view it as an agent satisfying the goal of writing some notes, for example, for its engaging agent, my colleague. Now, if I want to use her pencil, I can reason about the situation by considering the goal that the pencil is satisfying. First, I know that the goal was generated by my colleague, and I must therefore negotiate with her to secure use of the pencil. However, my knowledge of her motivations that generated the goal may lead me to decide that I will not succeed in securing use of the pencil for any number of reasons. If I know that the goal was generated because of an imminent important deadline, then I may decide that I will not be successful. Alternatively, if I know that the motivation for using the pencil was weak, then I may rate my chances of success highly.

It is important to note that the word, "pencil", of this example might just as easily be replaced by the word, "robot". The relationships described here are exactly the same.

4.1 Motivation

The key to the previously defined relations is motivation. Motivation is the 'force' that causes engagement chains to built up, satisfying goals that mitigate the motivation. In attempting to understand the nature of the relationships between entities in the world, and using or augmenting them, it is therefore necessary to be able to assess the relative strengths of motivation that caused those engagements. Similarly for cooperation, though here motivation plays an even greater role, since cooperation is symmetric and requires an assessment of motivation in all cooperating entities. In this paper, we are concerned not with investigating motivation, but in using it to describe engagement and cooperation. Related work has explored the nature of motivation and similar concerns in more detail [11, 9, 8], and we will only provide a simplified model here.

In order to retrieve goals to mitigate motivations, an autonomous agent must have some way of assessing the effects of competing or alternative goals. Clearly, the goals which make the greatest positive contribution to the motivations of the agent should be selected. The *AssessGoals* schema below describes how an autonomous agent monitors its motivations for goal generation. First, the *AutonomousAgent* schema is included, and a new variable representing the repository of available known goals, *goalbase* is declared. Then, the motivational effect on an autonomous agent of satisfying a set of new goals is given. The *motiveffect* function returns a numeric value representing the motivational effect of satisfying a set of goals with a particular configuration of motivations and a set of existing goals. The predicate part specifies all goals currently being pursued must be known goals that already exist in the goalbase. Finally, for ease of expression, we define a function related to *motiveffect* called *satisfy* which returns the motivational effect of an agent satisfying some goals.

AssessGoals

$$
\begin{array}{l}
AutonomousAgent \\
goalbase : \mathbb{P}\ Goal \\
motiveffect : \mathbb{P}\ Motivation \longrightarrow \mathbb{P}\ Goal \longrightarrow \mathbb{P}\ Goal \longrightarrow \mathbb{Z} \\
satisfy : \mathbb{P}\ Goal \longrightarrow \mathbb{Z} \\
\hline
goals \subseteq goalbase \\
\forall\, gs : \mathbb{P}\ goalbase \bullet satisfy\, gs = motiveffect\ motivations\ goals\ gs
\end{array}
$$

Therefore, $satisfy_x(gs)$ is the motivational effect on the autonomous agent x of satisfying the goals, gs. We further define the *complement* of a goal, which we write $\overline{goal}$, to be the goal to prevent that *goal* being achieved.

Consider the following situation in which, according to the model of some autonomous agent y, the autonomous agent x is directly engaging the server agent z for some goal g_x so that:

$(x, z, g_x) \in dengagement$

and further suppose that agent y wants to use z for some other goal g_y. There are several possible courses of action for y.

- y can persuade x to share z
- y can persuade x to release z.

- y can attempt to take z by force without x's permission.
- y can give x priority and find an alternative.

Any decision as to which alternative to take requires an analysis of both y's motivations and x's motivations. The analysis of their motivations below is solely from y's point of view.

- $satisfy_x\{g_y, g_x\} > satisfy_x\{g_x\}$. If g_y and g_x do not conflict, it is possible for z to adopt both of the goals of y and x without violating any motivational constraints. So long as the motivational effect on x of satisfying both goals is more than satisfying just her own, x will be disposed to share z.
- $satisfy_x\{g_y\} > satisfy_x\{g_x\}$. y understands that x stands to gain more from enabling y to satisfy y's goal than from satisfying its own goal. This is due to the effect that a positive change in y's motivations will have on x. This may require that y explains and persuades x of the degree of effect that g_y will have. For example, a friend may currently be reading a book that I want to borrow. My goal of borrowing the book may conflict with my friends goal, but because she wants to please me and does not need to read the book now, she happily lends the book to me.
- $satisfy_y\{g_y\} > satisfy_y\{\overline{g_x}\}$. It may seem obvious that the motivational effect on y of satisfying its goal should be greater than the motivational effect on y of satisfying x's goal. However, if x's goal is not satisfied, then the motivational effect on x will be negative, and this results in a state which must be considered in terms of its effect on y. (In other words, a negative motivational effect on y, particularly if it was a consequence of x, may result in a negative motivational effect on x.) Thus this alternative may be chosen if there is a positive motivational effect from y's goal being satisfied, *and* this is greater than the negative consequences of x's goal not being satisfied. Note that this relies on the relationship of y to x. Normally, y's motivations will be such that negative motivational effect on other agents will lead to some negative motivational effect on y itself. If, however, y is motivated by malicious concerns, then it is certainly possible that the consequences of x's goal not being satisfied may have a positive motivational effect on y. While we do not envisage such a situation arising regularly, and though this is a case typically not considered in related work, it ought to be possible within any formalism. By using motivations in the way we describe, we allow the possibility of perverse configurations leading to such malicious behaviour, but envisage appropriate design of motivations so that this does not arise. In summary, this captures normal social behaviour by which we act so as to avoid annoying others, but allows for situations where we may deliberately choose to annoy them. For example, if my friend is reading a book that I want to borrow, then I can simply take the book from her without permission. It only makes sense for me to do this, however, if the benefit I get from having the book is more significant than the bad feeling caused in my friend by my having taken it forcibly.

– $satisfy_y\{g_y\} < satisfy_y\{\overline{g_x}\}$. y understands that the motivational effect of satisfying its goal will be less than the effect of causing a negative motivational effect on x through g_x nor being satisfied. This affects y's behaviour, because its motivations are configured in such a way that y is concerned for x. Like the previous case, this describes the situation where we do not act if that action is likely to annoy others.

4.2 Applying Agent Models

Consider the situation in a library in which a user wants to locate a particular book. The autonomous agents in the library world include the user and the librarian. Now, suppose that in order to locate the book, the user requires the assistance of the librarian. Since the librarian is autonomous, this means that the user must persuade the librarian to *cooperate* with her in achieving her desired goal. In attempting to locate the book for the user, the librarian uses a terminal to invoke a computer program which, in turn, accesses the library databases and performs the relevant query. These relationships can be captured very easily in terms of the predicates defined earlier.

First, the librarian, L, cooperates with the user, U.

$(L, U) \in cooperates$

Then, the librarian uses the terminal, T, the program, P, and the database, D, to locate the book.

$(L, T) \in downs \wedge$
$(T, P) \in dengages \wedge$
$(P, D) \in dengages \wedge$
$\Rightarrow (L, D) \in engages$

In this scenario, the librarian owns (and also directly engages) the terminal, T, and this engagement makes T a server-agent. In turn, T directly engages P, making it a server-agent, and similarly, P engages D. This engagement chain is constructed through the engagement of agents by other agents in order to satisfy the goal of locating the book, and is shown in Figure 2.

Let us now suppose that the terminal can only be used by one person at a time, but the program and database can be used by many people at once. With this information, other agents in the library can effectively decide what courses of action to follow to achieve their aims. Another user, for example, $U2$, may also want to locate a book. $U2$ must decide what action to take, or agents to invoke, on the basis of the motivations of the autonomous agents involved in the engagement. Since the terminal cannot be used by multiple agents and is *owned* by the librarian, $U2$ cannot share it, but can choose to take it forcibly, to persuade L to release it, or to give L priority and either wait or find an alternative terminal. Note that this last possibility is not constrained by the program and database being engaged by L, since they are not owned by L. If $U2$ attempts to take the terminal forcibly, the consequences may be severe in terms of detrimental motivational effect on L. Persuading L to release the terminal requires another cooperation, between L and $U2$, which may not be

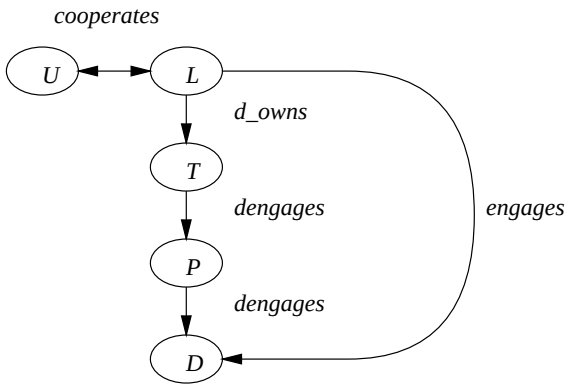

Fig. 2. An Engagement Chain for the Library Scenario

forthcoming if the motivational effect of $U2$'s goal on L is less than L's current goal. One of the alternative options is for $U2$ to ask another librarian, $L2$, to locate the book, while another solution would be for $L2$ to take part in separate cooperations with $U2$ and L so that the terminal is released for use in finding the second book. This is because the negative motivational effect on L of not cooperating with $L2$ will be significantly greater than the negative motivational effect of not cooperating with the user $U2$.

Figure 3 shows another situation that arises when $U2$ asks another librarian, $L2$, to locate a book, and $L2$ uses a different terminal, $T2$. Note that in order to avoid complicating the diagram, not all of the relations between the entities are shown, though they can be inferred. For example, L engages T and P and D, while $L2$ engages $T2$ and P and D.

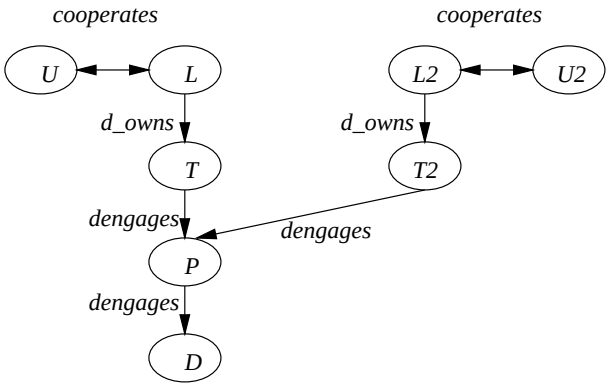

Fig. 3. Additional Engagement and Cooperation in the Library Scenario

5 Related Work

This research shares some common themes with the concepts of dependence networks [10]. Dependence networks use *external descriptions* which store information about other agents, comprising a set of goals, actions, resources and plans. The goals are those an agent wants to achieve, the actions are those an agent is able to perform, the resources are those over which an agent has control, and the plans are those available to the agent, but using actions and resources which are not necessarily owned by the agent. This means that one agent may *depend* on another in terms of actions or resources in order to execute a plan.

Using external descriptions, the authors distinguish three different forms of autonomy, as follows. An agent is *a-autonomous* if for a given goal, according to a set of plans, there is some plan in the set which consists of only actions of that agent. An agent is *r-autonomous* if, similarly, any resources needed belong to the agent. An agent is *s-autonomous* if it is autonomous in both of the above ways (and does not, therefore, need any help). If agents are not autonomous, then they may *depend*, for resources or actions, on other agents. By contrast, our work describes autonomy not as action or resource dependence, but as the ability to make one's own choices, to generate goals. This is a far stronger notion of autonomy. The fact that a pocket calculator has the resources and the actions necessary for adding some numbers surely does not make it autonomous. A more complete analysis of this work, and a reformulation of it in our framework can be found in [3].

Moreover, even though this kind of categorisation is useful, it is not clear how much help it would be to a motivated agent. An agent may believe that another agent will perform a required action for reasons of continuing friendship, trust, promise, and so on. The motivational context of these dependencies is what is important, and the target agent may not care to investigate whether there are any specific actions that it has promised to perform for an agent.

While our work has concentrated on defining the relationships that are involved in cooperation, for example, other related work has addressed the *processes* of cooperative problem solving, also in a formal way [15]. One possible way forward might be to investigate the possibility of integrating these two approaches.

6 Conclusions and Further Work

The work reported in this paper is part of a broader project with the aim of developing effective mechanisms for autonomous communication. To date, we have developed a formal framework structuring the world into object, agent and autonomous agent components. Based on those definitions, we are developing mechanisms for goal generation and transfer or adoption so that agents may interact with each other in useful ways. In particular, we describe and specify the differences that arise in these interactions which we characterise as *engagements* of non-autonomous agents, and *cooperation* between autonomous agents. The

next objective is to incorporate explicit mechanisms for deliberation, which will include both planning and communication. Our immediate aim is to provide a communication facility through the use of *knowledge interchange protocols* [1]. Finally, we intend to add a learning component which will dynamically acquire knowledge of the environment and of protocols so that more efficient behaviour can be obtained.

References

1. J. A. Campbell and M. P. d'Inverno. Knowledge interchange protocols. In Y. Demazeau and J. P. Muller, editors, *Decentralized Artificial Intelligence*. Elsevier North Holland, 1989.
2. P. R. Cohen and C. R. Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3):177–212, 1979.
3. M. d'Inverno and M. Luck. A formal view of social dependence networks. In *Proceedings of the First Australian DAI Workshop*. Springer Verlag, To appear, 1996.
4. R. Goodwin. Formalizing properties of agents. Technical Report CMU-CS-93-159, Carnegie-Mellon University, 1993.
5. Z. Kunda. The case for motivated reasoning. *Psychological Bulletin*, 108(3):480–498, 1990.
6. M. Luck and M. d'Inverno. A formal framework for agency and autonomy. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 254–260. AAAI Press / MIT Press, 1995.
7. M. Luck and M. d'Inverno. Goal generation and adoption in hierarchical agent models. In *AI95: Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence*. World Scientific, 1995.
8. D. Moffat and N. H. Frijda. An agent architecture: Will. In *Proceedings of the 1994 Workshop on Agent Theories, Architectures, and Languages*, 1994.
9. T. J. Norman and D. Long. A proposal for goal creation in motivated agents. In *Proceedings of the 1994 Workshop on Agent Theories, Architectures, and Languages*, 1994.
10. J. S. Sichman, Y. Demazeau, R. Conte, and C. Castelfranchi. A social reasoning mechanism based on dependence networks. In *ECAI 94. 11th European Conference on Artificial Intelligence*, pages 188–192. John Wiley and Sons, 1994.
11. A. Sloman. Motives, mechanisms, and emotions. *Cognition and Emotion*, 1(3):217–233, 1987.
12. R. G. Smith and R. Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man and Cybernetics*, 11(1):61–70, 1981.
13. J. M. Spivey. *The Z Notation*. Prentice Hall, Hemel Hempstead, 2nd edition, 1992.
14. M. J. Wooldridge and N. R. Jennings. Agent theories, architectures, and languages: A survey. In *Proceedings of the 1994 Workshop on Agent Theories, Architectures, and Languages*, 1994.
15. M. J. Wooldridge and N. R. Jennings. Formalizing the cooperative problem solving process. In *Proceedings of the Thirteenth International Workshop on Distributed Artificial Intelligence*, 1994.

This article was processed using the LaTeX macro package with LLNCS style