Annotating Cooperative Plans with Trusted Agents

Nathan Griffiths
Dept of Computer Science
University of Warwick
Coventry, CV4 7AL, UK
nathan@dcs.warwick.ac.uk

Michael Luck
Dept of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, UK
mml@ecs.soton.ac.uk

Mark d'Inverno
Cavendish School of Computer Science
Westminster University
London, W1M 8JS, UK
dinverm@wmin.ac.uk

ABSTRACT

Cooperation is the single most fundamental characteristic of multi-agent systems, and much work has been done on studying the various aspects involved, from general models of the overall structure of cooperation to detailed analyses of specific components. In our work, we aim to do both — we provide a general model and instantiate each stage in that model. We take the notions of *trust* and *motivation* to be fundamental to engendering successful cooperation between autonomous entities, and our model of cooperation accounts for the important roles played by these concepts. This paper focuses in particular on the details of how, based on trust, an agent chooses and keeps track of which agents it may use to assist in the performance of actions that make up a multi-agent plan, and how that information can be used in actually soliciting the assistance.

1. INTRODUCTION

1.1 Cooperation

Cooperation underpins multi-agent systems in which individual agents must interact for the overall system to function effectively. There has been much previous work in examination of different aspects of cooperation and cooperative activity, but only relatively little concerned with a detailed examination of the different stages involved in a full appreciation of it. In this paper we introduce an overall model of cooperation that is heavily dependent on the notion of *trust* as supplying the glue to support effective interaction. Trust is an issue that is little considered in this respect, but is involved in, and necessary for, the establishment of inter-agent cooperative relationships in dynamic and open environments, and has an important role to play. It is used to identify potential trusted partners for cooperation and to annotate plans with those partners so that the plans may be executed with reference to those agents already included in the plan.

Our model can be seen as instantiating previous work in a more general effort by Wooldridge and Jennings [15], and offering a detailed analysis of how cooperative plans can be constructed. It is not intended to offer a sophisticated representation for trust itself and how it is modified and updated, rather to show how it can be used to support cooperative activity. In this section we describe the basic outline of our model and introduce the fundamental aspects of it, before going into detail of cooperative plan annotation based on trust in the rest of the paper.

Adopting the BDI approach in which an individual agent comprises *beliefs* (about itself, others and the environment), *desires* (in terms of the states it wants to achieve) and *intentions* (as adopted plans for these desires) [2], a detailed model of agent behaviour can be constructed. However, in line with the views of some that motivation is an extra component need in such models, we include explicit *motivations*, which embody the high-level desires that guide behaviour, and at a fundamental level control an agent's reasoning cycle [10]. In this view, an agent responds to changes in its beliefs, resulting from perception, by generating a set of goals[1] according to its motivations and beliefs. It then determines which goals to commit to according to the motivational value they afford, and selects an appropriate plan to use (adopting it as an intention). Finally, an agent selects a particular intention to pursue and acts towards its achievement, again using motivational value as the guiding measure.

There are several distinct tasks surrounding the formation and execution of a cooperative intention, which we introduce in this section. If an agent is faced with a plan containing actions that require assistance, or actions that are beyond the extent of its capabilities, it must seek assistance from others and form an appropriate cooperative intention before that plan can be fully executed. This, in turn, gives rise to a second situation in which cooperation arises, which is in response to another's request for assistance. In both cases cooperation arises from a particular agent wishing to adopt a plan that contains actions it is unable to perform alone — in the first case the agent itself has the plan, while in second case it is another agent's plan that leads to the request for assistance. The notion of cooperation arising in response to a particular plan complements the approach taken in many BDI-based systems of constructing plans from a *library* of partial plans, rather than from first principles. Since we are adopting the BDI approach, and are not assuming that agents have the ability to plan from first principles, we take this view of cooperation arising in response to a particular plan, rather than focusing upon forming a cooperative group and then addressing the task of constructing a plan from scratch. Although we take this view, our approach does not preclude planning from first principles; the impact of trust and motivational value are

[1]We use the term goal rather than desire in order to make clear the distinction between the desire to bring about a specific situation (a *goal*) and a more general desire (a *motivation*). For example, the motivation of thirst may lead to the goal of drinking a cup of coffee.

similar in both approaches (a discussion of how these issues can be found in [5]). Note also that provision is made for an agent to seek assistance if it does not have a suitable plan for its goal, and our model supports the construction of suitable plans in a cooperative manner. However, we are not concerned with group planning *per se*, and this paper is orthogonal to work such as Grosz and Kraus' model of SharedPlans [7], and although the notions of trust and motivation are applicable to group planning, this is is beyond the scope of our work. The entire process of setting up cooperation among agents can be broken down into the stages of plan selection, intention adoption, and group action.

1.2 Plan Selection

An agent's motivations give rise to certain goals that must be adopted as intentions, by selecting an appropriate plan and forming a commitment to its execution. Now, the set of applicable plans for a particular goal may include plans containing actions that are beyond the agent's capabilities, or joint or concurrent actions. We refer to such plans as *cooperative plans* since they can only be executed through cooperation with others. If an agent selects a cooperative plan, it is electing to cooperate for the achievement of its goal. In order to select between plans, where the execution of those plans may require cooperation, an agent must consider the nature of the agents it may cooperate with; it should consider both the likelihood of finding agents to assist in achieving its goal and the likelihood that they will execute the plan successfully, i.e. their trustworthiness. However, although the characteristics of others are considered at selection time, the agent cannot decide which agents to seek assistance from since there may be a delay between selecting a plan and actually acting towards it, thus any annotation may be premature. Additionally, there is a computational cost to annotating a plan, and since there are typically several plans to choose between it is undesirable to annotate each to select between them. Rather, a coarser approach to plan selection is taken which minimises the computational cost. If there were a very small number of plans to choose from, and the delay between selection and execution could be guaranteed to be small then it may be computationally cheaper for an agent to annotate its plans at selection time. However, in general we cannot make such guarantees and so such pre-selection plan annotation is not appropriate. Note also that the cost of annotation is proportional to the number of plans, and since pre-selection plan annotation is only useful when there is a small number of plans, the benefits are also (relatively) small. Ongoing work aims to investigate whether an agent's knowledge about the evolution of its environment can be used to determine when to perform pre-selection plan annotation in a computationally efficient manner.

Since an agent's choice of plan determines whether it must cooperate to achieve its goal, and cooperation involves a certain degree of risk, then to choose the plan involving cooperation requires there to be some inherent advantage to that cooperation. The problem of plan selection amounts to choosing the best plan — the plan that is most likely to be successful, with least cost in terms of time and resources, and the least *risk*. When the plans involved do not involve other agents, standard plan selection criteria (or planning heuristics) can be used to assess cost. However, when plans involve others, an element of *risk* is introduced by the inherent uncertainty of interaction. In addition to a measure of the cost of a plan, therefore, we need to assess the likelihood of finding agents for actions required for successful plan execution; the likelihood that such agents will agree to cooperate; and the likelihood that the agents concerned will fulfill their commitments.

The notion of *trust* is recognised by several researchers as a means of assessing the perceived risk in interactions [3, 11]. The risk of whether to cooperate and with whom, may be determined by, among other things, the degree of confidence or *trust* in other agents. Despite the notion of trust being commonplace in our everyday interactions, there are few formal definitions. However, it is generally accepted that trust implies a form of risk, and that entering into a trusting relationship is choosing to take an uncertain path that can lead to either benefit or cost depending on the behaviour of others. The perceived risk of cooperating with a particular agent is determined by that agent's reliability, honesty, veracity, etc., embodied by the notion of *trust*. As an agent interacts with others it can ascribe trust values based on their previous behaviour, and over time improve its model of trustworthiness. Thus, in a sense, trust provides a mechanism for an individual agent to maintain its own view of the reputation of another. These values can be used as a means of assessing the risk involved in cooperating with others. Since, in general, agents do not necessarily have sufficient reasoning capabilities to assess the various facets of others that determine their trustworthiness, such as their honesty and veracity, a coarser mechanistic approach is taken.

Trust values are initially ascribed to others (and form part of its models of others) according to an agent's *disposition*: optimistic agents are likely to ascribe a high value, while pessimists are likely to give a low value. This disposition also determines how trust values are updated as a result of interactions with others [12]. After a successful interaction optimists increase their trust more than pessimists, and conversely, after an unsuccessful interaction pessimists decrease their trust more than optimists. The magnitude of change in trust is a function of a variety of factors depending on the agent concerned, including the current trust and the extent of the agent's optimistic or pessimistic disposition. However a simplistic approach, described in more detail in [5], is for an agent's disposition to be represented by two values, $trustIncrease$ and $trustDecrease$, which determine the proportion of current trust level to increase or decrease by respectively according to whether an interaction was successful or not.

We have described in [6] a mechanism for assessing the contributions contained in a plan in terms of the risk associated with the agents who are believed capable of executing them. This assessment is combined with more traditional standard planning heuristics (such as cost and plan length) to obtain a measure for selecting between plans that balances these, often contradictory, desires to minimise both cost and risk. Using this approach an agent's choice about whether to cooperate or not is embodied by its choice of plan.

1.3 Intention Adoption

After selecting a plan for its goal an agent must commit to its execution by forming an intention. If the plan does not require assistance from others then it can simply be adopted and action towards it can begin, otherwise the agent must solicit assistance from selected agents towards its execution. We refer to the agent that selects a cooperative plan, and attempts to gain assistance for its execution, as the *initiating* agent, or the *initiator*. In order to gain assistance, the initiator must first determine which agents to request assistance from, achieved by iterating through the steps of the plan, annotating each contribution with the identifier of the agent that the initiator considers the best to perform it, based on knowledge of their capabilities, and their believed reliability, etc. as determined by the trust value ascribed to them. The assistance of these agents can then be requested. On receiving a request for assistance, these agents in-

spect their own motivations and intentions to decide whether or not to agree, and send an appropriate response to the requesting agent; an agent's motivations determine whether it *wants* to cooperate, and its existing intentions determine whether it *can* cooperate (since intentions must be consistent). If sufficient agents agree then a commitment in the form of a cooperative intention can be established among them. However, if insufficient agents agree then either the plan can be reannotated, or failure is conceded.

1.4 Group Action

Once a group of agents have formed a cooperative intention they can execute it — each step of the plan in turn is either performed or elaborated according to whether it is an action or a subgoal, respectively. On the successful completion of the cooperative intention, the agents concerned dissolve their commitment and cooperation is finished. Alternatively, if execution of the intention fails, the agent that first comes to believe this informs the others in accordance with the conventions introduced in the previous section, and again their commitments are dissolved. In both cases agents can update the information they store about others to aid future decisions about cooperation, in particular the trust values ascribed to these agents are updated. For example, if cooperation fails due to the behaviour of a particular agent, others may be more wary of cooperating with that agent in future.

1.5 Cooperative Plan Annotation

These stages strongly relate to those contained in Wooldridge and Jennings' formalisation of cooperative problem solving, namely: recognition of the potential for cooperation, team formation, plan formation, and team action [15]. Their model is relatively abstract and, as they recognise, is intended to provide a top-level specification for a system, requiring more detail before it can be implemented. We view the work described in this paper as providing an instantiation for some of the details that were previously left abstract. Wooldridge and Jennings also recognise that although the stages in their model are presented as being sequential, in practice they may not occur strictly in the order they describe. Indeed, this is a significant difference between our model and theirs; in our approach an individual agent selects a plan that requires cooperation, and then seeks assistance, while in their approach an agent recognises the potential for cooperation, seeks assistance, and *then* the agents as a group form a plan.

This difference arises from our alternative view of the *potential for cooperation*, which in turn is a result of the nature of our agent architecture. They view the potential for cooperation as being where an agent has a goal that it is unable to achieve in isolation, or does not want to use the resources required to achieve it alone. Alternatively, in our framework the recognition of the potential for cooperation is implicit in an agent's choice of how to achieve its goal — an agent simply selects a plan to achieve its goal, which may or may not require cooperation to execute. Therefore, in our model an agent seeks assistance *after* a plan has been selected rather than before, since unless an agent knows how to achieve the goal it cannot consider what cooperation may result from that goal[2]. This is important since we are specifically concerned with *why* an agent might enter into cooperation.

[2]It is, however, possible for an agent to seek assistance if it has no explicit plan for its goal by using the plan containing just that goal as a plan step. However, due to space constraints we do not consider the details here (a discussion can be found in [5]).

In dynamic environments there is often a delay between between obtaining commitments from others and using them in plan execution. In general, the longer this delay, the more time there is for agents' motivations to change, thereby increasing the risk of failure. To address this, an initiating agent can choose between annotating a plan and soliciting assistance as soon as the plan is selected (an immediate commitment strategy), or waiting until execution time (a delayed commitment strategy). This is a choice about *when* to annotate its plan and obtain commitments, rather than *whether* to do so. Several factors are relevant in choosing between strategies, including the trust of others, the degree of environmental dynamism, and the nature of the domain itself. The degree of dynamism determines how likely others' motivations are to change, since the intensity of an agent's motivations are determined in response to its perceptions of the environment. The trustworthiness of others can can be used as an indication of the likelihood that their commitments will be fulfilled. Higher trust suggests a greater perceived likelihood of fulfilling commitments. If other agents are generally distrusted, therefore, obtaining commitments at plan selection time may be too expensive since they are more likely to renege on them. However, due to space constraints we do not describe how to make this choice here. Instead we simply note that this offers agents considerable flexibility in establishing cooperative activity.

Having previously considered the plan selection stage in some detail [6], this paper extends that, and is specifically concerned with intention adoption, focusing in particular on plan annotation. The next section introduces the notion of cooperative plans, which require a number of agents to execute. After plan selection an agent must seek assistance from others, beginning by annotating that plan with appropriate agents as described in Section 3. In Section 4 we introduce a number of approaches for requesting another's assistance with respect to an annotated plan. Finally, Section 5 concludes this paper.

2. COOPERATIVE PLANS

For an agent situated in a multi-agent environment to take advantage of others, its plans must include a means for it to interact with them. Cooperation may take the form of performing an action on behalf of another, a group of agents performing a (joint) action together, or a set of (concurrent) actions performed at the same time.

Our definitions of these actions build upon the notions of strong and weak parallelism described by Kinny *et al.* by decomposing joint actions into the specific component actions, or *contributions*, that comprise them [8]. This allows us to build a relatively simple, yet expressive, formalisation in which to represent cooperative plans. Although not as expressive as possible alternative approaches, such as directed graphs, the resulting plans are simpler to manipulate, and are sufficient for most situations. Moreover, the general principles of plan annotation expressed in this paper, could be equally well applied to an alternative representation of plans.

First, *individual actions* are those performed by an individual agent without the need for assistance, and may be executed by the agent owning the plan in which it is contained, or by another agent on its behalf. Now, in a cooperative domain, an agent needs to track who performs each action in a plan, so we represent each action as a *contribution*, which is a tuple comprising the action and a globally unique identifier corresponding to the agent that performs it. Using the Z notation, which is based on set-theory and first order logic [13], we write this formally as follows. (A full treatment of Z, together with explanations of its suitability for specification of

agent systems and its usefulness in moving from specification to implementation, is available elsewhere [4]; for reasons of brevity, however, we will not elaborate the use of Z further.)

$$
\begin{array}{|l}
\hline
Agent \\
\hline
agtId : AgentID \\
beliefs : \mathbb{P}\, Belief \\
goals : \mathbb{P}\, Goal \\
intentions : \mathbb{P}\, Intention \\
motivations : \mathbb{P}\, Motivation \\
capabilities : \mathbb{P}\, Act \\
\vdots \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
Contrb \\
\hline
act : Act \\
agtId : AgentID \\
\hline
act \in (agent\ agtId).capabilities \\
\hline
\end{array}
$$

Joint actions are composite actions, made up of individual actions that must be performed together by a group of agents. Each agent involved in executing a joint action makes a simultaneous *contribution* to the joint action, corresponding to the component action that it performs[3]. Note that the agents within any joint contribution must be distinct.

$$
\begin{array}{|l}
\hline
JointAct \\
\hline
contrbs : \mathbb{P}\, Contrb \\
\hline
\#contrbs \geq 2 \\
\forall\, c_1, c_2 : contrbs \mid c_1 \neq c_2 \bullet \\
\quad c_1.agtId \neq c_2.agtId \\
\hline
\end{array}
$$

Finally, *concurrent actions* are those that can be performed in parallel by different agents, without the need for synchronisation (except at the beginning and end of a set of concurrent actions). Concurrent actions can comprise both individual contributions and joint actions that are to be performed simultaneously, denoted by $singles$ and $joints$ in the schema $ConcAct$. As with joint actions, the action an agent performs as part of a set of concurrent actions is its *contribution*[4]. Unlike joint actions there is no requirement for the agents involved in a concurrent action to be distinct, although in practise they typically are (otherwise the components of the concurrent action must be executed sequentially).

$$
\begin{array}{|l}
\hline
ConcAct \\
\hline
singles : \mathbb{P}\, Contrb \\
joints : \mathbb{P}\, JointAct \\
allcontrbs : \mathbb{P}\, Contrb \\
\hline
\#singles + \#joints \geq 1 \\
allcontrbs = singles \cup \bigcup \{j : joints \bullet j.contrbs\} \\
\hline
\end{array}
$$

In common with the base BDI model, we take plans to be *partial* in that they are incomplete, and contain subgoals in addition to actions [2]. Additionally, since plans apply only to particular situations, they must also have a set of preconditions that define when they are applicable. Thus, we define a plan as sequence of steps, where a step is either an individual action, a joint action, a set of concurrent actions, or a subgoal.

$$
\begin{array}{rcl}
PlanStep & ::= & Individual\langle\langle Contrb\rangle\rangle \\
& \mid & Joint\langle\langle JointAct\rangle\rangle \\
& \mid & Conc\langle\langle ConcAct\rangle\rangle \\
& \mid & Subgoal\langle\langle Goal\rangle\rangle
\end{array}
$$

$$
\begin{array}{|l}
\hline
Plan \\
\hline
achieves : Goal \\
precon : \mathbb{P}\, Belief \\
body : \mathrm{seq}\, PlanStep \\
\hline
\end{array}
$$

3. PLAN ANNOTATION USING TRUST

Once an agent has selected a plan for its goal that plan must either be adopted as an intention if its execution does not require assistance, or the agent must initiate the process of forming a cooperative intention if others are required for the plan's execution. As described above, the first step in forming a cooperative intention is to determine which agents would best perform the contributions in the plan. Agents selected in this way are associated with a contribution by *annotating* it with the identifiers of the agents, and each cooperative action in the plan must be annotated in this way. Note that several agents may be able to perform the required contribution, and more than one may be listed in the annotation. In this case, there is a degree of redundancy to safeguard against the situation where some agents decline to cooperate, which we call *redundant annotation*. Conversely, we refer to the annotation of each contribution with just one agent as *minimal annotation*.

In pursuit of the desire to minimise the risk associated with electing to use a cooperative plan an agent uses its knowledge of others in selecting agents to cooperate with. In particular an agent can use knowledge based on its previous experience of others, in the form of the trust ascribed to them, in evaluating potentially cooperative partners. In general, each action is annotated with the n most trusted agents, where $n = 1$ in minimal annotation and $n > 1$ in redundant annotation. Note that if $n > 1$ and the number of agents having the required capabilities is less than n (but more than 1) the agent must simply annotate the plan with all those agents, rather than trying to find others with the required capabilities in order to annotate the plan with n agents. If no agents are known to have the required capabilities then plan annotation fails.

With redundant annotation, even if some of the chosen agents decline to cooperate, cooperation may still be successful. For example, suppose that for each action three agents are asked for assistance. If all three agents accept then the initiator can simply enter into cooperation with the most trusted agent (since it is perceived to involve the least risk). However, if two agents decline, then cooperation can still go ahead with the third agent. In general the initiator will enter into cooperation with the *most trusted combination* of agents from the redundant annotation that agree to cooperate. Unfortunately, this redundancy comes at a price, firstly, because the cost of communication and processing the responses will be increased over minimal annotation where a single agent might be asked for each action, in the ideal case of that agent accepting. Secondly, constantly requesting assistance but then not entering into cooperation with the agents that accede (for example because a more trusted agent agrees) might lead to others reducing their trust

[3]For example, if agents α_1 and α_2 perform the joint action of lifting a table, then α_1 must make the contribution of lifting one end of the table simultaneously with α_2 lifting the other.

[4]For example, if agents α_1 and α_2 each write a chapter for a book, and they perform their actions in parallel, then α_1 and α_2 perform concurrent actions where each agent's contribution is the action of writing the appropriate chapter.

of the initiator. Furthermore, using minimal annotation when some actions need to be reassigned, may still have reduced communication cost, since there may be fewer agents in total to send requests to. Note, however, that at a lower level redundant annotation offers more scope for optimisation, for example through the use of targeted broadcast messages (which may be cheaper than communicating with several agents individually). Thus, it is not necessarily true to say that redundant annotation, where n agents are asked for each action, is equivalent in communication cost to minimal annotation where the nth agent agrees, since it may be cheaper to send a single broadcast than to send n individual messages.

3.1 Choice of Annotation Strategy

At this point, it is useful to introduce the notion of a *closely coupled* and *loosely coupled* view of agent systems. Where we are concerned with the behaviour and performance of a multi-agent system as a whole rather than with a specific individual in that system, as in when designing a complete multi-agent system to perform a particular task, we say that we are taking a *closely coupled* view. Conversely, where we are concerned with maximising the performance of a particular agent, without concern for the effect on the system as a whole, as with an agent designed to compete against others, such as an auction agent, this is a *loosely coupled* view.

Now, in the closely coupled view, redundant annotation may have negative effects on the group's efficiency since there will obviously be some overhead involved in agents agreeing to cooperate. In particular, an agent may be unnecessarily constrained while committed to cooperating in this way (though perhaps not actually being needed), which may have prevented it from doing something else beneficial to itself or the group as a whole. Thus, although redundant annotation increases the likelihood of getting agreement to cooperate without reassigning actions, it may be counter-productive overall in this respect.

In the loosely coupled view, when concerned with maximising individual performance without consideration of others, redundant annotation may not be successful over a period of time. If an agent is asked for assistance and agrees to provide it, only to be turned down later, its trust of the requesting agent will tend to decrease, since the requester did not honour the request and may have cost the provider time and caused it to constrain its actions unnecessarily. While the effect may be negligible in the short term, over an extended period the decreased trust may cause the provider to decline to cooperate. Thus, if at a later point there is only one agent with the appropriate capabilities, that agent may refuse to cooperate because it does not trust the requester; it has been inconvenienced too many times.

Ultimately, the best strategy in terms of redundant or minimal annotation is determined by both the domain itself and the overall perspective (of maximising system or individual performance). Overarching these issues, however, is the importance to the initiator of its goal, since if a goal is important, redundant annotation may be justified despite any concern for the performance of the overall system. It is, therefore, desirable for an agent to be able to choose between these strategies dynamically, according to the current situation, and we consider both possibilities in the remainder of this chapter. In order to deal with this, we introduce the notion of a *redundancy threshold* to determine whether to use redundant annotation. If the motivational value of a goal is greater than this threshold then redundant annotation is used. However, since the redundant approach should only be used sparingly this threshold must be sufficiently high.

3.2 Annotating with Trusted Agents

Although this considers whether agents are trusted, it does not consider whether they are *distrusted* i.e. are trusted below some minimum trust threshold. If the only agents that are believed to have the required capabilities are distrusted, then it may be better for the assignment of agents to actions to fail, rather than enter into cooperation with a group of distrusted agents, since they are considered likely to renege on their commitments. Agents that are distrusted are not annotated to a plan; thus if all the agents capable of performing a particular action are distrusted then plan annotation fails. An agent is trusted if and only if the trust ascribed to it is above a minimal threshold. The minimal trust threshold is part of an agent's disposition, but is also affected in an inversely proportional manner by the importance of the current goal. Thus, if an agent's goal is sufficiently important to it, we can model the situation where it is better to have tried to achieve it, and failed, than to have not tried at all. The trust of an agent, along with its capabilities is embodied in a model of that agent, formalised as follows. Note that each agent has its own models, giving it an individual representation of others' capabilities and trustworthiness.

$$
\begin{array}{|l}
\hline
AgentModel \\
agtId : AgentID \\
trust : \mathbb{R} \\
capabilities : \mathbb{P}\, Act \\
\vdots \\
\hline
\end{array}
$$

For ease of specification we assume an injective function which maps each agent identifier onto the corresponding agent.

$$agent : AgentID \rightarrowtail Agent$$

This formalisation allows us to express complex trust relationships, and to express the web of trust that links agents together. Our approach is simplified, however, in that we do not consider *situational* trust where the trust associated with a particular agent varies according to the current situation [11]. For example, while an agent may trust another to extract product information from a database, it might not trust it to determine which product represents the best value for money. Conceptually situational trust is a more powerful mechanism than general trust, however the computational overhead involved in identifying and maintaining trust values for specific tasks can be prohibitive, and so we do not use it here.

3.3 Individual Action Annotation

Recall that a contribution is defined to be an action, along with the identifier of the agent that is to perform it. Where we are concerned with minimal annotation this is sufficient to represent the agent annotated to a contribution. However, when we consider redundant annotation, this is insufficient, since we need to associate a *set* of agent identifiers with a particular action. Therefore, before we can give the function for annotating a contribution we must introduce the notion of an annotated contribution, where an action is annotated with a set of agents. Clearly the action must be in the capabilities of each of the associated agents, according to the corresponding agent model.

AnntdContrb

$act : Act$
$agts : \mathbb{P}\, AgentID$

$act \in \bigcap \{a : agts \bullet (agent\ a).capabilities\}$

The annotation of a contribution is given below in the schema $AnntContrb$, in which max and t represent the number of agents with which to annotate a contribution and the minimum trust threshold, respectively. This function specifies that an individual contribution is annotated with the max most trusted agents, provided their associated trust values are greater than t.

$$
\begin{array}{l}
anntContrb : Contrb \rightarrow \mathbb{P}\, AgentModel \\
\quad \rightarrow \mathbb{Z} \rightarrow \mathbb{R} \rightarrow AnntdContrb \\
\hline
\forall\, c : Contrb;\ ms : \mathbb{P}\, AgentModel;\ max : \mathbb{Z}; \\
\quad t : \mathbb{R};\ anntc : AnntdContrb \bullet \\
\quad anntContrb\ c\ ms\ max\ t = anntc \Rightarrow \\
\qquad c.act = anntc.act \land \\
\qquad \#anntc.agts \leq max \land \\
\qquad (\forall\, agt : anntc.agts \bullet \exists\, m : ms \bullet \\
\qquad\quad m.agtId = agt \land \\
\qquad\quad c.act \in m.capabilities \land \\
\qquad\quad m.trust > t)
\end{array}
$$

The predicate part of this schema states that:

1. the action of the annotated contribution is the same as that of the contribution,
2. there are at most max number of agents in the annotated contribution,
3. for every agent in the annotated contribution there is an associated agent model in the original set of agent models, ms, from which we are choosing,
4. according to this model all agents have capabilities which contain the action of the original contributions, and
5. the trust value of the agent (in the corresponding model) is above the value t supplied as a function parameter.

3.4 Simultaneous Action Annotation

The approach described above is only applicable for plans that do not contain joint or concurrent actions. The main consideration in annotating a plan containing joint or concurrent actions is that an agent must not be required to execute two or more contributions simultaneously, since we assume that agents can only perform one action at a given time. In minimal annotation this is simply achieved by not annotating an agent to more than one simultaneous contribution. Annotated joint and concurrent actions can be constructed from annotated contributions, formalised below.

AnntdJointAct

$anntcontrbs : \mathbb{P}\, AnntdContrb$

$\#anntcontrbs \geq 2$

AnntdConcAct

$singles : \mathbb{P}\, AnntdContrb$
$joints : \mathbb{P}\, AnntdJointAct$
$allcontrbs : \mathbb{P}\, AnntdContrb$

$\#singles + \#joints \geq 1$
$allcontrbs = singles \cup$
$\quad \bigcup \{j : joints \bullet j.anntcontrbs\}$

A minimal annotation has only one agent associated with an action and, necessarily, all the agents must be distinct. Note that for a concurrent action the only constraint is that the component individual and joint actions are minimally annotated, since although the components of a concurrent action are typically executed simultaneously, this is not a formal requirement.

MinimalAnntdJointAct

$AnntdJointAct$

$\forall\, c : anntcontrbs \bullet \#c.agts = 1$
$\forall\, c_1, c_2 : anntcontrbs \mid c_1 \neq c_2 \bullet$
$\quad c_1.agts \neq c_2.agts$

MinimalAnntdConcAct

$AnntdConcAct$

$\forall\, c : singles \bullet \#c.agts = 1$

Redundant annotation, however, is more complex, because an agent might be annotated to several simultaneous contributions, and its assistance requested for all of them.

Since an agent can only perform one action at a time, and its intentions must be consistent, an agent asked to assist for several simultaneous contributions can only agree to one of them at most (according to its motivations and intentions), or its intentions would become inconsistent. Redundant annotation of an agent to several simultaneous contributions allows that agent to choose which contributions it performs. The key requirement when annotating the same agent to more than one simultaneous contribution is that agreement is necessary for at most one of them. For example, a joint action comprising two contributions each annotated with the same two agents is a valid annotation, because either agent can perform either contribution. Alternatively, a joint action comprising three contributions, each annotated with the same two agents, is not a valid annotation, since even if both agents agree to perform a contribution, there will be a third contribution for which no agent has agreed. (These are illustrated in Figure 1). Where we are concerned with annotating concurrent actions it is possible for an agent to be annotated to more than one thread of execution since synchronisation is only required at the beginning and end of a concurrent action block, and all contributions do not necessarily have to be performed simultaneously, although doing so may compromise efficiency. Formally, a valid annotation is one where it is possible to find a minimal interpretation by selecting appropriate agents, such that the minimal contribution has the same actions as the valid one, and the agent associated with each action in the minimal one is also one of the many associated agents with the same action in the redundant one[5].

[5] Since there is no requirement for the agents in a concurrent action to be distinct we do not need to consider whether a redundant annotation of concurrent action is valid.

Contribution	Annotation	Contribution	Annotation
$contrb_1$	α_1, α_2	$contrb_1$	α_1, α_2
$contrb_2$	α_1, α_2	$contrb_2$	α_1, α_2
		$contrb_3$	α_1, α_2
valid		invalid	

Figure 1: Valid and invalid joint action annotations

$$
\begin{array}{|l}
validjointannotation\ \_ : \mathbb{P}\ AnntdJointAct \\
\hline
\forall\ a : AnntdJointAct \bullet validjointannotation\ a \Leftrightarrow \\
\quad (\exists\ m : MinimalAnntdJointAct \bullet \\
\quad (\{c : m.anntcontrbs \bullet c.act\} = \\
\qquad \{c : a.anntcontrbs \bullet c.act\}) \wedge \\
\quad (\forall\ c_1 : m.anntcontrbs \bullet (\exists\ c_2 : a.anntcontrbs \bullet \\
\qquad c_1.act = c_2.act \wedge \\
\qquad c_1.agts \subset c_2.agts)))
\end{array}
$$

3.4.1 Joint Actions

In formalising the annotation of joint actions we rely on three auxiliary functions[6]. Firstly, the function $allValidAnntdJAs$ takes a joint action, a set of agent models and a minimal trust threshold, and returns all possible valid (minimal) annotations of that joint action, such that an agent is associated with a contribution if it can perform it and is trusted above the minimal trust threshold.

$$
\begin{array}{|l}
allValidAnntdJAs : JointAct \rightarrow \mathbb{P}\ AgentModel \\
\quad \rightarrow \mathbb{R} \rightarrow \mathbb{P}\ AnntdJointAct
\end{array}
$$

Secondly, $orderedAnntdJAs$ takes a set of possible annotations of a joint action and orders them according to the combined trust of the agents involved.

$$
\begin{array}{|l}
orderedAnntdJAs : \mathbb{P}\ AnntdJointAct \\
\quad \rightarrow \text{seq}\ AnntdJointAct
\end{array}
$$

Finally, $combineJA$ takes a sequence of minimal annotations and combines them into a single redundant annotation, such that each contribution in the redundant annotation is annotated with a set of agents corresponding to those agents that are associated with the same contribution in one of the minimal annotations.

$$
\begin{array}{|l}
combineJA : \text{seq}\ AnntdJointAct \\
\quad \rightarrow AnntdJointAct
\end{array}
$$

We can now formally describe the annotation of a joint action in the function $anntJointAct$ which takes a joint action, ja, a set of agent models, ms, the maximum number of agents to annotate a contribution with, max, and a minimum trust threshold, t, and returns an annotated joint action. The predicate part of this definition determines all possible valid annotations, orders them according to trust, and then takes the head of the sequence corresponding to the first max annotations from the front of the ordered sequence. Finally, the head of the sequence is combined into a single annotated joint action.

[6]For reasons of space we only give the function signatures here.

$$
\begin{array}{|l}
anntJointAct : JointAct \rightarrow \mathbb{P}\ AgentModel \\
\quad \rightarrow \mathbb{Z} \rightarrow \mathbb{R} \rightarrow AnntdJointAct \\
\hline
\forall\ ja : JointAct;\ ms : \mathbb{P}\ AgentModel; \\
\quad max : \mathbb{Z};\ t : \mathbb{R} \bullet \\
\quad anntJointAct\ ja\ ms\ max\ t = \\
\quad combineJA\ (\{i : \mathbb{Z} \mid i \leq max \bullet i\} \upharpoonright \\
\qquad (orderedAnntdJAs\ (\\
\qquad\quad allValidAnntdJAs\ ja\ ms\ t)))
\end{array}
$$

3.4.2 Concurrent Actions

In a similar manner, we make use of three auxiliary functions in formalising the annotation of concurrent actions. First, the function $allAnntdCAs$ takes a concurrent action, a set of agent models and a minimal trust threshold, and returns all possible annotations of the concurrent action, which associate an agent with a contribution if it can perform it and is trusted above the minimal trust threshold.

$$
\begin{array}{|l}
allAnntdCAs : ConcAct \rightarrow \mathbb{P}\ AgentModel \\
\quad \rightarrow \mathbb{R} \rightarrow \mathbb{P}\ AnntdConcAct
\end{array}
$$

Again, we make use of a function, $orderedAnntdCAs$, which takes a set of possible annotations and orders them according to the combined trust of the agents involved.

$$
\begin{array}{|l}
orderedAnntdCAs : \mathbb{P}\ AnntdConcAct \\
\quad \rightarrow \text{seq}\ AnntdConcAct
\end{array}
$$

Finally, we have a function $combineCA$ which takes a sequence of annotations and combines them into a single redundant annotation.

$$
\begin{array}{|l}
combineCA : \text{seq}\ AnntdConcAct \\
\quad \rightarrow AnntdConcAct
\end{array}
$$

Thus in a similar manner to joint actions the annotation of a concurrent action is given in the function $anntConcAct$, whose parameters are a set of agent models, the maximum number of agents to annotate a contribution with, and a minimum trust threshold.

$$
\begin{array}{|l}
anntConcAct : ConcAct \rightarrow \mathbb{P}\ AgentModel \\
\quad \rightarrow \mathbb{Z} \rightarrow \mathbb{R} \rightarrow AnntdConcAct \\
\hline
\forall\ ca : ConcAct;\ ms : \mathbb{P}\ AgentModel; \\
\quad max : \mathbb{Z};\ t : \mathbb{R} \bullet \\
\quad anntConcAct\ ca\ ms\ max\ t = \\
\quad combineCA\ (\{i : \mathbb{Z} \mid i \leq max\} \upharpoonright \\
\qquad orderedAnntdCAs\ (allAnntdCAs\ ca\ ms\ t))
\end{array}
$$

3.5 Annotated Plans

The notion of an annotated plan is formalised below in the schema $AnntdPlan$, in which all contributions are annotated with a *set* of agents. Each contribution is annotated with a set, rather than the individual agent that will execute it since, at this stage, the annotation represents the agents to request assistance from. Thus, to allow for redundant annotation, a contribution is associated with a set of agents rather than an individual. However, before a final cooperative intention can be formed, an agent must select one agent for each contribution and modify the annotated plan accordingly.

$$
\begin{array}{rcl}
APlanStep & ::= & AIndividual\langle\langle AnntdContrb\rangle\rangle \\
& | & AJoint\langle\langle AnntdJointAct\rangle\rangle \\
& | & AConc\langle\langle AnntdConcAct\rangle\rangle \\
& | & ASubgoal\langle\langle Goal\rangle\rangle
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{AnntdPlan} \\
\textit{achieves} : \textit{Goal} \\
\textit{precon} : \mathbb{P}\, \textit{Belief} \\
\textit{body} : \text{seq}\, \textit{APlanStep} \\
\hline
\end{array}
$$

The function $anntStep$ takes a plan step and applies the appropriate annotation function according to whether the step is an individual, joint or concurrent action (unless the step is a goal in which case it is not changed).

$$
\begin{array}{|l}
\textit{anntStep} : \textit{PlanStep} \to \mathbb{P}\, \textit{AgentModel} \to \mathbb{Z} \\
\quad \to \mathbb{R} \to \textit{APlanStep} \\
\hline
\forall\, ps : \textit{PlanStep};\ ms : \mathbb{P}\, \textit{AgentModel}; \\
\quad max : \mathbb{Z};\ t : \mathbb{R};\ aps : \textit{APlanStep} \bullet \\
\quad \textit{anntStep}\ ps\ ms\ max\ t = aps \Leftrightarrow \\
\qquad (\exists\, c : \textit{Contrb} \bullet \textit{Individual}(c) = ps \wedge \\
\qquad\quad aps = \textit{AIndividual}(\\
\qquad\quad \textit{anntContrb}\ c\ ms\ max\ t)) \vee \\
\qquad (\exists\, ja : \textit{JointAct} \bullet \textit{Joint}(ja) = ps \wedge \\
\qquad\quad aps = \textit{AJoint}(\\
\qquad\quad \textit{anntJointAct}\ ja\ ms\ max\ t)) \vee \\
\qquad (\exists\, ca : \textit{ConcAct} \bullet \textit{Conc}(ca) = ps \wedge \\
\qquad\quad aps = \textit{AConc}(\\
\qquad\quad \textit{anntConcAct}\ ca\ ms\ max\ t)) \vee \\
\qquad (\exists\, g : \textit{Goal} \bullet \textit{Subgoal}(g) = \\
\qquad\quad ps \wedge aps = \textit{ASubgoal}(g)) \\
\hline
\end{array}
$$

We can now formalise the annotation of a plan in the function $anntPlan$, which takes a plan and annotates each of its steps according to the supplied parameters, returning the corresponding annotated plan.

$$
\begin{array}{|l}
\textit{anntPlan} : \textit{Plan} \to \mathbb{P}\, \textit{AgentModel} \to \mathbb{Z} \\
\quad \to \mathbb{R} \to \textit{AnntdPlan} \\
\hline
\forall\, p : \textit{Plan};\ ms : \mathbb{P}\, \textit{AgentModel};\ max : \mathbb{Z}; \\
\quad t : \mathbb{R};\ ap : \textit{AnntdPlan} \bullet \\
\quad \textit{anntPlan}\ p\ ms\ max\ t = ap \Leftrightarrow \\
\qquad p.\textit{achieves} = ap.\textit{achieves} \wedge \\
\qquad p.\textit{precon} = ap.\textit{precon} \wedge \\
\qquad (\forall\, n : \mathbb{Z} \mid n \leq \#p.\textit{body} \bullet ap.\textit{body}\ n = \\
\qquad\quad \textit{anntStep}\ (p.\textit{body}\ n)\ ms\ n\ t) \\
\hline
\end{array}
$$

4. SOLICITING COMMITMENT

After deciding which agents to try to cooperate with (by annotating its plan), an agent must request assistance from those agents. There are several options for the level of information to include in a request for assistance. In particular an agent attempting to initiate cooperation can communicate either:

1. the whole plan, but without annotations,
2. just the actions it wants the potential participant to perform,
3. the goal for which assistance is required, along with the actions it wishes the potential participant to perform,
4. the whole plan, annotated only with the actions it wishes the potential participant to perform, or
5. the whole annotated plan.

These options provide varying degrees of information to the receiver, and support different objectives represented by the loosely coupled and closely coupled views, as we discuss below.

- The first alternative of communicating the whole plan without annotations, does not in general give sufficient information for the participant to make a decision about whether or not to cooperate, since it does not specify which actions it should perform. Without knowing which actions are requested of it, an agent cannot determine whether they will conflict with its intentions or their motivational value. There are a small number of exceptional circumstances in which an agent could make a decision; for example, if all actions in the plan and the goal it achieves are of motivational value, and the agent has no other intentions, then it can decide to cooperate. In general, however, this is not the case, and more information is required. Thus, we reject the first alternative.

- Remember that there must be some motivational justification for an agent choosing to perform a particular action, and although the overall goal must be of motivational value (or it would not have merited committing to), the particular actions required to achieve it might not be. For example, achieving the goal of getting a paper accepted for a conference is likely to have motivational value, but the actions involved in proof-reading and correcting are less likely to be valuable in themselves. Thus, while the *end* may have motivational value, the *means* may not if considered out of the context of the overall goal. In practice an agent's motivations are typically mitigated by the achievement of goals, rather than the performance of particular actions, although there are exceptions. Thus, an agent is unlikely to gain assistance for its goal if its request contains only the actions that it wishes to be performed, and not the goal that they achieve (as in the second alternative above). The exception to this is if the action is valued by the potential participant and the goal is not. For example, if you gain value from performing the action of driving, and I wish you to drive a getaway car in a robbery for me, then the negative motivational effect of achieving the goal would outweigh the benefit obtained from driving (assuming you are a law-abiding citizen). Thus, in this situation if I *believe* that the goal is of zero or negative motivational value to you, then I might make my request giving only the action for which assistance is sought.

- The third alternative requests assistance from the potential participant for a particular set of contributions, and towards a particular goal. This allows an agent to consider both the motivational value of the actions it is requested to perform, and the value it would gain if the overall goal is achieved.

- The fourth alternative also includes the complete plan, without the annotations related to other agents. This additional information can influence the potential participant's decision about whether to cooperate. If the participant is informed of the plan then it knows what other actions will be performed in the achievement of the goal. If it has a goal or intention that some action in the plan is not performed (by any agent), then it may refuse, even if it would otherwise have accepted based solely on the goal and actions *it* is to perform (in situations such as the getaway car example above).

- The final alternative includes both the plan, and the complete set of annotations; if the participant is informed of the other annotations in the plan, it is given information about which agents are likely to be involved in the cooperative interaction. If it has a goal or intention of not cooperating with another of the annotated agents then it may also refuse, even if would accept were its choice based only on the goal and actions

it is to perform. Note also that communicating *redundant* annotations makes recipients aware of the redundancy and the potential unnecessary constraints this may impose upon them. Thus, if the fifth alternative is used, the requesting agent must process the annotations contained in the request to remove redundant annotation of the potential participant[7].

In our framework, therefore, an agent has a choice of the latter four options. The choice about which of these approaches to use is a macro level consideration determined by the loosely or closely coupled approach being taken. We therefore simply assume that an agent uses one of them, without specifying which, leaving the agent's designer to select which is the most appropriate for the domain concerned.

Since cooperative intention establishment may involve many rounds of requesting, some agents may have already been asked for assistance for a previous action, in which case it is possible that an agent may have already accepted a request. Here, some form of commitment to perform the (previously requested) action will have been formed, and if an agent has agreed to perform some action to which it is no longer annotated in the latest plan annotation, it must be informed that its commitment is unnecessary. Similarly, if the agent has already agreed to perform the same action that it is currently annotated to then there is no need to ask it again

If the action is part of a joint or concurrent action which is currently annotated with a different group of agents, and the agent was informed of the original annotation, its decision to cooperate may be affected by the composition of the group (in particular its trust of the group members), and the agent must be informed of the changes. Each round of plan annotation involves forming a nominal commitment and requesting assistance. Therefore, if assistance has already been requested for a previous annotation of the plan, a nominal commitment will exist toward the agents whose assistance was requested. A new nominal commitment does not need to be formed; instead, the annotation of agents to whom the commitment is made are updated. Those agents that are not in the current annotation are removed from the commitment, since there is no need to inform them if assistance is no longer required, and any newly annotated agents are added. If no requests have previously been made for (a prior annotation of) the plan, then a new nominal commitment is formed to the agents contained in the current annotation. Consider the example of an agent requesting assistance, and forming a nominal commitment towards, three agents, α_1, α_2, and α_3. Now, suppose α_3 declines and the agent re-annotates its plan with agents α_1, α_2, and α_4, such that the former two are given the same tasks and α_4 assigned to the task for which α_3 declined. The initiator must update its nominal commitment to be towards this new set of agents, i.e. it must modify its commitment to α_3 to be towards α_4.

5. CONCLUSION

The problem of cooperation is complex, and comprises many distinct sub-problems, not least of which is the need to consider who to ask for assistance, and who to assist. Finding a cooperative plan to achieve a goal requires not just the selection of the plan based on the capabilities and trustworthiness of the agents that may perform the actions within it, but also a dynamic re-evaluation of these agents at the point at which it is executed. In this paper, we have described a procedure for selecting which agents to cooperate with by annotating a plan according to the capabilities and trustworthiness of others, and providing the capability for redundant annotation so that dynamic re-allocation of actions can take place. Importantly, this raises questions about what information to include in a request for cooperation in order to maximise the likelihood of success. This paper has described the model for these processes, with the associated implementation that has been constructed to demonstrate its validity being described elsewhere [5].

There are three particular areas of limitation of the work described in this paper that form the focus for ongoing work. The most significant area is the need to investigate a mechanism for introducing Marsh's notion of situational trust in a computationally practical manner [11]. As noted above, situational trust is a powerful mechanism that can give an agent valuable information in reasoning about others, but the cost of maintaining models of trust at a task specific level is prohibitively high. The primary problem in introducing situational trust is the need to determine the reason why a particular cooperative interaction failed. For an agent to maintain models of the trustworthiness of others at a task specific level it is necessary to know which task caused cooperation to fail. In some cases, where a particular agent is only responsible for performing a single action, this can be inferred from the plan. However, in general, determining the failure point requires agents to provide information about the failure. Our aim is to develop a means for an agent to have access to the kind of information provided by situational trust models, without the high cost of maintenance typically associated with them.

The second area of ongoing work is to use estimates of the expected quality of others' actions in considering requesting their assistance. Our current model simply uses knowledge about a particular agent being capable of a particular action, without considering the quality of execution that may result. In human interactions there is often a tradeoff between trust and the expected quality of the result. For example, one might ask assistance of a less trusted (in terms of reliability, or speed of response etc.) but highly knowledgeable expert rather than a highly trusted trainee. Similarly, when faced with a choice between two equally trusted agents the rational choice is to choose the one expected to result in the best quality outcome. Although related to situational trust, utilising the notion of quality of capabilities provides an additional metric in evaluating others.

The final area of current work is concerned with enabling agents to share information about the degree to which others are trusted. Where two or more strongly trusted agents cooperate they may share information about the trust they ascribe to others. Such information sharing allows agents to update their trust models in the light of others' experiences. Assuming agents are honest and have broadly the same aims (meaning that they enter into the same kind of cooperative interactions) then this approach enables agents to reduce the time take to obtain more accurate trust values of others. This is particularly beneficial in the case where an agent obtains information about another with whom it is yet to cooperate (recall that prior to interaction agents simply ascribe a default value to others). We noted earlier that trust can be seen as an individual's view of another's reputation. Similarly, the sharing of trust values in this manner can be thought of as a group of agents considering another's reputation in the view of the group. Clearly, many more complex and robust approaches to modelling reputation are possible, however it is our view that sharing trust in this manner can provide a computationally cheap mechanism for soliciting, from *trusted agents*, the opinions of others' trustworthiness.

[7]It could be argued that all redundant annotations should be removed in case an agent infers that if another is redundantly annotated, it may be treated similarly.

6. REFERENCES

[1] M. E. Bratman. Shared cooperative activity. *Philosophical Review*, 101(2):327–341, Apr. 1992.

[2] M. E. Bratman, D. Israel, and M. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355, 1988.

[3] C. Castelfranchi and R. Falcone. Principles of trust for MAS: Cognitive anatomy, social importance, and quantification. In *Proceedings of the Third International Conference on Multi-Agent Systems*, pages 72–79, Paris, France, 1998.

[4] M. d'Inverno and M. Luck. *Understanding Agent Systems*. Springer-Verlag, 2001.

[5] N. Griffiths. *Motivated Cooperation in Autonomous Agents*. PhD thesis, University of Warwick, 2000.

[6] N. Griffiths and M. Luck. Cooperative plan selection through trust. In F. J. Garijo and M. Boman, editors, *Multi-Agent System Engineering: Proceedings of the Ninth European Workshop on Modelling Autonomous Agents in a Multi-Agent World*. Springer, 1999.

[7] B. Grosz and S. Kraus. The evolution of SharedPlans. In A. Rao and M. Wooldridge, editors, *Foundations and Theories of Rational Agencies*, pages 227–262. Kluwer Academic Publishers, 1999.

[8] D. Kinny, M. Ljungberg, A. Rao, E. Sonenberg, G. Tidhar, and E. Werner. Planned team activity. In *Proceedings of the Forth European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 227–256, 1992.

[9] H. J. Levesque, P. R. Cohen, and J. H. T. Nunes. On acting together. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 94–99, Boston, MA, 1990.

[10] 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/The MIT Press, 1995.

[11] S. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, 1994.

[12] S. Marsh. Optimism and pessimism in trust. In *Proceedings of the Ibero-American Conference on Artificial Intelligence (IBERAMIA '94)*, 1994.

[13] J. M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall, Hemel Hempstead, 2nd edition, 1992.

[14] M. Wooldridge and N. R. Jennings. Formalizing the cooperative problem solving process. In *Proceedings of the Thirteenth International Workshop on Distributed Artificial Intelligence*, pages 403–417, Lake Quinhalt, WA, 1994.

[15] M. Wooldridge and N. R. Jennings. Cooperative problem-solving. *Journal of Logic and Computation*, 9(4):563–592, 1999.