Plan Analysis for Autonomous Sociological Agents

Michael Luck* Mark d'Inverno †

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

† Cavendish School of Computer Science, Westminster University, London W1M 8JS, UK
dinverm@westminster.ac.uk

Abstract. This paper is concerned with the problem of how effective social interaction arises from individual social action and mind. The need to study the *individual social mind* suggests a move towards the notion of *sociological* agents who can *model* their social environment as opposed to acting socially within it. This does not constrain social behaviour; on the contrary, we argue that it provides the requisite information and understanding for such behaviour to be effective. Indeed, it is not enough for agents to model other agents in isolation; they must also model the relationships between them. A *sociological agent* is thus an agent that can model agents *and* agent relationships. Several existing models use notions of autonomy and dependence to show how this kind of interaction comes about, but the level of analysis is limited. In this paper, we show how an existing agent framework leads naturally to the enumeration of a map of inter-agent relationships that can be modelled and exploited by sociological agents to enable more effective operation, especially in the context of multi-agent plans.

1 Introduction

Underlying all multi-agent systems are the notions of interaction and cooperation. Much research in the area of intelligent agents has thus sought to develop computational mechanisms for bringing about and sustaining these relationships in a dynamic and open world. The focus on mechanisms is just one part of the set of areas for investigation, however, and a separate and distinct strand of work aims to provide an understanding of the relationships themselves and how agents may influence each other through these relationships over the course of time. Over a number of years, one such effort has been in the work of Castelfranchi and Conte, which has focussed on issues relating to the *social foundations* of multi-agent systems. For example, they point out that the problem of how to allocate tasks and resources and how to coordinate actions is typically raised only after a collective or social problem or goal is assumed [3]. One key question in consideration of this is *how society is implemented in the minds of social agents.*

There are many definitions of a *social* agent. For example, Wooldridge states that any agent in a multi-agent system is necessarily social [14] and Moulin and Chaib-draa [10] take an agent to be social if it can model others. However, the term is more often associated with social *activity* such as provided by Wooldridge and Jennings [15] who refer to the process of interaction. Yet the need to study the *individual social mind* suggests a move towards the notion of *sociological* agents who can *model* their social environment as opposed to acting socially within it. This does not constrain social behaviour; on the contrary, we argue that it provides the requisite information and understanding for such behaviour to be effective. We argue that effective social agents must be sociological. These notions are implicit in the work of Castelfranchi, in particular, which has led to the construction and refinement of a theory of social relationships based on notions of inter-agent dependence[2, 4]. Taking as a base his Social Power Theory (SPT), he has sought to provide a computational model of autonomy and inter-agent relationships through the Social Dependence Networks (SDN)[12] that implement the constructs of SPT.

The point of this work is to identify dependence situations in which one agent depends on another for actions or resources, or is autonomous with respect to these components, according to a particular plan. Using these notions of dependence, the *negotiation power* of an agent can be found to represent how well an agent can sell itself on a market. Underlying this very powerful theory are plans, yet the analyses possible, while revealing certain information about inter-agent relationships, are still rather crude. In order to provide a more detailed and precise account of these dependencies, further analysis is required. SDN provides the motivation from a very particular perspective for considering agent plans, while more general social interaction provides a broader base of direction.

It is not enough for agents to model other agents in isolation; they must also model the relationships between them. A *sociological agent* is an agent that can model agents *and* agent relationships. In this paper, we show how an existing agent framework leads naturally to the enumeration of a map of inter-agent relationships that can be modelled and exploited by sociological agents to enable more effective operation, especially in the context of multi-agent plans. Importantly, the resulting strong model of individual agents could be applied to clarify several issues in SDN. Constraints of space, however, limit the potential for showing the particular impact on SDN, but we illustrate the potential benefits of the analysis with a detailed example at the end. The paper begins with a brief review of an agent framework and the relationships that arise within it, and then proceeds to provide a detailed description of single and multi-agent plans, identifying useful categories of plans and agents.

2 Preliminaries

Agent Framework Elsewhere, we have presented an agent framework to define entities, objects, agents and autonomous agents. Below we provide a brief overview of the main aspects of the framework [6, 8], and the relationships that arise within it [9], but we omit a detailed exposition. We use the Z notation [13] to formalise these notions and, though we assume some familiarity, the meaning should be clear.

As specified in Figure 1, an *entity* comprises a set of *motivations*, a set of *goals*, a set of *actions*, and a set of *attributes* such that the attributes and actions are non-empty. Entities can be used to group together attributes into a whole without any *functionality*. They serve as a useful abstraction mechanism by which they are regarded as distinct from the remainder of the environment, to organise perception. An *object* is then an entity with abilities that can affect the environment in which it is situated. Now, an *agent* is just an object either that is useful to another agent in terms of satisfying that agent's

$Entity$

$attributes : \mathbb{P}\, Attribute$
$capabilities : \mathbb{P}\, Action$
$goals : \mathbb{P}\, Goal$
$motivations : \mathbb{P}\, Motivation$

$attributes \neq \{\ \}$

$Object == [Entity \mid capableof \neq \{\ \}]$
$Agent == [Object \mid goals \neq \{\ \}]$

$AutoAgent == [Agent \mid motivations \neq \{\ \}]$
$NeutralObject == [Object \mid goals = \{\}]$
$ServerAgent == [Agent \mid motivations = \{\}]$

$MultiAgentSystem$

$entities : \mathbb{P}\, Entity$
$objects : \mathbb{P}\, Object$
$agents : \mathbb{P}\, Agent$
$autonomousagents : \mathbb{P}\, AutoAgent$
$neutralobjects : \mathbb{P}\, NeutralObject$
$serveragents : \mathbb{P}\, ServerAgent$

Fig. 1. Formal specification of the agent framework

goals, or that exhibits independent purposeful behaviour. In other words, an agent is an object with an associated set of goals, but one object may give rise to different instantiations of agents with different goals. This notion of agency relies upon the existence of other agents to provide the goals that are adopted to instantiate an agent. In order to escape an infinite regress of goal adoption, however, we can define *autonomous agents*, which are just agents that generate their own goals from motivations (not dissimilar to Antunes's notion of values [1]). We can also distinguish objects that are not agents, and agents that are not autonomous, as *neutral-objects* and *server-agents* respectively. An agent is then either a server-agent or an autonomous agent, and an object is either a neutral-object or an agent, and a multi-agent system simply contains a collection of these entities.

Agent Relations When an agent uses another non-autonomous entity, the entity adopts or satisfies the agent's goals and creates a *social* relationship [2] known as an *engagement*. In a *direct engagement*, a neutral-object or a server-agent adopts the goals of another. Chains of engagements are also possible, explicitly representing the goal and all the agents involved in the sequence of direct engagements. Since goals are grounded by motivations, the agent at the head of the chain must be autonomous.

Now, since it can generate its goals and decide when to adopt the goals of others, an autonomous agent is said to be *cooperating* with another autonomous agent if it has adopted the goal or 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. 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. (At any point in time, such autonomous agents may then be cooperating with others as well as attempting to satisfy their own self-motivated goals.) Further details of these relationships, specified in Figure 2, may be found elsewhere [9].

These inter-agent relationships are not imposed, but arise naturally from our view of agents as they interact (regardless of the manner of their initiation) and therefore underlie all multi-agent systems. They provide a means for analysing the interdependence of agents in terms of the goals some agents achieve for others. Moreover, from this

$$\begin{array}{l}
\text{--- } \mathit{DirectEngagement} \text{ ---}\\
\mathit{client} : \mathit{Agent}\\
\mathit{server} : \mathit{ServerAgent}\\
\mathit{goal} : \mathit{Goal}\\
\hline
\mathit{client} \neq \mathit{server}\\
\mathit{goal} \in (\mathit{client.goals} \cap \mathit{server.goals})
\end{array}$$

$$\begin{array}{l}
\text{--- } \mathit{Cooperation} \text{ ---}\\
\mathit{goal} : \mathit{Goal}\\
\mathit{generatingagent} : \mathit{AutoAgent}\\
\mathit{cooperatingagents} : \mathbb{P}_1\ \mathit{AutoAgent}\\
\hline
\mathit{goal} \in \mathit{generatingagent.goals}\\
\forall\, \mathit{aa} : \mathit{cooperatingagents} \bullet \mathit{goal} \in \mathit{aa.goals}\\
\mathit{generatingagent} \notin \mathit{cooperatingagents}
\end{array}$$

$$\begin{array}{l}
\text{--- } \mathit{EngagementChain} \text{ ---}\\
\mathit{goal} : \mathit{Goal}\\
\mathit{autoagent} : \mathit{AutoAgent}\\
\mathit{agentchain} : \mathrm{iseq}_1 \mathit{ServerAgent}\\
\hline
\mathit{goal} \in \mathit{autoagent.goals}\\
\mathit{goal} \in \bigcap\{s : \mathit{ServerAgent} \mid\\
\qquad s \in \mathrm{ran}\ \mathit{agentchain} \bullet s.\mathit{goals}\}
\end{array}$$

$$\begin{array}{l}
\text{--- } \mathit{AgentSociety} \text{ ---}\\
\mathit{MultiAgentSystem}\\
\mathit{dengagements} : \mathbb{P}\,\mathit{DirectEngagement}\\
\mathit{engchains} : \mathbb{P}\,\mathit{EngagementChain}\\
\mathit{cooperations} : \mathbb{P}\,\mathit{Cooperation}
\end{array}$$

Fig. 2. Formal specification of agent relationships

basic set of constructions, we can derive a detailed map of the relationships between individual agents for a better understanding of their current social interdependence. In particular, different situations of interdependence each suggest different possibilities for interaction. Below, we enumerate the various possible relations that might result in this view. Each relation is described and defined formally.

– The direct engagement relationship specifies the situation in which there is a direct engagement for which the first agent is the client and the second agent is the server. Thus, an agent, c, *directly engages* another server-agent, s, if, and only if, there is a direct engagement between c and s.

$$\begin{array}{l}
\forall\, c : \mathit{Agent};\ s : \mathit{ServerAgent} \bullet (c, s) \in \mathit{dengages} \Leftrightarrow\\
\qquad\qquad \exists\, d : \mathit{dengagements} \bullet d.\mathit{client} = c \wedge d.\mathit{server} = s
\end{array}$$

– The notion of direct engagement implies a tight coupling between the behaviours of the agents involved without an intermediate entity. However, there may be entities an agent engages that indirectly serve some purpose for it. An agent c *engages* another (server) agent s if there is some engagement chain ec that includes s, such that either c is before s in the chain or c is the autonomous agent of ec. (This requires the before relation, which 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. The details are omitted here.)

$$\begin{array}{l}
\forall\, c : \mathit{Agent}, s : \mathit{ServerAgent} \bullet (c, s) \in \mathit{engages} \Leftrightarrow\\
\qquad \exists\, \mathit{ec} : \mathit{engchains} \bullet (s \in (\mathrm{ran}\ \mathit{ec.agentchain}) \wedge c = \mathit{ec.autoagent}) \vee\\
\qquad\qquad\qquad (((c, s), \mathit{ec.agentchain}) \in \mathit{before})
\end{array}$$

– To distinguish engagements involving an intermediate agent we introduce the indirect engagement relation $\mathit{indengages}$. An agent *indirectly* engages another if it engages it, but does not *directly* engage it.

$$\mathit{indengages} = \mathit{engages} \setminus \mathit{dengages}$$

– If many agents directly engage the same entity, then no single agent has complete control over it. Any actions an agent takes affecting the entity may destroy or hinder the engagements of the other engaging agents. This in turn, may have a deleterious effect on the engaging agents themselves. It is therefore important to understand *when* the behaviour of an engaged entity can be modified without any deleterious effect (such as when no other agent uses the entity for a *different* purpose). In this case we say that the agent *owns* the entity. An agent, c, owns another agent, s, if, for every sequence of server-agents in an engagement chain, ec, in which s appears, c precedes it, or c is the autonomous client-agent that initiated the chain.

$$\begin{aligned}&\forall\, c : Agent;\ s : ServerAgent \bullet (c, s) \in owns \Leftrightarrow\\&\quad (\forall\, ec : engchains \mid s \in \operatorname{ran} ec.agentchain \bullet\\&\qquad ec.autoagent = c \ \vee\ ((c, s), ec.agentchain) \in before)\end{aligned}$$

– An agent, c, *directly owns* another agent, s, if it owns it, and directly engages it. Formally, this relation is the intersection of the direct engagement relation, $dengages$, and the generic ownership relation, $owns$.

$$downs = owns \cap dengages$$

– A further distinction of direct ownership can be made. Either no other agent directly owns the entity, or there is another agent who is also directly engaging that entity for the same purpose. The first case occurs normally but the second situation can occur if the entity is engaged by two agents, each for the same purpose as generated by a single autonomous agent. To distinguish these situations we define the *uniquely owns* relation, which holds when an agent *directly* and *solely* owns another. An agent c *uniquely owns* another s, if it directly owns it, and no other agent is engaging it.

$$\begin{aligned}&\forall\, c : Agent;\ s : ServerAgent \bullet (c, s) \in uowns \Leftrightarrow\\&\quad (c, s) \in\ downs \wedge \neg\ (\exists\, a : Agent \mid a \neq c \bullet (a, s) \in engages)\end{aligned}$$

– An agent may own another with respect to either multiple distinct goals (which may conflict) or a single goal. The distinction is important because achieving one goal may affect the achievement of another. An agent, c, *specifically owns* another agent, s, if it owns it, and c has only one goal.

$$\begin{aligned}&\forall\, c : Agent;\ s : ServerAgent \bullet\\&\quad (c, s) \in sowns \Leftrightarrow (c, s) \in\ owns \wedge \#(s.goals) = 1\end{aligned}$$

– Finally, an agent, b, *cooperates* with agent, a, if and only if both agents are autonomous, and there is some cooperation in which a is the generating agent and b is in the set of cooperating agents. Notice that the relationship is not symmetric: if b is cooperating with a, a need not be cooperating with b.

$$\begin{aligned}&\forall\, a, b : AutoAgent \bullet (a, b) \in cooperates \Leftrightarrow\\&\quad \exists\, c : cooperations \bullet a = c.generatingagent \wedge\\&\qquad b \in c.cooperatingagents\end{aligned}$$

This analysis of the relationships between *agents* provides computational entities with a means of determining how they should approach interactions with those agents. For example, if I own an entity, I can do as I please, and other agents would be ill-advised to attempt to use this entity for another purpose. If I only engage it, then I may be more constrained in my interaction and may anticipate other agents engaging it.

The framework described above, together with the relationships arising from it, are suitable for reasoning both *about* entities in the world, and *with* entities in the world. That is to say that in addition to providing us with a way of understanding and analysing agents, agents themselves can also use the entity hierarchy as a basis for reasoning about other agents within their environment. It may be relevant, for example, for them to consider the functionality of other agents and the likelihood, that these others may or may not be predisposed to help in the completion of certain tasks. In this section we describe how we enable the possibility within agents of such reasoning.

Models If agents are to model their environment, they need more than just actions, goals and motivations; they require an *internal store*. Agents without internal stores are extremely limited since their past experience cannot direct behaviour, and actions can only be selected *reflexively*. A store exists as part of an agent's state in an environment but it must also have existed *prior* to that current state. We call this feature an *internal store* or *memory*, and define *store agents* as those with memories. Formally, a store-agent is a refinement of an agent, with the addition of a $store$ variable represented as a non-empty set of attributes. We omit this simple schema due to space constraints.

The most obvious things to represent in such models are other agents. Thus we need to describe agents who can model others in their environment and, for sufficiently advanced agents, the autonomy of others. However, it is inadequate to model entities in isolation; the relationships between them must also be modelled. Unlike *social* agents that engage in interaction with others (or social activity), *sociological* agents also model agent relationships. It is a simple matter to define the model an agent has of another agent ($AgentModel$), or its model of a cooperation relationship ($CoopModel$) by re-using the agent framework components.

$$\ldots\ AgentModel == Agent \ldots$$
$$\ldots CoopModel == Cooperation\ \ldots$$

Even though the types of these constructs are equivalent to those presented earlier, it is useful to distinguish physical constructs from mental constructs such as models, as it provides a conceptual aid. Similarly, we can define the models agents have of entities, objects and autonomous agents as well as models of other relationships we have described (though only the model of a cooperation is given above). A sociological agent is thus specified as a refinement of the $Agent$ schema.

```
┌─ SociologicalAgent ─────────────────────────────────
│ Agent
│ ... agentmodels : ℙ AgentModel; ... coopmodels : ℙ CoopModel ...
└─────────────────────────────────────────────────────
```

3 Plans

In this section we develop our models of agents further to consider multi-agent plans. Sometimes, agents may select an action, or a set of concurrent actions, to achieve goals directly. At other times, however, there may not be such a simple correspondence between goals and actions, and appropriately designed agents may perform *sequences* of actions, or *plans*, to achieve their goals. In multi-agent systems, agents have at their disposal not only their own capabilities but, potentially, the capabilities of others. Agents with plans requiring actions outside their competence need models of others to consider making use of their capabilities. If these agents are sociological then they can evaluate plans with respect to their model of current agent relationships. For example, agents can decide to what extent plans can exploit current relationships as well as consider how they may *impinge* on them. In general, agents must reason about exploiting existing and potential relationships without inadvertently or unnecessarily destroying them.

In this work, we do not model the *process* of planning, but instead consider how plans can be modelled and how agents can evaluate plans using these models. Similar to the BDI view, we take an agent to have a repository of goals and a repository of plans that have either been designed before the agent starts executing [7] or acquired and developed over the course of the agent's life [11]. Each plan may be associated with one or more goals, identifying the plan as a potential means of achieving those goals, as used for example, by Georgeff [7].

There are many different notions of agent plans both at the theoretical and the practical level, and in order to substantiate the claim that the agent framework and ensuing models can be generally applied it is necessary that different representations of plans can all be equally accommodated. Whilst we do not specify every type of plan of which we are aware, we do intend to show *how* the agent framework can be extended to describe familiar notions of plans, and to impress upon the reader how other models of plans can be similarly accommodated. We aim to achieve this by specifying general theoretical plan representations that we call *total* plans, *partial* plans and *tree* plans.

One methodological characteristic of our work is the *incremental* development of the models in it. Therefore, we first construct a high-level model of *plan-agents*, which applies equally well to reactive or deliberative, single-agent or multi-agent, planners. It represents a high-level of abstraction because nothing is decided about the nature of the agent, the plan representation, or of the agent's environment; we simply distinguish *categories* of plan and possible relationships between an agent's plans and goals. Specifically, we define *active* plans as those identified as candidate plans not yet selected for execution; and *executable* plans as those active plans that have been selected for execution.

Formally, we initially define the set of all agent plans to be a given set ($[Plan]$), so that at this stage we abstract out any information about the nature of plans themselves. Our highest-level description of a *plan-agent* can then be formalised in the $PlanAgent$ schema below. Since plans must be encoded as aspects of an internal store, the $StoreAgentState$ schema is included.

$$\begin{array}{|l}
\multicolumn{1}{l}{\textit{PlanAgent}} \\
\hline
StoreAgent \\
goallib : \mathbb{P}\, Goal \\
planlib, activeplans, executableplans : \mathbb{P}\, Plan \\
activeplangoal, plangoallib : Goal \pfun \mathbb{P}\, Plan \\
\hline
\mathrm{dom}\ activeplangoal \subseteq goals \ \land\ \bigcup(\mathrm{ran}\ activeplangoal) = activeplans \\
\mathrm{dom}\ plangoallib \subseteq goallib \ \land\ \bigcup(\mathrm{ran}\ plangoallib) \subseteq planlib \\
goals \subseteq goallib \ \land\ executableplans \subseteq activeplans \subseteq planlib \\
\hline
\end{array}$$

The variables $goallib$, $planlib$, $activeplans$ and $executableplans$ represent the agent's repository of goals, repository of plans, active plans and executable plans, respectively. Each active plan is necessarily associated with one or more of the agent's current goals as specified by $activeplangoal$. For example, if the function contains the pair $(g, \{p_1, p_2, p_3\})$, it indicates that p_1, p_2 and p_3 are competing active plans for g. Whilst active plans must be associated with at least one active goal the converse is not true, since agents may have goals for which no plans have been considered. Analogously the $plangoallib$ function relates the repository of goals, $goallib$, to the repository of plans, $planlib$. However, not necessarily all library plans and goals are related by this function.

A plan consisting of a total order of plan-actions is a *total plan*, which is represented as a sequence of plan-actions. (Other types of plan can be defined similarly and accommodated easily within our specification.) In *single*-agent systems, all the actions of such plans must be within the agent's capabilities. However, plan-agents in *multi*-agent systems can consider executing plans containing actions not within their capabilities as long as they can model the capabilities of others as we have described. It is our claim that if plan agents are also *sociological* agents then they can make more informed choices about plan selection. (Of course, while this describes how an agent can consider its use of plans involving others, it does not impact the issues involved in whether those others choose to cooperate or not.) Agents can be constrained in their design by including additional predicates. For example, it is possible to restrict the active plans of an agent, with respect to a current goal, to those which are related to that goal by the function $plangoallib$. This can be achieved in the specification of such an agent by including the following predicate.

$$\begin{aligned}&\forall\, ag : PlanAgent;\ g : Goal;\ ps : \mathbb{P}_1\, Plan \bullet (g, ps) \in ag.activeplangoal \Rightarrow\\&\quad(\exists\, qs : \mathbb{P}\, Plan \mid qs \subset ag.planlib \bullet (g, (ps \cup qs)) \in ag.plangoallib)\end{aligned}$$

Multi-Agent Plans In order for agents to reason about plans involving others it is necessary to analyse the nature of the plans themselves. This involves defining first the *components* of a plan, and then the *structure* of a plan, as shown in Figure 3. The components, which we call *plan-actions*, each consist of a *composite-action* and a set of related entities as described below. The structure of plans defines the relationship of the component plan-actions to one another. For example, plans may be *total* and define a sequence of plan-actions, *partial* and place a partial order on the performance of plan-actions, or *trees* and, for example, allow choice between alternative plan-actions at every stage in the plan's execution.

We identify four types of action that may be contained in plans, called *primitive*, *template*, *concurrent-primitive* and *concurrent-template*. There may be other categories and variations on those we have chosen but not only do they provide a starting point for specifying systems, they also illustrate how different representations can be formalised and incorporated within the same model. A primitive action is simply a base action as defined in the agent framework. An action template provides a high-level description of what is required by an action, defined as the set of all primitive actions that may result through an instantiation of that action-template. For example, in dMARS [5], template actions represent action formulae containing free variables, and become primitive actions when bound to values. Concurrent-primitive and concurrent-template actions are primitive actions and action templates performed concurrently. We define a new type, $ActnComp$, as a *compound-action* to include all four of these types.

$$
\begin{array}{l}
Primitive == Action \\
Template == \mathbb{P}\, Action \\
ConcPrimitive == \mathbb{P}\, Action \\
ConcTemplate == \mathbb{P}(\mathbb{P}\, Action) \\
ActnComp ::= Prim\langle\!\langle Primitive\rangle\!\rangle \\
\quad \mid Temp\langle\!\langle Template\rangle\!\rangle \\
\quad \mid ConcPrim\langle\!\langle ConcPrimitive\rangle\!\rangle \\
\quad \mid ConcTemp\langle\!\langle ConcTemplate\rangle\!\rangle
\end{array}
$$

$$
\begin{array}{lll}
TotalPlan == \mathsf{seq}\ PlanAction & & \\
TreePlan & ::= & Tip\langle\!\langle PlanAction\rangle\!\rangle \\
 & \mid & Fork\langle\!\langle \mathbb{P}_1(PlanAction \times TreePlan)\rangle\!\rangle \\
Plan & ::= & Part\langle\!\langle PartialPlan\rangle\!\rangle \\
 & \mid & Total\langle\!\langle TotalPlan\rangle\!\rangle \\
 & \mid & Tree\langle\!\langle TreePlan\rangle\!\rangle
\end{array}
$$

$$
\begin{array}{|l}
planpairs : Plan \rightarrow \mathbb{P}\, PlanAction \\
planentities : Plan \rightarrow \mathbb{P}\, EntityModel \\
planactions : Plan \rightarrow \mathbb{P}\, Action
\end{array}
$$

$$
\begin{array}{l}
PlanAction == \mathbb{P}(ActnComp \times \mathbb{P}\, EntityModel) \\
PartialPlan == \{ps : PlanAction \leftrightarrow PlanAction \mid \forall\, a, b : PlanAction \bullet \\
\qquad\qquad (a, a) \notin ps^{+} \wedge (a, b) \in ps^{+} \Rightarrow (b, a) \notin ps^{+} \bullet ps\}
\end{array}
$$

Fig. 3. Plan components and structure

Actions must be performed by entities, so we associate every compound-action in a plan with a set of entities, such that each entity in the set can potentially perform the action. At some stage in the planning process this set may be empty, indicating that no choice of entity has yet been made. We define a *plan-action* as a set of pairs, where each pair contains a compound-action and a set of those entities that could potentially perform the action. Plan-actions are defined as a *set* of pairs rather than *single* pairs so that plans containing simultaneous actions can be represented.

The following examples illustrate this representation. First, action a_1 is to be performed by either the plan-agent itself or the entity $entity1$. The second example describes the two separate actions, a_{2_1} and a_{2_2}, being performed simultaneously by the two entities $entity1$ and $entity2$ respectively. Then, the third example states that the actions a_{3_1} and a_{3_2} are to be performed simultaneously. No entity has been established as a possibility to perform a_{3_1}, and a_{3_2} is to be performed by either $entity2$ or $entity3$.

1. $\{(a_1, \{self, entity1\})\}$
2. $\{(a_{2_1}, \{entity1\}), (a_{2_2}, \{entity2\})\}$
3. $\{(a_{3_1}, \{\ \}), (a_{3_2}, \{entity2, entity3\})\}$

We specify three commonly-found categories of plan according to their structure as discussed earlier. Other types may be specified similarly. A partial plan imposes a partial order on the execution of actions, subject to two constraints. First, an action cannot be performed before itself and, second, if plan-action a is before b, b cannot be before a. A plan consisting of a total order of plan-actions is a total plan. Formally, this is represented as a sequence of plan-actions. A plan that allows a choice between actions at every stage is a tree. In general, a tree is either a leaf node containing a plan-action, or a fork containing a node, and a (non-empty) set of branches each leading to a tree. These are formalised in Figure 3, replacing the definition of *Plan* as a given set by a free-type definition to include the three plan categories thus defined. For *single*-agent systems all the actions of an executable plan must be within its capabilities:

$$\forall\, sap : PlanAgent;\ plan : Plan \mid plan \in sap.planlib \bullet$$
$$planactions\ plan \subseteq sap.capabilities$$

However, plan-agents in *multi* agent systems can consider executing plans containing actions not within their capabilities as long as they can model the capabilities of others. If such agents are also *sociological*, then we claim that they can make more informed choices about plan selection, through a better analysis of situations, as discussed below.

```
┌─ SPAgent ─────────────────────────────────
│ SociologicalAgent
│ PlanAgent
└───────────────────────────────────────────
```

4 Sociological Agents

Sociological agents with the capacity to model agents, relationships and plans (especially multi-agent plans) have available to them an appropriate level of detail to coordinate their actions in order to achieve their local goals more effectively. This bold claim arises from the increased awareness and appreciation not of the agents in their environment, but of the *impact* of those agents. In particular, the description of the multi-agent environment advanced so far enables an analysis to reveal opportunities to exploit existing relationships and how to minimise effort and avoid conflict.

To illustrate this greater reasoning capacity of sociological agents, we describe below some examples of categories of goals, agents and plans (with respect to the models of the sociological plan-agent), that may be relevant to an agent's understanding of its environment. Each of the categories is formally defined in Figures 4 and 5, in which the sociological plan-agent is denoted as spa. Any variable preceded by $model$ denotes the models that spa has of some specific type of entity or relationship. For example, $spa.modelneutralobjects$ and $spa.modelowns$ are the set of neutral objects and ownership relations the sociological agent models in its environment.

– A *self-sufficient plan* is any plan that involves only neutral-objects, server-agents the plan-agent owns, and the plan-agent itself. Self-sufficient plans can therefore be executed without regard to other agents, and exploit current agent relationships. (The formal definition makes use of the relational image operator: in general, the relational image $R(\!|\ S\ |\!)$ of a set S through a relation R is the set of all objects y to which R relates some member x of S.)

s-suff plan	$\forall\, p \in spa.planlib \bullet selfsuff(p) \Leftrightarrow spa.planentities(p) \subseteq spa.modelneutralobjects \cup spa.modelself \cup spa.modelowns(\!|\ spa.modelself\ |\!)$
s-suff goal	$\forall\, g \in spa.goallib \bullet selfsuffgl(g) \Leftrightarrow (\exists\, p \in sag.plangoallib(g) \bullet p \in selfsuff)$
rel goal	$\forall\, g \in spa.goallib \bullet reliantgoal(g) \Leftrightarrow spa.plangoallib\ g \neq \{\,\} \wedge \neg\,(\exists\, p : spa.plangoallib\ g \bullet p \in selfsuff)$

Fig. 4. Sociological Agent Categories I

coop agents

$$\forall\, p : spa.planlib \bullet cooperatingagents(p) = \\ \{a : spa.modelautoagents \mid a \in spa.planentities\ p \bullet a\} \setminus spa.modelself$$

affected agents

$$\forall\, p : spa.planlib \bullet affectedagents\ p = \\ \{a : spa.modelautoagents \mid (\exists\, s : ServerAgent \bullet s \in spa.planentities\ p \land \\ (a, s) \in spa.modelengages) \setminus spa.modelself$$

least direct fuss plans

$$\forall\, g \in spa.goallib \bullet leastdirectfuss\ g = \\ \{p : Plan \mid (p \in spa.plangoallib\ g)\ \land \\ \neg\, (\exists\, q : Plan \mid q \in spa.plangoallib\ g \bullet \\ \#(cooperatingagents\ q) < \#(cooperatingagents\ p)) \bullet p\}$$

least fuss plans

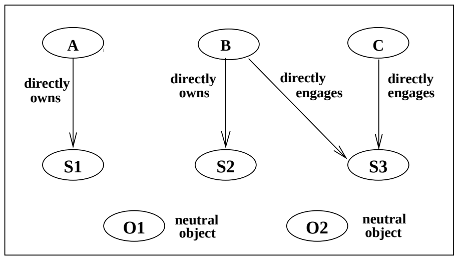
$$\forall\, g \in spa.goallib \bullet leastfuss\ g = \\ \{p : Plan \mid (p \in spa.plangoallib\ g) \land \\ \neg\, (\exists\, q : Plan \mid q \in spa.plangoallib\ g \bullet \\ \#(affectedagents\ q) < \#(affectedagents\ p)) \bullet p\}$$

Fig. 5. Sociological Agent Categories II

- A *self-sufficient goal* is any goal in the goal library that has an associated self-sufficient plan. These goals can then, according to the agent's model, be achieved independently of the existing social configuration.
- A *reliant-goal* is any goal that has a non-empty set of associated plans that is not self-sufficient.

For each plan that is not self-sufficient, a sociological plan-agent can establish the autonomous agents that may be affected by its execution, which is an important criterion in selecting a plan from competing active plans. An autonomous agent A may be affected by a plan in one of two ways: either it is required to perform an action directly, or it is engaging a server-agent S required by the plan. In this latter case, a sociological plan-agent can reason about either persuading A to share or release S, taking S without permission, or finding an alternative server-agent or plan. To facilitate such an analysis, we consider the following categories of agents and plans, formally defined in Figure 5.

- The *cooperating autonomous agents* of a plan are those autonomous agents, other than the plan-agent itself, that are involved in performing actions of that plan. They will need to cooperate with the plan-agent for the plan to be executed. Formally, an agent is a cooperating autonomous agent with respect to a plan, if it is contained in the set entities required for the plan. Note that an agent cannot guarantee that the identified cooperating agents of a plan will cooperate, only that cooperation is necessary for the plan to be performed.
- The *affected autonomous agents* of a plan are those autonomous agents, other than the plan-agent itself, that are engaging an entity required in the plan. Formally, an autonomous agent is *affected* with respect to a plan if a server-agent, contained in the set of entities required by the plan, is currently engaged by the autonomous



$$\begin{array}{l} \{A, B, C\} \in \mathit{modelautoagents} \\ \{A\} = \mathit{modelself} \\ \{S1, S2, S3\} \in \mathit{modelserveragents} \\ \{O1, O2\} \in \mathit{modelneutralobjects} \\ \{(A, S1), (B, S2), (B, S3), (C, S3)\} \subseteq \\ \qquad\qquad\qquad \mathit{modeldengagements} \\ \{(A, S1), (B, S2)\} \subseteq \mathit{modeldowns} \end{array}$$

Fig. 6. Example: A Sociological Agent's Model

agent. These agents *may* need to cooperate with the plan-agent. Notice that the affected autonomous agents do not include the cooperating agents.
- The *least-direct-fuss plans* for any reliant-goal are those plans that require the fewest number of cooperating agents.
- The *least-fuss plans* for any reliant-goal are those plans that require the fewest number of affected autonomous agents.

The categories described above are useful to an agent both at planning time and at run time. In the example below we consider their importance in the former case.

5 An Illustrative Example

To illustrate the value to an autonomous sociological plan-agent of being able to analyse plans using the categories above, consider an autonomous sociological plan-agent, A, and suppose that it models the agent relationships in its environment as follows. Autonomous agent B directly owns the server-agent $S2$ and directly engages $S3$; autonomous agent C directly engages $S3$; and A directly owns $S1$. In addition, in A's view, $O1$ and $O2$ are neutral-objects. This agent configuration can be seen in Figure 6 and would be represented in A's models as shown.

Consider also that agent A generates the goal, g_A, and activates four *total* plans p_1, p_2, p_3 and p_4 to achieve g_A as follows. The four plans are then in the set of active plans, and the pair $(g_A, \{p_1, p_2, p_3, p_4\})$ is in the function $\mathit{activeplangoal}$ relating current goals to candidate active plans. These goals and plans are shown in Figure 7. Notice that since in plan p_1, action a_1 can be performed by either agents B or C, and action a_3 by either $S2$ or $S3$, there are four possible ways of executing it, represented by p_{1_1}, p_{1_2}, p_{1_3} and p_{1_4} at the end of the figure. The agent then has seven alternative plans for execution selection.

Now, by inspection, the entities required by the plan p_2 are A, $S1$ and $O1$.

$\mathit{planentities}\ p_2 = \{A, S1, O1\}$

The previous definition of a self-sufficient plan for an agent A is any plan that only requires neutral-objects, agents owned by A, and A itself. In this case the union of the set of neutral-objects, owned agents and A itself is simple to calculate.

$\mathit{modelneutralobjects} \cup \mathit{modelself} \cup \mathit{modelowns}(\!|\ \mathit{modelself}\ |\!) = \{O1, O2, A, S1\}$

$$\begin{array}{l}
\{g_A\} \subseteq \mathit{goals} \\
\{p_1, p_2, p_3, p_4\} \subseteq \mathit{activeplans} \\
(g_A, \{p_1, p_2, p_3, p_4\}) \in \mathit{activeplangoal} \\[4pt]
p_1 = \mathit{Total}\ \{\{(a_1, \{B, C\}), (a_2, \{A\})\}, \{(a_3, \{S2, S3\})\}\} \\[4pt]
p_{1_1} = \mathit{Total}\ \{\{(a_1, \{B\}), (a_2, \{A\})\}, \{(a_3, \{S2\})\}\} \\
p_{1_2} = \mathit{Total}\ \{\{(a_1, \{B\}), (a_2, \{A\})\}, \{(a_3, \{S3\})\}\} \\
p_{1_3} = \mathit{Total}\ \{\{(a_1, \{C\}), (a_2, \{A\})\}, \{(a_3, \{S2\})\}\} \\
p_{1_4} = \mathit{Total}\ \{\{(a_1, \{C\}), (a_2, \{A\})\}, \{(a_3, \{S3\})\}\} \\[4pt]
p_2 = \mathit{Total}\ \{\{(a_{11}, \{O1\}), (a_1, \{A\})\}, \{(a_{12}, \{S1\}), (a_2, \{A\})\}, \\
\qquad \{(a_{13}, \{O1\}), (a_3, \{A\})\}, \{(a_{14}, \{S1\}), (a_4, \{A\})\}, \{(a_{15}, \{O1\}), (a_5, \{A\})\}\} \\
p_3 = \mathit{Total}\ \{\{(a_1, \{A\})\}, \{(a_2, \{S3\})\}\} \\
p_4 = \mathit{Total}\ \{\{(a_1, \{A\})\}, \{(a_2, \{S2\})\}\}
\end{array}$$

Fig. 7. Example agent goals and plans

The set of entities required by p_2 is a subset of this set which means that p_2 is self-sufficient, as is the associated goal g_A.

$$\{A, S1, O1\} \subseteq \{O1, O2, A, S1\} \Rightarrow p_2 \in \mathit{selfsuff}$$
$$p_2 \in \mathit{selfsuff} \Rightarrow g_A \in \mathit{selfsuffgl}$$

A is thus able to achieve g_A without affecting other autonomous agents and can act without regard to them whilst exploiting the current set of agent relationships. However, it may decide that, even though p_2 is self-sufficient, it is too costly, dismisses this possibility, and evaluates the six other alternatives to give the information shown in Table 1. It can be seen that the least-fuss and least-direct-fuss plans are as follows. Each of the least fuss plans affects only the one agent (in fact agent B in each case) as a result of requiring the server-agent S2 which is engaged by B. Both the direct least fuss plans require no cooperating agents.

$$\mathit{leastfuss}\ g_A = \{p_{1_1}, p_{1_3}, p_4\} \wedge \mathit{leastdirectfuss}\ g_A = \{p_3, p_4\}$$

Based on this analysis, p_4 may seem like the best candidate plan for execution since it does not involve the direct cooperation of other entities, and only affects one autonomous agent, B. The plan-agent can then analyse options concerning how to engage $S2$. Clearly, the final choice of plan must also consider other factors such as the motivations of the plan-agent, and its models of the motivations of others affected by plans. Nevertheless, this brief example illustrates just how sociological agents can use the

Plan	p_{1_1}	p_{1_2}	p_{1_3}	p_{1_4}	p_3	p_4
cooperatingagents	$\{B\}$	$\{B\}$	$\{C\}$	$\{C\}$	$\{\}$	$\{\}$
affectedagents	$\{B\}$	$\{B, C\}$	$\{B\}$	$\{B, C\}$	$\{B, C\}$	$\{B\}$

Table 1. Example: A Sociological Agent's Evaluation of its Plans

plan, goal and agent categories defined in this section as important criteria in evaluating alternative active plans. If agents can model the plans of others, or produce agreed *multi-agent plans*, then they can *coordinate* their actions in order to achieve their local goals more effectively. Agents can then take advantage of the plans of others to avoid duplication of effort and to avoid *conflict*, which arises, for example, when two agents require direct ownership of the same entity at the same time.

Once agents are designed with the ability to reason about the plans of other agents, bargaining can take place between agents able to help each other in their plans. As an example, suppose agent A has a plan that necessarily involves the cooperation of B. It may be appropriate for A to consider the plans of B that involve A's cooperation since A may then realise that B has a high-priority plan that can only be achieved with A's cooperation. In this case A would consider herself to be in a strong bargaining position. The actual level at which other agents are modelled is clearly critical in directing behaviour. For example, a sociological agent with models of other agents as non-sociological may realise that these agents are unable to recognise agent relationships. The sociological agent may then be concerned that these other agents may use entities that destroy their own existing agent relationships.

6 Conclusions

As Castelfranchi has shown in his work on Social Power Theory, the relationships between agents are critical for effective behaviour in dynamic and open environments composed of multiple autonomous agents. Without an adequate appreciation of them, opportunities for interaction to enhance and improve individual agent performance may be missed, and agents may not be duly exploited. In this paper we have extended previous work to show how detailed models of plans, and the categories that may be derived from them, can be used to map the social landscape effectively. This is vital — in our view, autonomy is an absolute, but it means that agents will seek to exert power over others. Just as Castelfranchi's notion of negotiation power provides a *measure* of the independence or autonomy of an agent in his view of the world, so it provides us with a test of whether an agent is autonomous. Power *will* be used by an agent that can influence others — an autonomous agent, in seeking to maximise its benefit, must make use of the information available in its models of agents and relationships.

The analysis we provide is not tailored to any pre-existing model, and is generally applicable, though it does arise naturally from our clean and simple agent framework. However, the particular value can be seen in applications that embody critical notions of dependence among agents such as, for example, Social Power Theory and its computational counterpart, Social Dependence Networks [12]. Our work provides two key benefits: first, it addresses some of the weaknesses in the SDN model in relation to the ambiguity of some constructs such as the nature of an *owned resource*, which we have clarified and tightened; second, it balances that earlier work which focussed on the problem *situations*, such as dependencies, by considering configurations of *solutions* to minimise effort and take advantage of opportunities in dealing with dependencies (through self-sufficient plans, for example). The next step is to explore the space of plans, goals and agents in more detail, based on the inter-agent relationships described above, and to show how further, more refined categories in the manner of those above, impact an agent's capacity to understand its environment and the agents within it.

References

1. L. Antunes, J. Faria, and H. Coelho. Improving choice mechanisms within the BVG architecture. In *Intelligent Agents VII. Agent Theories, Architectures and Languages–7th International Workshop, ATAL-2000, Proceedings.* LNAI, Springer-Verlag, 2001. In this volume.
2. C. Castelfranchi. Social power. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI — Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 49–62. Elsevier, 1990.
3. C. Castelfranchi and R. Conte. Distributed artificial intelligence and social science: Critical issues. In G. M. P. O'Hare and N. R. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, pages 527–542. Wiley, 1996.
4. C. Castelfranchi, M. Miceli, and A. Cesta. Dependence relations among autonomous agents. In E. Werner and Y. Demazeau, editors, *Decentralized AI 3: Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 215–231. Elsevier, 1992.
5. M. d'Inverno, D. Kinny, M. Luck, and M. Wooldridge. A formal specification of dMARS. In *Intelligent Agents IV: Proceedings of the Fourth International Workshop on Agent Theories, Architectures and Languages*, volume 1365, pages 155–176. Springer-Verlag, 1998.
6. M. d'Inverno and M. Luck. Development and application of a formal agent framework. In M. G. Hinchey and L. Shaoying, editors, *ICFEM'97: First IEEE International Conference on Formal Engineering Methods*, pages 222–231. IEEE Press, 1997.
7. M. P. Georgeff and F. F. Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 972–978, Detroit, MI, 1989.
8. M. Luck and M. d'Inverno. A formal framework for agency and autonomy. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 254–260. AAAI Press / MIT Press, 1995.
9. M. Luck and M. d'Inverno. Engagement and cooperation in motivated agent modelling. In C. Zhang and D. Lukose, editors, *Distributed Artificial Intelligence Architecture and Modelling: Proceedings of the First Australian Workshop on Distributed Artificial Intelligence, LNAI 1087*, volume 1087, pages 70–84. Springer Verlag, 1996.
10. B. Moulin and B. Chaib-draa. An overview of distributed artificial intelligence. In G. M. P. O'Hare and N. R. Jennings (eds), editors, *Foundations of Distributed Artificial Intelligence*, pages 3–56. John Wiley and Sons, 1996.
11. J. S. Rosenschein. Multiagent planning as a social process: Voting, privacy, and manipulation. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, page 431, June 1995.
12. J. S. Sichman, R. Conte, C. Castelfranchi, and Y. Demazeau. 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.
13. J. M. Spivey. *The Z Notation: A Reference Manual.* Prentice Hall, Hemel Hempstead, 2nd edition, 1992.
14. M. Wooldridge. *The Logical Modelling of Computational Multi-Agent Systems.* PhD thesis, Department of Computation, UMIST, Manchester, UK, October 1992. (Technical Report MMU–DOC–94–01, Department of Computing, Manchester Metropolitan University, UK).
15. M. J. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2), 1995.