In Distributed Artificial Intelligence Architecture and Modelling: Proceedings of the First Australian Workshop on Distributed Artificial Intelligence, C. Zhang and D. Lukose, (eds.), Lecture Notes in Artificial Intelligence, 1087, 115-129, Springer-Verlag, 1996

A Formal View of Social Dependence Networks

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

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

Abstract. In response to the problems that have arisen regarding the terminology and concepts of agent-oriented systems, previous work has described a formal framework for understanding agency and autonomy. In particular, this work made the claim that the framework could serve as a vehicle for the precise presentation and evaluation of models and theories of multi-agent systems. We support this claim by outlining the framework and refining it through adding further levels of detail to formalise the concepts of *external descriptions* and *social dependence networks*. Social Dependence Networks are a valuable source of information about the relationships within a multi-agent world. They allow agents to reason about the resources and capabilities of others in order that they may enter into a negotiation to persuade these others to assist them in completing their tasks. By formalising social dependence networks within the framework we are able to identify deficiencies in the original characterisation of the networks and the external descriptions of agents within them. We address these deficiencies, and offer a modified view which removes much of the ambiguity and presents a stronger and more consistent formal model. In reformulating these networks in this way, we also present a case study which shows how the formal framework that has been previously developed can be applied to provide an environment in which we can describe and reason about theories and models of multi-agent systems.

1 Introduction

There is a growing recognition within the multi-agent system (MAS) community of the need to harmonise the efforts being made in different sub-fields and so derive a well-defined discipline of MAS [15]. Previously, we have developed a principled theory of agency and autonomy through the provision of a formal framework which defines these concepts and specifies the relationship between them [8]. This framework was an attempt to provide strong definitions, not only to be precise about the meaning of terms which often have an ambiguous interpretation, but also to serve as an environment in which theories and models of multi-agents systems can be presented, evaluated and developed. In this paper we illustrate how this can be done by adding detail to the framework to describe *social dependence networks* (SDN) [12]. Specifying SDNs formally in this way

has allowed us to note inconsistencies and ambiguities in the work and suggest possibilities for its development as a useful mechanism for social agents.

As stated elsewhere[8], in the current work, we have adopted the specification language Z [14] for two major reasons. First, it provides modularity and abstraction and is sufficiently expressive to allow a consistent, unified and structured account of a computer system and its associated operations. Such structured specifications enable the description of systems at different levels of abstraction, with system complexity being added at successively lower levels. Second, we view our enterprise as that of building programs. Z schemas are particularly suitable in squaring the demands of formal modelling with the need for implementation by providing clear and unambiguous definitions of state and operations on state which provide a basis for program development. Thus our approach to formal specification is pragmatic — we need to be formal to be precise about the concepts we discuss, yet we want to remain directly connected to issues of implementation. Z provides just those qualities that are needed, and is increasingly being used for specifying frameworks and systems in AI [6, 3, 11] and related areas [4, 5].

The paper begins with a brief description of Social Dependence Networks [12], and continues with a very short outline of the agent hierarchy specified previously [8]. The next section extends and refines the specification of the agent hierarchy to specify SDNs formally. This allows us to evaluate and reason about these mechanisms in terms of our formal framework. We then develop and propose a refined model of dependence networks based on our notions of agency and autonomy. Lastly, we draw conclusions made from this case study of applying the framework we have developed.

2 External Descriptions and Dependence Networks

Dependence networks [12] are structures that form the basis of a computational model of Social Power Theory [1, 2]. They allow agents to reason about, and understand, the collective group of agents that make up the multi-agent world in which they operate. This section introduces dependence networks and external descriptions, data structures used to store information about other agents, based on the work reported by Sichman et al. [12].

External descriptions store information about other agents, and comprise a set of goals, actions, resources and plans for each such agent. The goals are those an agent wants to achieve, the actions are those an agent is able to perform, the resources are those over which an agent has control, and the plans are those available to the agent, but using actions and resources which are not necessarily available to the agent. This means that one agent may *depend* on another in terms of actions or resources in order for a plan to be executed.

An agent i is denoted by ag_i, and any such agent has a set of *external descriptions* of all of the other agents in the world, denoted by

$$Ext_{ag_i} \stackrel{\text{def}}{\equiv} \bigcup_{j=1}^{n} Ext_{ag_i}(ag_j)$$

where

$Ext_{ag_i}(ag_j) \stackrel{\text{def}}{\equiv} \{G_{ag_i}(ag_j), A_{ag_i}(ag_j), R_{ag_i}(ag_j), P_{ag_i}(ag_j)\}$

such that

$G_{ag_i}(ag_j)$ is the set of goals,

$A_{ag_i}(ag_j)$ is the set of actions,

$R_{ag_i}(ag_j)$ is the set of resources, and

$P_{ag_i}(ag_j)$ is the set of plans

that agent i believes agent j has.

Notice that an agent has a model of itself as well as others. The authors adopt what they call the *hypothesis of external description compatibility* which states that any two agents will have precisely the same external description of any other agent. This is stated as follows.

$$Ext_{ag_i}(ag_i) = Ext_{ag_j}(ag_i) \land Ext_{ag_i}(ag_j) = Ext_{ag_j}(ag_j)$$

Now, $P_{ag_i}(ag_j, g_k)$ represents the *set* of plans that agent i believes that agent j has in order to achieve the goal g_k. Each plan within this set is given by $p_{ag_{i_l}}$, defined below:

$$p_{ag_{i_l}}(ag_j, g_k) \stackrel{\text{def}}{\equiv} \{g_k, R(p_{ag_{i_l}}(ag_j, g_k)), I(p_{ag_{i_l}}(ag_j, g_k))\}$$

where $R(p_{ag_{i_l}})$ represents the set of resources required for the plan and $I(p_{ag_{i_l}})$ is a *sequence* of instantiated actions used in this plan. Each instantiated action within a plan is defined by the action itself and the set of resources used in the instantiation of this action:

$$i_m(p_{ag_{i_l}}(ag_j, g_k)) \stackrel{\text{def}}{\equiv} \{a_m, R_{am}(p_{ag_{i_l}}(ag_j, g_k))\}$$

Note that this makes the definition of the resources of a plan redundant: if you know the resources required by each action within a plan, then you must also know the set of all the resources required by the plan.

3 An Overview of the Framework for Agency and Autonomy

Before we can attempt to reformulate the work described above in a broader formal framework, we must first provide an overview of that framework, specified in Z. Our basic component is an *entity* [10]. An entity consists of four constituents as follows: a set of attributes, which are perceivable qualities of the entity; a set of actions, which define the basic capabilities of the agent; a set of goals, which are the goals that can be ascribed to the entity which characterise its *agency*; and a set of internal non-derivable motivations which define an entity's *autonomy*.

Entity

$attributes : \mathbb{P}\ Attribute$

$capableof : \mathbb{P}\ Action$

$goals : \mathbb{P}\ Goal$

$motivations : \mathbb{P}\ Motivation$

$attributes \neq \{\}$

Using this schema we can define certain categories of entity. In particular, an object is any entity with a non-empty set of capabilities, an agent is any object with a non-empty set of goals, and an autonomous agent is any agent with a non-empty set of motivations.

Object

$$Entity$$
$$capableof \neq \{\}$$

Agent

$$Object$$
$$goals \neq \{\}$$

AutonomousAgent

$$Agent$$
$$motivations \neq \{\}$$

A full treatment of the framework which has subsequently been refined and developed in a number of ways can be found in [10]. A model of how goals are generated by motivated agents (which we take to be the defining quality of autonomy), and subsequently adopted by non-autonomous agents has been constructed [9]. We have also shown how certain social structures — cooperation between autonomous agents, and engagements of non-autonomous agents — arise as a result of such goal generation and adoption [7].

In addition, we can easily refine components within the framework to provide, for example, a high level specification of an autonomous planning agent. Consider the next schema which describes such a refinement. A planning agent is an agent with a set of plans associated with a set of goals. Each plan in the set is a possible means of bringing about the associated goal. Some subset of these goals are ones that the agent currently desires; it might have plans for a goal it does not currently desire. We define a *complete plan* to be a *sequence* of actions. (There are, certainly, other types of plan, but this will be sufficient for the presentation of SDNs in this paper.)

$$Plan == \mathsf{seq}\, Action$$

PlanningAgent

$$Agent$$
$$plans : \mathbb{P}\, Plan$$
$$planforgoal : Goal \pfun \mathbb{P}\, Plan$$
$$goals \subseteq \mathsf{dom}\, planforgoal$$
$$\bigcup(\mathsf{ran}\, planforgoal) = plans$$

The schema states that all the plans of an agent must be associated with a goal, although it may be that the set of plans associated with a goal is the empty set. It might also be that a plan brings about more than one goal of the planning agent.

4 Dependence Networks within the Formal Framework

By using this framework, specified in Z, as a basis for reformulating the model of SDNs, we can provide a clear and unambiguous formal model, and highlight some of the potential ambiguities which arise within the existing model. We take actions, goals and plans in the original model to be actions, goals and plans in the Z framework. We take a *resource* to mean some entity — an object, agent or autonomous agent.

4.1 External Descriptions

To deal with an *external description*, we must refine our definition of a simple planning agent by including three additional variables. The first, $ownedresources$, represents the set of resources which an agent *owns*. The second, *instsreq*, models the set of resources needed to instantiate an action within a plan. The third, redundant variable, *resourcesofplan*, is included for readability and records the total set of resources required by a plan.

There are two predicates in the lower part of the schema which relate the variables in the schema as follows: stripping the set of entities away from each instantiated action gives the original plan; and the resources of a plan are the union of each set of entities associated with each action of the plan.

ExternalDescription

$$
\begin{array}{l}
PlanningAgent \\
ownedresources : \mathbb{P}\ Entity \\
instsreq : Plan \pfun (\mathsf{seq}\ (Action \times \mathbb{P}\ Entity)) \\
resourcesofplan : Plan \pfun \mathbb{P}\ Entity \\
\hline
plans = mapset\ (mapseq\ first)\ (\mathsf{ran}\ instsreq) \\
\forall\ p : Plan \bullet resourcesofplan\ p = \bigcup (\mathsf{ran}\ (mapseq\ second\ (instsreq\ p)))
\end{array}
$$

Now, since every external description of an agent is the same, we can model the formalism very simply. An agent, A, has associated with it an external description which is precisely the model that every agent (including agent A) has of agent A (according to the hypothesis of external description compatibility).

World

$$
\begin{array}{l}
extdes : Agent \pfun ExternalDescription
\end{array}
$$

Then, according to the external description of some agent, i,
$(extdes\ i).plans$ is its set of plans,
$(extdes\ i).capableof$ is its set of actions,
$(extdes\ i).ownedresources$ is its set of resources and
$(extdes\ i).goals$ is its set of goals.

Discussion There are several difficulties that become apparent when the SDN model is reformulated in this way. First of all, the distinction between a resource and an agent is not clear. For example, is a benevolent agent, who will always adopt the goals of another, a resource or an agent? It seems that some arbitrary

distinction, presumably, will have to be made. This distinction is important since the nature of a plan assumes that all of the *resources* of an action have already been identified, but the agents which could possibly perform some action have not. In this respect, a partial plan where the resources required (whatever they may be) have not yet been considered, cannot be represented.

It is also limiting in that two agents cannot perform the same action simultaneously. For example, the act of lifting a table might require two agents together performing a basic lift action. When reasoning about the multi-agent world in particular, where cooperation is likely to ensue, this seems a stringent restriction.

In addition, the notion of *ownership* in these external descriptions is not clear. We take it to mean that an agent *owns* another entity, if, for whatever the reason, that entity can be used for *any* action within its capabilities whenever the agent requires it. In other words, a resource in this formalism can be seen as a benevolent agent, adopting the goals of others' (to subsequently perform an action to satisfy those goals) whenever it can. Even then, however, there are further subtleties to consider. For example, many agents may *own* the same resource (such as a printer) but there is no mention of a shared resource. There may also be some *degree* of ownership in that my manager may always be able to use the printer before me, or some weaker notion of ownership like a desk in a shared office which can only be used by one of the occupants at a time. Clearly, a much richer notion of ownership is required. By contrast, the agent hierarchy allows us to be much clearer about the nature of these relationships which will depend on the type of entity and which goal dependence networks exist between the entities in the environment. If the entity required for some action is an object, then instantiating it as an agent is straightforward. If the entity is a non-autonomous agent, then the planning agent must reason about the nature of its agency further. (For example, can it share this agent? Can it persuade other agents that are currently engaging it to release it?) If the entity is an autonomous agent, then the planning agent will need to negotiate with the autonomous agent to persuade it to adopt its goal. More details of these social structures can be found in [9].

The *hypothesis of external description compatibility* ensures that any two agents will agree on the model of themselves and each other. Though the authors argue that there is no loss of generality, it is difficult to see how this can be so. A truly autonomous agent will have its own view of the world around it which may bear no relation to another agent's interpretation of its world. In general, we argue, any model of the world that an autonomous agent has of the world must be subjective. Certainly, a truly autonomous agent can never know the plans and goals of another agent; it may only infer them by evaluating the behaviour of the other agent. The authors themselves go some way along this path when they recognise in a later paper some of these difficulties [13], but they still require an agent to have complete (and correct) knowledge of other agents' plans, for example, which is untenable.

In Section 5, we will provide a formal specification which allows for concurrent actions in a plan, an important requirement of general purpose multi-agent

systems. We do not arbitrarily distinguish agents from resources, but instead consider agents with different functionalities. In this way we can provide a clearer and more intuitive representation of the social structures in the world since a planning agent would have to consider merely the set of *agents* that are required in a plan. Some of these agents might be invoked directly, some might be shared with some other agent, and some might be autonomous agents requiring negotiation. These ideas are developed further in [7].

4.2 Definitions of Autonomy

Using external descriptions, Sichman et al. distinguish three different forms of autonomy. An agent is *a-autonomous* for a given goal according to a set of plans of another to bring about that goal if there is a plan in this set that achieves the goal, and every action in each plan belongs to the capabilities of the agent. An agent is *r-autonomous* for a given goal according to a set of plans of another to bring about that goal if there is a plan in this set that achieves the goal, and every resource required by the plan is owned by the agent. An agent is *s-autonomous* for a given goal if it is both *a-autonomous* and *r-autonomous*.

In the following schema, we define these three classes of autonomy using of a new relation, achieves. The predicate, achieves (a, g, ps), holds precisely when an agent, a, has goal, g, and the non-empty set of plans associated with g in order to achieve it, is ps.

Thus in the schema below, the first predicate states that an agent, a, is *a-autonomous* with respect to some set of plans, ps, if and only if there is some agent, c, with goal, g, and plans, ps, to achieve g such that some plan, p in ps, contains actions all in the capabilities of a. Similar predicate are specified for *r-autonomous* and *s-autonomous*. Finally, the achieves predicate is specified as defined above.

AutonomyRelations

$$
\begin{array}{l}
World \\
\mathsf{aaut}\ \_,\ \mathsf{raut}\ \_,\ \mathsf{saut}\ \_,\ \mathsf{achieves}\ \_ : \mathbb{P}(Agent \times Goal \times \mathbb{P}\ Plan) \\
\hline
\forall\, a : Agent; g : Goal; ps : \mathbb{P}\ Plan \bullet \\
\quad \mathsf{aaut}\ (a, g, ps) \Leftrightarrow (\exists\, c : Agent \bullet \mathsf{achieves}\ (c, g, ps)) \wedge \\
\qquad\qquad (\exists\, p : ps \bullet (\mathsf{ran}\ p \subseteq (extdes\ a).capableof)) \wedge \\
\quad \mathsf{raut}\ (a, g, ps) \Leftrightarrow (\exists\, c : Agent \bullet \mathsf{achieves}\ (c, g, ps)) \wedge \\
\quad (\exists\, p : ps \bullet (extdes\ a).resourcesofplan\ p \subseteq (extdes\ a).ownedresources) \wedge \\
\quad \mathsf{saut}\ (a, g, ps) \Leftrightarrow \mathsf{aaut}\ (a, g, ps) \wedge \mathsf{raut}\ (a, g, ps) \wedge \\
\quad \mathsf{achieves}\ (a, g, ps) \Leftrightarrow g \in (extdes\ a).goals \wedge \\
\qquad\qquad (g, ps) \in (extdes\ a).planforgoal \wedge ps \neq \{\}
\end{array}
$$

In the definition of achieves, the expression $g \in (extdes\ a).goals$ states that an agent can only reason with respect to a set of plans associated with a *current* goal (i.e. one that it desires). However, in the original description, there is ambiguity about whether this must be so. The mathematical definitions make no mention of whether this proviso is part of the mechanism. If we are guided by the examples given by Sichman, however, it would appear that this proviso is, in fact, included.

Using the formal framework ensures that we are precise and unambiguous about any definitions presented within it. This is particularly important in this case, since whichever definition is used has ramifications for social dependence network categorisations. This is explored more fully in section 4.4.

According to these definitions, if agents are autonomous, then they may not *depend*, for resources or actions, on other agents. Consequently, the fact that a pocket calculator has the resources and the actions necessary for adding some numbers makes it autonomous. (By contrast, we have argued elsewhere that autonomy is not simply action or resource dependence, but involves the ability to make one's own choices, to generate goals [8].)

Nevertheless, these notions are useful to a motivated agent since in some motivational contexts, knowledge of the dependencies that exist between agents is important. They provide information as to when a goal can be satisfied (by performing the actions in a plan) without involving any other agents. Naturally, a planning agent may decide to pursue a plan that *does* involve others even if able to carry it out alone, for reasons of, for example, laziness, distribution of responsibility, efficiency, and so on.

4.3 Dependence Relations

Now we can consider the types of dependencies that exist between agents. An agent, A, *a-depends* on another agent, B, for a given goal, g, according to some set of plans of another to achieve g, if it has g as a goal, is not *a-autonomous* for g, and at least one action used in this plan is in B's capabilities. An agent, A, *r-depends* on another agent, B, for a given goal, g, according to some set of plans of another to achieve g, if it has g as a goal, is not *r-autonomous* for g, and at least one instantiation used in this plan is owned by B. An agent, A, *s-depends* on another agent, B, for a given goal, g, if it *r-depends* or *a-depends* on B.

The first predicate in the schema below states that given two agents, a and b, a goal, g, and a set of plans according to which a is not *a-autonomous* with respect to g, a *a-depends* on b for g with respect to ps, if and only if there is some agent, c, with the goal, g, and plans to achieve g, ps, such that at least one plan in ps has an action in the capabilities of agent b.

DependencyRelations

AutonomyRelations

$$\mathsf{adep}\ \_, \mathsf{rdep}\ \_, \mathsf{sdep}\ \_ : \mathbb{P}(Agent \times Agent \times Goal \times \mathbb{P}\ Plan)$$

$$\begin{aligned}
&\forall a, b : Agent; g : Goal; ps : \mathbb{P}\ Plan \mid a \neq b \wedge (g \in (extdes\ a).goals) \bullet \\
&\quad \mathsf{adep}\ (a, b, g, ps) \Leftrightarrow \neg\ \mathsf{aaut}\ (a, g, ps) \wedge \\
&\qquad (\exists c : Agent \bullet \mathsf{achieves}\ (c, g, ps) \wedge \bigcup \{p : ps \bullet \mathsf{ran}\ p\} \cap \\
&\qquad\qquad (extdes\ b).capableof \neq \{\}) \wedge \\
&\quad \mathsf{rdep}\ (a, b, g, ps) \Leftrightarrow \neg\ \mathsf{raut}\ (a, g, ps) \wedge \\
&\qquad (\exists c : Agent \bullet \mathsf{achieves}\ (c, g, ps) \wedge \\
&\qquad\quad (\exists p : ps \bullet ((extdes\ c).resourcesofplan\ p) \cap \\
&\qquad\qquad (extdes\ b).ownedresources \neq \{\})) \wedge \\
&\qquad \mathsf{sdep}\ (a, b, g, ps) \Leftrightarrow \mathsf{adep}\ (a, b, g, ps) \vee \mathsf{rdep}\ (a, b, g, ps)
\end{aligned}$$

This reformulation also highlights some difficulties. As stated earlier, at no point is it made clear whether two agents can share an action or a resource. Second, it makes little sense to say that I *a-depend* on an agent for some goal if the actions that achieve that goal are in my capabilities. Similarly, it also makes little sense to say that I *r-depend* on some agent for some resource if that resource is also owned by myself. A more intuitive definition might be

$$\mathsf{adep}\ (a, b, g, ps) \Leftrightarrow (\exists\, c : Agent \bullet \mathsf{achieves}\ (c, g, ps) \wedge (\exists\, x : \bigcup\{p : ps \bullet \mathsf{ran}\ p\} \bullet x \in (extdes\ b).capableof \wedge x \notin (extdes\ a).capableof))$$

However, even when an agent is capable of some action of which I am not capable, and which I require for some plan, it again makes little sense to say there is a dependency. It is more appropriate to say that there is a possibility of that agent being able to help in achieving a goal. There is no doubt that such reasoning will be useful in certain situations. A better notion of actual *dependency* with respect to a goal, would be if *every* plan in the set of plans required some agent's assistance. In this respect there would be a real dependency on this agent in order to achieve the goal.

$$\mathsf{adep}\ (a, b, g, ps) \Leftrightarrow (\exists\, c : Agent \bullet \mathsf{achieves}\ (c, g, ps) \wedge (\forall\, p : ps \bullet \exists\, x : \mathsf{ran}\ p \bullet x \in (extdes\ b).capableof \wedge x \notin (extdes\ a).capableof)))$$

These relations provide an agent with the structures that can be used to reason about others with a view to choosing an appropriate course of action in the context of its dependencies on others' goals, plans, resources, and so on.

4.4 Dependence Situations

Sichman proceeds to use these relations to classify distinct *dependency relations* which arise. This subsection considers these situations. We must first note that there is an ambiguity between Sichman's mathematical and textual descriptions of dependency [12]. In the mathematical description, the dependency refers to any set of plans which any agent has, but the textual description refers only to the plans of the reasoning agent. In the interpretation that follows, we adopt the more restrictive version since it is consistent with the given notions of *independence* and *unilateral dependence* discussed later, and is more intuitive in reflecting the nature of autonomous agents. (We might equally have chosen the other alternative, however.)

Consider the situation where we have two agents, A and B, where A is not *a-autonomous* for some goal, g_1, according to A's plans, ps_1, to achieve g_1. We can then recognise the following situations.

A is *independent* with respect to B for g_1 if, according to ps_1, it infers that it does not *a-depend* on B for g_1.

A is *unilaterally dependent* on B if, according to ps_1, A *a-depends* on B, but there is no goal for which B *a-depends* on A.

Two agents are *mutually dependent* if they *a-depend* on each other for the same goal g_1 according to ps_1.

If, in addition, B is not *a-autonomous* for some goal, g_2, according to A's plans to achieve g_2 then we can also write the following.

Two agents are *reciprocally dependent* if they *a-depend* on each other for two different goals, g_1 and g_2, according to two sets of plans, ps_1 and ps_2 respectively.

Thus, given two agents, A and B, where A is not *a-autonomous* for some goal, g, we define the previous dependence situations with respect to g in the schema below.

DependencySituations

$$
\begin{array}{l}
\textit{DependencyRelations} \\
\mathsf{ind}\_, \mathsf{ud}\_ : \mathbb{P}(\textit{Agent} \times \textit{Agent} \times \textit{Goal}) \\
\mathsf{md}\_ : \mathbb{P}(\textit{Agent} \times \textit{Agent} \times \textit{Goal} \times \mathbb{P}\ \textit{Plan}) \\
\mathsf{rd}\_ : \mathbb{P}(\textit{Agent} \times \textit{Agent} \times \textit{Goal} \times \textit{Goal} \times \mathbb{P}\ \textit{Plan} \times \mathbb{P}\ \textit{Plan}) \\
\hline
\forall a, b : \textit{Agent}; g_1, g_2 : \textit{Goal}; ps_1, ps_2 : \mathbb{P}\ \textit{Plan} \mid \\
\quad (a \neq b \land \mathsf{achieves}\ (a, g_1, ps_1) \land \neg\ \mathsf{aaut}\ (a, g_1, ps_1) \land g_1 \neq g_2) \bullet \\
\qquad \mathsf{ind}\ (a, b, g_1) \Leftrightarrow \neg\ \mathsf{adep}\ (a, b, g_1, ps_1) \land \\
\qquad \mathsf{ud}\ (a, b, g_1) \Leftrightarrow \mathsf{adep}\ (a, b, g_1, ps_1) \land \\
\qquad\quad \neg\ (\exists g : \textit{Goal}; ps : \mathbb{P}\ \textit{Plan} \mid \mathsf{achieves}\ (a, g, ps) \bullet \mathsf{adep}\ (b, a, g, ps)) \land \\
\qquad \mathsf{md}\ (a, b, g_1, ps_1) \Leftrightarrow \mathsf{adep}\ (a, b, g_1, ps_1) \land \mathsf{adep}\ (b, a, g_1, ps_1) \land \\
\qquad \mathsf{rd}\ (a, b, g_1, g_2, ps_1, ps_2) \Leftrightarrow \mathsf{achieves}\ (a, g_2, ps_2) \land \\
\qquad\qquad\qquad\qquad \mathsf{adep}\ (a, b, g_1, ps_1) \land \mathsf{adep}\ (b, a, g_2, ps_2)
\end{array}
$$

These definitions would be more sensible if they were based on dependencies for actions which an agent does not have. For example, if A is independent of B, it implies that there is no way that B could *help* A in performing an action. A more intuitive definition of *independence* would be that A does not *need* B.

Consider the definition of mutual dependence between A and B. It states that A and B both have a goal g_1, and according to A's plans to achieve g_1, there is some plan in which B could perform an action, and some plan (not necessarily the same plan) in which A could perform an action. What this categorisation describes is a potential for cooperation. A more intuitive definition of mutual dependence would be that every plan in the set *needs* both agents.

Reciprocal dependence occurs when, according to two sets of plans, A could help B achieve some goal $g1$, and B could help A achieve some goal $g2$. However, since the definition is with respect to A's plans, if we assume that agents can only reason with respect to sets of plans associated with a desired goal (as suggested by the authors), it must be that A currently desires *both* goals g_1 and g_2. This is very restrictive since it rules out the possibility of bargaining when A has only one goal, for example, and B has only one other goal. In such a case, both agents may then help each other by adopting the other's goals.

The mechanism is described as *social exchange*, and the authors state that "one of them will have to adopt the other's goal first in order to achieve his own one in the future". As we have seen in one interpretation of this mechanism, A must necessarily have both goals, so this scenario is inappropriate. Even then, it is too restrictive since both plans may be carried out concurrently.

4.5 Local and Mutual Belief

The dependencies described above can be *locally* or *mutually* believed. A dependence is *local* if it only exists with respect to A's plans but not with respect to B's, and it is *mutual* if it occurs with respect to both A's and B's plans.

DependencySituationsLocalandMutual

DependencySituations

$$\mathsf{lbmd}\ \_, \mathsf{mbmd}\ \_ : \mathbb{P}(Agent \times Agent \times Goal)$$
$$\mathsf{lbrd}\ \_, \mathsf{mbrd}\ \_ : \mathbb{P}(Agent \times Agent \times Goal \times Goal)$$

$$\forall a, b : Agent; g_1, g_2 : Goal; ps_1, ps_2, ps_3, ps_4 : \mathbb{P}\ Plan \mid$$
$$a \neq b \land g_1 \neq g_2 \land \mathsf{achieves}\ (a, g_1, ps_1) \land \mathsf{achieves}\ (a, g_2, ps_2) \land$$
$$\mathsf{achieves}\ (b, g_1, ps_3) \land \mathsf{achieves}\ (b, g_2, ps_4) \land \neg\ \mathsf{aaut}\ (a, g_1, ps_1) \bullet$$
$$\mathsf{lbmd}\ (a, b, g_1) \Leftrightarrow \mathsf{md}\ (a, b, g_1, ps_1) \land \neg\ \mathsf{md}\ (a, b, g_1, ps_3) \land$$
$$\mathsf{mbmd}\ (a, b, g_1) \Leftrightarrow \mathsf{md}\ (a, b, g_1, ps_1) \land \mathsf{md}\ (a, b, g_1, ps_3) \land$$
$$\mathsf{lbrd}\ (a, b, g_1, g_2) \Leftrightarrow \mathsf{rd}\ (a, b, g_1, g_2, ps_1, ps_2) \land \neg\ \mathsf{rd}\ (a, b, g_1, g_2, ps_3, ps_4) \land$$
$$\mathsf{mbrd}\ (a, b, g_1, g_2) \Leftrightarrow \mathsf{rd}\ (a, b, g_1, g_2, ps_1, ps_2) \land \mathsf{rd}\ (a, b, g_1, g_2, ps_3, ps_4)$$

More problems arise here, too. Notice, in particular, that both *local* and *mutual* belief require an analysis of both A's and B's plans, thus contradicting the following claim:

> "An agent locally believes a given dependence if he uses exclusively his *own plans* when reasoning about the others ..." [12]

In a subsequent paper, the authors drop the *hypothesis of external description compatibility* and instead concentrate on how they might detect *agency level inconsistency* resulting from two agents having different external description entries regarding each other [13]. However, it is also noticeable that their definition of mutually believed mutual dependence (MBMD) bears little relation to that proposed in the earlier paper [12]. The later definition is as follows:

> "As an example, if i infers a MBMD between himself and j for a certain goal g, this means he believes that (i) both of them have this goal and at least one plan to achieve it (ii) there is an action needed in this plan that he can perform and j can not perform (iii) there is an action needed in this plan that j can perform and he can not perform. " [13]

But the mathematical definition provides a different account of mutually believed mutual dependence: A and B both have goal g; according to A's plans A and B are not *a-autonomous* with respect to g; according to B's plans A and B are not *a-autonomous* with respect to g; there is some plan of A's which contains an action which B can do and a plan (possibly the same) which contains an action which A can do, and there is some plan of B's which contains an action which A can do and a plan (possibly the same) which contains an action which B can do.

In particular, the mathematical definition is *not* given in terms of an action not being in some agent's capabilities and, further, there is no mention of a *particular* plan within the set of plans, as required by the textual description

above. This is precisely the kind of inconsistency we hope to avoid by specifying the mechanism formally within our framework, since we are then able to provide a unified and complete account of a system.

In the same paper [13], the authors do not assume the *hypothesis of external description compatibility* but state the following.

> "For simplicity, let us consider that the plans of the agents are the same and both of them know the plans of the other."

Consequently, there can be agent level consistency in terms of what agents believe about the capabilities, resources and goals of each other, assuming they *know* the plans of every agent. This is evidently very useful, even though severely limiting, and further work explores agent reasoning about this class of problem.

5 A New Proposal

In this section we briefly describe a new proposal for external descriptions which allows for true autonomy (in the sense that an agent can never know the goals, actions and plans of another), simultaneous actions, active and non-active goals and plans, partial plans and a richer understanding of the social relationship between the entities in the world. In this respect, we can re-formulate the useful work of social dependence networks within a well-defined formal framework for agency and autonomy.

Consider fixing a screw into a block of wood. According to our hierarchical framework, this may require two agents: a screwdriver, and someone with the ability to use the screwdriver, both agents performing an action simultaneously. Every action in a plan must either be associated with the entity intended to to perform the action, or be associated with no entity if the entity involved in its instantiation has not yet been chosen.

In the following example, we illustrate our new representation of a plan for use in external descriptions. It consists of an action which I will perform, followed by two actions performed by two entities simultaneously, followed by three actions performed simultaneously by three entities (including one by me), followed by some action to be performed by an as yet unknown entity.

$$\langle \{(a_1, \{me\})\},\ \{(a_{2_1}, \{entity1\}), (a_{2_2}, \{entity2\})\},$$
$$\{(a_{3_1}, \{me\}), (a_{3_2}, \{entity2\}), (a_{3_3}, \{entity4\})\},\ \{(a_4, \{\})\}\rangle$$

A plan, therefore, has the following new type.

$$NewPlan == \mathsf{seq}\,(\mathbb{P}(Action \times (\mathbb{P}\ Entity)))$$

The schema below specifies an external description which includes the *current* goals of an agent. Associated with each such goal are a set of plans which together form the set of current plans. In addition, we also define the set of *all* goals of an agent — some of which are currently desired and some of which are not — and the corresponding set of *all* plans. (Note that in this respect we can be clear about categorisations based on certain types of plans, goals and so on.)

In addition, we define three useful but redundant variables which for each plan return the set of action-entity pairs, actions, and entities involved in the plan, respectively. The last predicate ensures that given an action-entity pair in a plan where the entity is defined, the action must be in the capabilities of the entity.

$$
\begin{array}{|l}
\hline
NewExternalDescription \\
\hline
Agent \\
plans : \mathbb{P}\ NewPlan \\
allgoals : \mathbb{P}\ Goal \\
allplans : \mathbb{P}\ NewPlan \\
planforgoal : Goal \pfun \mathbb{P}\ NewPlan \\
actionsofplan : NewPlan \fun \mathbb{P}\ Action \\
entitiesofplan : NewPlan \fun \mathbb{P}\ Entity \\
actionentities : NewPlan \fun \mathbb{P}(Action \times \mathbb{P}\ Entity) \\
\hline
goals \subseteq \mathsf{dom}\ planforgoal \\
plans = \{p : NewPlan; g : Goal \mid g \in goals \land p \in planforgoal\, g \bullet p\} \\
allgoals = \mathsf{dom}\ planforgoal \\
allplans = \bigcup(\mathsf{ran}\ planforgoal) \\
\forall p : NewPlan \bullet actionsofplan\, p = \{aes : actionentities\, p \bullet first\ aes\} \land \\
\qquad entitiesofplan\, p = \bigcup\{aes : actionentities\, p \bullet second\ aes\} \land \\
\qquad actionentities\, p = \bigcup(\mathsf{ran}\ p) \land (\forall aes : actionentities\, p; e : Entity \mid \\
\qquad\qquad second\ aes = \{e\} \bullet first\ aes \in e.capableof) \\
\hline
\end{array}
$$

Further work can then progress using the definitions given above to provide new social dependence network categorisations based on the original formalisms. As a small example, we can say that an agent is *t-autonomous* with respect to a plan if all the actions the plan contains are within its own capabilities. Essentially:

$$\mathsf{taut}\ (agent, plan) \Leftrightarrow actionsofplan\ plan \subseteq (agent.capableof)$$

6 Conclusions

Social Dependence Networks are a valuable source of information about the relationships within a multi-agent world, and provide the necessary structure that can be exploited by agents in order to function effectively. They allow agents to reason about resources and capabilities of others in order that they may enter into a negotiation to persuade these others to assist them in completing their tasks. This paper has described the work of Sichman et al. in developing computational models of dependence networks and has reformulated it in another, formal, framework. By reformulating it in these terms, we have been able to identify deficiencies in the original characterisation of dependence networks and the external descriptions of agents within the networks. We have addressed these deficiencies, and offer a modified view of external descriptions which removes much of the ambiguity and presents a stronger and more consistent, formal model which can easily be extended to define social dependence networks.

In reformulating dependence networks in this way, we have also presented a case study which shows how the formal framework that we have previously developed can be applied to provide an environment in which we can describe and

reason about theories and models of MAS. Moreover, we have highlighted inconsistencies and ambiguities, and outlined how such models may be incorporated within our framework.

Acknowledgements Many thanks to Rafael Bordini for detailed comments and suggestions on an earlier version of this paper.

References

1. C. Castelfranchi. Social power. In Y. Demazeau and J. P. Muller, editors, *Decentralized Artificial Intelligence*, pages 49–62. Elsevier North Holland, 1990.
2. C. Castelfranchi, M. Miceli, and A. Cesta. Dependence relations among autonomous agents. In E. Werner and Y. Demazeau, editors, *Decentralized Artificial Intelligence*, pages 215–231. Elsevier North Holland, 1992.
3. I. Craig. *Formal Specification of Advanced AI Architectures.* Ellis Horwood, 1991.
4. M d'Inverno and J. Crowcroft. Design, specification and implementation of an interactive conferencing system. In *Proceedings of IEEE Infocom, Miami, USA. Published IEEE*, 1991.
5. M. d'Inverno and M. Priestley. Structuring a Z specification to provide a unifying framework for hypertext systems. In J. P. Bowen and M. G. Hinchey, editors, *ZUM'95: 9th International Conference of Z Users, Lecture Notes in Computer Science*, pages 83–102, Heidelberg, 1995. Springer-Verlag.
6. R. Goodwin. Formalizing properties of agents. Technical Report CMU-CS-93-159, Carnegie-Mellon University, 1993.
7. M. Luck and M. d'Inverno. Engagement and cooperation in motivated agent modelling. In *Proceedings of the first Australian DAI Workshop.* Springer Verlag, 1995.
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*, pages 254–260. AAAI Press / MIT Press, 1995.
9. M. Luck and M. d'Inverno. Goal generation and adoption in hierarchical agent models. In *AI95: Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence.* World Scientific, 1995.
10. M. Luck and M. d'Inverno. Structuring a Z specification to provide a formal framework for autonomous agent systems. In J. P. Bowen and M. G. Hinchey, editors, *ZUM'95: 9th International Conference of Z Users, Lecture Notes in Computer Science*, pages 48–62. Springer-Verlag, 1995.
11. B. G. Milnes. A specification of the Soar architecture in Z. Technical Report CMU-CS-92-169, School of Computer Science, Carnegie Mellon University, 1992.
12. J. S. Sichman, Y. Demazeau, R. Conte, and C. Castelfranchi. A social reasoning mechanism based on dependence networks. In *ECAI 94. 11th European Conference on Artificial Intelligence*, pages 188–192. John Wiley and Sons, 1994.
13. J. S. Sichman and Yves Demazeau. Exploiting social reasoning to deal with agency level inconsistency. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 352–359. AAAI Press / MIT Press, 1995.
14. J. M. Spivey. *The Z Notation.* Prentice Hall, Hemel Hempstead, 2nd edition, 1992.
15. M. J. Wooldridge and N. R. Jennings. Applying agent technology. *Journal of Applied Artificial Intelligence, special issue on Intelligent Agents and Multi-Agent Systems*, To appear, 1995.

A Introduction to Z

The formal specification language Z is based on set theory, first order logic and predicate calculus. It extends the use of these languages by allowing an additional mathematical type known as the *schema type*. Z schemas have two parts: the upper, declarative, part which declares variables and their types, and the lower, predicate, part which relates and constrains those variables. The type of any schema can be considered as the cartesian product of the types of each of its variables, without any notion of order, but constrained by predicates. Modularity is facilitated in Z by allowing schemas to be included within other schemas. We can select a state variable, *var*, of a schema, *schema*, by writing *schema.var*.

To introduce a type in Z, where we wish to abstract away from the actual content of elements of the type, we use the notion of a *given set*. We may write *NODE* to represent the set of all nodes. If we wish to state that a variable takes on some set of values or an ordered pair of values we write $x : \mathbb{P}\, NODE$; $x : NODE \times NODE$, respectively. The generic functions *first* and *second* return the first element and second element of any ordered pair, respectively.

A *relation* type expresses some relationship between two existing types, known as the *source* and *target* type. When no element from the source type can be related to two or more elements from the target type, the relation is a *function*. A *total* function ($\longrightarrow$) is one where every element in the source set is related, while a *partial* function ($\pfun$) is where not every element in the source is related. Finally, A sequence (seq)is a special type of function where the domain is the contiguous set of numbers from 1 up to the number of elements in the sequence. For example, consider the following function which defines a relation between nodes: $Rel = \{(node1, node2), (node2, node3), (node3, node2), (node4, node4)\}$.

The *domain* (dom) of a relation or function is those elements in the source set which are related, and the *range* (ran) is those elements in the target set which are related. In this case $\mathsf{dom}\, Rel = \{node1, node2, node3, node4\}$ and $\mathsf{ran}\, Rel = \{node2, node3, node4\}$.

Sets of elements can be defined using set comprehension. For example, the following expression denotes the set of squares of natural numbers greater than 10 $\{x : \mathbb{N} \mid x > 10 \bullet x * x\}$. The way to write down predicates in Z is non-standard. To state that, say, any number greater than 10 has a square greater than 100, we write: $\forall\, n : \mathbb{N} \mid n > 10 \bullet n * n > 100$.

Lastly, we make use of *mapseq*, which takes a function and a sequence and applies the function to each element of the sequence and *mapset*, which takes a function and a set and applies the function to each element of the set.

$$
\begin{array}{|l}
\hline
[X, Y] \\
\hline
mapseq : (X \pfun Y) \pfun \mathsf{seq}\, X \pfun \mathsf{seq}\, Y \\
mapset : (X \pfun Y) \pfun \mathbb{P}\, X \pfun \mathbb{P}\, Y \\
\hline
\forall\, seqs : \mathsf{seq}\, X; xs : \mathbb{P}\, X; fun : X \pfun Y \bullet \\
\quad mapseq\, fun\, seqs = \{n : \mathbb{N} \mid n \in 1 \mathinner{..} \# seqs \bullet (n, fun\, (seqs\, n))\} \land \\
\quad mapset\, fun\, xs = \{x : X \mid x \in xs \bullet fun\, x\} \\
\hline
\end{array}
$$

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