A Framework for Norm-Based Inter-Agent Dependence

Fabiola López y López[*] Michael Luck[*] Mark d'Inverno[†]

[*] Department of Electronics and Computer Science, University of Southampton
Southampton, SO17 1BJ, UK {flyl00r,mml}@ecs.soton.ac.uk
[†] Cavendish School of Computer Science, University of Westminster, London, UK

Abstract. A significant class of agent architectures designed for operation in a multi-agent world choose their next actions or plans based on a limited analysis; they ignore considerations of the multi-agent world they inhabit, and the inter-agent relationship that might influence their choice of action. This paper addresses that problem, and focuses on the integration of BDI-like agent architectures with computational notions of norms and dependence to arrive at a computational multi-agent organisation model. We describe initial work in pursuit of that goal.

1 Introduction

Much research in the field of intelligent agents and multi-agent systems has been concerned with the development of computational agent architectures that are designed to solve particular problems or offer general agent-based solutions. Progress in this area has been significant, and there is now a range of architectures to suit a vast array of problem scenarios. Similarly, at the organisational level, a rich set of abstractions, which includes such concerns as norms and dependencies, has been identified and studied. However, these two important and complementary aspects of the field of agent-based systems sit some distance from each other, without adequate integration, particularly at the level of computational architecture. In particular, a significant class of agent architectures that are designed for operation in a multi-agent world choose their next actions or plans based on a very limited analysis (if any analysis at all) of these organisational issues. That is to say that although the intention is for multi-agent operation, these systems ignore considerations of the multi-agent world they inhabit, and the inter-agent relationship that might legitimately influence their choice of action. We target this problem, and focus on the integration of BDI-like agent architectures with computational notions of norms and dependence to arrive at a computational multi-agent organisation model. In this paper we describe initial work in pursuit of that goal.

In the next sections we present a particular kind of agent based on the BDI model of agents. Then, we describe important notions of power and dependence, and finally consider norms before summarising. The key point is that although each section focuses on different aspects, they are integrated both in the mathematical description and in the textual explanation, offering a unified account of architecture, power, and norms.

2 An Abstract BDI Agent

Perhaps the most common and most widely cited deliberative agent architecture is the BDI architecture in its many forms and guises. Based around intuitive folk psychology notions of belief, desire and intention, BDI has become almost a default architecture, or a base architecture from which to consider more advanced or more sophisticated aspects not addressed in the more simple versions. Our own view is that for these reasons and for others relating to applicability and generality, BDI is therefore an appropriate place to start. Yet the different versions of BDI architecture, even those from the same tradition, (e.g. [2, 13]) all have rather different specific architectural bases. As a means of building up a general model of power and norms, we start by offering an *abstract* BDI architecture, which does not fit with any specific model, but is inspired by previous work on dMARS [10] and AgentSpeak(L) [11]. It includes the salient features, but omits irrelevant details. The model itself follows this previous work and consequently we do not provide an extensive description. In what follows, we use the Z specification language to construct a formal model of the BDI agents and of inter-agent dependence and norms. Z is based on set-theory and first order logic, with full details available in [17]. For reasons of brevity, however, we will not elaborate the use of Z further.

Agent and Agent State In general, a deliberative agent is essentially defined by its plan library, which contains all the *recipes* for action the agent knows about, and its capabilities or specific actions.

$$\begin{array}{|l}\hline \mathit{DeliberativeAgent} \\ \mathit{planlibrary} : \mathbb{P}\, \mathit{Plan} \\ \mathit{capabilities} : \mathbb{P}\, \mathit{ExternalAction} \\ \hline \end{array}$$

At run-time, an agent will also have a set of events, beliefs, goals and intentions that are generated in response to the environment through the reasoning and action control cycle of the agent. These components define the agent acting in the world, and are the key artifacts that are manipulated to ensure effective behaviour. We consider each in turn.

$$\begin{array}{|l}\hline \mathit{Agent} \\ \mathit{DeliberativeAgent} \\ \mathit{events} : \mathbb{P}\, \mathit{Event};\ \mathit{beliefs} : \mathbb{P}\, \mathit{Belief} \\ \mathit{goals} : \mathbb{P}\, \mathit{Goal};\ \mathit{intentions} : \mathbb{P}\, \mathit{Intention} \\ \hline \end{array}$$

We start, however, by saying nothing about the nature of external actions or beliefs, which we parachute into our specification as given sets. Effectively, they are primitives (although beliefs are typically implemented as first-order predicates, and in a more sophisticated and detailed analysis we might equally do the same). External actions are simply those actions that change the state of the world, and beliefs are the representation of information about the world. The alternative to an external action is an internal one, which either removes or adds a belief to or from the agent's set of beliefs. Also key to many BDI architectures is the notion of an event, which is simply something that happens in the world that can be perceived, and it is either a goal or a belief. (It may be a new goal or belief or the removal of a goal or belief.) Finally, goals, which direct behaviour, and are desirable states to be achieved, are defined as either *achieving* some belief (which amounts to making some predicate true in general) or *querying* a belief (which amounts to testing if a predicate is true).

$$[\mathit{ExternalAction}, \mathit{Belief}]$$
$$\mathit{InternalAction} ::= \mathit{add}\langle\langle \mathit{Belief} \rangle\rangle \mid \mathit{remove}\langle\langle \mathit{Belief} \rangle\rangle$$
$$\mathit{Event} ::= \mathit{goal}\langle\langle \mathit{Goal} \rangle\rangle \mid \mathit{belief}\langle\langle \mathit{Belief} \rangle\rangle$$
$$\mathit{Goal} ::= \mathit{query}\langle\langle \mathit{Belief} \rangle\rangle \mid \mathit{achieve}\langle\langle \mathit{Belief} \rangle\rangle$$

Agent Plans Now, plans are the most sophisticated data structure in deliberative agents, and effectively specify the course of action to be taken by the agent in achieving goals.

They consist of several key components. First, the plan *body*, typically comprising a sequence of actions, encapsulates the agent's "know-how". This is the most important part of a plan, but we also need to specify the conditions under which a plan can be initiated or continue, and what happens when a plan fails or succeeds. To start, a plan requires a *triggering* event, which is simply an event that causes the plan to fire. Once this occurs, its applicability to the current situation must be determined through the *context*, which is a set of beliefs that are entailed by the current beliefs of the agent. At this point the plan becomes active (that is, it is relevant and possible). For the plan to continue, we also need to include the notion of *maintenance* conditions which must hold for the plan to be executable. Finally, we need internal actions to specify what happens when a plan *fails* or *succeeds*.

$Plan$

$trigger : Event;\ context : \mathbb{P}\,Belief$
$body : Body;\ maintenance : \mathbb{P}\,Belief$
$fail, succeed : \mathbb{P}\,InternalAction$

Now, although the most general type of a plan is a tree, with branches and choice points, such a structure makes the formal specification of dependencies unnecessarily complicated in what follows. We therefore specify a body of a plan to be a sequence of branches, which are either actions or goals, with actions being internal or external. There is no loss of generality here, as a tree can be considered to be a set of sequence of branches where each sequence is a possible path from the root to a leaf node.

$Body == \mathrm{seq}\,Branch$
$Branch ::= action\langle\langle Action\rangle\rangle \mid goal\langle\langle Goal\rangle\rangle$
$Action ::= internal\langle\langle InternalAction\rangle\rangle \mid external\langle\langle ExternalAction\rangle\rangle$

Plan Instances A *plan instance* is an active plan that has been instantiated as either an addition to an existing intention or as a new intention. This plan instance represents a *copy* of the original plan that now serves as a *mental attitude* directing behaviour as opposed to a *recipe* for behaviour. The distinction between plans as recipes and plans as mental attitudes is very important in the study of deliberative agents and in this specification we distinguish between calling the former plans and the latter plan instances. The only distinction we make in our abstract BDI model is that every plan instance has a status which determines whether the plan instance is currently executing.

$PlanInstance$

$Plan;\ status : Status$

$Status ::= active \mid suspended$

Now, once a goal is selected from those the agent desires, a plan is selected to achieve that goal; this plan forms the basis of the *intention* that will direct the future behaviour of the agent. In short, an intention is a sequence (or stack) of plan instances.

$Intention == \mathrm{seq}\,PlanInstance$

In order to select a plan as an intention in response to an event, we must first generate those plans that are relevant (those that are triggered by an event), and then the subset of these that are applicable with the current beliefs (those whose contexts are a subset of the current beliefs). The variables, *relevantplans* and *activeplans* associate each event with the current set of relevant and applicable plans. The *executing* plans of an agents are those plan instances that are at the top of each intention stack.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{AgentState}} \\
\hline
\textit{Agent} \\
\textit{genrelevantplans} : \textit{Event} \rightarrow (\mathbb{P}\,\textit{Plan}) \rightarrow (\mathbb{P}\,\textit{Plan}) \\
\textit{genapplicableplans} : (\mathbb{P}\,\textit{Plan}) \nrightarrow (\mathbb{P}\,\textit{Belief}) \rightarrow (\mathbb{P}\,\textit{Plan}) \\
\textit{relevantplans}, \textit{activeplans} : \textit{Event} \nrightarrow \mathbb{P}\,\textit{Plan} \\
\textit{allrelevantplans}, \textit{allactiveplans} : \mathbb{P}\,\textit{Plan} \\
\textit{executingplans} : \mathbb{P}\,\textit{PlanInstance} \\
\hline
(\mathrm{dom}\,\textit{relevantplans} \cup \mathrm{dom}\,\textit{activeplans}) \subseteq \textit{events} \\
\forall\, e : \textit{events} \bullet \textit{relevantplans}\ e = \textit{genrelevantplans}\ e\ \textit{planlibrary} \land \\
\quad \textit{activeplans}\ e = \textit{genapplicableplans}\ (\textit{relevantplans}\ e)\ \textit{beliefs} \\
\textit{allrelevantplans} = \bigcup\{e : \textit{events} \bullet \textit{relevantplans}\ e\} \\
\textit{allactiveplans} = \bigcup\{e : \textit{events} \bullet \textit{activeplans}\ e\} \\
\textit{executingplans} = \{i : \textit{intentions} \bullet \textit{first}\ i\} \\
\hline
\end{array}
$$

3 Dependence

The architecture described above is a generalised version of the standard BDI model that underlies numerous agent systems. It decides on its course of action by identifying relevant and applicable plans to instantiate as intentions for execution. The problem with this model as it stands is twofold: first, there may be multiple plans in the plan library that may be both relevant and applicable, and the model says nothing about how to choose between them; second, the model does not include any consideration of dependencies or other relationship between agents that may impact on the suitability of plans. In this section, we describe a simplified *power model* based on the work of Castelfranchi and others on social power theory and social dependence networks (SDN), that can be used as a basis for making such judgements between plans.

Agent Models Before we can consider dependence notions we first briefly turn our attention to the models agents must have of each other in order to be socially adequate in general. In order that agents can take advantage of the capabilities of others, they generally need models of them; moreover, for agents to investigate the various dependencies that each agent may have on others, a model of their goals, beliefs and intentions will also be required. Using the technique from the SMART agent framework [15, 12], agent models are constructed at this level of abstraction by application of the existing models for describing deliberative agent architectures.

$$\textit{AgentModel} == \textit{AgentState}$$

In general, agents have a model of every other agent and of themselves.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{AgentModels}} \\
\hline
\textit{AgentState} \\
\textit{models} : \mathbb{P}\,\textit{AgentModel}; \qquad \textit{self} : \textit{AgentModel} \\
\hline
\textit{self} \in \textit{models} \\
\hline
\end{array}
$$

Now that we have defined deliberative agents and their models, it is possible to investigate the notion of *dependence* between agents.

Action Dependence Castelfranchi and colleagues developed the *social power theory* [3, 5], which claims that by comparing the dependence of one agent on others with their dependence on it, social behaviour results. According to Castelfranchi et al. [5], an agent depends on another to perform an action useful for achieving one of its goals, if it is unable to perform the action while the other agent can. Based on this notion we formalise a more generic relationship among agents in our framework, by first defining a dependence relationship between agents with respect to an action in terms of the capabilities of both.

$$
\begin{array}{l}
\mathit{actionDepend\_} : \mathbb{P}(\mathit{AgentState} \times \mathit{AgentState} \times \mathit{ExternalAction}) \\
\hline
\forall\, ag_1, ag_2 : \mathit{AgentState};\ ac : \mathit{ExternalAction} \bullet \\
\quad \mathit{actionDepend}(ag_1, ag_2, ac) \Leftrightarrow \\
\qquad ac \notin ag_1.\mathit{capabilities} \land ac \in ag_2.\mathit{capabilities}
\end{array}
$$

Then, a dependence relation caused by an action exists if there is a current intention of the agent which can only be performed by another agent. This means that the action is included in the instantiated plan that comprises the intention as the executing plan. Also notice that the analysis is now from the perspective of the planning agent and must therefore be with respect to the model it has of other agents.

$$
\begin{array}{l}
\mathit{excPlanActionDepend\_} : \mathbb{P}(\mathit{AgentModels} \times \mathit{AgentState} \times \mathit{ExternalAction}) \\
\hline
\forall\, ag_1 : \mathit{AgentState};\ ag_2 : \mathit{AgentModels};\ ac : \mathit{ExternalAction} \bullet \\
\quad \mathit{excPlanActionDepend}(ag_1, ag_2, ac) \Leftrightarrow \\
\qquad ag_2 \in ag_1.\mathit{models} \land (\exists\, p : ag_1.\mathit{executingplans} \bullet \\
\qquad\quad ac \in (\operatorname{ran} p.\mathit{body}) \land \mathit{actionDepend}(ag_1, ag_2, ac))
\end{array}
$$

Plan Dependence As can be seen from the above and in line with the *social dependence networks* originally proposed by Sichman et al. [16], in general, an agent needs to reason about dependencies with respect to its *plans*, and we extend our definitions accordingly. Specifically, we can define several categories of plan dependence irrespective of whether that plan is active, relevant or executing. First, we consider the situation where the plan of one agent includes actions that are not in its capabilities so that it relies on the capabilities of another.

$$
\begin{array}{l}
\mathit{planDepend\_} : \mathbb{P}(\mathit{AgentState} \times \mathit{AgentState} \times \mathit{Plan}) \\
\hline
\forall\, ag_1, ag_2 : \mathit{AgentState};\ p : \mathit{Plan} \mid p \in ag_1.\mathit{planlibrary} \land ag_1 \neq ag_2 \bullet \\
\qquad \mathit{planDepend}(ag_1, ag_2, p) \Leftrightarrow (\exists\, ac : \mathit{ExternalAction} \bullet \\
\qquad\qquad ac \in (\operatorname{ran} p.\mathit{body}) \land \mathit{actionDepend}(ag_1, ag_2, ac))
\end{array}
$$

Further categories can also be defined easily. For example, reciprocal dependence among plans describes the situation where there is a plan in the library of an agent ag_1 that needs an action that can be achieved by ag_2, and a plan in the library of ag_2 that needs an action requiring an action of ag_1.

$$
\begin{array}{l}
\mathit{recPlanDepend\_} : \mathbb{P}(\mathit{AgentState} \times \mathit{AgentState} \times \mathit{Plan} \times \mathit{Plan}) \\
\hline
\forall\, ag_1, ag_2 : \mathit{AgentState};\ p1, p2 : \mathit{Plan} \mid \\
\quad p1 \in ag_1.\mathit{planlibrary} \land p2 \in ag_2.\mathit{planlibrary} \land ag_1 \neq ag_2 \bullet \\
\qquad \mathit{recPlanDepend}(ag_1, ag_2, p1, p2) \Leftrightarrow \\
\qquad\quad \mathit{planDepend}(ag_1, ag_2, p1) \land \mathit{planDepend}(ag_2, ag_1, p2)
\end{array}
$$

Clearly, the problem of dependence is most acute in a multi-agent system when there is *only one agent* with the required capability causing such a dependence. This means that an agent really does have power over another if ever the plan becomes (in increasing degree) relevant, applicable and chosen for execution.

$$
\begin{array}{|l}
\hline
\mathit{MultiAgentSystem} \\
\mathit{agents} : \mathbb{P}\,\mathit{AgentState} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\mathit{specPlanDepend}\_ : \mathbb{P}(\mathit{AgentState} \times \mathit{AgentState} \times \mathit{Plan}) \\
\hline
\forall\, ag_1, ag_2 : \mathit{AgentState};\ \mathit{mas} : \mathit{MultiAgentSystem};\ p : \mathit{Plan} \mid \\
\quad (\{ag_1, ag_2\} \subseteq \mathit{mas.agents}) \bullet \\
\quad\quad \mathit{specPlanDepend}(ag_1, ag_2, p) \Leftrightarrow \\
\quad\quad\quad \mathit{planDepend}(ag_1, ag_2, p) \wedge (\nexists\, ag_3 : \mathit{mas.agents} \bullet \\
\quad\quad\quad\quad (ag_3 \neq ag_2) \wedge \mathit{planDepend}(ag_1, ag_3, p))
\end{array}
$$

In summary, an agent needs to reason with respect to events (which include the goals of the agent) in order to make sensible judgements about its choice of plans to execute. More precisely, it must consider the relevant plans associated with an event, then the applicable plans, and finally the executing plans or intentions. In multi-agent systems agents also need to reason about the dependencies of plans at all stages in order to consider competing alternatives. In particular, if agent A depends on B for a relevant plan but B depends on A for an executing plan, then A can be said to have more power than B. *Equal* power occurs when two agents both have *executing* plans that depend on each other. Notice that this analysis is now based on the executing plans of agents and the models that agents have constructed about each other.

$$
\begin{array}{|l}
\hline
\mathit{ExecutingPlanReciprocation} \\
\mathit{AgentModels} \\
\mathit{reciprocate}\_ : \mathbb{P}(\mathit{Plan} \times \mathit{Agent} \times \mathit{Plan}) \\
\hline
\forall\, ag : \mathit{models};\ p_1, p_2 : \mathit{Plan} \bullet \\
\quad \mathit{reciprocate}\ (ag, p_1, p_2) \Leftrightarrow p_1 \in \mathit{executingplans} \wedge \\
\quad\quad p_2 \in \mathit{ag.executingplans} \wedge \mathit{recPlanDepend}\ (\mathit{self}, ag, p_1, p_2) \\
\hline
\end{array}
$$

Goal Dependence Not only may agents depend on the actions of others to achieve their intentions, it may also be that they rely on each other to achieve their *goals*. This occurs when an agent recognises that the body of a plan contains a step that is a goal for which it cannot currently generate any relevant or applicable plans. In this situation, the agent must either find another agent, which is sufficiently capable of generating appropriate active plans, to adopt its goal, or recognise that another agent currently has this goal so that the intention can be completed. (Note that we do not address the problem of how an agent finds another to adopt its goal, but are merely concerned with the dependence relations in this paper.) We formalise the first of these notions here by using the predicates *canPlan* and *noPlan* which determine whether an agent can generate any active plans for a particular goal.

$$
\begin{array}{|l}
\mathit{canPlan}\_, \mathit{noPlan}\_ : \mathbb{P}(\mathit{AgentState} \times \mathit{Goal}) \\
\hline
\forall\, ag : \mathit{AgentState};\ g : \mathit{Goal} \mid g \in \mathit{ag.goals} \bullet \\
\quad \mathit{canPlan}(ag, g) \Leftrightarrow \mathit{ag.activeplans}(\mathit{goal}\ g) \neq \{\} \wedge \\
\quad \mathit{noPlan}(ag, g) \Leftrightarrow \mathit{ag.activeplans}(\mathit{goal}\ g) = \{\}
\end{array}
$$

Then we can define the notion of an agent depending on the planning capabilities of another agent, based on the models that the first has of the second in order to achieve some executing plan, which is part of some current intention.

$$
\begin{array}{|l}
goalDepend\_ : \mathbb{P}(AgentState \times AgentState \times Plan) \\
\hline
\forall\, ag_1, ag_2 : AgentState;\ p : Plan \mid ag_2 \in ag_1.models \bullet \\
\qquad goalDepend(ag_1, ag_2, p) \Leftrightarrow \\
\qquad\qquad (\exists\, g : Goal \mid goal(achieve\ g) \in (\operatorname{ran} p.body) \bullet \\
\qquad\qquad\qquad noPlan(ag_1, g) \land canPlan(ag_2, g))
\end{array}
$$

4 Norms

Now, agent societies of all kinds typically include certain constraints on behaviour that are known as *norms*. Norms have been seen as a natural way of improving coordination and cooperation in multi-agent systems because they can restrict, and make more predictable, the behaviour of agents. Researchers use different forms to represent them, such as commitments as in [14], as mental attitudes [6], or as obligations, authorisations and conventions [8]. In addition, several different ways to reason about norms have been proposed [4, 7, 9, 1]. Although some of the main ideas of these research efforts have been taken into account in our model, we have chosen to start with our base in order to be able to integrate our notions with the architecture described earlier. Nevertheless, we agree with the basic underlying principle of the majority of such work that norms are mental attitudes directed at controlling the behaviour of agents; that is, norms are pro-attitudes.

Norms are simply rules in a society that must be complied with by one or more agents. In this section, we introduce and specify some norms that are common in agent societies, and show how they fit with the architecture described earlier. (The problems of how to use them in reasoning of them is left for future work). Specifically, an agent may have access to certain norms which are represented as data structures relating to social rules. These may be common to all agents (such as with a mutually understood social law) or only available to some. The most general structure of a norm we can represent contains the following specific components.

$$
\begin{array}{|l}
\hline
\quad NormStructure \\
normativegoal : Goal;\ context, exceptions : \mathbb{P}\, Belief; \\
addressees, beneficiaries, defenders : \mathbb{P}\, Agent \\
rewards, punishments : \mathbb{P}\, Goal \\
\hline
\quad context \neq \{\};\ \ addressees \neq \{\} \\
\hline
\end{array}
$$

First, there is a $normativegoal$, which is the goal that the relevant group of agents must seek to achieve. Then, the $context$ of a norm refers to the set of beliefs that must be true for the norm to be active, unless agents are in $exception$ states. Now, each norm applies to a certain set of agents in the world — it may be all agents, or it may only be a limited subset. In either case, however, the $addressee$ agents who should obey the norm must be specified. Typically, there is also a set of $benefeciary$ agents, which are those agents who might specifically gain from the addresses agents adopting the normative goal. Norms may be monitored by agents referred to as $defenders$, who can adopt the goal of initiating certain punishments if the norm is not complied with. Finally, it may be that any beneficiary agent is expected to reciprocate by adopting a goal from some pre-defined set of $rewards$.

In a society of autonomous agents, many kinds of norms can be found. However, we consider only obligations and social commitments due to space constraints.

Obligations and Social Commitments *Obligations* may be defined as the kind of norm which compels a group of agents to do something for another group of agents. Generally, obligations are adopted once agents become members of a society, and persist as along as they stay in the society. Their main characteristic is that agents are punished if the normative goal is not satisfied. Normative goals are satisfied either in order to avoid punishments or just because agents have a sense of high social responsibility.

$Obligation$

$NormStructure$

$punishments \neq \{\}; \quad defenders \neq \{\}$

The second category of norm is *social commitments*, which are norms created through agreements or negotiations between two or more agents. Indeed, they are part of a deal between two groups, the set of addressee agents (which could be only one) who are obliged to achieve a goal, and the set of beneficiary agents (who sometimes also become defender agents and are consequently responsible for monitoring the fulfillment of the social commitment). Generally, once the normative goal is achieved, a reward is claimed. In contrast to obligations, social commitments are just temporal, and may disappear once the committed duty becomes satisfied.

$SocialCommitment$

$NormStructure$

$\#beneficiaries = 1; \quad rewards \neq \{\}$

Further categories of norms, such as *prohibitions*, which may be defined as states to be avoided for a group of agents, or *social codes*, which suggest the inclusion of extra actions in plans, may also be defined similarly, but we will not consider them further here due to lack of space. Now, without eliminating the possibility of having other categories, we state that a norm can be an obligation or a social commitment.

$$Norm ::= obligation\langle\langle Obligation\rangle\rangle \mid socialcommitment\langle\langle SocialCommitment\rangle\rangle$$

In general, a *normative agent* has all the components we have previously described including goals, beliefs, intentions, relevant and applicable plans, models of the goals, beliefs etc., of other agents and a set of norms which includes the obligations and commitments that it has adopted.

$NormativeAgent$

$AgentState$

$norms : \mathbb{P}\, Norm;$

Social Powers The introduction of norms into this framework opens up new possibilities for considering the power (and dependence) an agent may have over other agents, not because of its sophisticated planning capabilities, but because of the social responsibilities arising from norms. Thus, societies with norms can result in situations in which, for example, more talented agents may help agents that are relatively less able. For example, an agent in a particular society may adopt a goal to help a certain beneficiary rather than being obliged to leave the group. In this case, the beneficiary has power over this agent with respect to the normative goal.

We can formalise these notions in the next schema. It states that an agent has *power* through an *obligation* over another agent with respect to a goal, if there exists some active social obligation of the second agent in which the first is one of the beneficiaries.

$$\begin{array}{l} powerbyobligation\_ : \mathbb{P}(NormativeAgent \times NormativeAgent \times Goal) \\ \hline \forall\, ag_1, ag_2 : NormativeAgent;\; g : Goal \bullet \\ \quad powerbyobligation(ag_1, ag_2, g) \Leftrightarrow (\exists\, o : Obligation \bullet \\ \qquad o \in ag_2.norms \land ag_1 \in o.beneficiaries \land g = o.normativegoal) \end{array}$$

The notion of social commitment between agents also introduces power. If an agent agrees to adopt some goal of another because of a previous agreement, then the second agent has *power* through a *social commitment* over the first.

$$\begin{array}{l} powerbycommitment\_ : \mathbb{P}(NormativeAgent \times NormativeAgent \times Goal) \\ \hline \forall\, ag_1, ag_2 : NormativeAgent;\; g : Goal \bullet \\ \quad powerbycommitment(ag_1, ag_2, g) \Leftrightarrow (\exists\, sc : SocialCommitment \bullet \\ \qquad sc \in ag_2.norms \land ag_1 \in sc.beneficiaries \land g \in sc.rewards) \end{array}$$

Further categorisation is possible here, but space constraints dictate that only these most important power relationships through norms can be presented.

5 Discussion

The analysis provided above seeks to show how we can move towards a unified account of social power that includes not only dependence relationships, but also other related concepts, particularly agent architectures and the important societal concerns of norms that will facilitate more sophisticated and effective social reasoning. Previous work in this area has focused on these concerns, but in a discrete and somewhat arbitrary fashion. In this work, we seek to join the different accounts in a seamless way, with an integrated framework that can be used to better understand and reason about the structures that arise from the relationships in social organisations. Indeed, we are not just integrating but also extending and filling the gaps that result from considering these important topics independently.

For instance, our social dependence concept is slightly different from that presented by Sichman et al. [16], mainly because the agent architecture itself is included. It allows a clear differentiation between relevant, active and executing plans making the calculus of dependence situations more accurate. In Sichman's work, for example, a false dependence relationship is found if the plan causing the dependence is not selected to achieve the considered goal. We address this problem by considering different categories of plans. In addition, in contrast to the static vision of built-in norms that must always be complied with, our work goes in the same direction as Boello & Lesmo [1], Dignum, Conte and Castelfranchi [8, 7] among others. That is, we consider norms as indispensable elements to avoid chaos in any society comprising multiple autonomous agents repsonsible for their own decisions.

Although much work regarding norms remains to be done, our initial work, in contrast to other proposals, includes a structure which allows different categories of norm to be represented. In addition, a taste of power relationships based on social regulations has been given as the first step towards an enhanced power model of relationships. Indeed, work is already underway on exploring questions relating to the origination of norms and relationships that give rise to power, as well as their use in reasoning about the choice of actions and plans to execute, and agents to cooperate with.

Acknowledgments: The first author acknowledges the funding of Faculty Enhancement Program (PROMEP) of the Mexican Ministry of Public Education(SEP) and the Benemérita Universidad Autónoma de Puebla (México).

References

1. G. Boella and L. Lesmo. Deliberative normative agents. In *Proceedings of the Workshop on Norms and Institutions*, Barcelona, Spain, 2000.
2. M. Bratman, D. Israel, and M. Pollack. Plans and resource bounded practical reasoning. *Computational Intelligence*, 4(4):349 –353, 1988.
3. C. Castelfranchi. Social power. a point missed in Multi-Agent, DAI and HCI. In Y. Demazeau and J.P. Müller, editors, *Decentralized A.I.*, pages 49–62. Elsevier Science Publishers, 1990.
4. C. Castelfranchi, F. Dignum, C. Jonker, and Treur J. Deliberative normative agents: Principles and architecture. In N. Jennings and Y. Lesperance, editors, *Intelligent Agents VI (ATAL99)*, LNAI 1757. Springer-Verlag, 2000.
5. C. Castelfranchi, M. Miceli, and A. Cesta. Dependence relations among autonomous agents. In E. Werner and Y. Demazeau, editors, *Decentralized A.I. 3*, pages 215–231. Elsevier Science Publishers, 1992.
6. R. Conte and C. Castelfranchi. Norms as mental objects. from normative beliefs to normative goals. In C. Castelfranchi and J. P. Müller, editors, *From Reaction to Cognition (MAAMAW'93)*, LNAI 957, pages 186–196. Springer-Verlag, 1995.
7. R. Conte, C. Castelfranchi, and F. Dignum. Autonomous norm-acceptance. In J. Müller, M. Singh, and A. Rao, editors, *Intelligent Agents V (ATAL98)*, LNAI 1555, pages 319–333. Springer-Verlag, 1999.
8. F. Dignum. Autonomous agents with norms. *Artificial Intelligence and Law*, 7:69–79, 1999.
9. F. Dignum, D. Morley, E. Sonenberg, and L. Cavendon. Towards socially sophisticated BDI agents. In *Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS2000)*, pages 111–118. IEEE Computer Society, 2000.
10. M. d'Inverno, D. Kinny, M. Luck, and M. Wooldridge. A formal specification of dMARS. In *Intelligent Agents IV (ATAL97)*, LNAI 1365, pages 155–176. Springer-Verlag, 1998.
11. M. d'Inverno and M. Luck. Engineering AgentSpeak(L): A formal computational model. *Journal of Logic and Computation*, 8(3):233–260, 1998.
12. M. d'Inverno and M. Luck. *Understanding Agent Systems*. Springer-Verlag, 2001.
13. M. Georgeff and A. Lansky. Reactive Reasoning and Planning. In *Proceedings of the Sixth National Conference of Artificial Intelligence*, pages 677–682. AAI Press/MIT Press, 1987.
14. N. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993.
15. M. Luck and M. d'Inverno. A conceptual framework for agent definition and development. *The Computer Journal*, 44(1):1–20, 2001.
16. J. Sichman, R. Conte, Y. Demazeau, and C. Castelfranchi. A social reasoning mechanism based on dependence networks. In A. Cohen, editor, *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI94)*, pages 188–192. John Wiley & Sons, 1994.
17. J. M. Spivey. *The Z Notation: A Reference Manual.* Prentice Hall, 1992.