
Artificial Intelligence and Law **7:** 81–96, 1999.

Hohfeld in cyberspace and other applications of normative reasoning in agent technology

CHRISTEN KROGH[1] and HENNING HERRESTAD[2]
[1]*SINTEF Telecom and Informatics, Department of Distributed Information Systems, P.O. Box 124 Blindern, 0314 Oslo, Norway*
[2]*Department of Philosophy, University of Oslo*

Abstract. Two areas of importance for agents and multiagent systems are investigated: design of agent programming languages, and design of agent communication languages. The paper contributes in the above mentioned areas by demonstrating improved or novel applications for deontic logic and normative reasoning. Examples are taken from computer-supported cooperative work, and electronic commerce.

Key words: agents, user interface metaphors, agent programming languages, agent communication languages, agent protocols, Hohfeld, formal theories of rights, normative structures, deontic logic, groupware, CSCW, electronic commerce.

1. Preamble

This paper has two main purposes. First, through discussing various possible applications of agent technology, it will answer what uses there are for agents. Second, it will discuss the role of deontic logic with respect to the following two important issues in agent technology:

- The design of agent programming languages.
- The design of agent communication languages.

2. Uses of agents

> *The [agent] metaphor has become so pervasive that we're waiting for some enterprising company to advertise its computer switches as empowerment agents.*
>
> – Wayner and Joch, [36, page 95]

For some time, critical voices have been raised against agents, agent technology and what passes for it in the popular media. It is hard not to agree with their criticism: it is no exaggeration to state that agents, agent technology and multiagent systems have been oversold.

To take one example, it has recently become quite popular to refer to the use of any odd search engine on the internet (such as Lycos, Altavista, or Excite) as using an agent. There is nothing wrong in calling a search engine (or a computer switch!) an agent. In Webster's dictionary [37, page 31]: one of many uses of the word 'agent' is as "An active power or cause; that which has the power to produce an effect; as, a physical, chemical, or medicinal agent; as, heat is a powerful agent." If heat is an agent, then any computer program can be an agent. But even though there is nothing wrong in calling any computer program an agent, there is nothing right in it, either. The label 'agent' should be used for a purpose. It should yield some interesting insight into the subject of discussion to call something an agent. Calling search engines agents is perhaps an example of a case where using agents as a metaphor confuses the user rather than helps him. This consideration holds for basic applications of so-called push-technologies as well. It is easy to agree with Pattie Maes in her stating that [24] "[. . .] current commercially available agents barely justify the name".

In the same manner, care should be taken when ascribing terms such as beliefs, obligations, actions, etc, to agents. Such ascription should be done only if it is useful in determining, understanding, or analyzing the behaviour of agents.

Therefore, there is reason to ask – and keep asking – what possible uses there are:

- for agents, and
- for calling something an agent.

First when these questions are answered satisfactorily should other matters be addressed (such as 'how can normative reasoning be applied to artificial agents?').

Two interesting uses of agents and their technology can be identified among the multitudes of recent writings and discussions on the topics:

1. agent technology as a programming methodology, and
2. agents as user interface metaphors.

Other uses of agents, e.g., such as employing them as vehicles for investigating human terms or relationships, will not be discussed here.

2.1. AGENT TECHNOLOGY AS A PROGRAMMING METHODOLOGY

Agent technology[1] as a programming methodology refers to the use of agent technology as a means of developing computer systems. Most considerations offering arguments in favour of agents as a programming methodology bears on analogues with object-orientation and distributed object systems. From this perspective, agents are special objects used as program components. The main advantages for treating them as such seem to be:

[1] 'Technology' may be considered as a collection of methods to reach certain related goals.

- increased flexibility resulting from employing agents as a novel programming abstraction,
- increased portability of applications resulting from employing mobile code, and
- more efficient systems resulting from computation being performed local to large information sources.

Employing agents as a programming methodology presupposes both languages for programming the agents, and architectures for executing them.

Agents are constructed by means of particular programming languages.[2] As is usual within the agent technology community, there are high expectations attached to the coming of such languages; caused by, e.g., Marc Andreessen proclaiming the next big thing to be [1]...

> A secure, truly mobile agent language – way beyond Java – [that] will eliminate the Tower of Babel that prevents us from harvesting more of the benefits of computing and communications today.

The nature of an agent programming language will to some extent depend on the application areas. Programming agents to support group collaboration, for instance, presupposes particular capacities in the programming languages.

Let *agent architecture* denote a system which supports execution of particular programs called agents. An agent architecture where one or more agents are currently executing is referred to as a *multiagent system*. The following are all examples of multiagent systems:

- A manufacturing shop-floor populated by robots, e.g. as presented in [31].
- A groupware system enhanced by small programs called 'agents', e.g. as mentioned in [35].
- The internet, e.g. as implied by the considerations in [9].[3]

The success of an architecture depends on such factors as support for mobility, ease of implementing agents, support for basic protocols, efficiency of agent communication languages, etc. There are many contending technologies for such architectures. The standardization communities are slowly moving to enhance distributed object architectures to become agent architectures. According to 'The Agents Society', [10], there are activities at FIPA (Foundation of Intelligent Physical Agents), OMG (Object Management Group),[4] W3C (World-Wide-Web Con-

[2] In principle any general purpose programming language would do. In practice some special support is needed.

[3] I.e., [9]: "Even as the world struggles to move to client/server computing, a new model has emerged: network centric computing. In this model, computing, communication, and content converge, and the network becomes the computer."

[4] See [35] for a note on OMG's approach.

sortium), and others (e.g. the KQML community). Agent architectures will not be explicitly discussed in this paper.

2.2. AGENTS AS USER INTERFACE METAPHORS

Agents as a user interface metaphor refers to human-computer interfaces (HCI) constructed in such a way that the user is invited to think of himself as interacting with an agent. The main reason for constructing such interfaces is what may be termed 'cognitive economy': confronted with a computer system, it is sometimes easier (or advantageous) for a user to consider that he is interacting with an agent, than e.g. parts of a complex information system. Some studies have been undertaken which support this view. An interesting result due to Nass et al [28] concludes that users are more positive towards systems which mimics their own personal characteristics. Bates' research in believable agents may also be seen to fall into this category (cf. [3]). It should be noted that 'agents' employed in this mode does not necessarily correspond to an identifiable object. Systems where users think of their interaction as interacting with an agent need not be multiagent systems.

A business object is an object which has a relevant presence in an organization using the information system containing the object.[5] In [7, section 2.1], typical examples of business objects are given as: employee, product, invoice, and payment. Examples of non-business objects are time-table-property and report-transaction-log. Employing this as an analogy to agents, business agents are particular objects called agents which has a relevant presence – as agents – in the organization which employs them. Examples of business agents are: meeting-planners, project-deadline-reminders and personal organizers. An examples of a non-business agent is a request-re-router. In lieu of (1) and (2) above, business agents can be understood as agents which are useful as a user interface metaphor, and has been developed by an agent-oriented programming methodology. Business agents seem to be among the most interesting candidate subjects for normative reasoning.

Groupies are business agents residing in groupware systems. Investigating the use of agents in groupware systems is not new. In particular, automating tasks such as meeting scheduling has been a topic for some time. According to Baecker [2, page 458], meeting scheduling is: '[a] classic process that seems ripe for automation through structured messages, agents and/or workflows [...]'. According to Virdhagriswaran [35, page 97], groupware vendors soon went past merely meeting scheduling, and started exploring '[...] the use of agents to automate organizational roles and encapsulate organizational knowledge'. Also other applications has been investigated (cf. [9]).

Group collaboration is dominated by complex relationships between the collaborators – relationships which often are of a normative character. Collaboration and coordination of collaboration often centers around who does what and who

[5] Cf. [7, section 2]: "[...] business objects represent things, processes or events that are meaningful to the conduct of the business".

ought to do what. In [21, (section 2)] Lai et al presents an approach to group work support where semi-autonomous agents assist in managing relationships between objects representing 'who requested what' and 'who did what'. Expressing normative relations and (speech) actions are also an important topic in the approach developed by Flores et al. [8].[6] Thus, two requirements for agent programming languages suited to develop support for group collaboration can be formulated:

1. adequate treatment of normative relations, and
2. the capacity for reasoning about actions.

These two requirements can be extended to also include the 'actions' and 'normative' relationship local to agents (i.e., not involving their owners).[7]

3. Designing agent programming languages

In this section an approach to designing agent-oriented programming languages will be investigated. A problem with how it treats normative relations and actions will be pointed out, and a fix will be proposed. It is argued that the fix both makes the approach more suitable for constructing agents in general, as well as agents to support group collaboration.

3.1. AGENT-ORIENTED PROGRAMMING

In the papers [29–31] and [34], Shoham proposes a framework for multiagent systems. This paper follows the presentation in [31]. Shoham calls his approach Agent Oriented Programming (AOP). He motivates his investigations by describing a scenario from a shop-floor where robots are collaborating between themselves in order to construct vehicles. We consider it likely that the framework (or something similar) is also applicable for designing agents that support human collaboration (e.g. softbots arranging meetings rather than robots painting cars).

A multimodal logic is constructed in order to design the agent language. This logic contains three basic operators: a belief operator, an obligation operator[8], and a capability operator. The operators, as well as facts, are relativised to points in time. Two other notions are defined by means of the basic operators: decision and ability. A separate operator for action is not available in the logic, but is added in the programming language as a kind of 'syntactic sugar'.

This vocabulary of operators would seem like good candidates for specifying languages to implement support for many interesting group tasks (assuming the

[6] Even though they have not focussed on agents in particular.

[7] Artificial agents does of course not bear obligations in a traditional sense. But it might be advantageous to consider it that way.

[8] The obligation operator is sometimes referred to as a commitment operator, even though there is arguably a difference between obligations and commitments. However, as Shoham describes no such differences, it will be assumed that he denotes one notion rather than two, and that this notion is one of obligation.

operators have reasonable counterparts in the programming language). However, a critical point is whether the weak support for reasoning about action suffices. Below, a discussion of the obligation operator will shed light on this.

According to Shoham, an agent comes under an obligation that something on condition of a mental state (a belief) and a message state (a request): if an agent i request j that it do A, and doing A 'conforms' to agent j's belief state, then j comes under an obligation towards i to do A.

Obligations are expressed by means of sentences of the form $OBL^t_{i,j}A$. The sentence $OBL^t_{i,j}$`meetingArranged(owner(i), owner(j))` is read as 'at time t, agent i is under an obligation towards agent j that there is a meeting arranged between the owner of i and the owner of j'. It seems prudent to consider i to be the bearer, and j to be the counterparty, of the obligation. That i should arrange the meeting at a specific time $t+1$ is expressed as follows:

$$OBL^t_{i,j}(\texttt{meetingArranged(owner(i), owner(j))})^{t+1}.$$

Literally, this expression should be read as 'at time t, agent i is under an obligation towards agent j that, at time $t+1$, there is a meeting arranged between the owner of agent i and agent j'. Here, the obligation is *at* time t, but *about* time $t+1$.

Shoham imposes the following constraint on the obligation operator:

$$\text{for any } t, i : \{A : OBL^t_{i,j}A \text{ for some } j\} \text{is consistent} \tag{1}$$

The following principle may be validated from this constraint:

$$\forall x.\forall y.\neg(OBL^t_{i,x}A \wedge OBL^t_{i,y}\neg A) \tag{2}$$

(2) states that the obligation-set for any one individual (in this case i) is consistent in the sense that there is no obligation of i that conflicts with another obligation of i. Arguments may be constructed in favour of such a requirement. The reader should note that due to the rather misleading placement of the existential quantification *for some* j in the constraint (1) there are also other, weaker, possibilities. These weaker interpretations do not seem viable, however, as they do not correspond well to either the concept of *internal consistency* supported by Shoham [31, page 64], or the motivating examples he gives (e.g. [31, 12:15, page 58]).

If (2) conveys the strongest notion of consistency requirement for the obligation operator, then two agents may bear conflicting obligations to a common third individual, or conflicting obligations to different (third) individuals. In other words, the following sentences appear to be satisfiable within the system:

$$OBL^t_{i,j}A \wedge \forall x \neq i.OBL^t_{x,j}\neg A \tag{3}$$

$$OBL^t_{i,j}A \wedge \forall x \neq i.\forall y \neq j.OBL^t_{x,y}\neg A \tag{4}$$

However, it shall be argued otherwise.

When considering the way agents are updated and allowed to interact with the environment in AOP, it is seen that when the time-point an obligation is about is reached, the obligation is 'executed' [31, page 68]. This 'execution' of the obligation at the timepoint it obtains means that if there is an obligation of the following sort $OBt^{t}_{i,j}(A)^{t}$, and time t obtains, then A obtains. This consideration validates a version of the T-schema[9].

Having a version of the T-schema makes all obligations very close to regimentations. If Shoham intended to formulate obligations, the conclusion is that he failed. As a principle of action, however, the T-schema is less controversial. Which leads to the conclusion that it is less problematic to view Shoham's system as without an explicit representation of obligation than as without an explicit representation of action.

There is, however, a more serious problem just around the corner: because what is obligatory obtains, *no two conflicting obligations between any pairs of individuals may exist at the time the obligations are about*. This problem is a rather serious shortcoming, not because it is counter-intuitive (and it is!), but because it assumes strict coordination in settings where lack of such coordination seems particularly prone.[10] In Shoham's framework, for instance, we would want to allow a representation of the fact that; at time t, both Ian is under an obligation to arrange a meeting between Ian and Jim, and Karl is under an obligation not to arrange a meeting between Ian and Jim. This would be an accurate description of differences in responsibility that may allow us to distinguishing Ian and Karl's organizational roles. Strict coordination would obliterate such distinctions.

3.2. THE FIX

The problems are diagnosed as that (i) there is no explicit representation of action, and (ii) obligations between individuals are not allowed to conflict with obligations between other individuals. Without fixing the problems, AOP does not seem suitable for developing agents for groupware support. For other application areas, however, it may be suited. Note that even robots painting cars would suffer from the problems above.

Solving Shoham's problem can be done by appealing to formalisms described elsewhere (cf. [11, 18, 19]). A short summary of them will suffice here: A classical modal operator of type ET, ${}_i\mathcal{S}$, is introduced as a new representation of action. Expressions of the type ${}_i\mathcal{S}A$ is read: 'The agent i sees to it that A'. Expressions of the type $OBL_{i,j}A$ are substituted with expressions of the type ${}_i\mathcal{O}_j(A)$. The operator ${}_i\mathcal{O}_j$ is defined employing two operators ${}_i\mathcal{O}$ and $\mathcal{O}_j$ as follows:

$$ {}_i\mathcal{O}_j(A) \stackrel{def}{=}_i \mathcal{O}({}_i\mathcal{S}A) \wedge \mathcal{O}_j({}_i\mathcal{S}A) \tag{5}$$

[9] I.e., a version of $\Box A \supset A$.

[10] Note, however, that no normative conflict occurs in Shoham's own examples.

The operator ${}_i\mathcal{O}$ expresses an obligation personal to agent i. It follows an individualized normal modal logic of type KD. A sentence ${}_i\mathcal{O}A$ is read as 'i is under an obligation that A'.

The operator $\mathcal{O}_j$ expresses a notion of something being *ideal* for j in the eyes of an implicit normative system. The sentence $\mathcal{O}_j A$ is read as 'it is ideal for j that A'. The operator $\mathcal{O}_j$ follows an individualized classical modal logic of type END.

Neither of the operators ${}_i\mathcal{O}$ or $\mathcal{O}_i$ has the generalized D-schema; i.e.:

$$\not\vdash \neg({}_i\mathcal{O}A) \wedge ({}_i\mathcal{O}\neg A) \tag{6}$$

$$\not\vdash \neg(\mathcal{O}_j A) \wedge (\mathcal{O}_j \neg A). \tag{7}$$

It is easy to see that the troublesome principle of strong consistency does not hold for the directed obligation operator, since the following expression is satisfiable (where $i \neq k$ and $j \neq l$):[11]

$$\_i\mathcal{O}_j(A) \wedge_k \mathcal{O}_l(\neg A) \tag{8}$$

Note that (8) correspond to (4). The following expression, which corresponds to (3), is also satisfiable (where $i \neq k$):

$${}_i\mathcal{O}_j(A) \wedge_k \mathcal{O}_j(\neg A) \tag{9}$$

The reason for this being satisfiable is that the operator $\mathcal{O}_j$ is not closed under consequence.

In summary, introducing an explicit notion of action, and an improved notion of directed obligation, solves AOP's problems. This will make it easier to design a better agent language. In addition, if support for the operators described above is implemented in the programming language, it seems that the language would be better suited to develop agents to support group work (cf. the requirements in the previous section). Therefore the following suggestion is proposed:

SUGGESTION 1 *Formal deontic notions are useful when designing agent programming languages to be used in group work environments, because they make it easier to specify normative relationships between agents, as well as between agents and their users. Care should be taken to match the formal operators with the program concepts.*[12]

[11] To see this, it is sufficient to appreciate that neither ${}_i\mathcal{O}$ nor $\mathcal{O}_i$ has the generalized D schema.

[12] We shall not go on to specify *how* care can be taken to match the formal operators with the program concepts. The present concern is only with the design, not with the implementation of agent programming languages. Our attempt to give an exact specification of distinctions that ought to be modeled in the actual programming language is only a necessary first step on the way to making an adequate language for programming business agents.

4. Designing agent communication languages

Assuming an established means for programming agents as well as an architecture to execute them on, there arises a problem of how to ensure that the agents behave properly in relation to each other. One of the first prerequisites for proper agent behaviour is their (or their developers) ability to agree upon conducts of behaviour before entering into complex interactions. This is an issue in the design of agent communication languages.

The importance of proper behaviour varies with the application area, and electronic commerce is perhaps one of the most critical. It is also an area that receives increasing attention: a recent study done by Nielsen [27] argues that active web-shoppers have more than doubled in the past 18 months.[13]

There have been several experiments with proto-agents for searching out the best offers described in a set of distributed, heterogeneous information sources on the internet. The Bargainfinder agent from Andersen [20] is an example of such an agent: it checks a number of predetermined internet-accessible music stores for prices and availability of a given CD, and returns the best bargain.

More interesting examples can be conceived, in particular if one assumes an electronic currency and adequate mechanisms for performing micro-transactions. Consider the following scenario:

> A piece of news transmitted over the internet costs only a micro-Euro (one thousandth of one Euro). Peter has a news-agent executing on his personal computer. At regular intervals, the news-agent checks a number of sites around the world for news updates. If it finds something matching Peters profile, it buys it. Once a newly bought piece of news is stored locally, the news-agent notifies Peter. Peter indicates whether he likes it or not, and the news agent updates Peters profile.

Buying the news amounts to going through a particular predetermined protocol. The structure of such a protocol as well as the internal structure of each step must be preprogrammed in both the selling and the buying agent. Below, an outline of a naive version of such a protocol is given, where the news-agent is denoted the buyer and the sales-agent is denoted the seller:

1. The buyer commits to buying the goods at a determined price.
2. The seller commits to selling the goods at the determined price.
3. The buyer transfers the money.
4. The seller makes available the goods (in this case, the news).

During the first two protocol steps (1) and (2), the agents agree upon what they shall ‘do’ in the next two protocol steps (transfer money, and make available the goods, respectively). An important observation is that the sequence of the first two protocol steps (1) and (2) is analogous to entering into a contract.

[13] cf. [27]: “[. . .] 39 percent of all Web users have searched for product information online prior to making a purchase, compared to 19 percent in Fall 95”.

Another important observation is that the above transaction involves money, even though arguably very little of it. Since there is real value in the transaction, real rights (of the agents' owners) are involved, and may thus be violated if something goes wrong.

For these reasons such a protocol will usually include the use of methods for digital signatures to obtain so-called non-repudiation[14]. Unfortunately, this does not solve all problems. Consider the following extension to the above scenario:

> The news bought by the news-agent turned out to be rather large (containing several high-quality video-clips and a complex 3D model). The news-agent does not discover that this is a problem before the sales-agent is mid-way in transferring the data. The news-agent attempts to call off the sale, but to no avail. The sales-agent continues to transfer the data (perhaps a way of halting the transfer has yet to be implemented). By continuing the transfer, the sales-agent forces the news-agents server to shut down its internet connection, and thus induces damage on the news-agent's owner.

When confronted with the damage induced on the owner of the news-agent, the owner of the sales-agent could defend himself in the following way: 'Your agent bought the news, therefore my agent was permitted to transfer it. It is not my problem that your agent orders more than you can consume'.

The problem in the news-/sales-agents scenario may be diagnosed to be a difference of 'understanding' between the agents concerning who is permitted to do what: is the sales-agent permitted to go on transferring the goods after news-agent have requested to cancel the transfer, or is it not?

Regardless of the seeming ease with which the problem may be resolved now, it would be preferable to have a systematic means of analyzing and structuring candidate normative relationships for regulating interaction between agents, prior to them entering into (the analogue) of a contract. This is important because it gives cues for how agents ought to behave during and after the sale. In the next section, such a systematic means will be described.

4.1. FROM HOHFELD'S RIGHTS TO NORMATIVE POSITIONS

At the beginning of this century, Wesley Newcomb Hohfeld outlined a theory of rights describing what he called the fundamental legal conceptions [12]: Right (called 'claim', below), Duty, NoRight, Privilege (called 'liberty', below), Power, Liability, Disability, and Immunity. The rights can be classified into two groups: purely deontic (claim, liberty, no-right, duty), and capacitive (power, immunity, disability, liability). A scheme for how the rights were interrelated was proposed along with the rights.

[14] Non-repudiation is defined as the impossibility of an agent to (reasonably) deny having participated in all or part of a communication [6, page 11].

Hohfeld's rights were intended to serve as the smallest common denominators in jurisprudential reasoning. Hohfeld's contribution was important as an attempt to make jurisprudential reasoning more subtle and therefore presumably more accurate and precise. According to, e.g., James B. Brady [5, page 247], Hohfeld was after clarity, not for its own sake, but for the definite solution of legal problems.

Hohfeld did not have access to the methods available to modern logicians, and even though highly acclaimed for his analytic talents (cf. [4]), he does not offer a logical tool applicable for studying rights.

Stig Kanger (see [14–17]) was probably the first to give a formal account of Hohfeld's theory. One of Kanger's ideas was that the obligation-operator of a deontic logic, together with an action operator, would enable the expression of all of Hohfeld's fundamental legal concepts. In [14], Kanger gives the following explication of Hohfeld's theory (here, the symbol $\mathcal{O}$ is used for the obligation operator, and ${}_i\mathcal{O}$ for the action operator):

$$\text{Claim}(i, j, A) \equiv \mathcal{O}({}_j\mathcal{S}A) \tag{10}$$

$$\text{Liberty}(i, j, A) \equiv \neg\mathcal{O}({}_i\mathcal{S}\neg A) \tag{11}$$

$$\text{Power}(i, j, A) \equiv \neg\mathcal{O}\neg({}_i\mathcal{S}A) \tag{12}$$

$$\text{Immunity}(i, j, A) \equiv \mathcal{O}\neg({}_j\mathcal{S}\neg A) \tag{13}$$

By employing the interrelationships between the rights suggested by Hohfeld, the explications of the rest of Hohfeld's rights follow (cf. [14, page 17]):

$$\text{NoRight}(i, j, A) \equiv \neg\mathcal{O}({}_j\mathcal{S}\neg A) \tag{14}$$

$$\text{Duty}(i, j, A) \equiv \mathcal{O}({}_i\mathcal{S}A) \tag{15}$$

$$\text{Disability}(i, j, A) \equiv \mathcal{O}\neg({}_i\mathcal{S}\neg A) \tag{16}$$

$$\text{Liability}(i, j, A) \equiv \neg\mathcal{O}\neg({}_j\mathcal{S}A) \tag{17}$$

In [5, page 258], James B. Brady argues that Hohfeld's fundamental legal concepts may be employed in combination with each other. He uses this argument to overcome a critique of Stone [32] concerning the incompleteness of Hohfeld's analysis. In [17], some years before Brady, Stig Kanger and Helle Kanger started to investigate how their explications might be combined. This investigation resulted in what the Kangers called the *atomic types of rights*.

By employing a few notational constructs, slightly modified by those proposed by Makinson in [25], Kanger's investigation of combinations of rights can be formulated quite easily. A **choice set**, $(X)\{Y\}$, is defined as the following lexicographic operation:[15]

[15] Note that the curly brackets will be dropped when no problem of ambiguity arises.

DEFINITION 4.1 $(X)\{Y\} \stackrel{def}{=} \{UV : U \in X \,\&\, V \in Y\}$.

A maxiconjunction is defined as follows:

DEFINITION 4.2 *A* **maxiconjunction** *is a set resulting from forming all possible conjunctions from a set of sentences, and removing internally inconsistent and redundant members.* $[[M]]$ *expresses the maxiconjunction obtained from* M.

The *atomic types of rights* can now be expressed as follows:

$$[[(^{+}_{-})\mathcal{O}(^{+}_{-})(^{j\,\mathcal{S}}_{j\,\mathcal{A}})(^{+}_{-})A]] \tag{18}$$

The atomic rights presumed to give a complete specification of the possible normative relationships relative to two persons and a state of affairs.

Kanger has been criticised for explicating Hohfeld's *capacitive* rights by means of the *deontic* notions of obligation and permission (cf. Nils Kristian Sundby's critique in [33, p. 413]). This critique seems justified. Instead of considering Kanger's 'rights' as pure explications of Hohfeld's rights, they will be considered as normative expressions 'in their own rights'; – inspired by Hohfeld.

Lars Lindahl [23] made the observation that some notions, both meaningful and logically possible, were missing from Kanger's set of atomic positions (cf. [25, page 406]). In order to express Lindahl's positions, a further piece of notation must be introduced. Let $\bigwedge$ be defined as an operation on two sets of sentences as follows:

DEFINITION 4.3 *Let* X *and* Y *be sets of sentences.* $X \bigwedge Y \stackrel{def}{=} \{Z : (Z = A \wedge B) \,\&\, (A \in X) \,\&\, (B \in Y) \,\&\, (Z \nvdash \bot)\}$.

Lindahl's set of positions regulating the actions of two persons can now be expressed as follows (where permission is defined as $\mathcal{P}(_i\mathcal{S}A) \stackrel{def}{=} \neg\mathcal{O}\neg(_i\mathcal{S}A)$):

$$[[(^{+}_{-})\mathcal{P}[[(^{+}_{-})_i\mathcal{S}(^{+}_{-})A]]]] \bigwedge [[(^{+}_{-})\mathcal{P}[[(^{+}_{-})_j\mathcal{S}(^{+}_{-})A]]]] \tag{19}$$

Lindahl calls the set specified by the above expression the set of *two-person, individualistic, normative positions*. This set is composed of 35 statements concerning the normative status of each of the two agents' acts.[16] They number 9 more than Kanger's atomic rights, simply because expressions such as $\mathcal{O}(\neg_i\mathcal{S}A \wedge \neg_i\mathcal{S}\neg A) \wedge \mathcal{O}(\neg_j\mathcal{S}A \wedge \neg_j\mathcal{S}\neg A)$ are satisfiable. In [13] Jones and Sergot proposes a more logic-independent way of formulating Lindahl's positions:[17]

$$[[(^{+}_{-})\mathcal{O}(^{+}_{-})[[(^{+}_{-})_i\mathcal{S}(^{+}_{-})A]]\,]] \bigwedge [[(^{+}_{-})\mathcal{O}(^{+}_{-})[[(^{+}_{-})_j\mathcal{S}(^{+}_{-})A]]\,]] \tag{20}$$

[16] See [18, Appendix B.1] for a complete list of Lindahl's positions.

[17] Note their method is not completely logic independent. Because the maxiconjunction operation removes internally inconsistent members, it does not work correctly for operators which do not have the N-schema, cf. [18, pp. 66–68].

Expressing normative positions for personal rather than impersonal deontic operators is rather trivial. Employing the personal deontic operator described in [19], the set of normative positions can be expressed as follows:

$$[[(^{+}_{-})_i \mathcal{O} i (^{+}_{-})[[(^{+}_{-})_i \mathcal{S}(^{+}_{-})A]]]] \bigwedge [[(^{+}_{-})_j \mathcal{O}(^{+}_{-})[[(^{+}_{-})_j \mathcal{S}(^{+}_{-})A]]]] \tag{21}$$

4.2. HOHFELD IN CYBERSPACE

Let us return to the scenario, and see whether the above position is appropriate for specifying an acceptable normative relationship between the news-agent and the sales-agent. Let n stand for the news-agent, s stand for the sales-agent, and A stand for 'the goods is delivered'. One of the normative positions obtained from the generation scheme (21) is the following:

$$_n\mathcal{P}(_n\mathcal{S}A) \wedge {}_n\mathcal{P}(_n\mathcal{S}\neg A) \wedge {}_n\mathcal{P}(_n \prod A) \wedge {}_s\mathcal{O}(_s \prod A) \tag{22}$$

The formula $_s \prod A$ is shorthand for $\neg_s \mathcal{S}A \wedge \neg_s \mathcal{S}\neg A$. It is read as 'the agent s is passive wrt A'. (22) can now be read as follows: 'n is permitted to see to it that A, and n is permitted to see to it that not-A, and n is permitted to be passive wrt A, and s is under an obligation to be passive wrt A'. It seems like a plausible candidate for formulating the normative relationship between the two agents in question.

The second and the fourth of these clauses are particularly interesting. The second clause permits the news-agent to decline to receive the goods. The fourth clause forbids the sales-agent to be active wrt delivering the goods. In practice this means that the goods should be transferred by means of *pull*, rather than *push* mechanisms.[18]

The above considerations may be used to refine the four-pass protocol described earlier in the following way:

1. The buyer commits to buying the goods at a determined price, **and** a normative position to regulate the transaction.
2. The seller commits to selling the goods at the determined price, **and** the normative position.
3. The buyer transfers the money.
4. The seller makes available the goods.

As was noted earlier: during the first two protocol steps (1) and (2), the agents agree upon what they shall 'do' in the next two protocol steps (transfer money, and make available the goods, respectively). Adding a clause about a normative position enables a more fine-grained specification of what the agents should 'do' (and not 'do'). For instance, above, it was argued that the sales-agent should not 'do' anything concerning transfer of the goods. As this 'doing' was shown to be

[18] See [26] for a hyped-up presentation of push/pull-technologies.

a problem with the first protocol, the more fine-grained specification increased the quality of the protocol. This leads to the following suggestion:

SUGGESTION 2 *The theory of normative positions is useful when designing domain-specific agent communication languages, because it enables faster recovery from fraud or mistakes. Domain-specific protocols should be enhanced by making the agents agree upon which normative position should regulate their interaction before entering into (the analogue) of a contract.*

Not all normative positions are applicable for use in commercial protocols as the one discussed above. A possible direction for future work would be to single out particularly interesting categories of protocols and investigate what normative positions are most likely to be needed in order to regulate the behaviour of the participating agents.

5. Conclusion

[. . .] so he went to the inventor and asked to be shown an artificial friend.
– Stanislaw Lem, [22, page 79]

In spite of its heritage in AI and other disciplines, agent technology is a rather new research field. Most papers specifically addressing the topic have been published during the last 5 years. During its relatively short life-span, agent technology has attracted a lot of public attention, due to the promised coming of successful, mass-market, internet-based agent-applications. Such applications have been oversold, endangering the more serious uses and results.

In this paper we have addressed a few of the serious and important problems in agent technology while retaining some of the flare which has made the field so popular. The following suggestions have been made:

SUGGESTION 1 *Formal deontic notions are useful when designing agent programming languages to be used in group work environments, because they make it easier to specify normative relationships between agents, as well as between agents and their users. Care should be taken to match the formal operators with the program concepts.*

SUGGESTION 2 *The theory of normative positions is useful when designing domain-specific agent communication languages, because it enables faster recovery from fraud or mistakes. Domain-specific protocols should be enhanced by making the agents agree upon which normative position should regulate their interaction before entering into (the analogue) of a contract.*

Acknowledgments

Andrew Jones has as usual given valuable comments on previous formulations of many of the ideas presented here. Also thanks to Odd-Wiking Rahlff and Jan Øyvind Aagedal for reading and commenting on the current paper.

References

1. Marc Andreessen 1996. The future of microcomputers, *Byte*.
2. Ronald M. Baecker (ed.) 1993. *Readings in Groupware and Computer-Supported Cooperative Work*, Morgan Kaufmann Publishers Inc..
3. Joseph Bates 1994. The role of emotion in believable agents. Technical Report CMU-CS-94-136, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.
4. Nuel Belnap 1991. Backwards and forwards in the modal logic of agency, *Philosophical and Phenomenological Research*, LI.
5. James B. Brady 1972. Law, language and logic: The legal philosophy of Wesley Newcomb Hohfeld, *Transactions of the Charles S. Peirce Society*, **8**, 246–263.
6. Ulf Carlsen 1994. *Formal Specification and Analysis of Cryptographic Protocols*, PhD thesis, l'Université Paris XI – Orsay.
7. Cory Casanave 1996. Business-object architectures and standards. Technical Report, Business Object Domain Task Force, OMG.
8. Fernando Flores, Michael Graves, Brad Hartfield, & Terry Winograd 1988. Computer systems and the design of organizational interaction, *Transactions of Office Information Systems* **6**(2), 153–172.
9. Don Gilbert, Manny Aparicio, Betty Atkinson, Steve Brady, Joe Ciccarino, Benjamin Grosof, Pat O'Connor, Damian Osisek, Steve Pritko, Rick Spagna, & Les Wilson 1995. Ibm intelligent agent strategy. White paper.
10. Agent Interop Working Group 1997. Draft report of 'design workshop on open intelligent agent platforms and protocols' – first meeting of the agent interop working group. http://www.agent.org/society/meetings/workshop9702/report.html
11. Henning Herrestad & Christen Krogh 1995. Obligations directed from bearers to counterparties. In *Proceedings from ICAIL'95*, ACM Press, Washington, pp. 210–218.
12. Wesley Newcomb Hohfeld 1966. *Fundamental Legal Conceptions as Applied in Judicial Reasoning and Other Legal Essays*, Yale University Press, New Haven, Connecticut.
13. Andrew J.I. Jones & M. Sergot 1993. On the characterization of law and computer systems: The normative systems perspective. In J.-J. Meyer & R.J. Wieringa (eds.), *Deontic Logic in Computer Science – Normative System Specification*, John Wiley, Chichester.
14. Stig Kanger 1957. New foundations for ethical theory. Technical Report, Stockholm University, Stockholm.
15. Stig Kanger 1972. Law and logic, *Theoria* **38**,105–132.
16. Stig Kanger 1985. On realization of human rights. In G. Holmström & A.J.I. Jones (eds.), *Action, Logic, and Social Theory*, Acta Philosophica Fennica, Vol. 38, Helsinki, pp. 71–78.
17. Stig Kanger & Helle Kanger 1966. Rights and parliamentarism, *Theoria* **6**(2), 85–115.
18. Christen Krogh 1997. *Normative Structures in Natural and Artificial Systems*, PhD thesis, Department of Philosophy, University of Oslo.
19. Christen Krogh & Henning Herrestad 1996. Getting personal. In Mark A. Brown & José Carmo (eds.), *Deontic Logic, Agency and Normative Systems*, Workshops in Computing, Springer, Berlin, pp. 134–153.
20. Bruce T. Krulwich 1996. The bargainfinder agent: Comparing price shopping on the internet. In Joseph Williams (ed.), *Bots and other internet beasties*, SAMS.NET, Macmillan, pp. 258–263.

21. Kum-Yew Lai, Thomas W. Malone & Keh-Chiang Yu 1977. Object lens: A “spreadsheet” for cooperative work. *Transactions of office information systems* **6**(4), 332–353.
22. Stanislaw Lem 1977. *Mortal Engines*, Avon Bookes, New York.
23. Lars Lindahl. *Position and Change*, D. Reidel Publishing Company, Dordrecht, Holland.
24. P. Maes 1995. Intelligent software, *Scientific American* **273**(3).
25. David Makinson 1986. On the formal representation of rights relations. *Journal of Philosophical Logic* **15**, 403–425.
26. Lee Marshall 1997. Kill your browser, *Wired Magazine* **5**(3). http://www.wired.com/wired/5.03/features/ff-push.html
27. Nilsen Media 1997. Press release, march. http://www.commerce.net/nielsen/press-97.html
28. Clifford Nass, Moon Youngme, B.J. Fogg, Byron Reeves, and D. Christopher Dryer 1995. Can computer personalities be human personalities, *int. J. Human–Computer Studies* **43**, 223–239.
29. Yoav Shoham 1990. Agent-oriented programming (revised). Technical Report STAN-CS-1335-90, Computer Science Department, Stanford University.
30. Yoav Shoham 1991. Implementing the intentional stance. In R. Cummins & J. Pollock (eds.), *Philosophy and AI: Essays at the Interface*, MIT Press, Cambridge, Massachusets, pp. 261–277.
31. Yoav Shoham 1993. Agent-oriented programming, *Artificial Intelligence* **60**, 51–92.
32. Roy Stone 1963. An analysis of Hohfeld, *Minnesota Law Review* **48**, 312–337.
33. Nils Kristian Sundby 1974. *Om Normer*, Universitetsforlaget.
34. Mark C. Torrance & Paul A. Viola 1991. The Agent0 manual. User manual, Computer Science Department, Stanford University.
35. S. Virdhagriswaran, D. Osisek & P. O’Connor. Standardizing agent technology, *Standard View* **3(3)**, 96–101.
36. P. Wayner & A. Joch. Agents of change. *Byte*.
37. Webster 1913. *Webster Dictionary*.