Modelling Communication between Cooperative Systems

Frank Dignum[1] and Hans Weigand[2]

[1] Eindhoven University of Technology, Dept. of Mathematics and Computer Science, P.O.box 513, 5600 MB Eindhoven, The Netherlands, tel.+31-40-473705, email: dignum@win.tue.nl

[2] Tilburg University, Infolab, P.O.box 90153, 5000 LE Tilburg, The Netherlands, tel.+31-13-662806, email: h.weigand@kub.nl

Abstract. In cooperative systems many of the obligations, prohibitions and permissions that govern the behaviour of the system exist as a result of communication with users and/or other systems. In this paper we will discuss the role of illocutionary logic and deontic logic in modelling these communication processes and the resulting norms. The combination of illocutionary and deontic logic can be used to reason about communication structures. It is also possible to model the authorization relations, on the basis of which orders and requests can be made, and the delegation of these authoritizations in this logic.

1 Introduction

In recent years there has been a growing interest in cooperative systems or cooperating *agents*(see e.g. [Po93]). However, little has been said about how these cooperating agents influence each other's behaviour. Each agent has certain capabilities, actions that it can perform. These actions can be material, such as opening a window, or communicative, such as providing some piece of information. Each agent also has an agenda containing the actions to be performed by the agent, instantly or at some designated time. In a normative system this agenda consists of the obligations of the agent. We assume that the agenda is not fixed but can be manipulated by the agent. The agent can add new obligations to the agenda (typically done on the request of another agent). He can also remove actions by performing them or by violating the obligation. In the latter case he usually is obliged to perform some new action that compensates for the violation. The description of the obligations and the manipulation of the obligations has been the subject of research in normative systems and deontic logic (see e.g. [MW93]).
The interaction between agents can most easily be seen in the cases where a particular service is rendered from one agent to the other. A (simple) example of such a service is illustrated in the following picture.

In general we can distinguish three phases of communication. The first phase is the negotiation about the terms of the contract. In this phase authorizations can be established on the basis of which some actions can be performed in the

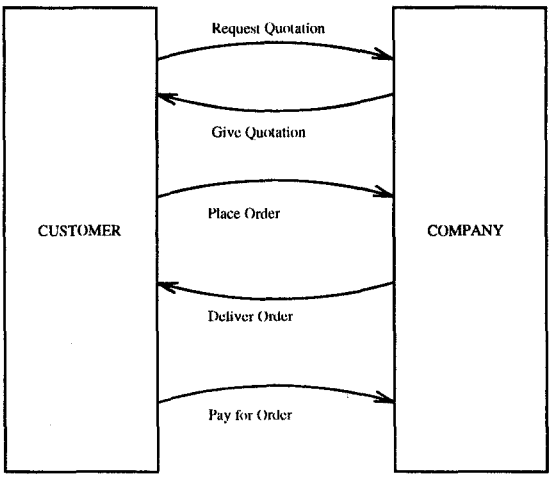

Fig. 1. ordering procedure

following phases. In the above figure this corresponds to the requesting and sending of a quotation. Here, sending a quotation implies authorizing the customer to order some products on some specified conditions. The second phase consists of the acceptance of the contract. I.e. the authorizations and obligations that follow from the contract. In the above example this is included in the order which implies an acceptance of the quotation. The last phase is the fulfilment of the contract. I.e. following a protocol according to the terms agreed upon in the contract. In the example this corresponds to the actual order, delivery and payment sequence.

In this paper we will show that the communication processes described above (including the resulting norms) can be modelled using illocutionary logic and deontic logic.
Illocutionary logic [SV85] is a logical formalisation of the theory of speech acts [Se69] and is used to formally describe the communication structure itself. A first attempt to model communication using speech acts is made in the method SAMPO [LL86], but this does not include the deontic aspects. In [Lee88], Lee presents a language and modelling technique for deontic rules, but these model procedures rather than communication structures.

The fundamental reason for the use of deontic concepts is that coordination of behaviour always requires some form of mutual commitment. If an agent does not execute an action, he has committed himself to, this causes a violation of the contract. Because the action should be executed in the future, it cannot be guaranteed, so the interpretation "it will happen in all future courses of events" is too strong, but the interpretation "it will happen in some course of events" is too weak. Interpreting the formula "α is obligatory" as: 'not doing α violates a commitment", we get a more precise meaning of what it is that something is on an agent's agenda.

In the next section we present a logical formalism that incorporates the speech acts into the dynamic deontic logic described in [Me88]. This language can be used to model the norms that result from the communications between agents in a normative system.
The combination of illocutionary logic and deontic logic is used to build communication protocols or contracts, that define the subsequent steps in the communication and how these steps are related. I.e. which are the allowed reactions to a certain action. E.g. After a product has been delivered the reaction should be a payment of the order. In section 3, a framework for describing contracts is presented on the basis of the logic defined in section 2.
Section 4 gives some conclusions and directions for future research.

2 Logical foundations for modelling communication

In this section we will present the logical foundation on the basis of which the communication processes between agents can be modelled. The logical language is based on the dynamic deontic logic described in [Me88], which is extended to include speech acts as formalized in illocutionary logic.

2.1 The language

We start with a language based on the Dynamic Deontic Logic Language given in [Me88]. We will only give a short overview.

We start by defining a language of parameterized actions L_{act}

Definition 2.1 The language L_{act} of actions is given by the following BNF:

$$\alpha ::- \ \underline{a}|\alpha_1 \cup \alpha_2|\alpha_1 \& \alpha_2|\overline{\alpha}|\mathbf{any}|\mathbf{fail}$$

The $\underline{a}$ stands for the atomic actions in the system, like "order(i,j,p)", which states that agent i orders p from agent j. The first parameter indicates the subject of the action. The meaning of $\alpha_1 \cup \alpha_2$ is a choice between α_1 and α_2. $\alpha_1 \& \alpha_2$ stands for the parallel execution of α_1 and α_2. The expression $\overline{\alpha}$ stands for the non-performance of the action α. The **any** action is a universal or "don't care which" action. Finally the **fail** action is the action that always fails (deadlock). This action does not lead to a next state.

The language L_{act} can be used to describe actions within dynamic deontic logic. The language of dynamic deontic logic (L_{dd}) is given in the following definition.

Definition 2.2 The language L_{dd} of dynamic deontic logic is given by the following BNF:

$$\Phi ::- \ \phi|\Phi \vee \Psi|\Phi \wedge \Psi|\neg\Phi|[\alpha]\Phi|B(i,\phi)|I(i,\alpha)|I(i,\phi)$$

Where ϕ is a first order logic formula and α an element of L_{act}.

The intuitive meaning of $[\alpha]\Phi$ is that after the execution of α, Φ necessarily holds. The meaning of $B(i,\phi)$ is that agent i believes ϕ. $I(i,\alpha)$ means that agent I intends to perform α and $I(i,\phi)$ means that agent i intends to bring ϕ about. Due to a lack of space we did not include temporal aspects in the logical language. One way to do this in a simple way is described in [WMW89].

The deontic operators are defined by the following abbreviations (cf. [Me88, WMW89]):

Definition 2.3

$$\begin{aligned} O_{ij}(\alpha(i)) &= [\overline{\alpha(i)}]Violation_{ij} \\ F_{ij}(\alpha(i)) &= [\alpha(i)]Violation_{ij} \\ &= O_{ij}(\overline{\alpha(i)}) \\ P_{ij}(\alpha(i)) &= \neg[\alpha(i)]Violation_{ij} \\ &= \neg F_{ij}(\alpha(i)) \end{aligned}$$

Where $Violation_{ij}$ are special predicates indicating the violation of an agreement between i and j. So, the agent i is obliged to agent j to perform the action $\alpha(i)$ if not doing $\alpha(i)$ by i leads to a violation of i with respect to j, i is forbidden to do $\alpha(i)$ by j if doing $\alpha(i)$ leads to a violation of i with respect to j. i is permitted to do $\alpha(i)$ by j if i is not forbidden to do $\alpha(i)$ by j.

In order to model the communication between agents in a normative system the language L_{dd} has to be extended to incorporate speech acts as described in illocutionary logic. Before we introduce the speech acts, first we introduce two special relations involving agents. One relation implements a power relation between two agents and the other one implements an authorization of an agent to perform some action.

The power relation is the most primitive relation of the two. There exists a power relation between the agent i and the agent j with respect to action α, if i has the power to order j to perform the action α. For instance, the boss can order his secretary to type a letter for him. Note that he might not have the power to order his secretary to make coffee for him! We assume that the power relation is persistent and is only changed in special occasions, like when a manager is appointed.

The power relation can also be defined with respect to a proposition. This means so much as that i has the power to convince j of the truth of ϕ. For instance, a student will (usually) consider the statements of a teacher to be true.

The power relation defines a partial ordering on the class of agents for every action α. This ordering is reflexive (self-power) and transitive but not necessarily total.

Notation: if i has power over j with respect to α we write: $j <_\alpha i$. If i has power over j with respect to the truth of ϕ we write $j <_\phi i$.

The second relation is the authorization relation. This relation can be established for a certain time with mutual agreement (under certain restrictions). For instance, I can agree that a company can order me to pay a certain amount of money after they delivered a product. This relation ends after I pay the money. The authorization relation is modelled using a special predicate.

Notation: if i is authorized to do α we write: auth(i,α).

We will now continue by extending the language of actions to include the speech acts. A speech act is formalised as an illocutionary point (indicating the goal of the speech act) with three parameters: the Speaker, the Addressee, and the content. We distinguish the following basic speech acts (based upon [SV85, Aus62]):

Definition 2.4
DIR(i,j,α) – i does a request to j for α
COM(i,j,α) – i commits himself to j to do α
ASS(i,j,ϕ) – i asserts to j proposition ϕ
DECL(i,j,ϕ) – i declares and informs j that ϕ holds from now on

From these basic speech acts we can construct other basic speech acts by e.g. using the logical negation of actions.

Definition 2.5
FOR(i,j,α) = DIR(i,j,$\overline{\alpha}$) – i forbids j to do α
PER(i,j,α) = DECL(i,j,$P_{j,i}(\alpha(j))$ – i permits j to do α

There might be some dispute over the question whether the declarative DECL has an Addressee parameter, since if it succeeds, the effect will be a change of the world and not of the knowledge of the Addressee only. Depending on the preparatory conditions, it is not necessary that there is an Addressee at all. However, in general it makes little sense to do a declarative speech act and not inform anybody. Hence the Addressee should be understood here as the agent (or set of agents) that is informed.

Declaratives can only be used for specific institutionalized speech acts, so the propositional content is usually rather restricted. In practice, a limited number of specific declaratives will be distinguished, such as the "authorization" action that we will introduce in section 2.2.

Speech acts can be grounded in three different ways: charity, power and authorization. (See also [DW92] for a similar distinction as an improvement to the completely power based CSCW tool Coordinator ([FGHW88])). For instance, a directive (DIR) can be made on the basis of charity, which means it is a request, or on the basis of a power relation or authorization (in which cases it is an order). Hence for each basic speech act we distinguish three variants, indicated by a subscript c,p or a. So, DIR_a stands for an authorized request, whereas DIR_p stands for an order based on power. Similarly for assertives and declaratives. For commissives, the distinction seems to be not very relevant and we ignore it here. Likewise, when the distinction of the powerbase of a speech act is not important, we will ignore the subscript.

The language of all acts is now defined in two steps. First we define the set of all speech acts L_{Sact}.

Definition 2.6
1. All basic speech acts are elements of L_{Sact}.
2. If $\alpha \in L_{Sact}$ then also $IP(i,j,\alpha) \in L_{Sact}$ and $IP(i,j,\overline{\alpha}) \in L_{Sact}$ where $IP \in \{DIR, COM\}$

Note that this is a recursive definition. So, we can have speech acts about speech acts, etc.
The language of actions L_{ACT} can now be defined as:

Definition 2.7

$$L_{ACT} = L_{act} \cup L_{Sact}$$

The speech acts of L_{Sact} as defined above do not contain all elements of speech acts that are identified in illocutionary logic.
The propositional content conditions are not modelled at this moment. They can be modelled through a refinement of the language L_{Sact} which renders only those speech acts syntactically correct that comply to the propositional content conditions. In an Information System environment, the propositional content conditions are contained in the data model.

The preparatory conditions (ϕ) and the intended effects (ψ) of a speech act (α) can be modelled through the following schema:

$$\phi \rightarrow [\alpha]\psi$$

Which means that if ϕ is true then ψ will hold after α has been performed. The intended effects of the speech acts are described by means of deontic and epistemic operators, while the preparatory conditions refer to either the authorization relation or the power relation. We have the following general preparatory conditions and intended effects for the basic speech acts. Of course, for speech acts mentioning specific actions there might be more conditions and effects.

Axiom 2.8
1. ($[DIR_p$ (i,j, $\alpha)]$ $O_{ji}(\alpha)) \leftarrow$ j $<_\alpha$ i
2. ($[DIR_a$ (i,j, $\alpha)]$ $O_{ji}(\alpha)) \leftarrow$ auth(i,DIR(i,j,α))
3. [COM(i,j,α)] $O_{ij}(\alpha)$
4. $[DECL_a$ (i,j,ϕ)] $\phi \leftarrow$ auth(i,DECL(i,j,ϕ))
5. $[DECL_p$ (i,j,ϕ)] $\phi \leftarrow$ j $<_\phi$ i
6. ($[ASS_a$ (i,j,ϕ)] B(j,ϕ)) $\leftarrow$ auth(i,ASS(i,j,ϕ))
7. ($[ASS_p$ (i,j,ϕ)] B(j,ϕ)) $\leftarrow$ j $<_\phi$ i

The last of the above properties expresses the fact that a person can be authorized to assert some facts. If this person asserts such a fact the effect is that the Addressee(s) will believe that fact (which is not the same as making the fact true, which happens with a declaration!).

The axioms describe the effects of power and authorization speech acts, but not of those based on charity. This is correct, although we might add some politeness rules that say that a message is always replied. For example, a request based on charity would be replied by either a commissive or an assertion of the effect that the agent does not commit himself:

$$[DIR_c \text{ (i,j, } \alpha)]\ O_{ji}(COM(j,i,\alpha) \cup ASS(j,i,\neg O_{ji}(\alpha))))$$

In a formal context we assume that an agent is always sincere and thus we have:

Axiom 2.9
[DIR(i,j,α)] I(i,α) – any DIR speech acts expresses that i intends α to happen
[DECL(i,j,ϕ)] I(i,ϕ) – any DECL speech acts expresses that i intends to bring about ϕ (by the speech act)
[ASS(i,j,ϕ)] B(i,ϕ) – any ASS speech act expresses that i believes ϕ

So the effect of a DIR_c is at least that the agent knows about the subjects intention, and this can trigger him to commit himself.

2.2 The dynamics of authorization

If the subject is not authorized, it can not issue a DIR_a speech act sucessfully. In that case, it can try to attain an authorization first. This can be done by means of a DIR_c (i,j,DECL(j,i,auth(i,DIR(..)))) , that is, a request for authorization of the other party. If the other party complies to the request, that is, grants the authorization, the subject gets authorized from that time on. This example shows that the system should not only formalize authorized behavior itself, but also the creation of authorizations, and, for that matter, the deletion. The crux of the formalization is that authorizations can only be made and retracted by an act of the other party. Because the establishment of authorizations is an important and frequently occurring speech act we introduce the following notation:

$$\text{AUT(i,j,}\alpha) == DECL_a\text{(i,j,auth(j,}\alpha))$$

So, AUT(i,j,α) means that i gives authorization to j to do α. Of course, this speech act is only successful if i is authorized to give this authorization. For that reason, we have to presuppose the following axiom:

Axiom 2.10
1. auth(i,AUT(i,j,DIR$_a$(j,i,α(i))))
2. auth(i,AUT(i,j,ASS$_a$(j,i,p)))

that says that each agent is authorized to authorize other parties as far as actions and beliefs of the agent himself are concerned. This is irrespective of whether the granting of authorizations is forbidden by for example a higher power. If that would be the case, the authorization would still be successful, although the agent might be punished for it.

The ability to retract an authorization should be left to the subject of the authorization. If i has granted j an authorization, it is only j who can retract the authorization. For this purpose, we introduce a new declarative RTR:

$$\text{RTR(i,j,}\alpha\text{)} == DECL_a\text{(i,j,}\neg\text{auth(i,}\alpha\text{))}$$

The preparatory condition of RTR is that the authorization does exist. By axiom, every agent is authorized to retract authorizations given to him. If an agent has first granted an authorization, and then wants to retract it, he must ask the other party to do so. Of course, the agents may have made appointments. For example, the agent who grants the authorization may ensure himself of the authorization to request the retracting. The effect is that he can have the authorization retracted whenever he wants.

2.3 "Ordering" in logic

In this section we will show how the example given in section 1, about ordering products, can be formally described in the logic developed sofar. Each of the arrows from figure 1 (i.e. each of the messages between the customer and the company) is modelled with a logical formula. The whole contract including the request for quotation can be modelled by the following formulas:

1. $[DIR_c$(i,j,give-quotation(j,i,g,p))$]O_{ji}$(give-quotation(j,i,g,p) $\cup$ refuse(j))
   After a request for a quotation (i.e. a directive based on charity) the company is obliged to give the quotation or send a refusal. Here we assume some business rule that such a request is always answered.
2. [give-quotation(j,i,g,p)] auth(i,DIR_a(i,j,deliver(j,i,g,p)))
   If a company gives a quotation for a certain price (p) the client is authorized to order the product (g) for that price.
3. auth(i,DIR_a(i,j,deliver(j,i,g,p))) $\rightarrow$
   $[DIR_a$(i,j,deliver(j,i,g,p))$](O_{ji}$(deliver(j,i,g,p)) $\wedge$
   [deliver(j,i,g,p)]auth(j,DIR_a(j,i,pay(i,j,p))))
   If a customer is authorized to order a product for a certain price (i.e. a quotation has been given for that price) then the company is obliged to deliver the product after the customer has ordered it. (This follows directly from the axioms 2.8.) But after delivery of the product, the company is authorized to order the customer to pay for it.
4. O_{ji}(deliver(j,i,g,p)) $\rightarrow$ [deliver(j,i,g,p)]$\neg$auth(i,DIR_a(i,j,deliver(j,i,g,p)))
   If a company has to deliver a product and actually does it, the customer is no longer authorized to request delivery of the product. (One might omit this formula or replace it with a formula that limits the validity of the quotation to a period of time).
5. auth(j,DIR_a(j,i,pay(i,j,p))) $\rightarrow$ $[DIR_a$(j,i,pay(i,j,p))$]O_{ij}$(pay(i,j,p))
   If an order has been delivered (and authority acquired to request payment) a request for payment induces an obligation for the customer to pay. (This follows directly from the axioms 2.8.)

6. O_{ij}(pay(i,j,p)) → [pay(i,j,p)]¬auth(j,DIR_a(j,i,pay(i,j,p)))
   Finally, after the customer has paid, the company cannot request another payment again.

Although the above formulas describe the exchange between the customer and the company exact and complete, they are not very readable. In the next section we will show a language that is based on the logic introduced in this section, but has a more readable syntax and structures the (speech) acts to form transactions and services.

3 Contracts and communication protocols

In this section we will give an overview of the formal specification language CoLa (Communication Language), which is based on the logic described in section 2 and on the language presented in [We93]. Due to space limitations, this overview will not be extensive neither complete, but will be given by illustrating how the example from section 1 can be modelled in this language.

The protocols that are specified in CoLa are independent from the applications, but in contrast to traditional communication protocols, they capture the complete communication logic, not just a set or ordered set of messages. The contracts specified in CoLa are managed by a Contract Manager, which is a process responsible for the proper dealing with the communication, including (a) the transaction management, including compensation; (b) the set-up of new contracts, and adaptation of active ones. It follows that these aspects therefore do not have to be specified in the contracts themselves. For instance, we do not have to specify in the contract what happens if a message does not arrive through some hardware failure. Handling this kind of communication problems is typically a task for the Contract Manager and can be specified (generically) in that place. We will not consider the Contract Manager in this paper, but assume it to be present.

The contracts as specified in CoLa are units of cooperation between two or more agents, and have a three-level organization: messages, transactions and services.

Messages are defined using primitives such as request,assert, and authorize. These message types have pre-defined semantics as described in section 2, in terms of obligations and authorizations.

The agents that are involved in the contract and the messages that each of them uses as explicit speech act during the execution of the contract are specified at the start of the contract. The messages that are not defined here are implemented as implicit speech acts executed through some other action. For instance, the commitment of the company to deliver a product in this contract is not modelled as an explicit speech act. It is only implicit through the actual delivery of the product. Note that this is a choice made for this contract indicating that in this case there is no need to send a message indicating that the company commits itself to deliver the product.

For our example the explicit speech acts are as follows:

```
agents X: customer; Y: company

MESSAGES(X to Y)

 request_quotation == quotation.request_c;
 place_order/3     == request_a(partno,quantity,price);
 place_order/2     == order.request_c(partno,quantity);

MESSAGES(Y to X)

 refuse_quotation  == quotation.refuse;
 give_quotation    == quotation.authorize;
 refuse_deliver    == order.refuse;
 request_payment   == pay.request_a;
```

The messages that are specified here correspond to the speech acts that are used in the contract (either explicitly or implicitly). For instance, "request_quotation" is defined as a request_c (corresponding to DIR_c in the logic), which is defined local within the transaction "quotation". We do not specify all actions that are used in the contract, but only the speech acts. So, the actions deliver and pay are not specified at this place!

Messages that are aimed at the establishing (or adapting) a certain deontic statement (that is, a conjunction of obligations and authorizations) are grouped together in transactions. A typical example of a transaction is a request of the client followed by a commit, or refuse, of the server. For each transaction, there is a successful termination, indicated by the goal of the transaction, and non-successful terminations. In the example we have the following transactions:

```
TRANSACTIONS
transaction quotation
  isa get_authorization(order(Partno, Quantity, Price));

transaction get_authorization(Alfa:action)
messages
    request_c(auth(Alfa));
    authorize(Alfa), refuse(authorize(Alfa));
goal
    authorize(Alfa)
end-transaction;
```

The above states that the transaction "quotation" is a specialization of the transaction "get_authorization". This generic transaction is defined as a request (based on charity) to get authorization, followed by an authorization or a refusal.

```
transaction order
messages
    request_c(deliver(Partno,Quantity));
```

```
    commit(deliver(Partno,Quantity,Price)),
    refuse(deliver(Partno,Quantity));
    accept(commit(deliver(Partno,Quantity,Price)))
goal
    {commit(deliver(Partno,Quantity,Price)),
        accept(commmit(deliver())) }
end_transaction;
```

The "order" transaction is defined for the cases where no quotation was given before. The request to deliver a product is therefore based on charity. The company then commits itself to deliver the product against a certain price (or refuses to deliver) after which the price has to be accepted by the customer.

```
transaction delivery (O: order)
messages
    declare(delivered(O));
    accept(delivered(O));
goal
    accept(delivered(O))
end-transaction;

transaction invoice(O: order)
messages
    request_a(transfer_money(O));
    commit(transfer_money(O));
goal
    commit(transfer_money(O))
end_transaction;
```

Finally, transactions that deal with the same deontic statements are grouped together in one service. The service, centered around an object, specifies how the object - the deontic statement - can be created, how it can be removed and, possibly, how it can be modified. In general, there are several ways in which the object can come into being or is removed. In the service, it is possible to specify also the consequences of a certain action, such as sanctions for not satisfying the obligation.

```
SERVICES
service quote
provided by Y
object
    auth(X,place_order/3)
attributes
    Partno: num(9);
    Quantity: integer;
    Price: money;
    Expiration_date: date
```

```
created by
    quotation
removed by
    expire(Expiration_date)
    withdraw            => {auth(X,request(sanction)) };
end-service;
```

The service "quote" is given by the company to the customer. It is centered around the authorization of the customer to order a product for a certain price. The authorization pertains to a certain product, quantity and price that are given in the attributes. The expiration date indicates when the offered quotation expires. The authorization is created by the quotation transaction and removed by the passing of the expiration date. It can also be removed by a withdrawal of the company, but this gives the customer the authorization to request some compensation.

```
service supply
provided by Y
object
    obl(Y,deliver)
attributes
    Partno: num(9);
    Quantity: integer;
    Price: money
created by
    place_order(Partno, Quantity, Price)
    order(Partno,Quantity)
removed by
    delivery        => {auth(Y,request_payment)};
    cancel_order    => {obl(X,pay(fine)) };
end-service;
```

The obligation for the company to deliver a product arises either directly from an authorized order message (after a quotation has been made) or from the order transaction (an order without previous quotation, followed by a commitment of the company).

```
service payment
provided by X
object
    obl(X,pay)
created by
    invoice
removed by
    transfer_money
    expire(date)    => {obl(X,pay(fine)) }
end-service
```

The last "service" of the contract concerns the payment of the order. After the delivery of the products, the company is authorized to request payment. This is done through the transaction "pay". This transaction always ends with an obligation of the customer to pay if the company is authorized to request payment. Note that if the message "request_payment" is used to create the obligation directly, this would result in an obligation to pay for the customer whenever the company asks for it, which is obviously not what we want.
The obligation to pay is not directly created by the delivery of the product, but arises from a request for payment which is authorized through the delivery of the product. This construction makes the contract more flexible than usually is the case where delivery of a product means a direct obligation to pay for it.

We now have described all the components of the contract in CoLa. The contract itself is the set of messages, transactions and services together.
It can be seen from the above that CoLa provides a means to describe the communications between agents and the resulting obligations and authorizations in a structured way, which makes it very suitable to describe contracts as given in the example above.

4 Conclusions

Although deontic logic has been applied in the field of Information Systems before, the dynamics of normative systems has received almost no attention. In this paper we have explored the way how deontic statements are created and adapted in communication processes, and the role they play in the regulation of communication itself. We have distinguished three different perspectives, corresponding with three validity claims that communicative agents can make: charity, authorization, power. As in human social systems, these perspectives can complement each other in the organization of interoperable computer systems.

We have shown how authorizations for specific acts can be requested, granted and also retracted, thereby creating a dynamic environment for establishing and derogating authorized norms.

Contracts were presented as a way of modularizing the normative specifications. The contracts can be described in CoLa, which provides a structured high level way to specify the contracts.
Both the negotiation phase to establish the contract as well as the contract itself can be modelled using our formalism.

Several open issues are left for further research. One is the use of conditions. In the deontic/illocutionary logic given in section 2, all deontic statements were unconditional. In practice, a certain obligation or authorization only obtains under certain conditions: a specific event, a specific time or time slot etc. In the directive that creates a certain deontic statement, an extra parameter "condition" should be added. We have not worked this out yet.

A second topic is the delegation and inheritance of authority. Under which conditions is it possible to delegate the authority to perform a certain action? Under which conditions is authority inherited through a power relationship?

A third topic of future research is the elaboration of contracts. In this paper, a contract is a set of norms. A contract gives rise to a certain communication process or protocol. Such a protocol can be implemented in a computer-computer or human-computer interface. Protocols are based on the essential speech acts given in the contract, but usually contain other speech acts as well, such as acknowledgement and reminding. A given protocol can be tested for completeness with respect to the basic contract, and also for internal completeness: whether every communicative obligation that is created, or can be created, is materialized in a possible communicative action.

A last topic for further research that we want to mention here is the relation between different contracts. One contract can be a specialisation of another contract or an extension of that contract. Also an agent can have contracts with different parties at the same time. We should be able to prove that these contracts are mutually consistent.

References

[Aus62] J.L. Austin *How to do things with words* Oxford: Clarendon Press, 1962.

[DW92] J.L.G. Dietz and G.A.M. Wiederhoven A comparison of the linguistic theories of Searle and Habermas as a basis for communication supporting systems *Linguistic Instruments in Knowledge Engineering (Riet, R.P van de, and R.A. Meersman (eds)* North-Holland, 1992.

[FGHW88] F. Flores, M. Graves,B. Hartfield,T. Winograd Computer Systems and the Design of Organizational Interaction *ACM Trans. on Information Systems* Vol.6, No.2, 1988.

[Lee88] R. Lee Bureaucracies as Deontic Systems, *Trans. on Office Information Systems* Vol.6, No2, 1988, pp87-108.

[LL86] E. Lehtinen, K. Lyytinen Action Based Model of Information Systems *Information Systems*, Vol.12, No3, 1986, pp299-317.

[Me88] J.-J.Ch. Meyer A different approach to deontic logic: deontic logic viewed as a variant of dynamic logic, *Notre Dame Journal of Formal Logic* 29(1), 1988, pp.109-136.

[MW93] J.-J.Ch. Meyer and R. Wieringa (eds.) *Deontic Logic in Computer Science* Wiley Professional Computing, 1993.

[Po93] R. Power *Cooperation among organizations : the potential of computer supported cooperative work* Springer, Berlin. 1993.

[SV85] J.R. Searle and D. Vanderveken *Foundations of illocutionary logic* Cambridge University Press. 1985.

[Se69] J.R. Searle *Speech Acts* Cambridge University Press. 1969.

[We93] H. Weigand Deontic aspects of communication *Deontic Logic in Computer Science (Meyer, Wieringa eds)* Wiley Professional Computing, 1993.

[WMW89] R.J. Wieringa, J.-J.Ch. Meyer and H. Weigand *Specifying dynamic and deontic integrity constraints* Data & Knowledge Engineering 4, 1989, pp.157-189.