Communication for Goal Directed Agents

Mehdi Dastani
Institute of Information and Computing Sciences
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands
mehdi@cs.uu.nl

Jeroen van der Ham
Institute of Information and Computing Sciences
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands
jham@cs.uu.nl

Frank Dignum
Institute of Information and Computing Sciences
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands
dignum@cs.uu.nl

ABSTRACT

This paper discusses some modeling issues concerning the communication between goal-directed agents. In particular, the role of performatives in agent communication is discussed. It is argued that the specification of the effect of performatives, as prescribed by FIPA, is sometimes too weak or unrealistic. The alternative, proposed in this paper, suggests a two phase modeling of the effect of the communication. A minimum effect can be hardwired in the semantics of sending and receiving messages. And the performative related part is achieved by executing a number of rules which are under the control of the agent and made accessible to the agent programmer. These issues are discussed in the context of 3APL which is a goal directed agent programming language.

1. INTRODUCTION

In any multi-agent system the communication between the agents will be an important aspect. Some work has been done on the formalization of the communication (see e.g. [1]). Most of the early work has concentrated on formalizing the messages, such that a precise and unambiguous meaning of them could be established. (see e.g. [2, 3]). However, one of the main problems was (and still is) to determine the exact effects of a message. If we look at the specification of the "inform" message in the FIPA agent communication language (ACL) it states only a precondition that the sender should believe the contents of the inform message and do not believe that the receiver has knowledge about the content already and a rational effect (which should be interpreted as an intended effect of the inform) that the receiver will believe the content of the inform message. FIPA does not, however, give a formal specification of what a "rational effect" exactly means. It is not a direct consequence of performing the communicative act, but seems to be more like a goal of the sender of the message.

On the other hand the FIPA specification states informally that the receiver is "entitled" to believe that the sender believes the contents of the inform message and wishes the receiver to believe the contents of the inform message as well. However, this effect is not specified in the formal specifications of the message. It is therefore unclear whether these points could/should be seen as effects of sending the message. The above example quite clearly shows some of the major problems in formalizing agent communication. Although the FIPA ACL specification already formalizes some aspects of the ACL it also contains some gaps at crucial points. Most of these gaps are related to the exact preconditions and postconditions of the communicative act. It has been argued in [7] that preconditions of a communicative act in which the sender is supposed to have knowledge about the state of mind of the receiver are not very realistic. These preconditions can never be checked, because the sender cannot verify whether they actually hold (she cannot "look inside the head of the receiver"). Therefore the preconditions are relatively weak.

The postconditions are also difficult to specify. As in the example above, the sender certainly has an intended purpose with sending the message, but one cannot guarantee that this purpose is actually achieved. This depends for a large part on the receiver, which is autonomous. So, the effect of receiving a message is for the largest part determined by the receiving agent. If we want to give strict postconditions for a communicative act this would also pose heavy constraints on the way agents would have to handle messages and the mental updates they have to make. This seems overly restrictive to be practical.

The crux of the matter seems to lie in the balance between the autonomy of the agents on the one hand and the wish to predict the effects of a communicative act on the other hand. The first is of prime importance for two reasons. First, because autonomy is one of the most important characteristics of agents. Secondly, in open agent systems one cannot predict how other agents work internally and therefore are seemingly completely autonomous. However, one also would like to give precise semantics for the messages and their effects in order to standardize agent communication and for agents to be able to reflect about communicative acts.

In this paper we explore the balance between the autonomy of the agents and agent communication in the practical setting of the agent programming language 3APL. We give a short overview of 3APL and its semantics in the next section. In section 3 we indicate the issues that have to be dealt with in order to extend 3APL with a communication component (without solving issues in the implementation in a way that is not covered by the formal semantics of 3APL). In section 4 we show how the practical reasoning rules of 3APL can be used to add (more restrictive) effects to the communicative acts in a stepwise way. This allows the programmer to implement an agent, which minimally fulfills the FIPA specification, to draw more elaborate conclusions. In section 5 we will some preliminary conclusions and indicate areas for further research.

2. 3APL SPECIFICATION

3APL is an implementation language for cognitive agents that have beliefs and goals as mental attitudes and can revise or modify their goals. Moreover, 3APL agents are assumed to be capable of performing a set of basic actions such as sensing actions or mental updates. Each basic action is defined in terms of pre- and post-conditions and can be executed if the pre-condition is true. After the execution of a basic action the post-condition is set to be true. For example, a 3APL agent may have a goal to buy a computer and thereafter buy a book. The agent has the capability of buying computers and books (basic actions). The agent may believe he has not enough money to buy the computer but enough to buy the book. The agent can also delay the purchase of the computer if he believes he has not enough money by doing other things first.

A 3APL agent starts its deliberation with a number of goals which it aims to achieve. If the goals are basic actions for which the pre-conditions are true, then the goals are achieved by executing the basic actions; otherwise the goals are revised. In the above example, a 3APL agent aims at buying the computer first, but it realizes that he has not enough money. Therefore, he delays the purchase of the computer and buy the book first. In the rest of this section, we will briefly explain the formal syntax and semantics of 3APL. The complete formal specification of 3APL is described in [5]. We introduce only the minimum definitions needed to explain the working of the agents and the links with the communication between the agents. Those who are familiar with the 3APL specification can skip this section.

2.1 3APL Syntax

A 3APL agent is specified in terms of its beliefs, goals, and goal revision rules. These rules are called practical reasoning rules. The entities are defined as follows.

DEFINITION 1 (BELIEFS). *Given a set of domain variables VAR and functions (constants are defined as zero-place functions), the set of domain terms is defined as usual. Let $t_1, \ldots, t_n$ be terms referring to domain elements and $Pred$ be a set of domain predicates, then the set of programming constructs for belief formula, BF, is defined as follows:*

- *if $p \in Pred$, then $p(t_1, \ldots, t_n) \in BF$.*
- *if $\varphi, \psi \in BF$, then $\neg\varphi, \varphi \wedge \psi \in BF$.*

All variables in the BF formula are universally quantified with maximum scope.

Agents are assumed to be able to perform a set of basic actions. These actions, which are also called basic capabilities, are parameterized actions.

DEFINITION 2. *Let T be the set of terms denoting domain elements and $A_1, \ldots, A_n$ be action symbols. Then, $A_i(t_1, \ldots, t_m)$ is a parameterized basic action. The set of basic actions is denoted by $Bact$.*

Goals in 3APL can be best understood as partial plans which should be performed by the agents. The goals are based on action operators, primitive actions, or other plans. We also assume a set of goal variables, $GVAR$ ($GVAR \cap VAR = \emptyset$) that can be used to refer to unidentified goals.

DEFINITION 3 (GOALS). *Let $Bact$ be a set of basic actions, BF be a set of belief sentences, $\varphi \in BF$, and $\pi_1, \pi_2 \in G$. Then, the set of 3APL goals (G) are defined as follows:*

- *$GVAR, Bact, BF \subseteq G$*
- *$(\varphi?)$, $(\pi_1; \pi_2)$, $(\pi_1 + \pi_2)$, $(\pi_1 || \pi_2)$ $\in G$.*

The first clause states that belief formula can be considered as goals. This type of goals are called achievement goals and they function as a kind of procedure names. The corresponding procedures are themselves arbitrary goals. The procedure call is then modelled by practical reasoning rules which is explained later. We assume that a belief formula cannot simultaneously be an achievement goal and part of the belief-base.

Practical reasoning rules are at the heart of the functioning of 3APL agents. For each goal of the agent, that is not a basic action, there should be a practical reasoning rule indicating how the goal can be achieved. However, the practical reasoning rules can also be applied to revise agent's goals that are not achievable, basic actions that are blocked, to optimize agent's goals, to generate some sort of reactive behavior, or to define achievement goals (i.e. procedures).

DEFINITION 4 (PR-RULES). *Let $\varphi \in BF$ and $\pi_h, \pi_b \in G$, then a practical reasoning rule is defined as follows: $\pi_h \leftarrow \varphi \mid \pi_b$. This rule can be read as follows: if the agent's goal is π_h and the agent believes φ, then π_h can be replaced by π_b.*

The 3APL agent functions as follows: At each cycle the agent checks what its current goal is and tries to execute the goal if it is a basic action or find a practical reasoning rule whose head can be unified with the current goal and whose guard is true. If the agent succeeds in finding such a rule, it rewrites the current goal to the body of the rule (given the unification of the variables in the goal and the rule). This process continues until the agent has no more goals left or no rule is applicable.

Given the definition of beliefs, goals and practical reasoning rules, a 3APL agent can be specified as follows:

DEFINITION 5. *A 3APL agent is a triple* $< \Pi, \sigma, \Gamma >$*, where* Π *is a set of goals,* σ *is a set of belief formula, and* Γ *is a set of practical reasoning rules.*

The following is an example of a 3APL agent.

EXAMPLE 1. *Let* $A()$ *be a basic action with pre-condition* $\neg p(a)$ *and postcondition* $p(a)$ *(i.e.* $\{\neg p(a)\}\ A()\ \{p(a)\}$ *), and* $B()$ *be an action with precondition* $p(a)$ *and postcondition* $\neg p(a)$ *(i.e.* $\{p(a)\}\ B()\ \{\neg p(a)\}$ *). The following agent has one goal which is "first do* $A()$ *and then* $B()$*". The agent also believes* $p(a)$*, and has a goal revision rule which states that whenever it has to do* $A()$ *and after that something else,* (X) *and also believes* $p(a)$ *(i.e. the precondition of* $A()$ *is not satisfied), then it delays the execution of* $A()$ *and does* X *first.*

$$\begin{array}{ll} < & \Pi = \{\ A(); B()\ \}\ , \\ & \sigma = \{\ p(a)\ \}\ , \\ & \Gamma = \{\ A(); X \leftarrow p(a) \mid X; A()\ \}\ \ > \end{array}$$

2.2 3APL Semantics

In [5] an operational semantics for the 3APL language is proposed which is defined by means of a transition system. This semantics specifies transitions between the agent's states by means of transition rules. The state of an agent is defined as follows:

DEFINITION 6. *The state of a 3APL agent is defined as a 4-tuple* $< \Pi, \sigma, \Gamma, \theta >$*, where* Π *is the set of the agent's goals,* σ *is the agent's beliefs,* Γ *is the set of practical reasoning rules, and* θ *is a substitution consisting of binding of variables that occur in the agent's beliefs and goals.(In the sequel we leave out* Γ *from the agent's states since this part does not change by transitions.)*

The binding of variables θ is generated and updated by transitions such as those responsible for the execution of test goals and the application of practical reasoning rules. The variable binding θ is passed through to other transitions. For the application of practical reasoning rules the guard of the rules that contain domain variables should be unified with a belief formula. Also, the heads of the rules containing both domain variables and goal variables should be unified with the goal-base. Both of these unifications result in substitutions which need to be passed to the next states. The set Π is a set of goals from which one goal can be executed. The transition rule on the agent level specifies the execution of a goal from Π.

DEFINITION 7. *Let* $\Pi = \{\pi_0, \ldots, \pi_i, \ldots \pi_n\} \subseteq G$*, and* θ *and* θ' *be ground substitutions. Then,*

$$\frac{< \{\pi_i\}, \sigma, \theta > \rightarrow < \{\pi_i'\}, \sigma', \theta' >}{< \{\pi_0, \ldots, \pi_i, \ldots \pi_n\}, \sigma, \theta > \rightarrow < \{\pi_0, \ldots, \pi_i', \ldots \pi_n\}, \sigma', \theta' >}$$

The above transition rule states that transitions of sets of goals can be accomplished by transitions on each goal separately. Thus, the agent develops by transitions and revision of each goal in its goal-base. The application of practical reasoning rules to the goal-base may introduce new occurrences of existing variables in the goal-base and therefore may cause implicit unwanted bindings between variables. These bindings can be prevented by assuming fresh variable names which replace all variables that occur in a practical reasoning rules before they are applied to the goal-base. For more details on variable bindings see [5].

Additional transition rules are given for the application of practical reasoning rules on a goal or the execution of a goal when it consists of a basic action. The transition rule for basic actions is formulated as follows.

DEFINITION 8. *Let* $A(\overrightarrow{t})$ *be a basic action with a list of parameters* $\overrightarrow{t}$ *and* τ *be an update function that specifies the effect of the basic action on beliefs. The semantics of basic actions is captured by the following transition rule.*

$$\frac{\tau(A(\overrightarrow{t})\theta, \sigma) = \sigma'}{< \{A(\overrightarrow{t})\}, \sigma, \theta > \rightarrow < \emptyset, \sigma', \theta >}$$

The complete set of transition rules can be found in [5].

3. EXTENDING 3APL WITH COMMUNICATION

The specification of 3APL agents is basically designed for single agents. As a consequence there is no account for agent communication. There are two ways to extend 3APL specification to account for communication. The first way is to extend the set of basic actions with synchronous communication actions such as `tell`(ψ) and `ask`(φ) or `offer`(ψ) and `require`(φ). The arguments φ and ψ of the synchronized actions are unified and the resulting variable bindings are the result of the communication and form the contents of the messages. In these approaches, the synchronized actions are matched on the basis of the performatives they enact. I.e. the `tell` action is a speech act with performative 'tell'. It should be matched with a speech act with the complementary performative 'ask'. This account of agent communication is proposed in [4, 6].

The above synchronous communication approach has as its big disadvantage that we would have to categorize all performatives in pairs that should be synchronized. Although a pair like `tell` and `ask` looks very natural it is difficult to model, e.g. all FIPA-ACL performatives in such a way. Moreover, in many multi-agent applications it is not realistic to determine the unique pairs of performatives beforehand as the agents need to deliberate on how to respond to a certain message. For example, when an agent receives a request it needs to deliberate on the content of the message to decide whether it reacts with an agree, refuse or not-understood message. If we consider all the different pairs possible then it becomes impossible to use the synchronous model described above. In that case we have to indicate with a particular instance of the `request` action which of the three possible answers should synchronize with this request.

I.e. we determine the answer on forehand when sending the request.

As a final disadvantage we want to mention that not all communication takes place in pairs as indicated above. Sometimes an agent just wants to inform another agent of some fact without the other agent having explicitly asked for the fact. These 'spur' messages are not be possible in the above synchronous model.

In this paper we propose an alternative approach to incorporate communication in the 3APL framework. The main feature is that it is based on asynchronous communication and supports modeling of FIPA-ACL performatives separately from the sending and receiving of the messages. In this approach, 3APL agents send and receive messages containing contents compliant to the FIPA specification. I.e. the message contains a message identifier, an explicit performative, the content, and the sender and receiver identifiers. In order to model asynchronous communication in the 3APL framework, the 3APL specification is extended with a buffer, called a message-base. The message-base of a 3APL agent contains messages that either are sent to other agents or are received from other agents. The message-base makes it possible for an agent to continue with its own goals after sending a message. It does not have to wait for the receiving agent to synchronize before it continues. In the same way, the receiving agent can receive messages at any time and does not have to form a goal to receive a message.

3.1 Communication Actions

Agents communicate by means of sending and receiving messages consisting of a message identifier, sender and receiver identifiers, a performative, and content information. The content information can be beliefs, basic actions, or goals. Sending and receiving messages is considered as communication actions (not basic actions) in the 3APL framework. The goal language, as defined in Definition 3, is extended to include these types of actions.

DEFINITION 9. *The set of possible 3APL goals G is extended with two communication actions* $\texttt{Send}(\iota, \alpha, \beta, \rho, \psi)$ *and* $\texttt{Receive}(\iota, \alpha, \beta, \rho, \psi)$, *where ι is a term that denotes the message identifier, α and β are terms denoting the identifiers for sender and receiver of the message, ρ denotes a FIPA performative, and ψ is the content information.*

We also extend the definition of an agent state in 3APL by including a message-base in the state.

DEFINITION 10. *The message-base Ω of an agent is a set consisting of sent and received messages. The elements have the format:* $\texttt{Sent}(\iota, \alpha, \beta, \rho, \psi)$ *and* $\texttt{Received}(\iota, \alpha, \beta, \rho, \psi)$, *where the arguments are the same as described in definition 9. The state of a 3APL agent can now be defined as a 5-tuple $< id, \Pi, \sigma, \theta, \Omega >$, where id is the agent's identifier, Π is the set of agent's goals, σ is agent's beliefs, θ is a substitution consisting of binding of first order variables that denote domain elements, and Ω is the message-base.*

We use a synchronization mechanism for sending and receiving messages. But this synchronization mechanism only takes care of simultaneously taking the messages from the sending agent's message-base and putting it in the receiving agent's message-base. At what time the receiving agent checks its message-base and how the messages are interpreted is treated in a completely asynchronous fashion.

This differs from other approaches, such as [4], since we do not synchronize performatives but only the sending and receiving actions of messages. As we see later, the exchange of information is based on unification of arguments of synchronized `Send` and `Receive` actions. The unification of the first four arguments of the communication actions is trivial since these arguments are terms. However, the fifth argument of these communication actions, the content, can be belief formula, basic actions, or even complex goals. For these complex objects unification is not defined standardly. For this reason, we introduce a set of so-called constraint variables that stand for belief formula, basic actions and complex goals, and assume that the fifth argument of one of the two synchronizing communication actions is such a variable. Therefore, the unification of arguments of synchronizing communication actions consists always of a pair $[X/c]$ where X is a constraint variable and c is the content information.

The semantics of `Send` and `Receive` is specified in terms of transition rules. The idea is that the sending agent removes the communication action `Send` from its goal-base after the execution of the send action, and stores the information concerning the sending action in its message-base. Although storing this information is not a part of the semantics of the send action, we believe that the agent may need this information for its future deliberation or its ongoing communications.

The receiving agent also stores its incoming messages in its message-base. In general, we assume that agents can receive messages at any moment. In fact, we assume that the goal of an agent a is of the form $\Pi \parallel \texttt{Receive}(\iota, a, \beta, \rho, \psi)$, which indicates that the agent is concurrently waiting to receive messages.

DEFINITION 11. *The following three transition rules specify the semantics for sending and receiving messages between agents and their synchronization, respectively.*

$$\frac{\varphi = < \iota, a, \beta, \rho, \psi >}{\begin{array}{c} < a, \Pi \cup \{Send(\varphi); \pi\}, \sigma, \theta, \Omega > \xrightarrow{\varphi!} \\ < a, \Pi \cup \{\pi\}, \sigma, \theta, \Omega \cup \{sent(\varphi)\} > \end{array}}$$

$$\frac{\varphi = < \iota, \alpha, b, \rho, \psi >}{\begin{array}{c} < b, \Pi \cup \{Receive(\varphi)\}, \sigma, \theta, \Omega > \xrightarrow{\varphi\tau?} \\ < b, \Pi \cup \{Receive(\varphi)\}, \sigma, \theta, \Omega \cup \{received(\varphi\tau)\} > \end{array}}$$

$$\frac{A \xrightarrow{\psi\tau?} A' \ , \ B \xrightarrow{\varphi!} B' \ , \ \psi\tau = \varphi}{M \cup \{A, B\} \rightarrow M \cup \{A', B'\}}$$

Note that in the second transition rule the concurrent receive action is not removed from the goal-base such that agents can continuously receive messages. Also note that the unification process takes care that an agent only receives messages (store them in his message-base) when they are sent to him. So, an agent does not have to check explicitly whether a certain message is meant for him.

4. INTERPRETING MESSAGES BY PRACTICAL REASONING-RULES

Once a message is stored in the message-base of an agent, it is interpreted by applying of practical reasoning rules. The effect of the specific performative in the message is thus achieved by the execution of practical reasoning rules of the receiving agent.

We can force the interpretation of the message (and thus the effects) to take place immediately by using so-called reactive rules. These rules do not have a head and are thus executed whenever the guard of the rule is true. In the case of practical reasoning rules pertaining to communication the guard of these rules is evaluated with respect to the message-base instead of the belief-base. For this reason, we index these rules with the label 'MB' to indicate that the guard should be evaluated with respect to the message-base rather than the belief-base. The rules have the following form:

$$\stackrel{MB}{\longleftarrow} \varphi \mid \pi$$

where φ refers to a received (or sent) message in the message-base and π will determine the effects of receiving (or sending) the message φ. E.g.

$$\stackrel{MB}{\longleftarrow} \mathit{received}(\iota, a, b, \mathit{inform}, p) \mid B_a(p)$$

which states that after an agent receives a message from agent a informing him that p, the agent will believe that agent a believes p (it assumes a to be sincere).

In many cases we do not want the agent to react immediately to the reception of a message. E.g. we only want an agent to react on an agree message if it is waiting for such a message. One could test a few of these things in the guard, but it seems more natural that some messages are handled as part of a protocol. The specification of the protocol in terms of subsequent goals of the agent leads to points where the agent will check its message-base for incoming messages. In order to model this situation we use (more general) rules of the following form:

$$\mathit{handle\_performative}(\overrightarrow{X}) \stackrel{MB}{\longleftarrow} \varphi \mid \pi$$

where $\mathit{handle\_performative}(\overrightarrow{X})$ can be used by the programmer to invoke the processing of a certain performative, π is the goal indicating how a message should be interpreted and φ is a logical formula that should be evaluated with respect to the message-base. The variable vector $\overrightarrow{X}$ is used to determine the parameters needed to execute a certain performative. We illustrate the use of these variables in the example given in the next subsection. Note that the programmer has to decide when messages with a certain performative should be processed since the head of the practical reasoning rule is not empty. Of course, these rules can also be reactive rules, in the sense that they can have empty head, such that the programmer has no influence on when a certain message is processed. Although both approaches are possible, we believe the programmer should be able to decide about the message processing.

Let us see how the process of interpreting a message in which the performative is a *request* can be specified by means of practical reasoning rules. According to the FIPA semantics for the request performative, a sending agent can request a receiving agent to perform an action. The receiving agent has to deliberate on the request to decide if it is granted or not. If so, the receiving agent sends an agree message to the sending agent, then performs the requested action, and finally informs the requesting agent that the action has been performed. This is achieved by applying the following rule.

$$\begin{array}{l}
\mathit{handle_request}(f) \stackrel{MB}{\longleftarrow} \\
\qquad \mathit{received}(\iota, \alpha, \beta, \mathit{request}, \mathit{Action}) \mid \\
\qquad (\neg f(\mathit{Action})?; \\
\qquad\quad \mathit{Send}(\mathit{reply}(\iota), \beta, \alpha, \mathit{refuse}, \mathit{Action})) \\
\qquad + \\
\qquad (\ f(\mathit{Action})?; \\
\qquad\quad \mathit{Send}(\mathit{reply}(\iota), \beta, \alpha, \mathit{agree}, \mathit{Action}); \\
\qquad\quad \mathit{Action}; \\
\qquad\quad \mathit{Send}(\mathit{reply}(\iota), \beta, \alpha, \mathit{inform}, \mathit{done}(\mathit{Action})))
\end{array}$$

This rule for request handling implements the protocol as shown in figure 1. The head of the rule is an achievement goal and has one argument, f. This function, $f : G \rightarrow BF$, maps a goal to a formula, which is considered as the constraint for granting requests. This function is used in test-goals, so that when used in combination with the choice (+), we can define what happens when an agent does not grant a request and when he does. In the first case he replies with a *refuse* message and goes his own way. In the latter case, he first replies with an *agree* message, executes the action and then informs the requester that the action has been done. Note that the *not-understood* message is not defined here, as it is only sent when the parsing of the message has failed.

4.1 An Example

In this section we show an example of a request conversation using the semantics as explained before. This example is not the trace of an execution of implemented agents, but it is constructed by hand to illustrate the expected trace. The semantics do not only follow the standard FIPA semantics, but also the standard FIPA request protocol as shown in Figure 1.

The conversation itself is in Table 1. This table has four columns, the first indicates the step-number, the second indicates the name of the mental states of the agents (GB:goal-base, MB:message-base) and the third and fourth column indicate the mental states of the communicating agents. For simplicity we assume that each agent executes one action at each step. This restriction is only for the ease of representation, the communication protocol does not require it.

During the conversation, the agents have the following belief-bases, unless noted otherwise: $\mathrm{BB}_a = \{$*salesPerson(b)*, *action(b,SellPC)*$\}$, meaning that agent a believes agent b is

		Agent a	Agent b
1.	GB:	buyPC(500)	startSelling()
2.	GB:	Send(1,a,b,request,SellPC(a,C,500))	handleRequest(f)
3.	GB:		handleRequest(f)
	MB:	sent(1,a,b,request,SellPC(a,C,500))	received(1,a,b,request,SellPC(a,C,500))
4.	GB:		(¬(f(SellPC(a,C,500)))?; Send(2,b,a,refuse,SellPC(a,C,500))) + (f(SellPC(a,C,500))?; Send(2,b,a,agree,SellPC(a,C,500)); SellPC(a,C,500); Send(2,b,a,inform,done(SellPC(a,C,500))))
	MB:	sent(1,a,b,request,SellPC(a,C,500))	received(1,a,b,request,SellPC(a,C,500))
5.	GB:		Send(2,b,a,agree,SellPC(a,c2,500)); SellPC(a,c2,500);
	MB:	sent(1,a,b,request,SellPC(a,C,500))	Send(2,b,a,inform,done(SellPC(a,c2,500)))
6.	GB:		SellPC(a,c2,500); Send(2,b,a,inform,done(SellPC(a,c2,500)))
	MB:	sent(1,a,b,request,SellPC(a,c2,500)), received(2,b,a,agree,SellPC(a,c2,500))	received(1,a,b,request,SellPC(a,c2,500)), sent(2,b,a,agree,SellPC(a,c2,500))
7.	GB:		Send(3,b,a,inform,done(SellPC(a,c2,500)))
	MB:	sent(1,a,b,request,SellPC(a,c2,500)), received(2,b,a,agree,SellPC(a,c2,500))	received(1,a,b,request,SellPC(a,c2,500)), sent(2,b,a,agree,SellPC(a,c2,500))
8.	GB:		
	MB:	sent(1,a,b,request,SellPC(a,c2,500)), received(2,b,a,agree,SellPC(a,c2,500)) received(3,b,a,inform,done(SellPC(a,c2,500)))	received(1,a,b,request,SellPC(a,c2,500)), sent(2,b,a,agree,SellPC(a,c2,500)) sent(3,b,a,inform,done(SellPC(a,c2,500)))

Table 1: Example of a request conversation

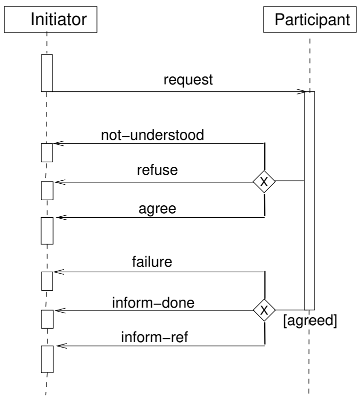

Figure 1: FIPA Request Protocol

a salesperson and he also believes that b is able to sell computers ('SellPC'). The belief-base for agent b: {customer(a), pc(c1), pc(c2), pc(c3), available(c1,400), available(c2,500), available(c3,600)}, meaning that agent b believes agent a is customer and b also believes that he has three different PCs, which are available for €400, €500 and €600 respectively. Finally, agent b has one basic action, SellPC:

$$\begin{array}{ll} PreCondition: & \{customer(A), pc(C), available(C,P)\} \\ & \quad SellPC(A,C,P) \\ PostCondition: & \{\neg available(C,P), sold(A,C,P)\} \end{array}$$

In the first step we see that agent a starts with buyPC(500) in his goal-base and agent b with the startSelling() goal. Agent a has one practical reasoning rule that matches with this goal:

$$\begin{array}{ll} buyPC(X) \leftarrow & salesPerson(B) \mid \\ & Send(1, a, B, request, SellPC(a, C, X)) \end{array}$$

This means that given a salesPerson, buyPC(500) is revised to the goal given in step 2. After which agent a uses the transition rule for the Send action to go to step 3, sending out a request for a computer for €500 to agent b. The fact that a sent a request to b is then kept in the message-base using the sent predicate.

Meanwhile, agent b starts off with a startSelling() achievement goal in step 1. This is revised at step 2 through the following rule:

$$startSelling() \leftarrow \ \top \mid handle\_request(condition)$$

with

$$\begin{array}{l} condition(SellPC(A,C,P)) \iff \\ \qquad customer(A) \wedge pc(C) \wedge available(C,P) \end{array}$$

The handle_request() is an achievement goal to handle the procedure of deciding to agree or refuse incoming requests as defined earlier. Here, the function *condition* is defined as the preconditions of the basic action SellPC, but in general this need not be the case. When for example an achievement goal is requested, which can trigger a whole set of basic actions, the relation between the actions and the preconditions may not be as straightforward as in this case.

In step 3 a request comes into the message-base of agent b and this triggers the revision of the handleRequest goal to the choice we see in step 4. And since agent b believes agent a is costumer and that he has a PC available for €500 (c2), he agrees to the request. Also due to this test-goal, he has now found a substitution for the variable C so that with the agree-message he can let agent a know what kind of PC to expect.

Going from step 4 to step 5, agent b has rewritten his goal-base because he has chosen to agree to the request. The first thing he has to do now is to inform agent a of his choice, the choice of agreeing to the request, as well as the choice for the PC he is going to sell to agent a. This message is received by agent a in step 6. He applies the substitution for C and waits for agent b to inform him about the result of the requested action.

Agent b on the other hand goes on processing his goal-base and is just about the execute this action. This results in step 7, where the belief-base of agent b is also updated: he removes the fact 'available(c2,500)' and adds 'sold(a,c2,500)'.

The last part of the request protocol is that agent b informs agent a that the action has been done. This is what happens when going from step 7 to step 8, where we see the final result. This, however, does not mean that the agents are finished now. They will still have to agree on a payment and delivery method, but that is outside the scope of this example.

The relations between the message-base and the belief-base have been left out from this example to keep it simple. Possible updates related to the incoming messages are for example after step 3. Agent b has received a request and could add $G_a buyPC(500)$ to his belief-base, meaning that agent a has the goal $buyPC(500)$. Similarly when agent a has received the request in step 6, he could add $G_b SellPC(a, c2, 500)$ to his belief-base. These possible belief-updates are all not defined in the FIPA semantics. The only belief-update that is defined in the FIPA ACL semantics is an update after an agent has received an `inform`, e.g. after step 8, agent a can add $B_b done(SellPC(a,\ c2, 500))$ to his belief-base. A possible update for the sender of this inform, like $B_a B_b done(SellPC(a, c2, 500))$ is, however, not defined in FIPA.

5. CONCLUSION

In this paper we have discussed some practical issues concerning the communication between goal directed agents. The first fundamental issue that we discussed is the fact that although the sending agent chooses the time to send a message, the receiving agent does not have the goal to receive that message. Therefore we created a message-base for each agent such that the transport of the message is performed synchronously while the handling of the message is asynchronous. We have shown how the specification of the effects of that depend on the performative of the message and how this is incorporated in practical reasoning rules that are invoked by the agents. The parts of the semantics of the communication that are undisputed and should take immediate effect are modelled using reactive rules that are invoked at the moment the message arrives in the message-base of the agent.

The parts of the intended effects of the communication over which the receiving agent needs to have control are modelled by reasoning rules that are invoked whenever the agent decides to pursue the trigger goal of that rule. E.g., the receiving agent decides itself when it wants to handle an inform message and whether it believes the contents of this message or even whether it believes that the sending agent believes the content.

By splitting up the 'standard' semantics of the messages given by FIPA in this way, it becomes possible to make agents that strictly conform to the FIPA specifications, but also deviate from this specification in circumstances where that might be necessary (e.g. the sincerity condition cannot be guaranteed in many open systems). The second advantage of incorporating the message handling in the practical reasoning rules is that the agent can incorporate them in a natural way among its other goals and can handle the messages at the point where it is most suitable for the agent.

Although the framework has been set up, a number of details still have to be taken care of. For instance, the unification of the content between the sent and received message is not trivial. Of course in this paper we only specified a few messages and their effects in 3APL. We strive to implement the complete set of FIPA messages in 3APL and implement the communication process in the next few months. Finally there is a practical issue of how large the message-bases should be and what happens if received messages are never handled by the agent.

6. ACKNOWLEDGMENTS

Special thanks go to Frank de Boer, Wiebe van der Hoek, Meindert Kroese, and John-Jules Meyer for many discussions and suggestions.

7. REFERENCES

[1] F. Dignum and M. Greaves. Issues in agent communication. In *LNCS-1916*. Springer-Verlag, 2000.

[2] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J. Bradshaw, editor, *Software Agents*. MIT Press, Cambridge, 1995.

[3] FIPA. http://www.fipa.org/.

[4] K. Hindriks, F. de Boer, W. van der Hoek, and J.-J. Meyer. Semantics of communicating agents based on deduction and abduction. In *IJCAI'99, Workshop on Agent Communication Languages*, 1999.

[5] K. V. Hindriks, F. S. D. Boer, W. V. der Hoek, and J.-J. C. Meyer. Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.

[6] R. van Eijk, F. de Boer, W. van der Hoek, , and J.-J. Meyer. Generalised object-oriented concepts for inter-agent communication. In *Intelligent Agents VII, Proceedings of 7th International Workshop on Agent Theories, Architectures, and Languages (ATAL 2000)*, volume 1986 of LNAI, pages 260–274. Springer-Verlag, 2001.

[7] M. Wooldridge. Semantic issues in the verification of agent communication languages. In *Autonomous Agents and Multi-Agent Systems*, volume 3, pages 9–32, 2000.