Communicating Agents

P. Dell'Acqua
Department of Information Science
Uppsala University
Box 311, S-751 05 Uppsala, Sweden
pier@csd.uu.se

F. Sadri, F.Toni
Department of Computing
Imperial College
180 Queen's Gate, London SW7 2BZ, UK
{fs,ft}@doc.ic.ac.uk

November 9, 1999

Abstract

We study the problem of endowing logic-based agents that can reason about their own beliefs as well as the beliefs of other agents with communication skills. We show how communication performatives from existing agent communication languages as well as their preconditions and effects can be expressed within logic-based agents in terms of the agents' beliefs. We illustrate the resulting language for programming logic-based agents by means of examples.

1 Introduction

In an earlier paper [6], we propose an approach to logic-based agents by combining the approach to agents by Kowalski and Sadri [11] and the approach to meta-reasoning by Costantini et al. [5, 4]. Similarly to Kowalski and Sadri's agents, the agents in [6] are *hybrid* in that they exhibit both *rational* (or *deliberative*) and *reactive* behaviour. The reasoning core of the agents is a proof procedure that combines forward and backward reasoning. Backward reasoning is used primarily for planning, problem solving and other deliberative activities. Forward reasoning is used primarily for reactivity to the environment, possibly including other agents. The proof procedure is executed within an "observe-think-act" cycle that allows the agent to be alert to the environment and react to it as well as think and devise plans. Both the proof procedure and the agents' architecture can deal with temporal information. The proof procedure (IFF proof procedure [9]) treats both inputs from the environment and agents' actions as *abducibles* (hypotheses). Moreover, by adapting techniques proposed in [4], the agents are capable of reasoning about their own beliefs and the beliefs of other agents.

In this paper, we extend the approach to agents in [6] by refining the inter-agent communication capabilities of the agents. In particular, we provide agents with "proactive" communication capabilities. The benefits of such "proactive" communication is widely documented in literature [3, 13, 14, 15, 16, 17]. For example, one can model agents that advertise their services so that other agents, possibly with the help of *mediators*, can find agents that provide services for them. In our resulting framework reactive, rational or hybrid agents can reason about their own beliefs as well as the beliefs of other agents and can communicate proactively with each other. In such a framework, the agents' behaviour can be regulated by condition-action rules such as: if I am asked by another, friendly agent about something and I can prove it from my beliefs, then I will tell the agent about it.

In the proposed approach, we adopt two primitives for communication, *tell* and *ask*, treated as abducibles within the "observe-think-act" cycle of the agent architecture in [11]. The predicate defined *told* is used to express both passive reception of messages from other agents and reception of information in response to an active request.

The remainder of the paper is structured as follows. In section 2 we give some the basic definitions for our logic-based agents. In section 3 we define our approach to communication in agents. In section 4 we illustrate the framework by means of an example. In section 5 we consider some communication performatives from exisiting agents' communication languages, in particular the FIPA ACL [8], and show how to express them and their semantics in our framework, in terms of the basic primitives *tell* and *ask*. In section 6 we conclude and discuss future work.

2 Agents via abductive logic programming

An **abductive logic program** [10] is a triple $\langle P, \mathcal{A}, I \rangle$, where P is a normal logic program, $\mathcal{A}$ a set of predicates in the language of P, referred to as the **abducible predicates**, and I a set of **integrity constraints**, i.e. implications of the form:

$$L_1 \wedge \ldots \wedge L_n \Rightarrow A \qquad (n \geq 0)$$

where $L_1, \ldots, L_n$ are literals and A is an atom, possibly *false*. All variables in an integrity constraint are implicitly universally quantified, with scope the entire integrity constraint. The atoms built from the abducible predicates are referred to as **abducibles**.

Without loss of generality (see [10]), we can assume that abducible predicates have no definitions in P. Abducibles can be thought of as hypotheses that can be used to extend the given logic program in order to provide an **explanation** for given **queries**, i.e. conjunctions of literals whose variables are implicitly existentially quantified. Explanations are required to "satisfy" the integrity constraints. Different notions of explanation and satisfaction have been used in the literature. The simplest notion of satisfaction is consistency of the explanation with the program and the integrity constraints.

In the sequel, constant, function and predicate symbols are written as any sequence of characters in `typewriter` style. Variables are written as any sequence of characters in *italic* style starting with a lower-case character. Thus, for example, `buys` may be a predicate symbol, `Tom` may be a constant symbol, while x and *city* are variables.

Abductive logic programs can be used to represent the body of information determining agents' behaviour. For example, in the abductive logic program $\langle P, \mathcal{A}, I \rangle$ given as follows

$$\begin{array}{rcl} P & = & \left\{ \begin{array}{l} \mathtt{has}(x) \leftarrow \mathtt{buys}(x) \\ \mathtt{has}(x) \leftarrow \mathtt{steals}(x) \end{array} \right\} \\ \mathcal{A} & = & \{ \ \mathtt{buys}, \mathtt{steals}, \mathtt{no\_money} \ \} \\ I & = & \{ \ \mathtt{no\_money} \wedge \mathtt{buys}(x) \Rightarrow \mathtt{false} \ \} \end{array}$$

P represents the belief of an agent that, in order to have an object, the agent can either buy the object or steal it. $\mathcal{A}$ represents the possible actions the agent can perform (i.e. to buy or steal) as well as the possible observations the agent can perform (i.e. that it has no money); I represents a prohibition, that the agent is not allowed to buy anything having no money. In the agent setting, queries represent both goals of agents and actual observations they have performed, whereas explanations represent plans of actions agents might want to take to achieve their goals, given the observations.

Then, given the query $Q_1 = \mathtt{has}(\mathtt{computer})$, both sets of abducibles $\{\mathtt{buys}(\mathtt{computer})\}$ and $\{\mathtt{steals}(\mathtt{computer})\}$ are explanations for Q, satisfying I, whereas given the query $Q_2 = \mathtt{has}(\mathtt{computer}) \wedge \mathtt{no\_money}$, the set of abducibles $\{\mathtt{steals}(\mathtt{computer}), \mathtt{no\_money}\}$ is the only possible explanation, as $\{\mathtt{buys}(\mathtt{computer}), \mathtt{no\_money}\}$ is inconsistent with I.

Given an abductive logic programming formulation of agents, their behaviour can be controlled by the execution of **abductive proof procedures** for the automated computation of explanations for queries. The agent architecture of [11] adopts this view, and uses the specific abductive proof procedure of [9] as its underlying engine. This architecture relies upon an *observe-think-act* cycle, interleaving the execution of the abductive proof procedure (*think*) with the observation of events taking place in the environment in which the agent is situated (*observe*) and the execution of actions generated by the abductive proof procedure as part of (partial) plans (*act*). The partial plan from which agents are selected and executed represents an *intention*. The interleaving allows agents to be reactive. The environment may include other agents, to possibly interact with.

The abductive logic program can be a temporal theory, for example, the event calculus [12]. Then, actions that are generated in an intention may have times associated with them. The times may be absolute, for example $\mathtt{happens}(\mathtt{ring\_bell}, 3)$, or may be within a constrained range, for example $\mathtt{happens}(\mathtt{step\_forward}, t) \wedge (1 < t < 10)$.

3 Communication for abductive logic programming agents

Within abductive logic programming agents, we represent communication performatives via abductible atoms, which thus can be part of explanations (i.e. plans of action) for queries (i.e. observations and goals). In this paper we only consider the most basic communication performatives between agents, namely `ask`, for *active request* of information, and `tell`, for *voluntary provision* of information. The atoms representing the two performatives are (respectively) $\mathtt{ask}(A, B, X, ID, T)$, for "agent A `ask`s agent B at time T whether X is the case, by a message whose identifier is ID", and $\mathtt{tell}(A, B, X, ID, T)$, for "agent A `tell`ss agent B at time T that X is the case, by a message whose identifier is ID". Note that this formalisation allows asking "what", "when" and "who" questions, for example:

$$\mathtt{ask}('\mathtt{agent}_1, '\mathtt{agent}_2, '\mathtt{manager}(Y, '\mathtt{Jim}), '\mathtt{id\_1}, 10)$$

represents a question by $\mathtt{agent}_1$ to $\mathtt{agent}_2$ at time 10 on who *Jim*'s manager is, to which the following answer might be given

$$\mathtt{tell}('\mathtt{agent}_2, '\mathtt{agent}_1, '\mathtt{manager}('\mathtt{tim}, '\mathtt{Jim}), '\mathtt{id\_1}, 13).$$

Reception of information, either in response to an active request or by voluntary provision by other agents, can be represented via the predicate `told`, which can be defined in terms of `ask` and `tell` as follows:

$$\begin{array}{lll}
(TOLD_1) & \mathtt{told}(A, B, X, T_2) \leftarrow & \mathtt{ask}(A, B, X, ID, T_1) \wedge \\
& & \mathtt{tell}(B, A, X, ID, T_2) \wedge \\
& & T_2 > T_1 \\
(TOLD_2) & \mathtt{told}(A, B, X, T) \leftarrow & \mathtt{tell}(B, A, X, ID, T).
\end{array}$$

Namely, an agent A is told of X by another agent B at some time if either

- A actively asks B about X at some earlier time via a message whose identifier is ID and B gives a positive answer to the question in the message identified by ID, by confirming X, or
- B tells A of X at a time T, of its own initiative, in reply to no message.

Note that the rule $(TOLD_1)$ is logically redundant but it helps clarifying the use of the communication primitives in the context of "conversations". It is also useful in an abductive agent setting, where it allows to generate a plan for an agent to ask another agent for information that it requires (or wants to be told). Explicit time is needed to represent appropriately the order of question and answer in a "conversation".

Note also that, from the viewpoint of the cycle of agent A, $\mathtt{tell}(A, B, X, ID, T)$ and $\mathtt{ask}(A, B, X, ID, T)$ are actions to be selected and executed, whereas $\mathtt{tell}(B, A, X)$ and $\mathtt{ask}(B, A, X)$ are not actions that can be executed by agent A; they can be solved only if they are matched by inputs.

Communication can give rise to changes in the beliefs of agents. The changes can be represented by means of a predicate `solve` which allows us to extend at the *meta-level* the intended meaning of predicates (partially) defined at the *object-level.* In the sequel,

$$\mathtt{solve}(A, X, T)$$

represents the fact that agent A believes X at time T.

Then, one can adopt the unconditional, monotonic belief revision strategy expressed by

$$\texttt{solve}(A, X, T) \leftarrow \texttt{told}(A, B, X, T)$$

namely, any agent A believes (at the meta-level) anything it is told by any agent B, at any time. An alternative monotonic, but more refined strategy is expressed by

$$\texttt{solve}(A, X, T) \leftarrow \texttt{told}(A, B, X, T) \wedge \texttt{reliable}(B, T)$$

namely, at any time, any agent A believes (at the meta-level) anything it is told by any agent B that is reliable at the given time.

The introduction of the `solve` predicate within abductive logic programming gives rise to a metalogic programming language. In the extended language, terms need to be defined differently in order to include *names* that are intended to represent at the meta-level the expressions of the language itself. In this paper, we adopt the naming conventions of Reflective Prolog [5], by making a distinction between variables and *metavariables* and by introducing *metaconstants*. Metavariables can only be substituted with names of sentences, and metaconstants are intended as names for constants, function and predicate symbols. If `c` is a constant, a function or a predicate symbol, then we write $'\texttt{c}$ as a convenient way to represent the metaconstant that names `c`. Similarly, if $'\texttt{c}$ is a metaconstant, then its name is $''\texttt{c}$, and so on. Metavariables are written as any sequences of characters in *italic* style starting in the upper-case. Compound terms and atoms are represented at the meta-level as *name terms*. For example, the name of the term $\texttt{f}(\texttt{a}, x)$ is the name term $'\texttt{f}('\texttt{a},'x)$, where $'\texttt{f}$ and $'\texttt{a}$ are the metaconstants that name the function symbol `f` and constant `a`, respectively, and $'x$ stands for the name of the value of the variable x.

Note that the introduction of the meta-predicate `solve` raises the problem of linking object- and meta-level beliefs. In [6], ignoring the time parameter T in `solve`, we present a transformation that achieves the effect of the following (inter-level) reflection axioms:

$$X \leftarrow \texttt{solve}('A,'X) \quad \text{and} \quad \texttt{solve}('A,'X) \leftarrow X,$$

for all (abducible and non-abducible) atoms X in the knowledge base of each agent A.
The first asserts that whenever an atom of the form $\texttt{solve}('A,'X)$ is provable at the meta-level, then X is provable at the object-level. In [6], these axiom schemata are not explicitly present in any given abductive logic program, but (finitely many) appropriate logic programming rules and integrity constraints guaranteeing them are.

It is an advantage to have time as an explicit parameter in `solve` atoms, to have correct representation of dynamic beliefs. For simplicity, we will assume that agents' beliefs can only grow monotonically over time, namely

$$\texttt{solve}(A, X, T+1) \leftarrow \texttt{solve}(A, X, T).$$

The adoption of a more general temporal reasoning theory is beyond the scope of this paper.

Finally, we assume that the meta-predicate `solve` takes only names of atoms as their second argument, and the meta-predicates `told`, `tell` and `ask` take only names of atoms as their third arguments.

In the sequel we will omit the message identifier in atoms in the predicates `tell` and `ask`, unless the omission causes a loss of clarity. Moreover, until section 5, we will omit the time argument from `tell`, `ask`, `told` and `solve` and we will ignore temporal reasoning issues. Furthemore, we will abbreviate $\texttt{tell}(\textit{Agent}, X, Y)$ (resp. `ask` and `told`) and $\texttt{solve}(\textit{Agent}, X)$ within the program P of *Agent* itself as $\texttt{tell}(X, Y)$ and $\texttt{solve}(X)$, unless confusion arises.

Example 3.1 Let $\texttt{agent}_1$ be represented by the abductive logic program $\langle P, \mathcal{A}, I\rangle$ with:

$$P = \left\{ \begin{array}{l} \texttt{told}(A,X) \leftarrow \texttt{ask}(A,X) \wedge \texttt{tell}(A,X) \\ \texttt{told}(A,X) \leftarrow \texttt{tell}(A,X) \\ \texttt{solve}(X) \leftarrow \texttt{told}(A,X) \\ \texttt{desire}(y) \leftarrow y = \texttt{car} \\ \texttt{good\_price}(p,x) \leftarrow p = 0 \end{array} \right\}$$

$$\mathcal{A} = \{ \ \texttt{tell}, \texttt{ask}, \texttt{offer} \ \}$$

$$I = \left\{ \begin{array}{l} \texttt{desire}(x) \wedge \texttt{told}(B, '\texttt{good\_price}('p, 'x)) \\ \quad \Rightarrow \texttt{tell}(B, '\texttt{offer}('p, 'x)) \end{array} \right\}.$$

The third clause in P says that $\texttt{agent}_1$ believes anything it is told (by any other agent A). The fourth and fifth clauses in P say, respectively, that the agent desires a car and that anything that is free is at a good price. The integrity constraint says that, if the agent desires something and it is told (by some other agent B) of a good price for it, then it makes an offer to B, by telling it. Note that a more accurate representation of the integrity constraint should include time.

Note that, in our approach:

- Agents are known by their symbolic names.
- When an agent sends a message, it directs that message to a specific addressee.
- When an agent receives a message, it knows the sender of that message.
- Messages may get lost.

Both `tell` and `ask` are treated as actions. In an agent $\texttt{agent}_1$'s observe-think-act cycle, everytime (the cycle of) the agent selects a communicative action, i.e., an action of the form

$$\texttt{ask}('\texttt{agent}_2, A) \quad \text{or} \quad \texttt{tell}('\texttt{agent}_2, A),$$

$\texttt{agent}_1$ will attempt to execute it. If the attempt is successful, the record

$$\texttt{ask}('\texttt{agent}_2, A) \quad \text{or} \quad \texttt{tell}('\texttt{agent}_2, A)$$

is conjoined to the observations of agent $\texttt{agent}_1$, as an additional input. If the attempt is not successful, the record

$$\texttt{ask}('\texttt{agent}_2, A) \Rightarrow \texttt{false} \quad \text{or} \quad \texttt{tell}('\texttt{agent}_2, A) \Rightarrow \texttt{false}$$

is conjoined to the observations of agent $\texttt{agent}_1$.

4 Example

We borrow an example from [6] to demonstrate the execution of the proof procedure within the agent cycle, action selection, proactive communication whereby one agent volunteers information to another, and how during the planning phase such information can help in the choice of intention.

The example is as follows: an agent, Peter, wishes to register for a conference, let us say Jelia, that is to take place in Paris on the 10th and to make travel arrangements to go to Paris on the 10th (for simplicity we omit the month). So Peter's original goal is represented by the conjunction:

$$\texttt{register}(\texttt{Jelia}) \wedge \texttt{travel}(\texttt{Paris}, 10).$$

Peter has the following program, P:

C_1 $\quad$ $\texttt{travel}(\mathit{city}, \mathit{date}) \leftarrow$
$\qquad\qquad \texttt{go}(\mathit{city}, \mathit{date}, \texttt{train})$
C_2 $\quad$ $\texttt{travel}(\mathit{city}, \mathit{date}) \leftarrow$
$\qquad\qquad \texttt{go}(\mathit{city}, \mathit{date}, \texttt{plane})$
C_3 $\quad$ $\texttt{early\_registration}(\texttt{Jelia}) \leftarrow$
$\qquad\qquad \texttt{send\_form}(\texttt{Jelia}, \mathit{date}) \wedge$
$\qquad\qquad \mathit{date} < 3$
C_4 $\quad$ $\texttt{go}(\mathit{city}, \mathit{date}, \mathit{means}) \leftarrow$
$\qquad\qquad \texttt{book}(\mathit{city}, \mathit{date}, \mathit{means})$
C_5 $\quad$ $\texttt{book}(\mathit{city}, \mathit{date}, \mathit{means}) \leftarrow$
$\qquad\qquad \texttt{available}(\mathit{city}, \mathit{date}, \mathit{means}) \wedge$
$\qquad\qquad \texttt{tell}('\texttt{Peter}, '\texttt{ticket\_agent}, '\texttt{reserve}('\mathit{city}, '\mathit{date}, '\mathit{means}), \mathit{ID}) \wedge$
$\qquad\qquad \texttt{told}('\texttt{ticket\_agent}, '\texttt{Peter}, '\texttt{confirmed}(''\texttt{Peter}, '\mathit{city}, '\mathit{date}, '\mathit{means}), \mathit{ID})$
C_6 $\quad$ $\texttt{solve}(X) \leftarrow$
$\qquad\qquad \texttt{trust\_on}(A, T) \wedge$
$\qquad\qquad \texttt{told}(A, '\texttt{Peter}, X, \mathit{ID}) \wedge$
$\qquad\qquad \texttt{related}(X, T)$
C_7 $\quad$ $\texttt{trust\_on}('\texttt{ticket\_agent}, '\texttt{travelling})$
C_8 $\quad$ $\texttt{related}('\texttt{available}('\mathit{city}, '\mathit{date}, '\mathit{means}), '\texttt{travelling})$
C_9 $\quad$ $\texttt{credit\_card}('\texttt{Peter}, \texttt{Visa}, \texttt{30-02-99})$

together with the two clauses $TOLD_1$ and $TOLD_2$ (defined on page 3). Clauses C_1 and C_2 say that one travels to a *city* on a given *date* if one goes there on that *date* by train or by plane. C_3 says that the deadline for early registration for Jelia is the 3rd. C_4 says that one goes to a *city* on a given *date* by some *means* if one makes a booking for that journey. C_5 says that to make a booking, Peter has to know the availability of the ticket, to tell the ticket agent to make the reservation, and to get confirmation of it. The subgoal `available` is solved with help of clause C_6. Note that in a more thorough representation we would represent the transaction time of when a booking is made (which must be before the time of travel). In that case we will have an extra argument in `tell`, `ask` and `told` that represents such transaction times. We will ignore this issue here for simplicity. C_6 says that Peter believes X if he has been told X by an agent A, Peter trusts everything he is told by A regarding a topic T, and X is related to topic T. Clauses C_7 and C_8 say that the ticket agent is trustworthy on the subject of travelling, and the availability of tickets is related to travelling. Finally, C_9 provides the details of Peter's credit card.

Peter has the following set of integrity constraints I:

I_1 $\quad$ $\texttt{register}(\mathit{conference})$
$\qquad\qquad \Rightarrow \texttt{early\_registration}(\mathit{conference})$
I_2 $\quad$ $\texttt{tell}('\texttt{Peter}, \mathit{Anyone}, '\texttt{reserve}('\mathit{city}, '\mathit{date}, '\mathit{means}), \mathit{ID}) \wedge$
$\qquad\quad \texttt{strike}(\mathit{city}, \mathit{date}, \mathit{means})$
$\qquad\qquad \Rightarrow \texttt{false}$
I_3 $\quad$ $\texttt{tell}('\texttt{Peter}, '\texttt{ticket\_agent}, '\texttt{reserve}('\mathit{city}, '\mathit{date}, '\mathit{means}), \mathit{ID}) \wedge$
$\qquad\quad \texttt{ask}('\texttt{ticket\_agent}, '\texttt{Peter}, X, \mathit{ID}) \wedge$
$\qquad\quad \texttt{solve}(X)$
$\qquad\qquad \Rightarrow \texttt{tell}('\texttt{Peter}, '\texttt{ticket\_agent}, X, \mathit{ID}).$

I_1 states the Peter's departmental policy that anyone registering at a conference must take advantage of early registration. I_2 states that one cannot make a reservation with a mode of transport which is subject to a strike. I_3 states that if Peter requests a reservation and the ticket agent asks Peter something with respect to this request, then Peter should answer the ticket agent, if he knows the answer.

Finally, Peter has the following set of abducibles $\mathcal{A}$:

$$\{\texttt{send\_form}, \texttt{tell}, \texttt{ask}, \texttt{register}, \texttt{available}, \texttt{reserve}, \texttt{strike}\}.$$

The `ticket_agent` has the following program, Q:

D_1 $\quad \texttt{trust}(A) \leftarrow$
$\qquad \texttt{old\_custumer}(A)$
D_2 $\quad \texttt{old\_custumer}('\texttt{John})$
D_3 $\quad \texttt{related}('\texttt{strike}('city,'date,'means),'\texttt{available}('city,'date,'means))$
D_4 $\quad \texttt{strike}(\texttt{Paris}, 10, \texttt{plane})$
D_5 $\quad \texttt{available}(\texttt{Paris}, 10, \texttt{train})$
D_6 $\quad \texttt{respond}(A,'\texttt{reserve}('city,'date,'means), ID) \leftarrow$
$\qquad \texttt{trust}(A) \wedge$
$\qquad \texttt{reserve}(A, city, date, means, ID)$
D_7 $\quad \texttt{respond}(A,'\texttt{reserve}('city,'date,'means), ID) \leftarrow$
$\qquad \texttt{paid}(A) \wedge$
$\qquad \texttt{reserve}(A, city, date, means, ID)$
D_8 $\quad \texttt{respond}(A,'\texttt{reserve}('city,'date,'means), ID) \leftarrow$
$\qquad \texttt{not trust}(A) \wedge \texttt{not paid}(A) \wedge$
$\qquad \texttt{told}(A,'\texttt{ticket\_agent},'\texttt{credit\_card}('A,'type,'expiry), ID) \wedge$
$\qquad \texttt{reserve}(A, city, date, means, ID)$

D_6–D_8: the `ticket_agent` responds to an agent's request for a reservation by reserving a ticket for that agent if he trusts that agent or if the agent has paid or if the agent has told his credit card details to him.

The `ticket_agent` has the following set of integrity constraints.

L_1 $\quad \texttt{ask}(A,'\texttt{ticket\_agent}, X, ID) \wedge$
$\quad \texttt{solve}(X)$
$\qquad \Rightarrow \texttt{tell}('\texttt{ticket\_agent}, A, X, ID)$
L_2 $\quad \texttt{ask}(A,'\texttt{ticket\_agent}, X, ID) \wedge$
$\quad \texttt{not solve}(X) \wedge$
$\quad \texttt{solve}(Y) \wedge$
$\quad \texttt{related}(Y, X)$
$\qquad \Rightarrow \texttt{tell}('\texttt{ticket\_agent}, A, Y, ID)$
L_3 $\quad \texttt{reserve}(A, city, date, means, ID)$
$\qquad \Rightarrow \texttt{tell}('\texttt{ticket\_agent}, A,'\texttt{confirmed}('A,'city,'date,'means), ID)$
L_4 $\quad \texttt{tell}(A,'\texttt{ticket\_agent},'\texttt{reserve}('city,'date,'means), ID)$
$\qquad \Rightarrow \texttt{respond}(A,'\texttt{reserve}('city,'date,'means), ID)$

L_2: If `ticket_agent` agent is asked about something which it does not know, it tries to be helpful by telling the inquirer something that the ticket agent does know and that is related to the original query.
L_3: The `ticket_agent` sends a confirmation for every reservation that it makes.
L_4: The `ticket_agent` responds to requests for reservations.

Below we give a brief representation of the cycles of the two agents running in parallel and the communications that take place between the two agents. We have omitted many details of the goals in order to provide an overview. The goals that are underlined are actions that are selected for performance by the respective agents. For the purposes of this illustration, we assume that all the actions, including the communication actions, are successful.

Peter	**Ticket_agent**
register(Jelia) ∧ [go(Paris, 10, train) ∨ go(Paris, 10, plane)]	
send_form(Jelia, 2)	
go(Paris, 10, train) ∨ go(Paris, 10, plane)	
[ask('Peter,' ticket_agent, 'available('Paris,'10,'plane),'id_1) ∧ . . .] ∨ go(Paris, 10, train)	
	tell('ticket_agent,' Peter, 'strike('Paris,'10,'plane),'id_1)
go(Paris, 10, train)	
ask('Peter,' ticket_agent, 'available('Paris,'10,'train),'id_2)	
	tell('ticket_agent,' Peter, 'available('Paris,'10,'train),'id_2)
tell('Peter,' ticket_agent, 'reserve('Paris,'10,'train),'id_3)	
	ask('ticket_agent,' Peter, 'credit_card("Peter,' *type*,' *expiry*),' id_3)
tell('Peter,' ticket_agent, 'credit_card("Peter,'Visa,'30-02-99),'id_3)	
	reserve('Peter, Paris, 10, train,'id_3)
	tell('ticket_agent,' Peter, 'confirmed("Peter,' Paris,'10,'train),'id_3))

5 Other communication primitives

Up until now we have considered only two communication performatives, `tell` and `ask`. Existing communication languages, e.g., KQML [7] and FIPA ACL (Agent Communication Language) [8], include additional performatives, such as `ask-if`, `ask-one`, `ask-all`, `deny`, `achieve` and `unachieve` in KQML and `inform`, `query-if`, `request`, `request-whenever`, `propose` in FIPA ACL, to mention a few. In this section we will show how some of the performatives in FIPA ACL can be expressed via definitions in our framework. We will ignore those parameters of FIPA ACL performatives, such as `ontology` and `language`, which do not play any role in the formal semantics of the permormatives.

Given an agent communication language, the problem arises of how to specify unambiguosly the semantics of performatives in the language and to check that agents' behaviour conforms to such semantics. KQML and FIPA ACL have been given semantics by expressing preconditions and effects of the performatives in a multi-modal language, distinct from the communication language, e.g., [3] (for KQML) and [1] (for FIPA ACL). In this section we will express preconditions and effects of some performatives of the FIPA ACL via integrity constraints and definitions, within our framework.

In the remainder of this section, we will use a time argument in `solve`, `told`, `tell` and `ask` atoms (see section 3).

The performative inform in the FIPA ACL directly corresponds to our performative tell:

$$\texttt{inform}(A, B, X, T) \leftarrow \texttt{tell}(A, B, X, T).$$

The preconditions for inform, that agents should not inform other agents of something they themselves do not believe or of something they believe the other agents already believe, can be expressed via the integrity constraints:

$$\texttt{inform}(A, B, X, T) \wedge \texttt{not}\,\texttt{solve}(A, X, T) \Rightarrow \texttt{false}$$
$$\texttt{inform}(A, B, X, T) \wedge \texttt{solve}(A, \texttt{solve}(B, X, T), T) \Rightarrow \texttt{false}.$$

Note that, by introspection, the second integrity constraint can be replaced by

$$\texttt{inform}(A, B, X, T) \wedge \texttt{solve}(B, X, T) \Rightarrow \texttt{false}.$$

The "rational" effect of inform, that every agent believes what it is informed about, can be expressed by the clause

$$\texttt{solve}(A, X, T + 1) \leftarrow \texttt{inform}(B, A, X, T).$$

Note that, in our approach, more refined versions of this clause can be given, e.g.,

$$\texttt{solve}(A, X, T + 1) \leftarrow \texttt{inform}(B, A, X, T) \wedge \texttt{trustworthy}(B)$$

or

$$\texttt{solve}(A, X, T + 1) \leftarrow \texttt{inform}(B, A, X, T) \wedge \texttt{trust-on}(A, B, S) \wedge \texttt{related}(X, S)$$

namely, agent A believes anything that B tells it, provided B is a trustworthy agent or A trusts B on subject S and X is related to S.

The performative query-if in the FIPA ACL directly corresponds to our performative ask:

$$\texttt{query-if}(A, B, X, T) \leftarrow \texttt{ask}(A, B, X, T).$$

The preconditions for query-if, that agents asking about a proposition should not have any prior knowledge of it and should believe that the agents they are asking know about it, can be expressed via the integrity constraints:

$$\texttt{query-if}(A, B, X, T) \wedge \texttt{solve}(A, X, T) \Rightarrow \texttt{false}$$
$$\texttt{query-if}(A, B, X, T) \wedge \texttt{not}\,\texttt{solve}(A, \texttt{certain}(B, X, T), T) \Rightarrow \texttt{false}$$

where $\texttt{certain}(B, X, T)$ stands for "B has certain knowledge about X at time T".

The effect of query-if, that the agent being asked replies with a positive answer if it believes the proposition asked about, can be expressed via the integrity constraint:

$$\texttt{query-if}(A, B, X, T) \wedge \texttt{solve}(B, X, T) \Rightarrow \texttt{inform}(B, A, X, T).$$

The performatives inform-ref and query-ref, for informing and quering, respectively, about domain objects corresponding to a given descriptor also correspond to tell and ask, respectively:

$$\texttt{inform-ref}(A, B, X, D(X), T) \leftarrow \texttt{tell}(A, B, D(X), T)$$
$$\texttt{query-ref}(A, B, X, D(X), T) \leftarrow \texttt{ask}(A, B, D(X), T).$$

In inform-ref A informs B of a value of an X that satisfies property $D(X)$, and in query-ref A asks B for such a value.

The preconditions for `inform-ref` and `query-ref` can be expressed via the integrity constraints:

$$\texttt{inform-ref}(A, B, X, D(X), T) \wedge \texttt{not}\,\texttt{solve}(A, D(X), T) \Rightarrow \texttt{false}$$
$$\texttt{inform-ref}(A, B, X, D(X), T) \wedge \texttt{solve}(A, \texttt{solve}(B, D(X), T), T) \Rightarrow \texttt{false}$$
$$\texttt{query-ref}(A, B, X, D(X), T) \wedge \texttt{solve}(A, D(X), T) \Rightarrow \texttt{false}$$
$$\texttt{query-ref}(A, B, X, D(X), T) \wedge \texttt{not}\,\texttt{solve}(A, \texttt{solve}(B, D(X), T), T) \Rightarrow \texttt{false}.$$

As before, by introspection, the second integrity constraint can be replaced by

$$\texttt{inform-ref}(A, B, X, D(X), T) \wedge \texttt{solve}(B, D(X), T) \Rightarrow \texttt{false}.$$

To express other performatives of the FIPA ACL within our framework, we need to extend it so that clauses and integrity constraints can be communicated. For example, the performatives `request` and `request-whenever`, by means of which agents request other agents to perform a given action, unconditionally or whenever some property holds, could be expressed via clauses:

$$\texttt{request}(A, B, X, T) \leftarrow \texttt{tell}(A, B, \texttt{true} \Rightarrow X, T)$$
$$\texttt{request-whenever}(A, B, P, X, T) \leftarrow \texttt{tell}(A, B, P \Rightarrow X, T)$$

This extension of our approach is subject for future work.

In order to express the performative `confirm`, we extend our framework by an additional meta-predicate `uncertain`, such that $\texttt{uncertain}(A, X, T)$ stands for "agent A is uncertain about X at time T". Then, the perfomative `confirm` can be expressed by

$$\texttt{confirm}(A, B, X, T) \leftarrow \texttt{tell}(A, B, X, T)$$

with preconditions expressed by integrity constraints

$$\texttt{confirm}(A, B, X, T) \wedge \texttt{not}\,\texttt{solve}(A, X, T) \Rightarrow \texttt{false}$$
$$\texttt{confirm}(A, B, X, T) \wedge \texttt{not}\,\texttt{solve}(A, \texttt{uncertain}(B, X, T), T) \Rightarrow \texttt{false}$$

i.e. A confirms a proposition X to B only if A believes X, and A believes that B is uncertain about X.

6 Conclusions

We have presented an approach to agents that can reason about their own beliefs as well as beliefs of other agents and that can communicate with each other. The approach extends by means of communication the approach to agents and meta-reasoning in [6]. We have illustrated the approach by means of a number of examples.

The approach presented in this paper relies upon two basic communication primitives, `tell` and `ask`. We have shown how other communication performatives, taken as primitive in existing communication languages for agents, can be expressed in terms of `tell` and `ask` via definitions and integrity constraints in our framework. More importantly, their semantics can be directly expressed in our framework too, and conformance of the behaviours of the agent to the semantics is obtained as a by-product of the soundness of our the proof procedure applied within the agents' cycle. We are currently investigating whether the remaining additional performatives taken as primitive in existing communication languages for agents can be defined within (extensions of) our framework, allowing, in particular, for integrity constraints to be communicated amongst agents alongside with atoms.

For simplicity, we have ignored the treatment of time. However, time plays an important role in most agent applications and should be explicitly taken into account.

We have implicitly assumed that different agents share the same content language. However, this assumption is not essential. Indeed, "translator agents" could be defined, acting as mediators between agents with different content languages. This is possible by virtue of the metalogic features of the language.

An interesting application of agents is that of a virtual marketplace on the Web where users create autonomous agents to buy and sell goods on their behalf. Chavez & Maes [2], for example, propose a marketplace, where users can create selling and buying agents by giving them a description of the item they want to sell or buy. The main goal of the approach is to help users in the negotiations between buyers and sellers, and to sell the goods better (i.e., at a higher price) than the user would be able to otherwise, by taking advantage of their processing speed and communication bandwidth. Chavez & Maes' agents are (*i*) proactive: "...they try to sell themselves, by going into a marketplace, contacting interested parties (namely, buying agents) and negotiating with them to find the best deal", and (*ii*) autonomous: "...once released into the marketplace, they negotiate and make decisions on their own, without requiring user intervention". Chavez & Maes point out that their agents' lack rationality:

> "Our experiment demonstrated the need and desire for 'smarter' agents whose decision making processes more closely mimic those of people and which can be directed at a more abstract, motivational level."

Our approach allows for agents that are autonomous, proactive, reactive to inputs as well as rational. The adoption of our approach for applications such as the above is an issue for future research.

Acknowledgments

The authors are grateful to the anonymous referees for their comments. The first author was supported by a PRAXIS scholarship. The last two authors were supported by the UK EPSRC project "Logic-based multi-agent systems".

References

[1] P. Bretier and D. Sadek. A rational agent as the kernel of a cooperative spoken dialogue system: Implementing a logical theory of interaction. In J.P. Müller, M.Wooldridge, and N.R. jennings, editors, *Intelligent Agents III.* Springer Verlag, LNAI 1193, 1997.

[2] A. Chavez and P. Maes. Kasbah: An agent marketplace for buying and selling goods. In B. Crabtree and N. Jennings, editors, *Proc. 1st Intl. Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology*, pages 75–90. The Practical Application Company, 1996.

[3] P. R. Cohen and H. J. Levesque. Communicative actions for artificial agents. In V. Lesser, editor, *Proc. 1st Intl. Conf. on Multiagent Systems*, AAAI Press, pages 65–72. MIT Press, 1995.

[4] S. Costantini, P. Dell'Acqua, and G. A. Lanzarone. Reflective agents in metalogic programming. In Alberto Pettorossi, editor, *Meta-Programming in Logic*, LNCS 649, pages 135–147, Berlin, 1992. Springer-Verlag.

[5] S. Costantini and G. A. Lanzarone. A metalogical programming approach: Language, semantics and applications. *Int. J. of Experimental and Theoretical Artificial Intelligence*, 6:239–287, 1994.

[6] P. Dell'Acqua, F. Sadri and F. Toni. Combining Introspection and Communication with Rationality and Reactivity in Agents. In U. Furbach and L. Farinas del Cerro editors, *Proc. JELIA'98*, 6th European Workshop on Logics in Artificial Intelligence. LNAI 1489, Springer Verlag, pages 17–32, 1998.

[7] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J. Bradshaw editor, *Software Agents.* MIT Press, 1997.

[8] The Foundation for Intelligent Physical Agents. FIPA'97 Specification Part 2 — Agent Communication Language. http://drogo.cselt.it/fipa/spec/fipa97.htm

[9] T. H. Fung and R. Kowalski. The IFF proof procedure for abductive logic programming. *J. Logic Programming*, 33(2):151–165, 1997.

[10] A. C. Kakas, R. A. Kowalski, and F. Toni. The role of abduction in logic programming. In D. Gabbay, C. Hogger, and A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 5, pages 235–324. Oxford University Press, UK, 1998.

[11] R. A. Kowalski and F. Sadri. From Logic Programming to Multi-agent Systems. *Annals of Mathematics and Artificial Intelligence* 1999 (To appear)

[12] R. A. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[13] Y. Labrou and T. Finin. Semantics and conversations for an agent communication language. In M. N. Huhns and M. P. Singh, editors, *Readings in Agents*, pages 234–242, San Francisco, 1997. Morgan Kaufmann.

[14] M. P. Singh. Towards a formal theory of communication for multiagent systems. In *Proc. 12th Intl. Joint Conf. on Artificial Intelligence*, pages 69–74, Sydney, Australia, 1991. Morgan Kaufmann.

[15] B. van Linder, W. van der Hoek, and J.-J. Ch. Meyer. Communicating rational agents. In B. Nebel and L. Dreschler-Fischer, editors, *KI-94: Advances in Artificial Intelligence*, LNAI 861, pages 202–213, Berlin, 1994. Springer-Verlag.

[16] E. Verharen and F. Dignum. Cooperative information agents and communication. In P. Kandzia and M. Klusch, editors, *Cooperative Information Agents*, LNAI 1202, pages 195–208, Berlin, 1997. Springer-Verlag.

[17] G. Wagner. Multi-level security in multiagent systems. In P. Kandzia and M. Klusch, editors, *Cooperative Information Agents*, LNAI 1202, pages 272–285, Berlin, 1997. Springer-Verlag.