Towards an Alternative Semantics for FIPA ACL *

Mirko Viroli, Andrea Omicini
DEIS, Università degli Studi di Bologna,
via Rasi e Spinelli 176, 47023 Cesena (FC), Italy
email: {mviroli,aomicini}@deis.unibo.it

Abstract

We show how a formal framework for *observation* in computer systems can be used for the specification of agent's observable behaviour, and provide for an architecture-independent approach alternative to the one currently used for FIPA ACL. Since the agent architecture induced by our model is more abstract than the one assumed by FIPA, our specification tool is likely to be applicable to a wider set of agents' architecture, thus better supporting FIPA standardisation aims. Some application examples are shown, describing how the current semantics of FIPA ACL could be adapted and specified within our framework.

1 Motivation

Agent-based systems are typically highly dynamic and complex systems, which often do not allow their behaviour to be fully modelled. On the one hand, each agent can encapsulate a complicated machinery, e.g. supporting intelligent behaviours, planning, and inference activities. On the other hand, the agent abstraction is often exploited as a mean for dealing with the unknowability of software components – that is, as a wrapper for *legacy* systems whose behaviour is no longer completely known, but which should be anyway integrated within an existing multi-agent system (MAS).

As a result, a crucial issue in agent-based systems is to abstract away from agent's internal behaviour, while associating a precise semantics to their communicative acts, in terms of both how they affect the receiving agent, and by which means the sender agent emits them. In general, such a communication language's specification should make it possible to understand the behaviour of a MAS by simply observing its communication events and abstracting away from the inner details of each individual agent's dynamics.

One of the most relevant examples of this methodology is the specification of the FIPA Agent Communication Language (ACL) [FIPA, 2000], which focuses on the relations between agents' communications and agents' *mental states* [Sadek, 1992]. Each FIPA communicative act's specification is equipped by a *feasibility precondition* (FP) that must hold for the sender, and a *rational effect* (RE) that the sender may suppose to occur on the receiver (even though such an effect is not actually mandatory for the receiver). Both these specifications, as well as the actual messages' content, are given in terms of a quantified, multi-modal logic with modal operators for *beliefs* (B), *desires* (D), *uncertain beliefs* (U), and *intentions* (I), called *Semantic Language* (SL) [FIPA, 2000], which originated from the work on the BDI framework [Cohen and Levesque, 1990]. Despite FIPA does not mandate any actual architecture for agents, FIPA ACL semantics implicitly assumes that the agent's observable behaviour can be interpreted in terms of a BDI-like architecture, which can be pictorially represented as shown in Figure 1. Notice that adhering to ACL's semantics is mandatory for an agent using the FIPA ACL communicative acts in a FIPA-compliant environment.

This specification approach may provide a suitable support for standardising the cooperation amongst BDI-agents. However, it might turn out to be an ineffective tool for the agents built upon different frameworks – such as agents wrapping legacy systems, agents built as simple active objects, or agents with limited (or even no) intelligent behaviour living on environments such as mobile devices, and so on. In all these cases, which are definitely relevant means for building complex software systems in the Internet era, it is mostly unclear whether the current specification may provide some help in the design, implementation, and validation of the agents' communications.

Inadequacies of current FIPA ACL specification have been argued in previous works as well: [Pitt and Mamdani, 1999; Singh, 1998; Wooldridge, 1998] to cite some. The general idea is that directly relating agent's communications to agent's mental states – whose dynamics are typically described only through rather complex modal logics – seems generally to poorly support a true engineering approach to the construction of MASs. Clearly, choosing a given model for an agent's observable behaviour, and using it as the core paradigm for defining a standardised specification, is

*This work has been partially supported by MIUR, and by Nokia Research Center, Burlington, MA, USA.

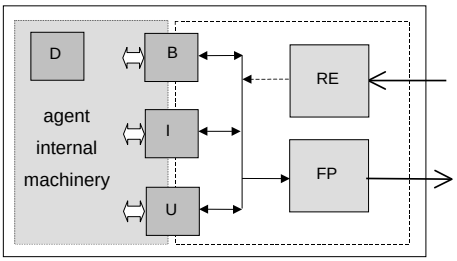

Figure 1: Agent abstract architecture implicitly assumed by FIPA Semantic Language specification

likely to implicitly promote implementors to rely on the corresponding agent's actual architecture. We believe that choosing the ACL specification model is crucial for the standardistation aims of FIPA ACL. So, in this paper we develop on the idea of modelling an agent's observable behaviour in an abstract and architecture-independent way, promoting an alternative approach for the specification of FIPA ACL.

In [Viroli *et al.*, 2001] we described the *observation issue* within computer systems in general, by providing a common ontology and a formal framework as its conceptual and scientific foundations. While the ontology is meant to characterise the paradigms used to observe a generic software component's status, the formal framework provides a tool for precisely specifying these observation patterns, and for describing a software component's interactions by interpreting them as related to observation [Viroli and Omicini, 2001]. The core idea of this approach is to model a software component as an observable knowledge source, interacting with others by providing and exploiting services involving the observation of the source's status and its dynamics. On the one hand, this model abstracts away from any agent's definition, so it provides a uniform framework for the many different agent's models, architectures, and implementations currently existing. On the other hand, our model should be effective also whenever *agentification* is exploited to tackle with legacy systems. We claim that this framework is suitable for reasoning about an agent's observable behaviour since it is grounded on top of an ontology for observation, so as to take core concepts related to observation as first-class paradigms of the model [Viroli and Omicini, 2001].

The remainder of the paper is organised as follows. Section 2 presents the observation framework for modelling agents' observable behaviour, by describing an agent abstract architecture and the corresponding operational semantics. Section 3 elaborates on the idea of exploiting the observation framework to the specification of FIPA ACL communicative acts. This is done by re-interpreting current specification and providing a new semantics grounded on the formal framework introduced. Whereas general guidelines are outlined, the specification provided here sticks to a small yet relevant subset of FIPA ACL. Section 4 discusses related works, and provides perspectives on future directions in this research.

2 Modelling Agents as Observable Sources

The ontology for observation presented in [Viroli *et al.*, 2001] interprets computer systems as made of three kinds of entity: *observable sources*, *observers*, and *coordinators*. Sources are able to provide their knowledge and services to interested observers by *manifesting* an observable behaviour through chunks of information delivered in form of messages. Coordinators are entities that can configure sources so as to produce particular kinds of manifestation to observers, namely by *conditioning* the sources' observable behaviour. Observers and coordinators can be considered as the ends of the observation patterns, so they can be easily modelled as black-box entities represented in terms of their outputs – coordinators producing conditionings – and inputs – observers receiving manifestations. On the other hand, sources are the core of the observation pattern, so a source's inner state and dynamics are worth to be explicitly represented, at least as far as they affect the way in which the source participates to the observation pattern. As a result, the core idea of our framework based on observation is to represent components as sources and their interactions in terms of the observation pattern [Viroli and Omicini, 2001].

2.1 The Source Abstract Architecture

On top of this ontology we define an abstract architecture for agents interpreted as *(observable) sources*, focussing on representing the agents' interactions with their environment and their relation to the agents' status changes.

An agent is conceptually divided in two parts, the *agent (observable) core* and the *agent internal machinery*, as show in Figure 2. The agent's core represents the part of the agent directly involved in how the agent's behaviour can be perceived by and can affect its environment. In order to clearly represent how the agent internal machinery may affect the observable behaviour, the core is itself divided in two parts: *position* (P) and *configuration* (C). The agent's position represents the part of the agent's status affecting its observable behaviour; the agent's configuration is the part keeping track of the interactions the agent is in charge of handling.

According to this structure, the agent's dynamics can be described as follows. The agent is said to be in *equilibrium* when its core does not change. Because of the architecture shown in Figure 2, this does not model agent's inactivity, but rather the internal machinery's activity not perceivable by the environment. Equilibrium can be broken in one of two ways. On the one hand, a coordinator entity – e.g. another agent – can change the agent's configuration through a conditioning (Cnd), modelling the agent receiving a message. On the other hand, the agent can perform a so-called spontaneous move (Spn), representing the internal machinery's activity causing a change on the place. Spontaneous moves are used to model the agent proactively willing to manifest some kind of observ-

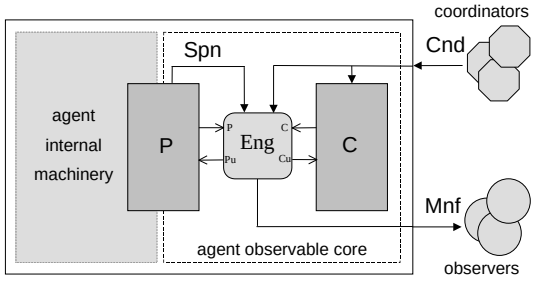

Figure 2: Agent abstract architecture in the observation framework

able behaviour. As a result of one of these two events, the agent enters in *motion* and performs an *observation action*, that is, it *evaluates* the current state of place and configuration, producing: (i) a change on their state – namely a *place update* (Pu) and a *configuration update* (Cu) – and (ii) a set of messages sent out to some observers – called the agent's manifestations (Mnf). This evaluation is performed by a logical agent's component called the *core engine* (Eng). In particular, the evaluations fired as response to a conditioning corresponds to manifestations of a reactive behaviour, whereas those due to spontaneous moves corresponds to a proactive behaviour. After processing an observation action, the agent returns in equilibrium waiting for a new conditioning or spontaneous move.

2.2 Notation

Given any set X, ranged over by variable x by default, the set of multisets over X is denoted by $\overline{X}$, and is ranged over by variable $\overline{x}$. The union of two multisets is represented through the binary operator '|', as in $\overline{x} = \overline{x}'|\overline{x}''$, while the void multiset is denoted by $\bullet$.

Given any set X, symbol $\perp_X$ is used to denote an exception value in set $X_\perp$ defined as $X \cup \{\perp_X\}$. Variable $\widehat{x}$ is automatically defined that ranges over set $X_\perp$. The set of functions associating elements of the set *Dom* to elements of the set *Range* is denoted by $Dom \mapsto Range$; function updating is defined by the operator $f[d \mapsto r]$, defined as $f \setminus \{(d \mapsto r') \in f\} \cup \{d \mapsto r\}$, which denotes a function obtained from f by updating (or adding) $f(d)$ to r.

A finite *sequence* (or *tuple*) of elements $x_1, .., x_n$ is represented by the symbol $\langle x_1, .., x_n\rangle$ ranging over the set $\langle X_1, .., X_n\rangle = X_1 \times .. \times X_n$. A *record* R containing a tuple of sets $\langle X_1, .., X_n\rangle$, each tagged by a different label l_1,...,l_n of any sort, is denoted by the syntax $\langle l_1 : X_1, .., l_n : X_n\rangle$ ranged over by elements $r = \langle l_1 : x_1, .., l_n : x_n\rangle$. R is defined as the function $(\{l_1\} \mapsto X_1) \cup ... \cup (\{l_n\} \mapsto X_n)$, so that $r(l_i) = r_{l_i} = x_i \in X_i$.

In this paper, we exploit the formal framework of transition systems, which is widely used in the field of process algebra [Bergstra *et al.*, 2001] to specify the operational semantics of interactive systems. A transition system over set X is a triple $\langle X, \rightarrow, Act\rangle$ where Act is called the set of actions and $\rightarrow \subseteq X \times Act \times X$ is a relation. The occurrence of $\langle x, act, x'\rangle$ in $\rightarrow$ means that the software component of interest may move from state x to state x' processing action act – modelling either an internal computation or an interaction with environment –, and is represented by notation $x \xrightarrow{act} x'$.[1]

2.3 Operational Semantics

In this section we provide the operation semantics of the agent abstract architecture described in Figure 2. The specification of an agent's observable behaviour is given in terms of a triple $\langle \mathcal{P}, \mathcal{C}, \mathcal{E}\rangle$.

$\mathcal{P} = \langle P, Spn, PU, \longrightarrow_{\mathcal{S}pn}, \longrightarrow_{\mathcal{P}u}\rangle$ defines the structure of the agent's place and its dynamics through spontaneous moves and place updates. Set P defines the states of the agent's place, and is briefly referred to as the set of places; Spn is the set of spontaneous moves and PU is the set of place updates. $\longrightarrow_{\mathcal{S}pn}$ defines the effects of spontaneous moves on agent's place, so transition system $\langle P, \longrightarrow_{\mathcal{S}pn}, Spn\rangle$ gives semantics to notation $p \xrightarrow{spn}_{\mathcal{S}pn} p'$ – representing place p moving to p' due to spontaneous move spn. Analogously, $\longrightarrow_{\mathcal{P}u}$ defines the effects of place updates to places, so that transitions system $\langle P, \longrightarrow_{\mathcal{P}u}, PU\rangle$ gives semantics to notation $p \xrightarrow{pu}_{\mathcal{P}u} p'$ – representing place p moving to p' due to place update pu.

$\mathcal{C} = \langle C, Cnd, CU, \longrightarrow_{\mathcal{C}nd}, \longrightarrow_{\mathcal{C}u}\rangle$ defines the structure of the agent's configuration and its dynamics through conditionings and configuration updates. Set C defines the states of the agent's configuration, and is briefly referred to as the set of configurations; Cnd is the set of conditionings and CU is the set of configuration udpates. Analogously to the case of places, $\longrightarrow_{\mathcal{C}nd}$ and $\longrightarrow_{\mathcal{C}u}$ define the effects of conditionings and configuration updates by means of transitions systems $\langle C, \longrightarrow_{\mathcal{C}nd}, Cnd\rangle$ and $\langle C, \longrightarrow_{\mathcal{C}u}, CU\rangle$, giving semantics to notations $c \xrightarrow{cnd}_{\mathcal{C}nd} c'$ and $c \xrightarrow{cu}_{\mathcal{C}u} c'$.

The evaluation of spontaneous moves and conditionings is defined by the third element $\mathcal{E} = \langle \pi, \rho\rangle$, where π and ρ are relations of the kind:

$$\begin{aligned} \pi &\subseteq \langle P, C, Spn\rangle \times \langle PU_\perp, CU_\perp, \overline{Mnf}\rangle \\ \rho &\subseteq \langle P, C, Cnd\rangle \times \langle PU_\perp, CU_\perp, \overline{Mnf}\rangle \end{aligned}$$

π accepts a triple $\langle p, c, spn\rangle$ where $\langle p, c\rangle$ models the current equilibrium state – also denoted by $p \lceil c \rfloor$ – and spn is the spontaneous move firing the evaluation. ρ accepts a similar triple $\langle p, c, cnd\rangle$ representing equilibrium state $p \lceil c \rfloor$ and conditioning cnd. Then, both relations non-deterministically associates to that triples tuples of the kind $\langle \widehat{pu}, \widehat{cu}, \overline{mnf}\rangle$. There, $\widehat{pu}$ and $\widehat{cu}$ represent place and configurations updates that are applied to the agent's core as a result of evaluation – with $\perp_{PU}$ and $\perp_{CU}$ denoting that no changes affect place and configuration, respectively. $\overline{mnf}$ represents the multiset of manifestations sent out due to the evaluation. Tuple $\langle p, c, spn, \widehat{pu}, \widehat{cu}, \overline{mnf}\rangle$ occurring in π is denoted by symbol $\langle p, c, spn\rangle \lhd \rhd_\pi \langle \widehat{pu}, \widehat{cu}, \overline{mnf}\rangle$, and analogously, $\langle p, c, cnd, \widehat{pu}, \widehat{cu}, \overline{mnf}\rangle$ occurring in ρ is denoted by $\langle p, c, cnd\rangle \lhd \rhd_\rho \langle \widehat{pu}, \widehat{cu}, \overline{mnf}\rangle$.

[1] Here, relation $x \xrightarrow{\perp} x$ is supposed to hold by default.

Given an agent specification $\langle \mathcal{P}, \mathcal{C}, \mathcal{E} \rangle$, the corresponding operational semantics (its interactive behaviour over time) is given by transition system $\mathcal{O}bs$:

$$\mathcal{O}bs = \langle P \times C, \longrightarrow_{\mathcal{O}bs}, \{cnd \triangleright \overline{mnf}, spn \diamond \overline{mnf}\} \rangle$$

$\mathcal{O}bs$ describes the core engine's behaviour, that is, how an agent's equilibrium state – also called *position* – evolves through two kinds of interaction with the environment: the agent spontaneously sending manifestations (symbol $\diamond$), or the agent reacting to a conditioning sending manifestations (symbol $\triangleright$). This behaviour is described by the rules:

$$\frac{p_0 \xrightarrow{spn}_{\mathcal{S}pn} p' \qquad \langle p', c_0, spn \rangle \lhd\rhd_\pi \langle \widehat{pu}, \widehat{cu}, \overline{mnf} \rangle \qquad p' \xrightarrow{\widehat{pu}}_{\mathcal{P}u} p \qquad c_0 \xrightarrow{\widehat{cu}}_{\mathcal{C}u} c}{p_0 \lceil c_0 \rfloor \xrightarrow{spn \diamond \overline{mnf}}_{\mathcal{O}bs} p \lceil c \rfloor}$$

$$\frac{c_0 \xrightarrow{cnd}_{\mathcal{C}nd} c' \qquad \langle p_0, c', cnd \rangle \lhd\rhd_\rho \langle \widehat{pu}, \widehat{cu}, \overline{mnf} \rangle \qquad p_0 \xrightarrow{\widehat{pu}}_{\mathcal{P}u} p \qquad c' \xrightarrow{\widehat{cu}}_{\mathcal{C}u} c}{p_0 \lceil c_0 \rfloor \xrightarrow{cnd \triangleright \overline{mnf}}_{\mathcal{O}bs} p \lceil c \rfloor}$$

Transitions can be of two kinds:

$$p_0 \lceil c_0 \rfloor \xrightarrow{spn \diamond \overline{mnf}}_{\mathcal{O}bs} p \lceil c \rfloor$$
$$p_0 \lceil c_0 \rfloor \xrightarrow{cnd \triangleright \overline{mnf}}_{\mathcal{O}bs} p \lceil c \rfloor$$

In the former case, the agent in position $p_0 \lceil c_0 \rfloor$ moves to position $p \lceil c \rfloor$ producing manifestations $\overline{mnf}$. In particular, as the place changes due to spontaneous move spn, relation π provides (i) a place update leading to p, (ii) a configuration update leading to c, and (iii) a set of manifestations $\overline{mnf}$. In the latter case, the agent in position $p_0 \lceil c_0 \rfloor$ receives conditioning cnd, thus moving to position $p \lceil c \rfloor$ and producing manifestations $\overline{mnf}$. In particular, as the configuration is conditioned by cnd, relation ρ provides (i) a place update leading to p, (ii) a configuration update leading to c, and (iii) a set of manifestations $\overline{mnf}$.

3 FIPA ACL in the observation framework

In this section we provide a semantics for a small subset of FIPA ACL, aiming at providing some guidelines and hints on how it can be specified in a more architecture-independent way. First of all, since here we try to abstract away from the agent's architecture, we avoid to represent those details of the FIPA ACL semantics that strictly relates to the agent's mental state. Roughly speaking, we stick to the operational aspects of such details.

The two basic communicative acts we consider are INFORM and REQUEST. Along with CONFIRM and DISCONFIRM they can be used to define any other act – the latter two not considered here because they are just variations of INFORM involving a different mental state in the sender, namely what the sender believes the receiver knows. In our model, both INFORM and REQUEST are sent as results of a spontaneous move in the agent, thus abstracting away from their actual cause. While INFORM carries a generic *fact* $f \in F$, modelling some information the agent may carry – e.g., in SL a proposition to believe or not –, REQUEST carries a generic *action* $a \in A$ the agent can be able to internally execute. We consider that $F = S \times V$ is an association between *state-variables* $s \in S$ and *values* $v \in V$. Receiving an INFORM or a REQUEST through a conditioning causes a change on the agent's place, namely starting either an internal action execution or evaluating how to consider the fact (or whether to do so or not), which we uniformly refer to as a *processing*.

To this simple behaviour we add the communicative acts QUERY-REF and REQUEST-WHEN, which we use as paradigmatic examples of interaction patterns deserving a different treatment. On the one hand, a QUERY-REF specifies a state-variable s for which a corresponding value $v \in V$ is requested. Instead of relying on the REQUEST/INFORM pattern, QUERY-REF will be handled as a conditioning immediately reading the value from the agent's place and then sending an INFORM message – which in the observation framework is a well defined pattern called *direct reactive observation* [Viroli *et al.*, 2001]. This example should generalise the case in which a part of an agent is directly observable: allowing other entities to access it is generally easier and less expensive for the agent, requiring no special processing or deliberation. Furthermore, it also shows how to handle the case in which it is necessary to constraint an agent to answer to a message.

On the other hand, REQUEST-WHEN specifies an action $a \in A$ and a condition, here expressed as a fact $f \in F$ the receiving agent may includes or not: only when the condition is true the action is executed. This is explicitly modelled as (*i*) the corresponding conditioning that adds the request in the agent's configuration, (*ii*) such request remaining there until the condition becomes true, and (*iii*) then causing the action to be executed as in the case of REQUEST – the whole pattern referred to as an *indirect one-time-proactive observation* [Viroli *et al.*, 2001]. This example should emphasise how the agent's configuration can be used to explicitly represent the interactions the agent is in charge of handling – e.g., information on the status of these interactions within their communication protocol (following the protocol-oriented approach discussed in [Pitt and Mamdani, 1999]).

The whole behaviour can be formalised in our framework as follows. Agents identifiers are denoted by variables i, j. Communicative acts are elements $ca \in Ca$ somewhat resembling actual FIPA ACL syntax, which can be either `req(`i`,`f`,`a`)`, `inf(`i`,`f`)`, and `qry(`i`,`s`)`, the former elements – also denoted by variable ca_ρ – modelling a REQUEST if f is the exception value $\perp_F$, or a REQUEST-WHEN otherwise. Variable $a \in A$ ranges over the set of *actions* an agent can execute. Manifestations are elements `<`i`,`ca`>` where i is the sending agent. The set of places is defined as:

$$P ::= \langle \alpha : \overline{\texttt{<}i\texttt{,}ca\texttt{>}}, \omega : \overline{\texttt{<}i\texttt{,}ca\texttt{>}}, \sigma : 2^F \rangle$$

Given a place $p \in P$, *(i)* the component p_α de-

$$p \xrightarrow{\texttt{spn\_chg}(F')}_{\mathcal{S}pn} p[\sigma \mapsto F'] \qquad p \xrightarrow{\texttt{spn\_proc}}_{\mathcal{S}pn} p[\alpha \mapsto \bullet] \qquad p \xrightarrow{\texttt{spn\_issue}(\overline{\texttt{<}i,ca\texttt{>}})}_{\mathcal{S}pn} p[\omega \mapsto \overline{\texttt{<}i,ca\texttt{>}}] \quad \text{(SPN)}$$

$$p \xrightarrow{\texttt{pu\_proc}(\overline{\texttt{<}i,ca\texttt{>}})}_{\mathcal{P}u} p[\alpha \mapsto p_\alpha | \overline{\texttt{<}i,ca\texttt{>}}] \qquad p \xrightarrow{\texttt{pu\_rst}_\omega}_{\mathcal{P}u} p[\omega \mapsto \bullet] \qquad p \xrightarrow{\texttt{pu\_rst}_\alpha}_{\mathcal{P}u} p[\alpha \mapsto \bullet] \quad \text{(PU)}$$

$$c \xrightarrow{\texttt{cnd}(\texttt{<}i,ca\texttt{>})}_{\mathcal{C}nd} c \qquad c \xrightarrow{\texttt{cu\_add}(ca_\rho)}_{\mathcal{C}u} ca_\rho | c \qquad c \xrightarrow{\texttt{cu\_drop}(\overline{ca_\rho})}_{\mathcal{C}u} c \setminus \overline{ca_\rho} \quad \text{(CND,CU)}$$

$$\frac{f = \bot_F \quad \vee \quad f \in F}{\langle p, c, \texttt{cnd}(\texttt{<}i,\texttt{req}(i,f,a)\texttt{>})\rangle \lhd \rhd_\rho \langle \texttt{pu\_proc}(\texttt{<}i,\texttt{req}(i,f,a)\texttt{>}), \bot_{CU}, \bullet\rangle} \quad \text{(REQ)}$$

$$\frac{f \neq \bot_F \quad \wedge \quad f \notin F}{\langle p, c, \texttt{cnd}(\texttt{<}i,\texttt{req}(i,f,a)\texttt{>})\rangle \lhd \rhd_\rho \langle \bot_{PU}, \texttt{cu\_add}(\texttt{<}i,\texttt{req}(i,f,a)\texttt{>}), \bullet\rangle} \quad \text{(REQW)}$$

$$\frac{\langle s, v\rangle \in p_\sigma}{\langle p, c, \texttt{cnd}(\texttt{<}i,\texttt{qry}(j,s)\texttt{>})\rangle \lhd \rhd_\rho \langle \bot_{PU}, \bot_{PU}, \texttt{<}j,\texttt{inf}(i,\langle s, v\rangle)\texttt{>}\rangle} \quad \text{(QRY)}$$

$$\langle p, c, \texttt{spn\_issue}(\overline{\texttt{<}i,ca\texttt{>}})\rangle \lhd \rhd_\pi \langle \texttt{pu\_rst}_\omega, \bot_{CU}, \overline{\texttt{<}i,ca\texttt{>}}\rangle \quad \text{(SND)}$$

$$\langle p, c, \texttt{spn\_proc}\rangle \lhd \rhd_\pi \langle \texttt{pu\_rst}_\alpha, \bot_{CU}, \bullet\rangle \quad \text{(PRC)}$$

$$\frac{\overline{\texttt{<}i,ca_\rho\texttt{>}} = \{\texttt{<}i,\texttt{req}(j,f,a)\texttt{>} \in c : f \in F'\}}{\langle p, c, \texttt{spn\_chg}(F')\rangle \lhd \rhd_\pi \langle \texttt{pu\_proc}(\overline{\texttt{<}i,ca_\rho\texttt{>}}), \texttt{cu\_drop}(\overline{\texttt{<}i,ca_\rho\texttt{>}}), \bullet\rangle} \quad \text{(CHG)}$$

Figure 3: Formal semantics

notes the requests to be processed, *(ii)* p_ω the messages the agent proactively decided to send out, and *(iii)* p_σ the observable status, modelled as a set of facts. Spontaneous moves are either *(i)* updates in the observable part $\texttt{spn\_chg}(F')$ (with $F' \subseteq F$), *(ii)* events denoting the start of processing $\texttt{spn\_proc}$, and *(iii)* the issuing of messages to be sent $\texttt{spn\_issue}(\overline{\texttt{<}i,ca\texttt{>}})$. Place updates involve resetting components ω and α in the place, orderly through $\texttt{pu\_rst}_\omega$ and $\texttt{pu\_rst}_\alpha$, and adding processing requests in α through $\texttt{pu\_proc}(\overline{\texttt{<}i,ca\texttt{>}})$.

The set of configurations is $C = \overline{ca_\rho}$: thus at any time the configuration contains a multiset of pending requests REQUEST-WHEN waiting to be served. Conditionings are of the kind $\texttt{cnd}(\texttt{<}i,ca\texttt{>})$, and configuration updates are either $\texttt{cu\_add}(ca_\rho)$ $\texttt{cu\_drop}(\overline{ca_\rho})$, respectively adding a request or dropping a set of requests from the configuration. The simple semantics of spontaneous moves ($\rightarrow_{\mathcal{S}pn}$), place updates ($\rightarrow_{\mathcal{P}u}$), conditionings ($\rightarrow_{\mathcal{C}nd}$), and configuration updates ($\rightarrow_{\mathcal{C}u}$), is formally reported in the upper side of Figure 3.

In the lower side, instead, we reported the actual evaluation semantics, that is, element $\mathcal{E}$ of the specification. Rule (REQ) describes the acceptance of a REQUEST communication act or a satisfiable REQUEST-WHEN, which simply inserts the action in the α component of the place. Rule (REQW) handles the case where a REQUEST-WHEN is added to the configuration for it is not currently satisfiable. Rule (QRY) describes the acceptance of QUERY-REF, causing an INFORM to be immediately sent.

Rule (SND) describes the behaviour of an agent proactively sending messages outside, which is fired by spontaneous move $\texttt{spn\_issue(.)}$ also causing component ω to be reset through place update $\texttt{pu\_rst}_\omega$. Proactive rule (PRC) is about the agent consuming processing requests currently pending in the place, producing no manifestations. Proactive rule (CHG) handles the case where, due to a state change, requests pending in the configuration are served.

The communicative acts not presented here can be managed similarly, e.g. acts CANCEL, REQUEST-WHENEVER, SUBSCRIBE following the examples reported in [Viroli and Omicini, 2002].

4 Discussion and Related Works

In this paper we define a framework for reasoning about an agent's observable behaviour in an abstract and architecture-independent way. This is done by defining an abstract and parametric machine grounded on top of the observation ontology presented in [Viroli *et al.*, 2001], and defining its operational semantics through a transition systems-based specification. The suitability of this model for agent-based systems is studied in [Viroli and Omicini, 2001], where typical properties of agents such as reactivity, proactiveness, autonomy, and social ability are accounted for in the observation framework. In this paper, instead, we focus on defining a semantics for FIPA ACL in the observation framework, by describing through an operational semantics which conditions cause a communicative act to be sent and what is the effect on the receiver, similarly to the existing FIPA ACL specification. Most of these ideas are appliable to other ACL as well, such as KQML [Labrou and Finin, 1997]. Here we concentrated on the case of FIPA, since we believe the motivations of our work are fairly crucial for an organisation aiming at defining a standardisation of agent-based systems.

Critics to current FIPA ACL specification and different semantics are introduced also in other works.

In [Pitt and Mamdani, 1999], it is argued that the semantics of a standardised ACL, which in the case of FIPA only concerns *intentional* aspects internal to each agent, should instead be more protocol-oriented (as claimed also in [Singh, 1998; Wooldridge, 1998]). In their proposal each communicative act is seen in the context of a protocol, and therefore it is possible to associate to each message receiving a corresponding message sending the agent is obliged to perform. Moreover, each agent has to take into account the current state of the protocols it is currently involved in. We believe that the framework we described in this paper can indeed be a suitable basis for specifying a protocol-oriented semantics to communications as well, since it easily allows both to models the required causality relations between messages receiving and sending (such as in our specification of QUERY-REF), as well as the concept of a message as part of a protocol (exemplified by our management of REQUEST-WHEN) – thanks to the powerful abstraction of the agent's configuration. However, providing more details on this issue and a complete specification in the observation framework is left as future work.

Another well known approach to modelling an agent communication language has been developed by van Eijk et al. In [van Eijk *et al.*, 2000], for instance, a programming language is defined where agents are characterised by their mental state – including belief states and a goal state – and where agent communications follow the rendezvous schema, a version of the classic remote procedure call (RPC) where the target processes requests using an interleaved pattern. Then, the semantics of the language is defined through an operational semantics, describing how agents evolve by internal computations (such as believes update) and through communications. The aim of that work, however, is not to reconsider agent communication languages, but rather to give an operational semantics to existing ones. From a more technical point of view, the main difference with respect to our approach is that the observation framework relies on a parametric machine – i.e., the abstract architecture – which can be specialised to different behaviours, the specification of FIPA ACL outlined here being just one of them.

One of the most important applications to an ACL specification is as a reference for implementors, thus, it is crucial to provide a validation test stating if an agent implementation actually conforms to the ACL specification. However, there are serious doubts about finding an adequate conformance test for current FIPA ACL specification. While this is considered a future work by FIPA organisation, Wooldridge argued it is likely that such a specification is not verifiable at all [Wooldridge, 1998].

Currently, we have not faced the problem of testing conformance in our framework, however, we argue that this should be easier than in a specification based on SL language. On the one hand, as in [Viroli and Omicini, 2001; 2002] we showed how the observation framework is generally suitable for modelling an agents' observable behaviour independently from its architecture. On the other hand, modelling an interactive software component through an operational semantics is one of the classical approaches on which existing techniques for validating interaction protocols are based on – e.g. in the field of process algebras [Bergstra *et al.*, 2001]. Addressing the conformance issue, and analising in detail the specification of FIPA ACL and their adaptation to the observation framework are our main research directions in this field.

References

[Bergstra *et al.*, 2001] Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors. *Handbook of Process Algebra*. North-Holland, 2001.

[Cohen and Levesque, 1990] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

[FIPA, 2000] FIPA. FIPA communicative act library specification. `http://www.fipa.org`, 2000. Doc. XC00037H.

[Labrou and Finin, 1997] Yannis Labrou and Tim Finin. A proposal for a new KQML specification. Technical Report TR-CS-97-03, University of Maryland Baltimore County, 1997.

[Pitt and Mamdani, 1999] Jeremy Pitt and Ebrahim Mamdani. A protocol-based semantics for an agent communication language. In Thomas Dean, editor, *16th Intl. Joint Conf. on Artificial Intelligence (IJCAI '99)*, pages 486–491, Stockholm, Sweden, 1999. Morgan Kaufmann.

[Sadek, 1992] M. David Sadek. A study in the logic of intention. In *3rd Conf. on Principles of Knowledge Representation and Reasoning*, pages 462–473, Cambridge (MA), USA, 1992.

[Singh, 1998] Munindar P. Singh. Agent communication languages: Rethinking the principles. *IEEE Computer*, 31(12):40–47, 1998.

[van Eijk *et al.*, 2000] Rogier M. van Eijk, Frank S. de Boer, Wiebe van der Hoek, and John-Jules Ch. Meyer. Operational semantics for agent communication languages. In Frank Dignum and Mark Greaves, editors, *Issues in Agent Communication*, volume 1916 of *LNAI*, pages 80–95. Springer, 2000.

[Viroli and Omicini, 2001] Mirko Viroli and Andrea Omicini. Multi-agent systems as composition of observable systems. In *WOA 2001 – Dagli oggetti agli agenti: tendenze evolutive dei sistemi software*, Modena, Italy, 4–5 September 2001. Pitagora Editrice Bologna.

[Viroli and Omicini, 2002] Mirko Viroli and Andrea Omicini. Specifying agents' observable behaviour. In *1st Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 2002)*, Bologna, Italy, 15–19 July 2002. ACM.

[Viroli *et al.*, 2001] Mirko Viroli, Gianluca Moro, and Andrea Omicini. On observation as a coordination pattern: An ontology and a formal framework. In *16th ACM Symposium on Applied Computing (SAC 2001)*, pages 166–175, Las Vegas (NV), 11–14 March 2001. ACM.

[Wooldridge, 1998] Michael Wooldridge. Verifiable semantics for agent communication languages. In Yves Demazeau, editor, *3rd Intl. Conf. on Multi Agent Systems (ICMAS '98)*, Paris, France, 4–7 July 1998. IEEE Press.