The Knowledge Engineering Review, Vol. 21:1, 49–69. © 2006, Cambridge University Press
doi:10.1017/S0269888906000774 Printed in the United Kingdom

Operating instructions for intelligent agent coordination

MIRKO VIROLI[1], ALESSANDRO RICCI[2] and ANDREA OMICINI[3]

[1]*DEIS, Alma Mater Studiorum—Università di Bologna, via Venezia 52, 47023 Cesena, Italy;*
e-mail: mirko.viroli@unibo.it
[2]*DEIS, Alma Mater Studiorum—Università di Bologna, via Venezia 52, 47023 Cesena, Italy;*
e-mail: a.ricci@unibo.it
[3]*DEIS, Alma Mater Studiorum—Università di Bologna, via Venezia 52, 47023 Cesena, Italy*
e-mail: andrea.omicini@unibo.it

Abstract

In contrast to standard approaches based on agent communication languages (ACLs), environment-based coordination is emerging as an interesting alternative for structuring interactions in multiagent systems (MASs). In particular, the notion of coordination artifacts has been proposed as an engineering methodology to build runtime abstractions effectively providing collaborating agents with specifically designed coordination tasks.

In this paper, we study the semantics for the interaction of agents with coordination artifacts playing the same role of ACL semantics, that is, supporting semantic interoperability between agents developed by different parties through the connection between rationality and interaction. Our approach is rooted on the notion of *operating instructions* of coordination artifacts, which—as with a manual for a human exploiting a device—describe the interaction protocols the agent can follow as well as the mentalistic semantics of each single interaction. By tackling some of the most relevant issues raised in the context of ACL semantics, our framework allows intelligent, BDI-like agents to carry on complex interactions through coordination artifacts in a rational way.

1 Introduction

In contrast to standard approaches based on agent communication languages (ACLs) (Searle, 1969), environment-based coordination (Parunak *et al*., 2003; Bonabeau *et al*., 1999; Weyns *et al*., 2005) is emerging as an interesting alternative for structuring interactions in multiagent systems (MASs). In particular, Ricci *et al*. (2003) propose the notion of coordination artifact as an engineering methodology to build runtime abstractions effectively providing collaborating agents with given coordination tasks.

Unlike a middle agent (Sycara, 2001), a coordination artifact is not a rational entity: its behaviour is not understood in terms of achieving a goal in autonomy. Rather, it is an entity realizing a specific coordination task without the freedom of autonomy, but exporting a specific *usage interface*, some *operating instructions* for agents and a coordinating behaviour that can be flexibly adapted by a human/agent manager according to emerging needs. Most relevant here, coordination artifacts neglect proactivity, which significantly affects the way agents interact with them: instead of sending and receiving messages, agents execute *actions* over artifacts that eventually *complete*, possibly involving a perception.

In order to study how agents can rationally exploit coordination artifacts to achieve their goals, we study a semantics for the interaction of agents with coordination artifacts, conceptually playing the same role of existing ACL semantics (Labrou & Finin, 1997; Singh, 1998; Verdicchio &

Colombetti, 2003; van Eijk *et al.*, 2000; Parsons *et al.*, 2004). Our approach is rooted on the notion of operating instructions for coordination artifacts, which—as with an operating manual for a human using a device—describes the interaction protocols that an agent is allowed to follow when exploiting the artifact, along with the mentalistic/rational meaning of each single interaction. In particular, our semantic approach is based on the idea that following some operating instructions basically means intending to execute the associated interaction protocol on a step-by-step basis, and relying on mentalistic preconditions/effects to connect mind to interaction so as to maximize the benefits of coordination.

We develop our approach formally. Operating instructions are given process-algebraic semantics similarly to concurrent languages such as CCS by Milner (1989). As the use of process algebras in the context of MASs is relatively new and unexplored—see Viroli & Omicini (2005) for a deeper discussion on this—we rely on it because it easily deals with the step-by-step descriptive character of operating instructions. The operational semantics of operating instructions is then naturally defined and is used by rational agents for practical reasoning. The abstract architecture of rational agents following operating instructions is described by relying on an operational semantics as well, serving as a specification of the rational viewpoint over interactions.

The remainder of the paper is organized as follows. Section 2 motivates this research line, discusses the role of the environment and describes the framework of coordination artifacts. Section 3 provides an abstract model for coordination artifacts and Section 4 formally presents our semantic approach to agent interaction with coordination artifacts. Section 5 provides examples showing the usefulness of our approach, Section 6 discusses related work in the context of ACL semantics and finally Section 7 concludes by providing closing remarks.

2 Environment and artifacts

2.1 Motivation

One of the key principles of the research on agent-based systems is the intentional stance (Dennett, 1987): as far as the complexity of MAS design is concerned, agent behaviour is better understood, predicted and analysed in terms of mental properties—such as beliefs, desires and intentions (fears, hopes, and so on)—instead of just relying on the actual agent design. This approach has then promoted a connection between the agent abstraction and human behaviour (and between MASs and social groups of people), which drove a number of typical design choices in the agent field. Most notably, these include the idea of basing agent communications on the speech act theory (Searle, 1969): each communication act is directed to another agent and is an utterance whose meaning is associated to a human-like kind of speech, also called *performative*—for example, it can be an *inform*, a *request* or a *propose*. This model also made into the agent mainstream: in the FIPA standard, agents interact with each other through direct communication via speech acts, exploiting the so-called FIPA ACL (FIPA, 2000).

However, direct communication is not the only viable approach: new models are emerging in different research areas of MASs, for example, in environment-based coordination such as stigmergy and, more generally, mediated-interaction frameworks and infrastructures based on forms of coordination/cooperation without direct communication (Parunak *et al.*, 2003; Fenster *et al.*, 1997; Ciancarini *et al.*, 2000; Bonabeau *et al.*, 1999; Odell *et al.*, 2003). Even if we stick to human behaviour, we observe that direct speaking is not the only way of cooperating.

Rather, as emphasised in the Activity Theory (AT)—(a social-psychological theory about dynamics in human activity), see for instance Nardi (1996)—humans often use *mediating artifacts* such as blackboards, semaphores, mailboxes, form sheets and maps to make cooperation easier, more flexible and effective. So it is clear that other forms of interactions might be interesting, forms that do not involve two directly communicating agents but rather an agent acting through an environmental abstraction that provides a suitable mediation for agent interactions—an

abstraction that might not be fruitfully understood in terms of the intentional stance, and which is natural to be seen as part of the agent environment.

Also, AT suggests a meta-model where MAS entities are essentially of two different sorts: agents, featuring autonomy and possibly being proactive, and artifacts, providing functions and services to agents and essentially reactive (Ricci *et al*., 2005). Coordination artifacts, therefore, do not imply that MAS coordination does not, in principle, require deliberation, intention or autonomy anyway. Instead, they are the instrument for automating MAS coordination whenever a law of coordination can be conceived as a computable service provided by some suitably-engineered component (Viroli & Omicini, 2003)—so, whenever an agent is not required, in a sense.

2.2 Related literature

While the study of environment has only recently become a key issue in the MAS field (Weyns *et al*., 2005), it has often and generally traversed a number of related research fields such as Robotics, Artificial Intelligence (AI), Computer-Supported Cooperative Work (CSCW), Human–Computer Interaction (HCI), as well as Cognitive and Social Sciences.

It was the work by Brooks (and the so-called 'behaviour-based AI' view) that first put the environment at the core of the studies on intelligent systems (Brooks, 1991). There, intelligence is not 'conceptually contained' within an artificial agent, but is rather the result of agent interaction with the environment. In this acceptation, the ability to shape the space of agent interaction is an essential pre-condition to make intelligence emerge (Omicini & Papadopoulos, 2001).

In the AI field, this departure from the 'classic AI' paradigm has generated the 'representations versus no representations' argument (Agre, 1995): that is, should agents maintain a representation of their environment or should they instead interact and behave without a mental representation of the environment? The two corresponding streams of research either implicitly or explicitly referred to two extreme views of environment, where cognitive agents either interact in a trivial setting where they represent the only source of disruption, or live 'in the wild'—that is, in a mostly unstructured, unpredictable and even hostile environment. Only recently, it was finally recognized that agents more often live in complex yet well-structured environments, where 'the structure of the world compensates for the weaknesses of cognitive architectures' (Agre, 1995, page 13). Therefore, in a huge number of real-world application scenarios, the environment is a potential subject of the engineers' work, aimed at structuring the agents' surroundings so as to support their goal-oriented activity.

From the field of Cognitive Sciences also comes the notion of 'adapting the environment instead of oneself' (Kirsh, 1996). This roughly corresponds to the idea that the environment is not something that should be taken as it is by agents that have thus to adapt in order to survive and possibly achieve their own goals: instead, the environment is something that can be suitably designed and engineered according to the agents' purposes (Kirsh, 1995).

This is more than understood in the field of CSCW, where authors such as Schmidt have matched 'articulation of work' (Schmidt & Bannon, 1992) with 'coordinative artifacts' (Schmidt & Simone, 2000), converging to a reflection on the role of the environment in supporting and promoting collaboration activities (Schmidt & Wagner, 2004). From these works the need for shaping the environment in order to make it fit for individual and social activity clearly emerges: human organizations largely depend on the structuring of the working space, supporting/compensating the cognitive (in)abilities of intelligent agents participating in the organization activities.

Once it is conceived as a workplace (Kirsh, 1995), or a field of work (Susi & Ziemke, 2001), the environment for cognitive agents can be seen as articulated in *artifacts*—for example, coordinative artifacts and representation artifacts in Schmidt & Wagner. Artifacts encapsulate the environment responsibilities to support individual and social activities within a collaboration setting, and embody a history of social practise within an organization.

In the field of MAS, the idea that the environment is the place where to establish the laws regulating agent behaviours and MAS interactions is the basis of both research on Coordination Models and Languages (Omicini & Papadopoulos, 2001), situated MASs (Ferber & Müller, 1996), and also stigmergy-coordinated MASs (Parunak, 1997). When encapsulated in suitably expressive abstractions, rules and laws are usually enforced by MAS infrastructures (Omicini *et al*., 2004a), as in the case of field-based coordination (Mamei *et al*., 2003). Infrastructures are in fact the most systematic way for MAS engineers to shape agent environment and provide appropriate solutions to recurring MAS application problems.

Research on computational institutions—such as electronic institutions (Noriega & Sierra, 2002), logic-based institutions (Vasconcelos, 2004) and normative MAS (Boella & van der Torre, 2003)—is developing the notion of regulating infrastructure along this very line. Computational institutions allow MAS engineers to super-impose laws and norms to agents of a MAS. Norms can be enacted in a prescriptive way (only admissible behaviours are possible) or unruly behaviours can be simply detected and sanctioned by the infrastructure. Computational institutions can then be seen as a possible peak along the line of structuring MAS environment for cognitive agents: there are in fact, laws and norms governing the life of agents in the structured environment that can either be made explicitly available for agent reasoning, or be induced by agent interpretation—as in Boella & van der Torre (2003).

2.3 The coordination artifact abstraction

Ricci *et al*. (2003) introduce the notion of *coordination artifacts* to model persistent entities specialized in providing a coordination service in a MAS (Ricci *et al*., 2003; Omicini *et al*., 2004b). The term coordination should be here understood according to its more general acceptation as the management of dependencies among separate activities (Malone & Crowston, 1994), shaping and constraining the (agent) interaction space (Ciancarini *et al*., 2000). Coordination artifacts are *infrastructure* abstractions meant to improve coordination activities automation; they can be considered then as basic building blocks for creating effective shared collaborative working environments, alleviating the coordination burden for the involved agents.

Basically, a coordination artifact entails a form of mediation among the agents using it, and effectively embeds and enacts some coordination policies. Accordingly, two basic aims can be identified: *constructive*, as an abstraction essential for creating/composing social activities, and *normative*, as an abstraction essential for ruling social activities.

Coordination artifacts are first-class entities specialized in automating a coordination task, and are therefore used by software engineers to directly design, develop and support the coordination inside a system at execution time. Accordingly, the engineering of agents and coordination artifacts can be conceptually separated: on the one hand, individual agents are designed and developed for achieving some specific goal and for executing some specific task, on the other hand, coordination artifacts are useful to support agent coordination and achieve the global tasks of the systems.

The basic properties of the agent abstraction have been extensively described in literature, in terms of autonomy, pro-activeness, reactivity, social ability and so on (Wooldridge & Jennings, 1995). Analogously, it is possible to sketch the basic properties of coordination artifacts as well, which are indeed different from the agent properties. First, coordination artifacts realize a coordination activity by applying some kind of (coordination) laws ruling agent interactions. To this end, they typically adopt a computational model suitable for effective and efficient interaction management, whose semantics can be easily expressed with concurrency frameworks such as process algebras, Petri nets, or Event–Condition–Reaction rules. In other words, they are specialized in effectively specifying and applying pre-designed interaction rules: their behaviour is not characterized as 'pro-actively achieving a goal in autonomy' like for the agent abstraction, but to react to agent actions and apply forms of synchronization, scheduling, knowledge mediation, workflow management and so on. Then, coordination artifacts are meant to be fully *inspectable* and

controllable: they provide interfaces for inspecting as well as controlling (testing, debugging, diagnosing, etc.) their internal state in order to support runtime management. Also, *malleability* is an important property for coordination artifacts exploited in open agent systems characterized by unpredictable events and dynamism: they provide the means for dynamically adapting and changing their coordinating behaviour, either by engineers (humans) willing to sustain the MAS behaviour or by agents responsible of managing the coordination artifact. In principle, this allows for flexibly facing possible coordination breakdowns or evolving/improving the coordination service provided.

Finally, *infrastructures* play a key role in supporting the coordination artifact framework at runtime. As mentioned previously, coordination artifacts are meant to be not only tools for the design of MAS, but first-class entities populating the MAS environment. The aim of *keeping the abstraction alive* (from design to execution time) is essential for governing the complexity of system engineering, reducing the gap between the various engineering stages and being able to identify and manipulate directly design abstractions—such as agents and coordination artifacts—at runtime (Omicini *et al*., 2004a). In order to give some concreteness to the abstract notion of coordination artifacts, we conclude the section with an example of an existing model/infrastructure (TuCSoN[1]) implementing most of the coordination artifact framework.

2.4 Coordination artifacts in practise: TuCSoN

TuCSoN is an infrastructure for MAS coordination (Omicini & Zambonelli, 1999). TuCSoN agents interact by exchanging *tuples*—ordered collections of heterogeneous information chunks—through coordination artifacts called *tuple centres*. As with Linda tuple spaces (Gelernter, 1985), tuple centres are accessed by agents via simple communication operations (*out*, *rd*, *in*, *inp*, *rdp*) writing, reading and consuming tuples. Unlike tuple spaces, tuple centres are *programmable*: their behaviour in response to communication events can be programmed through *specification tuples* so as to encapsulate coordination laws—for all the details on tuple centres, we forward the interested reader to Omicini & Denti (2001).

Then, tuple centres can be conceived as general-purpose coordination artifacts, which can be customized (programmed, tuned) dynamically to entail a specific coordinating behaviour. Generally speaking, tuple centres exhibit the properties that characterize coordination artifacts: they provide different levels of inspectability—both the communication and the coordination state can be inspected at runtime—and different levels of malleability and controllability, both by dynamically changing their coordinating behaviour and by controlling its execution by means of proper infrastructure tools (Denti *et al*., 2002).

3 An abstract model

According to Omicini *et al*. (2004b), the basic abstract model for coordination artifacts features:

- a *usage interface*, defined in terms of a set of *operations*;
- a set of *operating instructions*, describing how to use the artifact in order to exploit its coordination service;
- a *coordinating behaviour specification*, describing the coordinating behaviour of the artifact, in terms of coordination rules required for enacting the coordination service.

In this section, we will focus on usage interface and operating instructions, which are central for defining a semantics for agent interactions through coordination artifacts—the reader interested in a more comprehensive treatment of coordination artifacts can refer to Omicini *et al.* (2004b).

[1] TuCSoN technology is available as an open-source project at the TuCSoN Web site `http://tucson.sourceforge.net`.

3.1 Usage interface

Each coordination artifact defines a *usage interface* describing the allowed interactions with agents. This is expressed as a set of *operations*, which can be executed by the agent and eventually complete providing information on the action outcomes. Thus, the coordination service provided by an artifact is exploited by the agent in two steps: (i) an action is executed by the agent over the artifact, specifying which operation is involved, and (ii) eventually the execution of the operation ends, causing the agent to perceive the *completion* to the executed action. Correspondingly, information flows from the agent to the coordination artifact are taken into account by actions over the artifact, while information flows from the coordination artifacts to the agent are carried by (perceptions of) completions.

From the agent viewpoint, coordination artifacts are very much like physical resources: they reside into its environment, and the agent acts upon and senses them. Thus, actions over coordination artifacts are more similar in nature to physical acts rather than communicative acts (Omicini *et al.*, 2004c): an agent can execute an action only if the corresponding operation is allowed by the coordination artifact. After execution, the agent will later perceive that the operation is completed: this perception possibly carries a reply information from the artifact—for example, information on the action outcome, on success, failure and so on.

The reason for this specific interaction schema is that coordination artifacts are not proactive entities, thus they are always seen as the target of an interaction and never as its source. In other words, action and perception are modelled in a conceptually atomic way supporting the idea that the agent acts upon the artifact and never the opposite—which would indeed clash with the very notion of agent autonomy.

3.2 Operating instructions

Other than usage interface, coordination artifacts introduce the idea of *operating instructions*, or sometimes *instructions* for short, which is at the root of the semantic approach described in this paper. An agent can take part to a collaborating group through a coordination artifact by playing one of the specific roles—in the broad use of the term—that the artifact supports. With any of these roles, operating instructions are associated—inspired by what a manual does for physical devices—that describe all the details about how an agent playing the role can/has to exploit the artifact.

On the one hand, operating instructions specify the interaction protocol to be used, namely, the precise sequences of actions and perceptions allowed to the agent. Notice that at any given time the agent might be allowed to execute more than one action, each involving a different protocol continuation—naturally, since a protocol is not simply a rigid sequence of actions but rather the prescription of a set of admissible sequences.

On the other hand, operating instructions are also used to connect interactions to agent rationality, which is a necessary tool to achieve *semantic interoperability* (Bergenti, 2003)—to let agents understand the rational meaning of interactions. Inspired by purely mentalistic approaches to ACL semantics, such as FIPA ACL (FIPA, 2000) and KQML (Labrou& Finin, 1997), but overcoming some of their drawbacks, as discussed in Section 6.2, by annotating actions and perceptions by *preconditions* and *effects* respectively, both expressed in terms of mental states. In particular, preconditions to actions specify the beliefs that the agent should have when the action is executed, and conversely, effects to perceptions specify the beliefs that should be added to the agent's mental state as the completion is sensed.

However, in a departure from the above mentioned ACL semantics, we do not consider preconditions and effects as prescriptive and subject to compliance verification. Instead, they are better understood as suggestions for the individual agent, with the goal of maximizing its (social) advantages when using the coordination artifact: adhering to preconditions and effects guarantees

the agent participating meaningfully in the collaboration. Prescription can be instead applied to the interaction protocol, its compliance can be not only verified but also enforced by the infrastructure—exploiting, for example, the agent coordination context (ACC) abstraction introduced by Omicini (2002) and widely discussed by Ricci *et al.* (2004).

It is worth mentioning here how an agent can become aware of its operating instructions, even though this issue is almost orthogonal to our semantic discussion. In a closed scenario, we may assume that the information on coordination artifacts—which coordination artifacts populate the agent environment, which roles they support and which operating instructions are associated to each role—is hard-coded inside the agent. In the more interesting case of an open environment, we suppose that agents negotiate with the infrastructure for the visibility/creation of some coordination artifacts (and for the permission to access them), and can then inspect an artifact to understand its operating instructions.

4 Semantics

In general, the semantics of agent interaction concerns both subjective and objective aspects (Omicini *et al.*, 2004c). While the former has to do with the individual agent viewpoint, and is typically tackled by connecting interactions with the agent rationality (mental state), the latter is more concerned with social aspects, involving issues related to interaction protocols, semantic compliance (Singh, 1998) and checking for runtime violations.

A key feature of our approach is that these two aspects are both considered but are also clearly separated. Objective aspects of the semantics are tackled by the interaction protocol specified into operating instructions, which is enforced by the coordination artifact or by the MAS infrastructure supporting it: the agent is correctly participating to the coordination task if it is interacting according to one of the admissible sequences of interactions—namely, if it is following the operating instructions. Subjective aspects are instead dealt with by preconditions and effects, which relate interactions to agent beliefs. As the need for one such division has somehow emerged in previous work criticising FIPA ACL semantics (Singh, 1998; Pitt & Mamdani, 1999), we believe that the novelty of the coordination artifact framework plays a crucial role in making this property fully effective.

4.1 A model for operating instructions

According to the typical approach used in the distributed systems field, where interaction is one of the main subjects of investigation, we introduce a process algebra that can act as a specification language for operating instructions. As the language abstract syntax is given by a formal grammar as usual, operational semantics is given in terms of a labelled transition system (Glabbeek, 2001), which precisely describes the step-by-step interactive behaviour allowed to agents by the coordination artifact. This language can be seen as a process algebra since instructions represent a true abstract process—an interactive behaviour that the agent should adhere to. A specification in this process algebra can be used, both by the agent and by the coordination artifact, to represent the state of operating instructions at a given time, and to know the next available actions and perceptions. In our semantic study, we address the case where the agent is exploiting one coordination artifact, the extension with more artifacts being a mere adaptation with semantic insights that are not crucial here.

4.1.1 Notation and syntax

Let t be a meta-variable over first-order logical terms, built over functions, variables and constants ranged over by meta-variables f, v and c respectively. Meta-variable θ ranges over variable-to-term substitutions: write θt for the term obtained from t by applying substitution θ to all the variables in it. As usual, a term t' is said to be an instance of t if and only if there exists a substitution θ such

as $\theta t = t'$, and a substitution θ is said to be more general than θ' if and only if there exists a substitution θ'' such as $\theta \circ \theta'' = \theta'$ (operator $\circ$ being usual function composition). Notation $[t'/t]$ is used for the most general substitution θ that applied to t yields t': this is seen as a partial function, so that symbol $[t'/t]$ makes sense only if the most general substitution actually exists. The notation is then abused by writing $[t'_1/t_1, \ldots., t'_n/t_n]$ for the most general substitution of $t_1, \ldots., t_n$ to $t'_1, \ldots., t'_n$.[2]

Among all the function names, we let *op* and *cm* be meta-variables over functions, and correspondingly introduce the meta-variables α and π as follows:

$$\alpha ::= op(t_1, \ldots., t_n),\ \pi ::= cm(t_1, \ldots., t_n).$$

A term α is used to model an agent action towards the coordination artifact, where its function name *op* is the name of the operation invoked; the term π is the perception of the completion of a previously executed action, and *cm* represents the name of the completion. Notice that it is not our goal to introduce a new complete theory of agent actions; rather, we aim at providing the few hypotheses over which our formal framework for interaction with artifacts can be defined—an exhaustive comparison with existing proposals is left as future work.

We define operating instructions as the terms defined by the grammar

$$O ::= 0 \mid !\alpha.O \mid \underline{\alpha}.O \mid ?\pi.O \mid O + O \mid (O \| O) \mid \mathcal{D}(t_1, \ldots., t_n)$$

thus, we see 0 as a constant, $!./2$, $\underline{.}/2$, $?./2$, $+/2$, $\|/2$ as functions, and $\mathcal{D}$ as a meta-variable over a subset of functions called *definition names*.

Instructions $O \in \mathcal{O}$ are defined inductively from *atomic instructions* through *structured instructions*. Atomic instructions include the void behaviour (0), the execution of an action α followed by continuation O ($!\alpha.O$), an action α just executed but not yet completed followed by continuation O ($\underline{\alpha}.O$) and a perception π of action completion followed by continuation O ($?\pi.O$). Structured instructions are the choice between two instructions ($+$) and the parallel composition of two instructions ($\|$).

Also, to support recursive behaviours, for example, we rely on the definition mechanism of standard calculi such as CCS (Milner, 1989).[3] We write a *definition* $\mathcal{D}(v_1, \ldots., v_n) := O$ to mean that, in given operating instructions, the use of term $\mathcal{D}(v_1, \ldots., v_n)$ is to be interpreted as instructions O—modulo substitution of the variables $v_1, \ldots., v_n$, which typically occur in O as well. We shall define the semantics of this mechanism so that, for example, given the definition $\mathcal{D}(v) := !o(v).?c(v).0$, then instructions $\mathcal{D}(t) + \mathcal{D}(t')$ are to be interpreted as $!o(t).?c(t).0 + !o(t').?c(t').0$.

For the sake of simplicity, and with no risk of ambiguity, we avoid the '.0' notation representing void continuation, writing for example, $!\alpha.?\pi$ for $!\alpha.?\pi.0$.

Simple examples of operating instructions are given below.

$!\alpha_1.?\pi_1$—means *first* executing action α_1 and *then* perceiving its completion π_1.
$!\alpha_1.?\pi_1.!\alpha_2.?\pi_2$—means executing α_1, then perceiving its completion π_1, then executing α_2 and then perceiving its completion π_2.
$!\alpha_1.(?\pi_1.O_1 + ?\pi_2.O_2)$—means first executing α_1, then if π_1 is perceived O_1 carries on, if π_2 is perceived O_2 carries on.
$!\alpha_1.?\pi_1 \| !\alpha_2.?\pi_2$—means concurrently executing the two sub-instructions $!\alpha_1.?\pi_1$ and $!\alpha_2.?\pi_2$.

[2] The reader familiar with logics might see differences between this setting and the usual logic one. In particular, we rely on a notion of substitution instead of unification. This is because the notion of substitution is more standard for the process-algebraic approach we take here.
[3] This mechanism is better known as 'agent definition'—a terminology we shall not use in the MAS context!

$\mathcal{D}$—(with the recursive definition $\mathcal{D} := !a_1.?\pi_1.\mathcal{D}$) means indefinitely executing the sequence of interactions $!a_1$ and $?\pi_1$ ($!a_1,?\pi_1,!a_1,?\pi_1,!a_1,?\pi_1, \ldots .$)—in this case the definition name $\mathcal{D}$ has arity zero.
$\mathcal{D}(t)$—(with the recursive definition $\mathcal{D}(v) := !op(v).?\pi.\mathcal{D}(v)$) means indefinitely executing action $op(t)$ and then perceiving the completion $[t/v]\pi$—in this case the definition name $\mathcal{D}$ has arity one.

4.1.2 Operational semantics

Semantic aspects of this language are described by three inter-related ingredients: the set of well-formed initial instructions, a congruence relation for instructions and a transition system semantics for instructions.

The set of well-formed initial states for operating instructions basically amounts to the surface language that has to be used to define the operating instructions for a given artifact. This is defined by the syntax

$$O_I ::= 0 \mid !a.(?\pi^1.O_I^1 + \cdots + ?\pi^k.O_I^k) \mid O + O \mid (O \| O) \mid \mathcal{D}(t_1, \ldots, t_n)$$

where $k>0$, $n\geq 0$, and definitions are of the kind $\mathcal{D}(v_1, \ldots, v_n) := O_I$. In particular, it is worth noting that the $\underline{a}.O$ construct does not belong to this language, it can be temporarily produced as instructions evolve—see the operational semantics below. Moreover, each prefix $!a$ must be followed by a choice of instructions, or possibly just one instruction ($k=1$), each prefixed by a perception π.

We then introduce a congruence relation equating operating instructions that, as far as the operational semantics are concerned, have to be considered syntactically equivalent—the term 'congruence' means such rules can be applied wherever one side of the equation appears inside instructions (Glabbeek, 2001). The congruence relation for instructions is defined as the smallest congruence relation satisfying the rules:

$$0 + O \equiv O \quad O + O' \equiv O' + O \quad O + (O' + O'') \equiv (O + O') + O''$$

$$0 \| O \equiv O \quad O \| O' \equiv O' \| O \quad O \| (O' \| O'') \equiv (O \| O') \| O''$$

$$\mathcal{D}(t_1, \ldots, t_n) \equiv [t_1/v_1, \ldots, t_n/v_n]O \text{ if } \mathcal{D}(v_1, \ldots, v_n) := O.$$

In particular, the above formulation states commutativity and associativity of '+' and '||', makes both absorbing 0, and provides the semantics of definition invocation: invoking a definition by $\mathcal{D}(t_1, \ldots, t_n)$ is considered equivalent to executed instructions $[t_1/v_1, \ldots, t_n/v_n]O$, given that the definition $\mathcal{D}(v_1, \ldots, v_n) := O$ has been provided. Notice that, since operating instructions are terms, substitutions can be applied to them in the standard way, for example, $\theta(O+O')$ is equal to $(\theta O)+(\theta O')$.

The language for operating instructions described so far can be given an operational semantics, describing: (i) what are the interactions allowed to an agent at a given time according to the current state of instructions, and (ii) how instructions evolve as the agent interacts with the coordination artifact. The set Δ of interactions, ranged over by meta-variable δ, is formed by the terms with the syntax

$$\delta ::= a \mid a{:}\pi.$$

Namely, interactions are either of the kind a, meaning execution of action a, or $a{:}\pi$, meaning perception π of the completion of action a. Operational semantics is then formalized by a labelled transition system $\langle \mathcal{O}, \rightarrow_{\mathcal{O}}, \Delta \rangle$, where the notation $O \overset{\delta}{\rightarrow}_{\mathcal{O}} O'$ is used as a shorthand for $\langle O, \delta, O' \rangle \in \rightarrow_{\mathcal{O}}$, and means that instructions $O \in \mathcal{O}$ move to new state $O' \in \mathcal{O}$ by the occurrence of

interaction $\delta \in \Delta$. The transition relation $\rightarrow_{\mathcal{O}}$ is the smallest relation satisfying the following operational rules:

$$!\alpha.O \xrightarrow{\alpha'}_{\mathcal{O}} \underline{\alpha'}.[\alpha'/\alpha]O \qquad \text{[ACT]}$$

$$\underline{a}.((?\pi.O) + O') \xrightarrow{\alpha:\pi'}_{\mathcal{O}} [\pi'/\pi]O \qquad \text{[COM]}$$

$$O + O' \xrightarrow{\delta}_{\mathcal{O}} O'' \quad \text{if } O \xrightarrow{\delta}_{\mathcal{O}} O'' \qquad \text{[CHO]}$$

$$O \| O' \xrightarrow{\delta}_{\mathcal{O}} O'' \| O' \quad \text{if } O \xrightarrow{\delta}_{\mathcal{O}} O'' \qquad \text{[PAR]}$$

$$O \xrightarrow{\delta}_{\mathcal{O}} O' \quad \text{if } O \equiv O_0,\ O' \equiv O'_0 \text{ and } O_0 \xrightarrow{\delta}_{\mathcal{O}} O'_0 \qquad \text{[CGR].}$$

Each rule defines the semantics of a different operator or construct of the language. The [ACT] rule is responsible for executing actions and handling instructions of the kind $!a.O$. Any instance a' of a can be actually executed, and the resulting state of operating instructions is obtained by the prefix $\underline{a'}$, meaning the completion to action a' is pending, followed by the continuation O to which substitution $[a'/a]$ is applied. Note that a' is guaranteed to be an instance of a since the notation $[a'/a]$ makes sense only in that case. For instance, we have that $!a(v).O \xrightarrow{a(1)}_{\mathcal{O}} \underline{a(1)}.[v/1]O$.

Similarly, the [COM] rule deals with perceptions, and handles instructions of the kind $\underline{a}.((?\pi.O)+O')$, namely, a pending action $\underline{a}$ followed by a choice. An interaction $\xrightarrow{a:\pi}_{\mathcal{O}}$ (completion π' to previously executed action a) is allowed if one of the available choices is guarded by a perception π such that π' is an instance of it. In this case this choice can be taken and the others are excluded, moving to state $[\pi'/\pi]O$. For instance, we have that $\underline{a(1)}.(?b(v).I_1 + ?c(v).I_2) \xrightarrow{a(1):b(2)}_{\mathcal{O}} [v/2]I_1$, while for example an interaction $a(1):d(2)$ would simply not be allowed.

The [CHO] rule defines the semantics of exclusive choice: if one choice allows for an action selected for execution, then the other choices are automatically excluded. The [PAR] rule sets interleaving as the concurrency model (Glabbeek, 2001): in the case of several parallel instructions, only one is allowed to execute at a given time and the others remain unchanged. Finally, the [CGR] rule is used to have the congruence relation making into the operational semantics. Owing to this rule, for instance, we do not require symmetric versions for rules [COM], [CHO] and [PAR], for congruence states commutativity of operators '||' and '+'.

The correctness of the semantics is stated by the following result, saying that from any well-formed initial state of operating instructions no (non-void) deadlock state is reached.

THEOREM *For any well-formed initial state of operating instructions* O_{I}^0 and for any allowed sequence of transitions

$$O_I^0 \xrightarrow{\delta^0}_{\mathcal{O}} O^1 \xrightarrow{\delta^1}_{\mathcal{O}} O^2 \xrightarrow{\delta^2}_{\mathcal{O}} \cdots \xrightarrow{\delta^n}_{\mathcal{O}} O^n$$

either $O^n \equiv 0$ *or for some* O^{n+1} *we have* $O^n \xrightarrow{\delta}_{\mathcal{O}} O^{n+1}$.

Proof. Because of rules [CGR], [PAR] and [CHO], any operating instructions are either void or allow for a transition due to prefixed instruction (either $!a.O$, $\underline{a}.O$ or $?\pi.O$). For the initial state of operating instructions O_I^0, this is of the kind $!a.(?\pi^1.O_I^1 + \ldots + ?\pi^k.O_I^k)$ by construction. Therefore, for any substitutions θ and θ' and for any i, two transitions are surely allowed, labelled θa and $\theta a : (\theta \circ \theta')\pi^i$. Because of rules [ACT] and [COM], in particular, this couple of transitions lead to instructions $(\theta \circ \theta')O_I^i$, which is a well-formed initial state of operating instructions again.

4.1.3 Examples

We provide here some examples of operating instructions specification, emphasizing the corresponding sequences of interactions allowed by means of the operational semantics. In the following, we denote function names by typetext fonts (as in `a`, `b`, etc.).

Specification $!\mathtt{a}(v).?\mathtt{b}(v)$ means that the agent can execute an action of the kind $\mathtt{a}(v)$ and then perceive the corresponding completion $\mathtt{b}(v)$—that is, $\mathtt{b}(v)$ is precisely the completion to action $\mathtt{a}(v)$,

modulo substitution. In particular, any instance of $\mathtt{a}(v)$ can be actually executed, where variable v is substituted by a term, for example, $\mathtt{a(val)}$ for some constant `val`. Hence, due to rule [ACT], action `a(val)` can, for example, be executed (α stands for $\mathtt{a}(v)$, α' for `a(val)` and O for $?\,\mathtt{b(v)}$) leading to the following transition:

$$!\mathtt{a}(v).?\mathtt{b}(v) \xrightarrow{\mathtt{a(val)}} \mathcal{O}\,\underline{\mathtt{a(val)}}.?\mathtt{b(val)}.$$

The new state of instructions $\underline{\mathtt{a(val)}}.?\mathtt{b(val)}$ means that action `a(val)` has just been executed, and the perception `b(val)` is expected. To model perception `b(val)` consider rule [COM] (with α as `a(val)`, π and π' as `b(val)`, O and O' as 0 and notice that $\underline{\alpha}.\pi \equiv \underline{\alpha}.(\pi.0+0)$): we have the transition

$$\underline{\mathtt{a(val)}}.?\mathtt{b(val)} \xrightarrow{\mathtt{a(val):b(val)}} \mathcal{O}\,0$$

meaning that as completion `b(val)` to action `a(val)` is perceived, the instructions move to the terminated state 0.

A more interesting example is given through the definition

$$\mathcal{D} := !\mathtt{ask}(v_q).(?\mathtt{fail}.\mathcal{D} + ?\,\mathtt{rep}(v_r).!\mathtt{ack}(v_q, v_r).?\mathtt{end}).$$

Now consider the operating instructions specified as $\mathcal{D}$ (invocation of the above definition), an agent can issue a specific query `q` towards the coordination artifact by an action `ask(q)` as seen before:

$$\mathcal{D} \xrightarrow{\mathtt{ask(q)}} \mathcal{O}\,\underline{\mathtt{ask(q)}}.(?\mathtt{fail}.\mathcal{D} + ?\mathtt{rep}(v_r).!\mathtt{ack}(\mathtt{q}, v_r).?\mathtt{end}).$$

Notice that variable v_q has been substituted throughout the continuation with term `q`. Now the agent can either perceive a failure by `fail` or a reply by `rep`. In the case of a failure we have

$$\underline{\mathtt{ask(q)}}.(?\mathtt{fail}.\mathcal{D} + ?\mathtt{rep}(v_r).!\mathtt{ack}(\mathtt{q}, v_r).?\mathtt{end}) \xrightarrow{\mathtt{ask(q):fail}} \mathcal{O}\,\mathcal{D}$$

in that the continuation to `?fail` carries on by excluding the other choice (see rule [COM]): the instructions move to the new state $\mathcal{D}$ so that the whole protocol can be executed again. If, on the other hand, the agent perceives a reply `r` to the query (`rep(r)`) we have

$$\underline{\mathtt{ask(q)}}.(?\mathtt{fail}.\mathcal{D} + ?\mathtt{rep}(v_r).!\mathtt{ack}(\mathtt{q}, v_r).?\mathtt{end}) \xrightarrow{\mathtt{ask(q:rep(r)}} \mathcal{O}\,!\mathtt{ack(q, r)}.?\mathtt{end}$$

that is, the continuation `!ack(q, r).?end` carries on (v_r being substituted by `r`): the agent should acknowledge the reply by a new action `ack` and then perceive its completion `end` as usual. Note that no completions other than $\mathtt{rep}(v_r)$ and fail are allowed by these operating instructions, by construction.

Finally, the parallel composition operator allows for those cases where more actions are concurrently executed, with the perceptions later arriving in any order. This is obtained, for example, by the operating instructions

$$!\mathtt{a}_1(v).?\mathtt{b}_1(v) \| !\mathtt{a}_2(\omega).?\mathtt{b}_2(\omega)$$

which allows, for example, the evolution

$$
\begin{array}{lll}
!\mathtt{a}_1(v).?\mathtt{b}_1(v) \| !\mathtt{a}_2(w).?\mathtt{b}_2(w) & \xrightarrow{\mathtt{a}_1(1)}_{\mathcal{O}} & \underline{\mathtt{a}_1(1)}.?\mathtt{b}_1(1) \| !\mathtt{a}_2(w).?\mathtt{b}_2(w) \\
 & \xrightarrow{\mathtt{a}_2(0)}_{\mathcal{O}} & \underline{\mathtt{a}_1(1)}.?\mathtt{b}_1(1) \| \underline{\mathtt{a}_2(0)}.?\mathtt{b}_2(0) \\
 & \xrightarrow{\mathtt{a}_2(0):\mathtt{b}_2(0)}_{\mathcal{O}} & \underline{\mathtt{a}_1(1)}.?\mathtt{b}_1(1) \\
 & \xrightarrow{\mathtt{a}_1(1):\mathtt{b}_1(1)}_{\mathcal{O}} & 0.
\end{array}
$$

4.2 Agent operational semantics

Different frameworks can be used to formally express the relationship between agents with a mental state of beliefs and intentions and the occurrence of interactions—actions and perceptions. A standard framework in the MAS community is the framework of logic, which allows for an abstract reasoning about behaviour. A notable example is Propositional Dynamic Logic and Beliefs and Intentions (PDL-BI) introduced by Paurobally *et al.* (2005), which allows to express the dynamics of beliefs and intentions. In the KARO logic by van Linder *et al.* (1996), similarly, formulas include modalities for knowledge ability, opportunity and selected wishes, along with an operator of achievement of results by actions—such actions being true protocols featuring sequential composition, choice, repetition and so on.

Another possible framework is that of operational semantics, expressing some rules of the step-by-step evolution of an agent state, relying on abstract operations on mental properties (queries and updates). In this paper we adopt the latter approach, which despite being less standard fits the semantics of operating instructions well—exploiting the framework of logics is an interesting direction for future work.

The mental state of an agent, ranged over by meta-variable φ, is here expressed as a first-order logic formula featuring modalities for beliefs (B) and intentions (I). Similarly to FIPA SL content language (FIPA, 2000), we use the syntax

$$\varphi ::= p(t_1, \ldots, t_n) \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid B\varphi \mid I\varphi$$

where p is a meta-variable over predicate names and t_i are terms.

The existence of a binary operator $\circ$ over formulas is assumed, which abstracts away from details of mental state updates and queries. The $\varphi \circ \varphi'$ notation is used to both denote the mental state including the formula φ' (querying for φ') and for the mental state obtained from φ by adding formulas in φ' (updating with φ'). The actual definition of operator $\circ$ is intentionally neglected here as it involves aspects related to the internal agent machinery and is therefore out of the semantic scope of operating instructions.

We introduce the predicates *instr*/1, *done*/1 and *comp*/2, so that, for example, B *instr*(O) means that the agent believes O to be the current state of operating instructions, I *done*(α) means that the agent intends action α to be executed and B *comp*(α, π) means that the completion π to action α has been perceived. As for operator $\circ$, the logic semantics of such predicates is neglected for it involves inner details of the agent behaviour: their impact on operational semantics is instead reported as follows. We write *Bif* φ for $B\varphi \vee B\neg\varphi$.

Analogously to FIPA ACL, or agent languages such as 3APL (Hindriks *et al.*, 1999), we fill the gap between agent interactions and mental state by assuming that operating instructions attach preconditions and effects to agent actions and perceptions. A *precondition* function *Pre* is defined that takes an action α and yields a formula φ, which represents a condition on beliefs that should hold in the agent when the action is executed. Similarly, an *effect* function *Eff* is defined that takes a perception π and yields a formula φ representing agent beliefs that are applied when the perception occurs.

Accordingly, the following operational rules describe the admissible behaviours of the rational agent exploiting the coordination artifact, in terms of how the mental state (beliefs and intentions) evolves as interactions occur:

ACTION	Precondition	PERCEPTION	Effect
`ask`(ϕ)	$\neg Bif\,\phi$	`ask_rep`(ϕ')	$B\phi'$
`get`	—	`get_rep`(ϕ)	—
`tell`(ϕ)	$B\phi$	`tell_rep`	—

Figure 1 Operating instructions for the Query-if coordination artifact: preconditions and effects

$$\frac{-}{\phi \xrightarrow{\alpha:\pi} \phi \circ B\ comp(\alpha, \pi)} \quad \text{[A-SNS]}$$

$$\frac{O \xrightarrow{\alpha:\pi}_{\mathcal{O}} O'}{\phi \circ B\ instr(O) \circ B\ comp(\alpha, \pi) \xrightarrow{\tau} \phi \circ B\ instr(O') \circ B\ Eff(\pi)} \quad \text{[A-PER]}$$

$$\frac{O \xrightarrow{\alpha}_{\mathcal{O}} O'}{\phi \circ B\ instr(O) \xrightarrow{\tau} \phi \circ B\ instr(O) \circ I\ done(\alpha)} \quad \text{[A-INT]}$$

$$\frac{O \xrightarrow{\alpha}_{\mathcal{O}} O'}{\phi \circ B\ instr(O) \circ I\ done(\alpha) \circ B\ Pre(\alpha) \xrightarrow{\alpha} \phi \circ B\ instr(O') \circ B\ done(\alpha) \circ B\ Pre(\alpha).} \quad \text{[A-ACT]}$$

The [A-SNS] rule describes agent sensing: when an action a reaches a completion π ($\xrightarrow{a:\pi}$), the agent mental state is updated by adding the formula $B\ comp(a, \pi)$. Then, such an event is managed by the agent as described by rule [A-PER]: the current knowledge about the state of operating instructions moves from O to O' according to the perception occurred ($O \xrightarrow{a:\pi}_{\mathcal{O}} O'$), and its effects $Eff\,(\pi)$ on the agent beliefs are applied as well. On the proactiveness side, rule [A-INT] deals with the generation of intentions from operating instructions: given the current instructions O, the agent can produce any intention $I\ done(a)$ provided that action a is allowed by O. Notice that, in this case, instructions do not yet move from O to O', for this is the case only when the action is actually executed. Accordingly, rule [A-ACT] says that when an agent intends to execute action a, and its preconditions $Pre(a)$ hold, then the action can be actually executed ($\xrightarrow{a}$), causing the update on the current operating instructions as well as the agent believing $done(a)$.

This model captures the main relationships between mental states and interactions, as well as the way agents could rely on operating instructions to meaningfully exploit coordination artifacts. Moreover, it abstracts away from most issues concerning proactiveness, which—even though they are relevant to an agent designer—do not have to make into semantics of interaction. These include, for example, the choice of which intention is to be generated, that is, which instance of rule [A-INT] is to be applied. This aspect in fact concerns internal planning and scheduling activity and can, for example, leverage agent intelligence—an agent could choose an action by taking into account the effects that can be obtained later in the interaction protocol.

5 Application examples

5.1 A request/response scenario

As a first, simple use case for our semantics we consider an interaction protocol resembling the usage of *query-if* FIPA performative. For this purpose, a coordination artifact is conceived supporting two roles, which we here call *client* and *server*: the former is played by an agent willing to receive information about the validity of a formula φ of interest, the latter by an agent willing to provide information about its beliefs on such formulas.

Allowed actions and perceptions, along with their preconditions and effects are summarized in Figure 1. The coordination artifact provides three kinds of action.

- The `ask`(φ) action is used by the client to query the artifact about the validity of formula φ, and has the precondition $\neg Bif\varphi$: the agent should execute the action if it neither believes φ nor $\neg\varphi$.

By the corresponding completion `ask_rep`(φ'), the agent perceives the validity of formula φ', hence we have the effect $B\varphi'$: the agent should believe the formula φ' perceived.

- The `get` action is used by the server to ask for a new pending query about the validity of a formula (with no precondition). The corresponding perception (with no effect) is of the kind `get_ rep`(φ), where φ is a formula whose validity has previously been questioned.
- Finally, the `tell`(φ) action is used by the server to state it believes the validity of formula φ, and is hence given precondition $B\varphi$. Its corresponding perception is `tell_rep` (with no effect).

Operating instructions for the two roles are expressed by the definitions below:

$$\begin{aligned}\mathcal{D}_c \;:=\; & !\texttt{ask}(\phi).(\\ & \quad ?\texttt{ask\_rep}(\phi).\mathcal{D}_c +\\ & \quad ?\texttt{ask\_rep}(\neg\phi).\mathcal{D}_c\\ &)\end{aligned}$$

$$\begin{aligned}\mathcal{D}_s \;:=\; & (!\texttt{get}.?\texttt{get\_rep}(\phi).(\\ & \quad !\texttt{tell}(\phi).?\texttt{tell\_rep} +\\ & \quad !\texttt{tell}(\neg\phi).?\texttt{tell\_rep} +\\ & \quad \mathcal{D}_s\\ &))\|\mathcal{D}_s.\end{aligned}$$

The client agent issues the request for information about the validity of a formula φ, eventually receiving a reply specifying validity of φ or $\neg\varphi$; in both cases the client agent keeps executing the protocol $\mathcal{D}_c$. On the other hand, a server agent first issues a request for receiving information on any pending query: as this is perceived the server can either notify validity of the formula $!\texttt{tell}(\varphi)$, notify validity of the negation $!\texttt{tell}(\neg\varphi)$ or simply ignore the request and proceed with the next request (by recursive invocation to $\mathcal{D}_s$). As the whole protocol is composed in parallel with $\mathcal{D}_s$, the server is allowed to handle multiple requests concurrently.

Actions and perceptions of the operating instructions also account for the relationship with agent beliefs, through preconditions and effects—as expressed above. The client should ask for the validity of a formula φ by action `ask`(φ) only if it has no information about φ ($\neg Bif\varphi$), and should believe the reply it receives ($B\varphi'$). The server gets information about a request automatically—by virtue of the interaction protocol and without any rational effect/cause. However, it may provide information about the validity of a formula φ by using the action $!\texttt{tell}(\varphi)$ only if it believes the formula φ ($B\varphi$).

This choice for preconditions and effects can be shown to allow agents with the operational semantics described in Section 4.2 to carry on these interaction protocols in a meaningful way. Suppose that for a given formula f, a client agent (willing to play instructions $\mathcal{D}_c$) initially intends to believe either f or $\neg f$, and a server agent (willing to play instructions $\mathcal{D}_s$) believes $\neg f$. The client agent can schedule any action `ask`(φ): by looking at the expected effects of doing so, as reported in its instructions, it can notice that it will eventually come to believe either φ or $\neg\varphi$. Hence, it is rational for the agent to execute action `ask`(f) and wait for a perception. On the other hand, the server has the only possibility of executing action get: the coordination artifact will match the two requests and make the server perceive the pending query through completion `get_rep`(f). Now the server has three possibilities: execute `tell`(f), execute `tell`($\neg f$) or simply ignore the request. Supposing it is social enough to be willing to reply to pending queries, it can however only execute `tell`($\neg f$), since only this action's preconditions are currently verified. Again, the coordination artifact matches the client request with the server reply, and lets the client perceive the completion `ask_rep`($\neg f$). By applying the corresponding effects, the client now believes $\neg f$, so that its initial intentions have been satisfied.

The actual behaviour of the coordination artifact is not reported for brevity, but its details are of no concern for the interaction semantics of agents. In fact, as promoted by the engineering approach underlying coordination artifacts (Ricci *et al.*, 2003; Omicini *et al.*, 2004b), the coordination rules inside the artifact could be adapted and accommodated because of emerging needs without the agents being necessarily aware of them. For instance, the artifact could itself handle a local knowledge base and reply to requests, keep track of previous replies, notify requests

```
D_p :=
    !getCFP.?newCFP(v).(
      !refuse(v).?ok +
      !propose(v).(
        ?rejected +
        ?accepted.(
          !failure.?ok +
          !result(v).?ok
    )))

D_i :=
    !CFP(v).?ok.D_h(v, v)

D_h(v, w) :=
    !getProp.(
        ?finished +
        ?newProp(v).(
          (!stop.?ok + D_h(w, w)) ‖ (
            !refuseProp(v).?ok +
            !acceptProp(v).(
              ?failure +
              ?result(v)
    ))))
```

Figure 2 Operating instructions for the Contract-Net Coordination Artifact: Interaction Protocols

to more servers applying majority voting and so on. Interestingly, these adaptations never concern the client and server agents, even though they could be known by inspection.

5.2 A Contract-Net scenario

The value of our approach can be better understood by considering a more complex coordination pattern, such as, for example, the well-known Contract-Net scenario (Smith, 1980): here, a coordination artifact mediates the interactions between an initiator and a number of participants. We assume that at the beginning the initiator and the participants negotiate with the infrastructure for a coordination artifact realizing the Contract-Net protocol. Therefore, they receive information about the operating instructions they have to use, and then simply start following them rationally.

The initiator issues call for proposals (CFP) by specifying an action it wants to be executed. To simplify our discussion without loss of generality, we suppose that such an action is a term with two variables: (i) one is bounded by the participant with information about its proposal, and is used by the initiator to evaluate which proposals have to be accepted, and (ii) the other is bounded by the participant after executing the action, and will contain the result of the execution. For instance, suppose a participant gets aware of a CFP for buying goods, due to the perception `newCFP(buy(good_name, v_price, v_qty))`; it makes a proposal (with 1000 as price) by action `propose(buy(good_name, 1000, v_qty))`, and later notifies the result of buying five items by `result(buy(good_name, 1000, 5))`. This allows us to specify the operating instructions for the participant with the general sequence ? `newCFP`(*v*).!`propose`(*v*).!`result`(*v*) as the variable *v* can get specialized further by substitution of variables to terms at each step, thanks to the operational rules [ACT] and [COM].

Figure 2 reports the detailed definitions of the operating instructions, where $\mathcal{D}_p$ and $\mathcal{D}_i$ are the instructions for participants and initiator, Figure 3 reports preconditions and effects to actions and perceptions.

5.2.1 Participant protocol

The participant first executes action `getCFP`, in order to receive information about a currently pending CFP: the completion `newCFP`(*v*) is eventually perceived, where variable *v* holds information on the requested proposal. As the CFP has been received, the agent may either refuse the CFP by the action `refuse`, or make a proposal through the action `propose`. In the latter case, the proposal can either be accepted or rejected, which is perceived by completions `accepted` and `rejected`. In the case of acceptance, the participant is meant to execute the requested action, eventually notifying the result by action `result`, or a failure by `failure`. For the sake of simplicity, we assume that at the end of this protocol instance the whole protocol is not iterated again, that is the operating instructions prevent any other actions—for example, new operating instructions have to be negotiated with the infrastructure.

ACTION	Precondition	PERCEPTION	Effect
getCFP	—	newCFP(v)	—
refuse(v)	B ¬ feasible(v)	ok	—
propose(v)	B feasible(v)	rejected	—
		accepted	—
failure	¬ B feasible(v)	ok	—
result(v)	B done(v)	ok	—
CFP(v)	¬ B done(v)	ok	—
getProp	—	finished	—
		newProp(v)	—
stop	—	ok	—
acceptProp(v)	—	failure	—
		result(v)	B done(v)
refuseProp(v)	—	ok	—

Figure 3 Operating instructions for the Contract-Net Coordination artifact: preconditions and effects

5.2.2 Initiator protocol

The initiator issues a CFP by executing action CFP(v), specifying the requested proposal. Then, the recursive invocation to the reply handler $\mathcal{D}_h$ causes a number of proposals to be considered. $\mathcal{D}_h$ is meant to receive two (initially equal) terms as arguments; one is used to bound the current proposal with its result, the other is left fresh to later iterate a similar handler. In particular, each time the initiator asks for a new proposal by executing the getProp action, waiting for its completion. If the finished completion is perceived, no more proposals will be processed for this CFP. Instead, if a proposal is perceived by completion newProp, the initiator executes two processes in parallel: on the one hand the current proposal is served, on the other hand by using ! stop.?ok+$\mathcal{D}_h(w, w)$ either the reception of proposals is aborted or the handler for a new proposal is created. We use this schema to stress the idea that the initiator may even decide not to respond to a proposal without invalidating the protocol, since—as we see later—the coordination artifact can automatically reject a proposal. In this case, the initiator instead decides to handle a proposal, it can either accept or refuse the proposal (executing actions acceptProp or refuseProp), and if it accepts the proposal it will receive either a failure or the result (perceiving failure or result(v)).

5.2.3 Mentalistic semantics

The execution of actions by participants and initiator is not only driven by the requirement to follow the operating instructions—which would hardly be sufficient to make the agent meaningfully carry on the protocol—but again it also leverages preconditions and effects. Notice that in our approach, since intentions are driven by operating instructions and preconditions/effects relate interactions only to agent beliefs, then such preconditions and effects are sensibly simpler and cleaner than, for example, the semantics of FIPA performatives involved in the Contract-Net protocol (FIPA, 2000). More specifically, an action is attached a precondition only if it involves the communication of some information from the agent to the coordination artifact, in which case it should be properly connected to the agent beliefs. For the sake of simplicity, we denote by B done(a) the fact that the agent believes action a has already been executed, and by B feasible(a) that it might be able to execute action a. In our case of the Contract-Net protocol, a participant: (i) refuses to make a proposal if it believes it cannot execute the action (B¬feasible(v)), (ii) makes a proposal if it believes it can execute its action (B feasible(v)), (iii) produces a failure message if it no longer believes the action can be executed (¬B feasible(v)), and (iv) it provides a result only if it believes the resulting action has been executed (B done(v)). Notice, for example, that without this last precondition, it would have been impossible for the agent to understand from the operating instructions what to inform, which is now clear from the conjunction of preconditions, effects and operating instructions.

Analogously, effects have to be specified only when the interaction protocol reaches a point where the agent is receiving some information that should change its perception of the world—not all the times it changes its assumptions about the protocol state as, for example, in FIPA ACL. Therefore, for the Contract-Net protocol case, effects have to be specified only when the initiator perceives the positive completion to the `acceptProp`: in this case the operating instructions tell the agent to believe the result.

5.2.4 Role of the coordination artifact

Finally, it is interesting to point out the role played by the coordination artifact in this coordination scenario. As far as providing a coordination task is concerned, the coordination artifact can be understood in terms of an interactive behaviour (Viroli & Omicini, 2003), namely allowing agents to execute actions and providing the proper perceptions consistently with respect to the operating instructions of each agent. This is obtained by means of the proper coordination rules stored and realized in the artifact. Provided that such coordination rules are quite fundamental to implement an effective and efficient Contract-Net protocol scenario, one should notice that they are not included in our semantics of interaction, since the agent only perceives their effect through the operating instructions. In particular, in our framework, agents participate in a collaboration without a necessary knowledge about the identity (and even the presence) of other agents: their viewpoint over coordination is rather subjective, for the artifact has the burden of the objective part of coordination (Ricci *et al*., 2003).

This is not a limitation but rather a feature of our approach, for the artifact encapsulates and hides a significant coordination burden that would otherwise charge each involved agent. In our example, for instance, the initiator is not required to reply to each incoming proposal (and in principle, it could be even completely released from this burden): the coordination artifact can simply keep track of each such proposals and refuse them within a given timeout.

Other useful features of the coordination artifact for the Contract-Net protocol include its intrinsic decoupling properties. On the one hand, it decouples the information flow from participants and initiator, for example, the initiator could handle a large amount of participants without significant design changes: indeed, our approach scales well with the size of the collaborating group. On the other hand, the initiator is not required to initially know the identity (and even the number) of participants: the coordination artifact is in charge of making participants perceiving CFPs, allowing the initiator to perceive only meaningful proposals—for example, automatically refusing late ones, bad ones, proposals from agents with bad reputation and so on.

6 Discussion

6.1 Comparison with mentalistic semantics

In spite of the fact that this work has many relationships with existing ACL semantics, it should be noted that it is not in competition with them—or of other related researches such as those on argumentation (Parsons & McBurney, 2003)—and that the particular structure of interactions between agents and artifacts calls for a new framework. Notable differences include the facts that: (i) interactions are not ACL (speech) acts, but rather action/completion couples, (ii) agents interact with artifacts, that are non-intentional entities, and (iii) the protocol to be used is reified in operating instructions, and agents can reason about them.

Indeed, we believe that our approach based on artifacts and operating instructions can be used to restructure some agent collaborations, resulting in new architectural solutions (see the examples in previous section) enjoying some advantages over speech-act based interactions. This is true in particular for ACLs with a purely mentalistic semantics, such as those of FIPA ACL (FIPA, 2000) and KQML (Labrou & Finin, 1997), which were initially evaluated for a standardization but are

now criticized for their inadequate support to interaction protocols and compliance verification (Pitt & Mamdani, 1999; Singh, 1998).

We take FIPA ACL as a case study. Feasibility preconditions (FPs) and rational effects (REs) are associated to each communicative act based on its performative, and are expressed in terms of beliefs and intentions (and uncertain beliefs). Since they are the only means by which an agent can rationally carry on an interaction protocol, their specification can easily become complicated and hard to apply in practise—especially as far as complex protocols are concerned. To show the advantages of our approach, we compare our example in Section 5.1 with the corresponding FIPA ACL solution. We consider the performative *query-if* executed by client agent c towards server agent s, and the communicative act $a=\langle c,\ query\text{-}if(s,\ \varphi)\rangle$. For c, the only RE is $I_c done(b)$ where $b=\langle s,\ inform\text{-}if(c,\ \varphi)\rangle$, that is, c should intend s to perform the reply. In general, this is not sufficient to motivate c to execute a: indirectly, c should also intend the REs to b, which are $Bif_c\varphi$. With a similar result but a more coherent deliberation, we observe that by exploiting the coordination artifact, instead, the agent c executes the `ask`(φ) action since—due to the structure of operating instructions $\mathcal{D}_c$—it entails either $B_c\varphi$ or $B_c\neg\varphi$, i.e. $Bif_c\varphi$. In some sense, the motivational part is encoded in the protocol of operating instructions, which resembles an actual plan for the agent.

Moreover, in FIPA ACL there are two kinds of FPs that apply to a: (i) $B_c Agent(s,b)$, c must believe that s is the agent that will execute b, and (ii) $\neg B_c I_s done(b)$, c does not believe that s is already willing to execute b. The former should entail that s may be willing to execute b (Bergenti *et al.*, 2003); again, by virtue of the coordination artifact, the same property is achieved in a more coherent way. First, c is not aware of any specific agent providing the reply. Second, the intention of any server s to provide replies is not to be checked, since it is ensured by the fact that s accepted to play the server role in the collaboration. On the other hand, the latter FP is used to avoid c executing action a indefinitely: we enforce this condition since the operating instructions make c wait for a reply before issuing the next request, by the time c will already believe φ or $\neg\varphi$.

6.2 *Comparison with other semantic approaches*

In practise, one of the main advantages of our approach is that it is intrinsically protocol-driven. Instead of relying on a fixed set of primitive actions composed so that the resulting semantics fit a given interaction goal—for example, composing a *request* and an *inform-if* to obtain a *query-if*—the approach in nature is quite different. Agent interactions are handled by a specific coordination artifact—either derived as the result of the MAS design, or dynamically constructed by cooperating agents. Such a coordination artifact specifies operating instructions that the agent can become aware of by inspection, and that can be used to more clearly understand the relation-effect cause of given actions. To this extent, our approach is similar to that by Pitt & Mamdani (1999), where incoming messages are connected to the corresponding admissible replies: however, our operating instructions are more flexible as protocols are expressed using powerful process-algebraic constructs. Moreover, dealing with coordination artifacts ensures those decoupling properties that simplify the need to connect interactions and mental states.

Currently, one of the most promising approach to ACL semantics is based on social commitments (Verdicchio & Colombetti, 2003; Pitt *et al.*, 2001), where communicative acts are understood as a manipulation of agent commitments relative to other agents. Therefore, since communicative acts are observable, an agent's behaviour can be checked for compliance, supporting interceptions of violations. Indeed, the framework of coordination artifacts is capable of addressing similar issues. An agent interacting with a coordination artifact can be considered as committed, relative to the artifact, to follow its operating instructions. Commitments are also dynamically adapted/refined, since executing a new action means to commit to follow the corresponding continuation of the instructions. Executing actions not allowed by the coordination artifact can be intercepted, tracked and pursued as violations of legality (Viroli & Ricci, 2004).

Unlike purely mentalistic approaches, in our semantics we do not mean preconditions and effects to be normative, i.e., to be subject to compliance verification. Rather, they are meant to be used by an agent so as to exploit the coordination service provided by the artifact in a meaningful way with respect to its own goals. Whereas adhering to them allows an agent to correctly interpret perceptions and to provide a significant service to others through actions, failing to do so never impacts the safety of the whole coordination task. Interactions are to be judged as correct or wrong based solely on their admissibility according to the operating instructions.

7 Conclusion and future works

In this paper, we introduce a semantic framework for the interaction of agents with coordination artifacts, based on the notion of operating instructions. Our framework combines a process-algebraic approach to specify interaction protocols—enforced by means of operating instructions—and modal logics for beliefs to connect interactions and the agent mental state. Even though our semantics is not meant to compete with existing ACL semantic approaches—for it applies to a mediated-interaction scenario rather than a direct, ACL-based scenario—it tackles in an effective way most of the basic issues they raise, including protocols and social commitments support.

Future work along this line will be devoted to the extension of operating instructions with the notion of timeout violations and guarantees, and to better evaluate the applicability of our framework to complex negotiations and existing agent architectures supporting intelligence.

References

Agre, PE, 1995, Computational research on interaction and agency. *Artificial Intelligence* **72**(1–2), 1–52. Special volume on computational research on interaction and agency, part 1.

Bergenti, F, 2003, A discussion of two major benefits of using agents in software development. In Petta, P, Tolksdorf, R and Zambonelli, F (eds) *Engineering Societies in the Agents World III (Lecture Notes in Computer Science, 2577)*. Berlin: Springer, pp. 1–12.

Bergenti, F, Botelho, LM, Rimassa, G and Somacher, M, 2003, A FIPA compliant goal delegation protocol. In Huget, M-P (ed.) Communications in Multiagent Systems. *Agent Communication Languages and Conversation Policies (Lecture Notes in Artificial Intelligence, 2650)*. Berlin: Springer, pp. 223–238.

Boella, G and van der Torre, LW, 2003, Attributing mental attitudes to normative systems. In Rosenschein, JS, Wooldridge, MJ, Sandholm, T and Yokoo, M (eds) *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*. New York: ACM Press, pp. 942–943. Poster.

Bonabeau, E, Dorigo, M and Theraulaz, G, 1999, *Swarm Intelligence: From Natural to Artificial Systems*. Oxford: Oxford University Press.

Brooks, RA, 1991, Intelligence without representation. *Artificial Intelligence* **47**(1–3), 139–159.

Ciancarini, P, Omicini, A and Zambonelli, F, 2000, Multiagent system engineering: the coordination viewpoint. In Jennings, NR and Lespérance, Y (eds) *Intelligent Agents VI. Agent Theories, Architectures, and Languages (Lecture Notes in Artificial Intelligence, 1757)*. Berlin: Springer, pp. 250–259.

Dennett, D, 1987, *The Intentional Stance*. Cambridge, MA: Bradford Books/MIT Press.

Denti, E, Omicini, A and Ricci, A, 2002, Coordination tools for MAS development and deployment. *Applied Artificial Intelligence* **16**(9/10), 721–752. Special Issue: Engineering Agent Systems—Best of 'From Agent Theory to Agent Implementation (AT2AI-3)'.

Fenster, M, Kraus, S and Rosenschein, JS, 1997, Coordination without communication: Experimental validation of focal point techniques. In Huhns, MN and Singh, MP (eds) *Readings in Agents*. San Francisco, CA: Morgan Kaufmann, pp. 380–386.

Ferber, J and Müller, J-P, 1996, Influences and reaction: a model of situated multiagent systems. In *Proceedings of the 2nd International Conference on Multi Agent Systems (ICMAS'96)*. MIT Press, pp. 72–79.

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

Gelernter, D, 1985, Generative communication in Linda. *ACM Transactions on Programming Languages and Systems* **7**(1), 80–112.

Glabbeek, RV, 2001, The linear time—branching time spectrum I. The semantics of concrete, sequential processes. In Bergstra, JA, Ponse, A and Smolka, SA (eds) *Handbook of Process Algebra*. Amsterdam: North-Holland, ch. 1, pp. 3–100.

Hindriks, KV, de Boer, FS, van der Hoek, W and Meyer, J-JC, 1999, Agent programming in 3APL. *Autonomous Agents and Multi-Agent Systems* **2**(4), 357–401.

Kirsh, D, 1995, The intelligent use of space. *Artificial Intelligence* **2**(72), 31–68.

Kirsh, D, 1996, Adapting the environment instead of oneself. *Adaptive Behaviour* **4**(3–4), 415–452.

Labrou, Y and Finin, T, 1997, Semantics and conversations for an agent communication language. In Huhns MN and Singh, MP (eds) *Readings in Agents*. San Francisco, CA: Morgan Kaufmann, pp. 235–242.

Malone, T and Crowston, K, 1994, The interdisciplinary study of coordination. *ACM Computing Surveys* **26**(1), 87–119.

Mamei, M, Zambonelli, F and Leonardi, L, 2003, Co-fields: towards a unifying approach to the engineering of swarm intelligent systems. In Petta, P, Tolksdorf R and Zambonelli, F (eds) *Engineering Societies in the Agents World III (Lecture Notes in Computer Science, 2577)*. Berlin: Springer, pp. 68–81.

Milner, R, 1989, *Communication and Concurrency*. Englewood Cliffs, NJ: Prentice Hall.

Nardi, BA, 1996, *Context and Consciousness: Activity Theory and Human-Computer Interaction*. MIT Press.

Noriega, P and Sierra, C, 2002, Electronic institutions: future trends and challenges. In Klusch, M, Ossowski S and Shehory, O (eds) *Cooperative Information Agents VI (Lecture Notes in Computer Science, 2446)*. Berlin: Springer, pp. 14–17.

Odell, J, Van Dyke Parunak, H, Fleischer, M and Brueckner, S, 2003, Modeling agents and their environment. In Giunchiglia, F, Odell, J and Weiß, G (eds) *Agent-Oriented Software Engineering III (Lecture Notes in Computer Science, 2585)*. Berlin: Springer, pp. 16–31.

Omicini, A, 2002, Towards a notion of agent coordination context. In Marinescu, DC and Lee, C (eds) *Process Coordination and Ubiquitous Computing*. CRC Press, ch. 12, pp. 187–200.

Omicini, A and Denti, E, 2001, From tuple spaces to tuple centres. *Science of Computer Programming* **41**(3), 277–294.

Omicini, A and Papadopoulos, GA, 2001, Why coordination models and languages in AI? *Applied Artificial Intelligence* **15**(1), 1–10. Special Issue: Coordination Models and Languages in AI.

Omicini, A and Zambonelli, F, 1999, Coordination for Internet application development. *Autonomous Agents and Multi- Agent Systems* **2**(3), 251–269.

Omicini, A, Ossowski, S and Ricci, A, 2004a, Coordination infrastructures in the engineering of multiagent systems. In Bergenti, F, Gleizes, M-P and Zambonelli, F (eds) *Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook*. Kluwer Academic Publishers, ch. 14, pp. 273–296.

Omicini, A, Ricci, A, Viroli, M, Castelfranchi, C and Tummolini, L, 2004b, Coordination artifacts: environment-based coordination for intelligent agents. In Jennings, NR, Sierra, C, Sonenberg, L and Tambe, M (eds) *Proceedings of the 3rd international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, vol. 1. New York: ACM Press, pp. 286–293.

Omicini, A, Ricci, A, Viroli, M, Cioffi, M and Rimassa, G, 2004c, Multi-agent infrastructures for objective and subjective coordination. *Applied Artificial Intelligence* **18**(9/10), 815–831. Special Issue: Best papers from EUMAS 2003: The 1st European Workshop on Multi-agent Systems.

Parsons, S and McBurney, P, 2003, Argumentation-based communication between agents. In Huget, M-P (ed.), *Communication in Multiagent Systems, Agent Communication Languages and Conversation Policies (Lecture Notes in Computer Science, 2650)*. Berlin: Springer, pp. 164–178.

Parsons, S, McBurney, P and Wooldridge, MJ, 2004, The mechanics of some formal inter-agent dialogues. In Dignum, F (ed.) *Advances in Agent Communication, International Workshop on Agent Communication Languages (ACL 2003) (Lecture Notes in Computer Science, 2922)*, Berlin: Springer, pp. 329–348.

Parunak, HVD, Brueckner, S, Fleischer, M and Odell, J, 2003, A preliminary taxonomy of multi-agent interactions. In Rosenschein, JS, Sandholm, T, Wooldridge MJ and Yokoo, M (eds) *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*. New York: ACM Press, pp. 1090–1091.

Parunak, VD, 1997, Go To The Ant: engineering principles from natural agent systems. *Annals of Operations Research* **75**, 69–101.

Paurobally, S, Cunningham, J and Jennings, NR, 2005, A formal framework for agent interaction semantics. In Dignum, F, Dignum, V, Koenig, S, Kraus, S, Singh MP and Wooldridge, M (eds) *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005)*, 25–29 July 2005, Utrecht, The Netherlands. New York: ACM Press, pp. 91–98.

Pitt, J and Mamdani, A, 1999, A protocol-based semantics for an agent communication language. In Dean, T (ed.) *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*. San Francisco, CA: Morgan Kaufmann, pp. 486–491.

Pitt, J, Kamara, L and Artikis, A, 2001, Interaction patterns and observable commitments in a multi-agent trading scenario. In André, E, Sen, S, Frasson, C and Müller, JP (eds) *Proceedings of the 5th International Conference on Autonomous Agents (AGENTS'01)*. New York: ACM Press, pp. 481–488.

Ricci, A, Omicini, A and Denti, E, 2003, Activity theory as a framework for MAS coordination. In Petta, P. Tolksdorf, R and Zambonelli, F (eds) *Engineering Societies in the Agents World III (Lecture Notes in Computer Science, 2577)*. Berlin: Springer, pp. 96–110.

Ricci, A, Viroli, M and Omicini, A, 2004, Agent coordination context: from theory to practice. In Trappl, R (ed.) *Cybernetics and Systems 2004*, vol. 2, Austrian Society for Cybernetic Studies, Vienna, Austria, pp. 618–623. 17th European Meeting on Cybernetics and Systems Research (EMCSR 2004), Vienna, Austria, 13–16 April 2004.

Ricci, A, Viroli, M and Omicini, A, 2005, Programming MAS with artifacts. In Bordini, RP, Dastani, M, Dix J and El Fallah Seghrouchni, A (eds) *Proceedings of the 3rd International Workshop 'Programming Multi-Agent Systems' (PROMAS 2005)*, AAMAS 2005, Utrecht, The Netherlands, pp. 163–178.

Schmidt, K and Bannon, L, 1992, Taking CSCW seriously: supporting articulation work. *Computer Supported Cooperative Work (CSCW)* **1**(1–2), 7–40.

Schmidt, K and Simone, C, 2000, Mind the gap! towards a unified view of CSCW. In *Proceedings of the 4th International Conference on the Design of Cooperative Systems (COOP 2000)*, INRIA.

Schmidt, K and Wagner, I, 2004, Ordering systems: coordinative practices and artifacts in architectural design and planning. *Computer Supported Cooperative Work (CSCW)* **13**(5–6), 349–408.

Searle, J, 1969, *Speech Acts: An Essay in the Philosophy of Language*. Cambridge: Cambridge University Press.

Singh, MP, 1998, Agent communication languages: rethinking the principles. *IEEE Computer* **31**(12), 40–47.

Smith, RG, 1980, The Contract Net Protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers* **29**, 1104–1113.

Susi, T and Ziemke, T, 2001, Social cognition, artefacts, and stigmergy: a comparative analysis of theoretical frameworks for the understanding of artefact-mediated collaborative activity. *Cognitive Systems Research* **2**(4), 273–290.

Sycara, K, 2001, Multi-agent infrastructure, agent discovery, middle agents for Web services and interoperation. In Carbonell, JG and Siekmann, J (eds) *Multi-agents Systems and Applications*. Berlin: Springer, pp. 17–49.

van Eijk, RM, de Boer, FS, van der Hoek, W and Meyer, J-JC, 2000, Operational semantics for agent communication languages. In Dignum, F and Greaves, M (eds) *Issues in Agent Communication (Lecture Notes in Computer Science, 1916)*. Berlin: Springer, pp. 80–95.

van Linder, B, van der Hoek, W and Meyer, J-JC, 1996, How to motivate your agents. In Wooldridge, MJ, Müller, J-P and Tambe, M (eds) *Intelligent Agents II (Lecture Notes in Artificial Intelligence, 1037)*. Berlin: Springer, pp. 17–32.

Vasconcelos, WW, 2004, Logic-based electronic institutions. In Leite, JA, Omicini, A, Sterling, L and Torroni, P (eds) *Declarative Agent Languages and Technologies (Lecture Notes in Artificial Intelligence, 2990)*. Berlin: Springer, pp. 223–242. 1st International Workshop (DALT 2003), Melbourne, Australia, 15 luglio 2003. Revised Selected and Invited Papers.

Verdicchio, M and Colombetti, M, 2003, A logical model of social commitment for agent communication. In Rosenschein, JS, Sandholm, T, Wooldridge, M.J and Yokoo, M (eds) *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*. New York: ACM Press, pp. 528–535.

Viroli, M and Omicini, A, 2003, Coordination as a service: ontological and formal foundation. In *FOCLASA 2002—Foundations of Coordination Languages and Software Architecture (Electronic Notes in Theoretical Computer Science, 68(3))*. Elsevier Science B. V.

Viroli, M and Omicini, A, 2005, Process-algebraic approaches for multi-agent systems: an overview. *Applicable Algebra in Engineering, Communication and Computing* **16**(2–3), 69–75. Special Issue: Process Algebras and Multi-Agent Systems.

Viroli, M and Ricci, A, 2004, Instructions-based semantics of agent mediated interaction. In Jennings, NR, Sierra, C, Sonenberg, L and Tambe, M (eds) *Proceedings of the 3rd international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, vol. 1. New York: ACM Press, pp. 102–110.

Weyns, D, Parunak, HVD and Michel, F. (eds.) 2005, *Environments for Multi-Agent Systems (Lecture Notes in Artificial Intelligence, 3374)*, 1st International Workshop (E4MAS 2004), New York, NY, USA, July 2004. Berlin: Springer. Revised Selected Papers.

Wooldridge, MJ and Jennings, NR, 1995, Intelligent agents: theory and practice. *The Knowledge Engineering Review 10*(2), 115–152.