Using OWL 2 DL for Expressing ACL Content and Semantics

Nicoletta Fornara[1,⋆], Daniel Okouya[1,⋆⋆], and Marco Colombetti[2,⋆⋆⋆]

[1] Università della Svizzera Italiana,
via G. Buffi 13, 6900 Lugano, Switzerland
{nicoletta.fornara,daniel.okouya}@usi.ch
[2] Politecnico di Milano, Piazza Leonardo Da Vinci 32, Milano, Italy
marco.colombetti@polimi.it

Abstract. The design and implementation of open interaction systems is widely recognized to be a crucial issue in the development of innovative applications on the Internet. In this paper we pursue the goal of enhancing interoperability and openness in open interaction systems by systematic use of web standards. We propose a way of using the semantic web language OWL 2 DL to represent both the content of an ACL message, whose structure is compatible with FIPA-ACL, and the meaning of the whole message, adopting a commitment-based semantics, in such a way that OWL reasoning on message meaning is made possible. To this purpose we specify a number of ACL conventions regarding the domain independent components of the content language and the semantics of messages; we describe a set of supporting OWL ontologies, and exemplify our proposal with the analysis of a commissive message: a promise to perform a certain action within a given deadline if certain conditions hold. We then describe a demonstrative prototype of a system where those conventions are concretely implemented that is based on Web Service technologies (WSDL, SOAP, and HTTP for message transport).

1 Introduction

The design and implementation of *open interaction systems* (OISs) is widely recognized to be a critical issue in the development of innovative applications on the Internet, like e-commerce systems, e-marketplaces, and applications for the management of virtual enterprises. Basically, we can conceive an OIS as a distributed system which different computer programs (regarded as autonomous agents) may freely enter or leave, with the purpose of achieving their individual goals by interacting with other agents according to shared rules. In an OIS,

⋆ Supported by the Hasler Foundation project Nr. 11115-KG and by the SER project Nr. C08.0114 within the COST Action IC0801 Agreement Technologies.
⋆⋆ Supported by the Swiss National Science Foundation project Nr. 200020_134790.
⋆⋆⋆ This research has been carried out as part of the activities of the author at Università della Svizzera italiana, Switzerland.

rules cover several aspects of the interaction, and in particular the *communication conventions* regulating the exchange of messages and their semantics, and the *artificial institutions* [7,5,3] specifying the roles, powers, and norms that constitute the social environment within which the interactions are carried out.

It has long been recognized that the set of conventions regulating the exchange of messages is particularly important. Since the 1990s this fact has led to a flourishing of studies on Agent Communication Languages (ACLs), and there are signs that this area of research may take front stage once again [1]. Indeed, in an OIS the *communication conventions* have to cover several layers, all of which are crucial to the successful exchange of messages between agents; moving bottom-up, such layers concern:

1. *connectivity*: the ability of agents to exchange messages as binary data with other agents, typically running on different and possibly heterogeneous platforms;
2. *data format*: the concrete serialization of the binary data that represent messages;
3. *message structure*: the application-independent component of messages (i.e., the abstract syntax);
4. *terminology*: the application-independent and the application-dependent terms contained in a message;
5. *semantics*: the contribution of the message structure and of the application-independent and application-dependent terminology to the meaning of the message;
6. *conversation management*: the rules concerning how a conversation should be initiated, carried out, and terminated.

In Section 2 and 3 we mainly focus on the message structure, terminology, and semantics, whereas in Section 4 we propose an approach for the connectivity and data format based on web services technologies. As far as possible, we follow the spirit of FIPA-ACL specifications[1]. In particular, we take from such specifications that:

1. a message is articulated in a set of sections, each of which is introduced by a specific *parameter* (like sender, receiver, etc.);
2. every message realizes a *communicative act*, whose type (like inform, request, etc.) is explicitly represented in the message;
3. the content of the message is expressed in a suitable Content Language (CL).

The main goal of this paper is to propose an approach to agent communication that maximizes *interoperability*, at both the syntactic and the semantic level, as a means to achieve true openness. For this reason we conform, as far as possible, to the suitable W3C recommendations, and in particular to those concerning the Semantic Web and the Service Oriented Architecture (SOA). More precisely, we show how agent communication can be based on OWL ontologies, used to specify the semantics of the application-independent component of the content language,

[1] `http://www.fipa.org/repository/aclspecs.html`

the application-dependent terminology, and the semantics of the whole message (inclusive of the illocutionary force of the message).

In particular in Section 2 we define a minimal set of *conventions* concerning the message structure, the content language, and the semantics of performatives, that would guarantee agent interoperability if adopted as interaction standards. In Section 3 we present the semantics of commissive communicative acts. In Section 4 we describe a demonstrative prototype which implements the various layers of communication conventions for an OIS. Finally, in Section 5 we discuss the contribution of this paper with respect to other relevant proposals available in the literature and draw some conclusions.

2 Message Structure and Content Language

An OIS is a system where agents can interact to achieve their individual goals by coordinating and negotiating their activities. Contrary to more traditional distributed systems, the distinctive feature of an OIS is that it allows external agents to enter the system, participate in the activities, and then leave the system at will. This brings to the foreground the problem of *interoperability*, because for the interaction to be successful all agents have to comply with a set of *shared conventions*, which will have to be taken as *standards* by the designers of the agents. In this section we introduce a set of possible conventions, covering the structure of the messages (i.e. their *abstract syntax*), the syntax and semantics of the application-independent part of the *content language*, and the syntax and semantics of the application-dependent *terminology* used in the content expression.

Regarding the structure of messages and the content language, when possible we will try to roughly follow FIPA-ACL specifications[2]. Regarding the semantics of the message, as discussed in [12,4], we depart significantly from the semantics of FIPA-ACL; this because, among other problems, FIPA-ACL is unable to account for the normative consequences of message exchanges, which are essential to a satisfactory treatment of agent interactions. More specifically, we adopt commitment-based semantics for communicative acts and institutional semantics for declarations. In the past we have expressed the semantics of communicative acts using the Event Calculus [5]; however, given the current importance of exploiting industry-level technology, in this paper we consider the application of Semantic Web technologies [9]. For the moment, in the next section we propose a semantics of commissive communicative acts, using an extension of the OWL ontology of obligations presented in [3]. We plan to further extend this ontology in the future, to express the semantics of *directive* and *assertive* communicative acts and of *declarations*.

2.1 Message Structure

In this section we propose the message structure of our ACL. Like in FIPA-ACL, a message contains an acl-envelope, used to correctly route the message

[2] http://www.fipa.org/repository/aclspecs.html

to the final destination and comprised of a number of attributes, including to, from, length, encoding, and others. The message also contains an acl-payload, characterized by the following components:

– performative: a symbol denoting the type of the communicative act, that in our ACL may be: promise, inform, request, agree, refuse, cancel, query-ref, query-if, declare; further types may be added and specified according to need;
– sender: the identifier of the agent that sends the message;
– receiver: the identifier of the recipients of the message;
– content: a complex expression that can represent: a state of affair if the performative is assertive (like inform); an action with a deadline if the performative is commissive (like promise) or directive (like request); and an institutional action if the performative is declare;
– reply-by: the time within which it is possible to answer to a request;
– msg-id: the unique identifier of the message, generated by the sender of the message by using its name as namespace followed by a progressive number, for example John:001.

The following parameters are used to describe the content of the message:

– language: denotes the name of the formal language in which the value of the content parameter is expressed;
– ontology: denotes the name of the ontology that specifies the concepts used in the content expression.

FIPA-ACL defines other parameters (like reply-to, protocol, conversation-id, reply-with, in-reply-to) that can be used for conversation management but that are nor relevant for the goals of this paper.

2.2 Content Language

Content expressions are sentences belonging to a *content language* (CL). This consists of: (i) *application-independent* terms, like for example those used to describe the logical structure of *actions* and *events*; and (ii) *application-dependent* terms used in relation to specific domains, for example to describe actions of payment, delivery, bidding, and so on. All such terms must merge to form a unique formal language.

As we have already remarked, in the attempt to maximize interoperability we conform as far as possible to widely adopted standards, like those recommended by W3C. A similar attempt has been made by FIPA, which proposes FIPA-RDF[3], an application-independent content language whose application-independent classes and properties are defined using RDF Schema. However RDF-based solutions, while very flexible and expressive, are limited in the complexity of the inferences they can support. This is due to the limited expressive power of RDF Schema, which for example does not allow one to specify the

[3] http://www.fipa.org/specs/fipa00011/

disjointness of classes, cardinality restrictions of properties, and formal characteristics of properties like transitivity.

We thus opted for OWL (more precisely, OWL 2 DL[4] that is briefly described in Appendix A) as the formal language for the specification of both the application-independent and the application-dependent terms of our content language. The reasons of our choice are that: (i) OWL, recommended by W3C, is by now a fairly well-known standard; (ii) OWL is a very expressive, but still decidable, logical language, which licenses powerful reasoning procedures; (iii) many open source tools are already available for editing OWL ontologies, carrying out reasoning, and realizing applications like Development Environments and APIs[5]; (iv) there are many available OWL ontologies, which can be easily integrated in our proposal thanks to the fact that two ontologies can be usually merged by simply taking the union of their axioms; (v) as discussed in Section 3, OWL 2 DL can be used to specify both the semantics of the content of messages and the semantics of the whole messages (inclusive for example, of its performative).

In this section we describe the application-independent concepts that can be used in the content of ACL messages: the specification of these concepts makes up what we call the *CL Ontology* (i.e., the Content Language Ontology). In turn, this ontology imports the OWL *Time Ontology*[6] that defines classes like $\mathsf{Instant} \sqsubseteq \mathsf{TemporalEntity}$, $\mathsf{Interval} \sqsubseteq \mathsf{TemporalEntity}$, $\mathsf{ProperInterval} \sqsubseteq \mathsf{Interval}$, Discla(ProperInterval,Instant), and properties like hasBeginning: TemporalEntity $\rightarrow$ Instant, hasEnd: TemporalEntity $\rightarrow$ Instant (for connecting a temporal entity to its start and end instant of time) and before: TemporalEntity $\rightarrow$ TemporalEnty (for expressing that a temporal entity is before another temporal entity).

The Content of Assertive Messages. The content of assertive messages, like for example an *inform* message, is the description of a state of affairs, that is, a *proposition*. For the moment, in our content language we consider only a restricted type of propositions, namely those that can be expressed as a set of positive or negative OWL assertions (i.e., the elements of an OWL ABox). Such assertions in the set are implicitly in logical conjunction. For example, the proposition "John owns a red car" must first be analysed as the conjunction of four atomic propositions, and then represented as the following set of OWL assertions:

Agent(john-001), Car(car-001), owns(john-001,car-001), Red(car-001),

where: the Agent class is defined in the *CL Ontology*; the Car and Red classes, and the property owns: Agent $\rightarrow$ Car, are defined in what we call the *Domain Ontology*. As we shall see, similar OWL assertions appear also as parts of the content of non-assertive messages.

The Content of Directive and Commissive Messages. The content of a *directive* or *commissive* message (like a request and a promise, respectively) includes the description of an *action*, which ought to be carried out in the future.

[4] http://www.w3.org/2007/OWL/wiki/OWL_Working_Group
[5] http://www.w3.org/2007/OWL/wiki/Implementations
[6] http://www.w3.org/TR/owl-time/

The semantics behind the exchange of a promise (as we shall see in Section 3) is that the sender is in charge of performing, within a given deadline, an instance of the action described in the message.

In the *CL Ontology* we define the Action class for representing the description of actions. The Action class is the domain of the following properties that can be used to characterize the action that has to be performed:

– The property hasActor, used to represent the agent responsible for the execution of the action; the range of this property is the application-independent class Agent.
– The property hasDeadline, that represents the instant of time within which the action has to be performed. It is empty if the deadline for the performance of the action is not known when the message is sent. In this case the deadline depends on the time when some other event happens; for example, in the promise to pay a given book within one week from delivery, the deadline depends on the date of delivery. The range of this property is the Instant class.
– The property hasDuration, which can be used to specify the interval of time within which the action has to be performed, starting from the instant of time when the event described in the condition component of the message will happen. Its range is the DurationDescription class.
– Other properties that can be defined in the *Domain Ontology* for describing specific types of actions. For instance, in an application where agents talk about payment and delivery, a *Commercial Ontology* will define the class Pay $\sqsubseteq$ Action and Deliver $\sqsubseteq$ Action (for representing the action of paying and delivering), with properties hasAmount and hasRecipient, hasObject, and the class Item for representing the items exchanged in the commercial transaction.

In many commissive and directive messages, the action is specified as a *conditional action*, that is, as an action that has to be executed if certain conditions obtain. The ConditionalAction class is used for representing an *action* that ought to be executed, on condition that a given *condition event* occurs; this is very frequent in both *request* and *promise* communicative acts. The content and the condition part are inserted in the message by introducing the property hasContentPart, whose range is the Action class, and the property hasConditionPart, whose range is the Event class.

To this purpose we introduce in the *CL Ontology* the application-independent Event class that generalizes the class Action $\sqsubseteq$ Event (in fact an action is regarded as an event with an actor). The Event class has some subclasses, and in particular: the class TimedEvent $\sqsubseteq$ Event, used to represent those events that are connected through the property atTime: TimedEvent $\rightarrow$ Instant to exactly one instant of time, as specified by the following axiom: TimedEvent $\sqsubseteq$ =1atTime.Instant; and the class TimeEvent $\sqsubseteq$ TimedEvent, used to specify as condition event the elapsing of a given instant of time.

Finally, we require that the content of directive or commissive messages are a set of OWL assertions specifying *exactly one* individual belonging to the ConditionalAction class.

Example: The Promise Communicative Act. To exemplify the proposed message structure and content language we formalize a type of commissive act, the *promise*. We use as an example the following promise that can be useful in an electronic commerce application: "agent John promises to agent Mary that John will pay 5 euro to Mary within 2 days from the delivery of book1". The message with the content expressed in the *Commercial Ontology* which imports the *CL Ontology*, which in turn imports the *Time Ontology* is:

```
(promise
 :sender John          :receiver Mary
 :language OWL-2-DL    :ontology Commercial Ontology
 :msg-id John:001
 :content
    ConditionalAction(condAct),
     hasActionPart(condAct, promisedAction),
       Pay(promisedAction), hasActor(promisedAction, John),
       hasAmount(promisedAction, 5), hasDuration(promisedAction, 2),
     hasConditionPart(condAct, cond),
       Deliver(cond), hasActor(cond, Mary),
       hasRecipient(cond, John), hasObject(cond, book1)
   )
```

3 Message Semantics

Basically, the meaning of a message derives from a combination of the message's performative and content expression. Different performatives combine with the content expression in different ways; this approach is compatible with the idea, which is fundamental in Speech Act Theory, that the meaning of a message depends on its illocutionary force (denoted by the performative) and propositional content (represented by the content expression) [11].

One of our assumptions is that the content expression of a message is a set of OWL axioms, which represent the *propositional content* of the speech act performed by sending the message. More precisely, such propositional content coincides with the meaning that the set of OWL axioms have under standard OWL 2 DL semantics [9] in the context of the *Domain Ontology* referred with the ontology parameter of the message. Such a propositional content is suitably transformed to obtain the *meaning of the whole message*, in a way that depends on the message *performative*. What remains to be decided is how the performative transforms the propositional content to produce a representation of the meaning of the whole message and how to represent this meaning.

The semantics of the whole message that we adopt in this paper is inspired by the commitment-based semantics that we firstly presented in [4], enriched with the semantics of declarations expressed via institutional concepts as presented in [5]. In this work we only formalize the semantics of the promise communicative act; we plan to cover other types of communicative acts, extending the approach proposed in this section, in our future works.

There is, we believe, no single solution to the problem of choosing a representation for the message meaning: the choice depends on what the representation is going to be used for. For the same reasons expounded in the previous section, we chose to base our representation of message meaning on OWL. That is, we propose to express the semantics of the whole message as a set of OWL axioms, which extend a pre-existing ontology representing the meanings of the previous messages belonging to the same conversation (this aspect is not treated in the current paper).

In the case of a promise (and, more generally, of commissive messages), an important point is the ability to represent, and reason on, the obligations brought about by making the promise. The matter here is more complex than with the representation of propositional content (i.e., of the meaning of the content expression). The reason is that a representation of obligations is already problematic in full First Order Logic (FOL), and even more so in the fragment of First Order Logic covered by OWL 2 DL. A suitable representation can be developed, however, if there is a clear specification of the reasoning tasks that the representation is intended to support. At the present stage of our research, we intend to use the representation of message meanings for *monitoring* the temporal evolution of commitments, obligations, etc., that are incurred by the agents as an effect of communication (either in a real OIS or in simulations). Therefore, to deal with the meaning of promises we designed an OWL representation of obligations that allows us to exploit standard OWL reasoning to monitor the temporal evolution of an obligation, as part of the *state of the interaction* of certain agents at any time instant.

We represent the state of an interaction (which includes the relevant events and actions that happen in the system) in a specific OWL ontology, the *State Ontology*. This ontology imports the *Domain Ontology* (introduced in the previous section) and the *Obligation Ontology*, which specifies the concepts needed to represents obligations of the interacting agents. The overall picture of the ontologies used and their dependencies is depicted in Figure 1.

Many classes and properties of the *Obligation Ontology* were formalized in [3], and some of them have been customized to the need of representing the semantics of communicative acts. More precisely the *Obligation Ontology* defines the classes and properties for the management of the obligations derived from the exchange of certain messages (like promises) and for the monitoring of their *state* (from activated to fulfilled or violated). It imports the *CL Ontology*, because this specifies some classes (like Agent, Action, Event) that are used as domain or range of properties related to obligations. The *Obligation Ontology* defines the

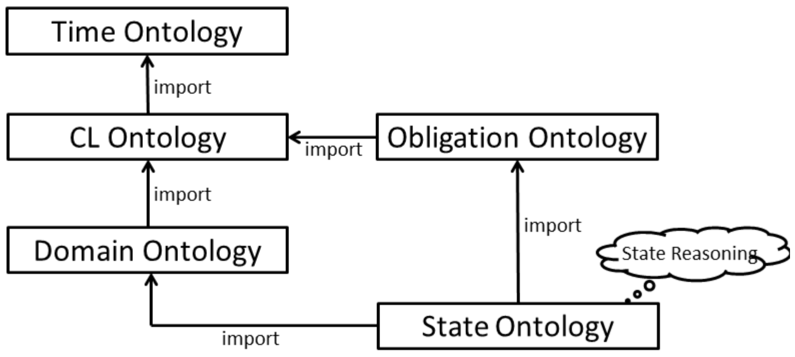

Fig. 1. Overall picture of the OWL ontologies used and their dependencies

class Obligation, which is the domain of a set of properties used to represent the *debtor* and the *creditor* of the obligation (their range being the Agent class). The *start event* of an obligation is a subclass of the Event class: when an event that belongs to the start event class of an obligation takes place, the obligation becomes activated. The *content* of the obligation is a subclass of the Action class: when an action that belongs to the content class of an obligation is executed within a given deadline the obligation becomes fulfilled.

The *State Ontology* is used to represent and monitor the temporal evolution of the interaction, therefore it defines the class $\mathsf{Elapsed} \sqsubseteq \mathsf{Instant}$, used to model that an instant of time is elapsed. As this ontology is used to represent the events and the actions that happen during an interaction, we introduce the class $\mathsf{OccurredEvent} \sqsubseteq \mathsf{TimedEvent}$ defined by the following axiom: $\mathsf{OccurredEvent} \equiv \forall\ \mathsf{atTime.Elapsed}$; this class contains the events that happened up to an elapsed instant of time. Finally we introduce the class PerformedAction defined by the following axiom: $\mathsf{PerformedAction} \equiv \mathsf{Action} \sqcap \mathsf{OccurredEvent}$; it is necessary for distinguishing between the description of an action used in the content of messages, and an action that has been executed, and is actually related to its elapsed instant of execution by means of the atTime property. The main classes and properties of the *State Ontology* (some of them are imported from the other ontologies) are represented in Figure 2.

As we already said, the *State Ontology* is used to represent at run-time the state of the interaction among agents; this is the ontology on which we run a reasoner for deducing relevant facts. As already proposed in [3], a program is in charge of representing the elapsing of time and the functionalities necessary to perform closed-world reasoning on certain classes, with the goal of monitoring the state of obligations and reacting to their fulfillment or violation. This program is also in charge of inserting into the *State Ontology* the new assertions and axioms used for representing the semantics of the messages.

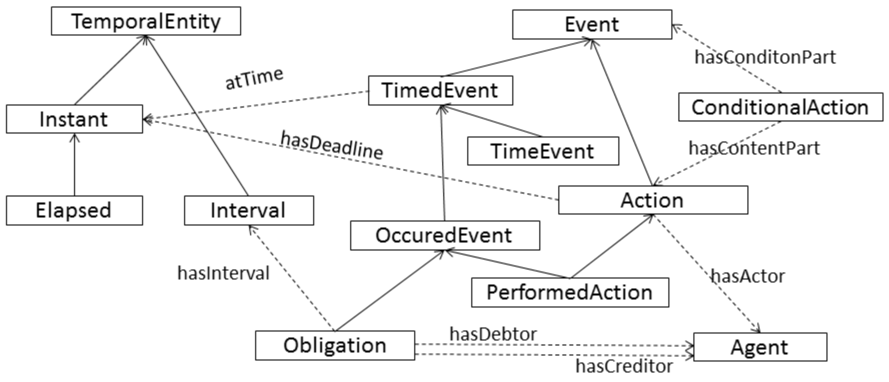

Fig. 2. Graphical representation of the main classes and properties of the *State Ontology* (some of them imported). Properties are represented with dotted lines, solid lines are used for subclasses.

3.1 Semantics of the Promise Communicative Act

A communicative act performed by exchanging a message is characterized by: a set of *preconditions* that need to be evaluated for the successful performance of the communicative act, and a set of *effects* that affect the state of the interaction if the communicative act is successfully performed and represent its meaning.

For a *promise* to be a valid communicative act it is necessary that: (i) the content belongs to the ConditionalAction class, with its action part belonging to the Action class and its condition part, if specified, belonging to the Event class; (ii) the action has to be performed in the future (i.e., if a deadline is specified it has to be in the future of the message exchange); and (iii) the sender of the message has to coincide with the actor of the promised action.

Following the commitment-based semantics for ACL proposed in [5], the performance of communicative acts has the effect of creating or changing *social commitments* [4]. The semantics of a promise communicative act is represented by a commitment of the message sender to perform the communicated action; this commitment can be viewed as a special case of social commitment, and coincides with the obligation to perform the action. Therefore in our proposal the effects of a promise are represented by creating in the state of the interaction an obligation of the message sender, directed to the receiver, to perform the action described in the content of the message within the specified deadline, if a certain communicated condition holds. Those effects are represent in the *State Ontology* by a program that is in charge of extending its *ABox* and *TBox* with a set of OWL axioms used to define the new obligation generated by the message and to monitor its state.

We now exemplify the axioms that need to be defined by using the promise message presented in the previous section; this procedure can be easily generalized and automatically applied to every promise message received by an agent. First of all we insert in the *ABox* of the *State Ontology* the following assertions:

Obligation(obl-1), Instant(instant1), atTime(obl-1,instant1)
hasDebtor(obl-1,John), hasCreditor(obl-1,Mary),
ProperInterval(interval1), hasInterval(obl-1,interval1),
Instant(instant2), hasEnd(interval1, instant2),
days(duration1, 2), hasDurationDescription(interval1,duration1),

The intuitive meaning of these assertions is as follows. A new individual obl-1 belonging to the class Obligation is created at the time instant instant1, equal to the instant of time when the message is received. The debtor of the obligation is the sender of the message and its creditor is the receiver of the message. The obligation is related to an interval of time that starts at the time instant when the obligation is activated and ends in the time instant communicated in the message as the deadline (if any). The instant of time at which this interval starts is computed as soon as an individual starts to belong to the Start-Event-1 class (defined below). If the message communicates a duration for the interval (in this case for example the duration is 2), the end instant of time is computed summing the duration to the start instant of time as soon as it becomes known.

The *TBox* of the *State Ontology* has also to be updated with the axioms necessary for representing the start event and the content of the newly created obligation. A crucial aspect of our model is that the start event and the content of an obligation are represented as OWL classes. Any instance belonging to such classes will satisfy the corresponding representations. The fact that a concrete action belongs to the start event class or to the content class can be established through an OWL 2 DL reasoner; therefore, an agent may exploit reasoning to choose which action to perform in order to fulfill a given obligation. We define the StartEvent-1 class of obligation obl-1 as a class of possible events. This class is defined as the intersection of a set of classes, defined using the properties communicated in the condition event part of the message content, and some other classes that are necessary to express the fact that the start event has to occur after the creation of the obligation:

$$\begin{aligned}\mathsf{StartEvent\text{-}1} \equiv{}& \mathsf{Deliver} \sqcap \mathsf{hasActor}{\ni}\mathsf{Mary} \sqcap \mathsf{hasRecipient}{\ni}\mathsf{John} \sqcap \\ & \mathsf{hasObject}{\ni}\mathsf{book1} \sqcap \mathsf{OccurredEvent} \sqcap (\exists \mathsf{evBefore}^{-}{\ni}\mathsf{obl\text{-}1}).\end{aligned}$$

The content of the obligation, the Content-1 class, is defined as the intersection of a set of classes that are defined using the properties communicated in the content of the message, the class of the content, and the PerformedAction class:

$$\begin{aligned}\mathsf{Content\text{-}1} \equiv{}& \mathsf{Pay} \sqcap \mathsf{hasActor}{\ni}\mathsf{John} \sqcap \mathsf{hasRecipient}{\ni}\mathsf{Mary} \sqcap \mathsf{hasAmount}{\ni}5 \sqcap \\ & \mathsf{PerfomedAction}.\end{aligned}$$

The Deadline-1 class of the obligation obl-1 contains only the time event that happens at the instant of time which terminates the interval of the obligation (which may be known or unknown when the obligation is created), and it is defined by the following axiom:

$$\mathsf{Deadline\text{-}1} \equiv \exists\ \mathsf{atTime}.(\exists\ \mathsf{hasEnd}^{-}.(\mathsf{hasInterval}^{-} \ni \mathsf{obl\text{-}1}))$$

For those obligations whose deadline event is a fixed time event, it is important to check that the start event happens before the end event. By introducing the following axiom, if the deadline time event is before or equal to the start time event, then the ontology becomes contradictory:

$$\mathsf{Deadline\text{-}1} \sqcap (\mathsf{evBefore.StartEvent\text{-}1} \sqcup \mathsf{evSameTime.StartEvent\text{-}1}) \sqsubseteq \bot$$

The $\mathsf{EndEvent\text{-}1}$ class for this obligation is equivalent to the empty set: $\mathsf{EndEvent\text{-}1} \equiv \bot$ because in this example the message does not specify an event that will terminate the obligation.

When a new obligation is created, we introduce in the *TBox* of the *Sate Ontology* also the following four axioms, necessary to deduce the state of a given obligation, that is, to deduce if the obligation belongs to the Activated, Cancelled, Fulfilled, or Violated classes. As we have already shown in [6], we need to perform some form of closed-world reasoning on the Cancelled and Fulfilled classes; this is done by a program, which computes the class $\mathsf{KCancelled} \sqsubseteq \mathsf{Cancelled}$ as the class that contains all obligations that are known to be in the Cancelled class, and the class $\mathsf{KFulfilled} \sqsubseteq \mathsf{Fulfilled}$ as the class that contains all obligations that are known to be in the Fulfilled class.

$$\{\mathsf{obl\text{-}1}\} \sqcap \neg\, \mathsf{KCancelled} \sqcap (\exists \mathsf{evBefore.}(\mathsf{StartEvent\text{-}1} \sqcap \exists \mathsf{atTime.Elapsed}) \sqcup \exists \mathsf{evSameTime.}(\mathsf{StartEvent\text{-}1} \sqcap \exists \mathsf{atTime.Elapsed})\ ) \sqsubseteq \mathsf{Activated}$$

$$\{\mathsf{obl\text{-}1}\} \sqcap \exists \mathsf{evBefore.}(\mathsf{EndEvent\text{-}1} \sqcap \exists \mathsf{atTime.Elapsed}) \sqsubseteq \mathsf{Cancelled}$$

$$\{\mathsf{obl\text{-}1}\} \sqcap \mathsf{Activated} \sqcap (\exists \mathsf{evBefore.}(\mathsf{Content\text{-}1} \sqcap \exists \mathsf{atTime.Elapsed}) \sqcup \exists \mathsf{evSameTime.}(\mathsf{Content\text{-}1} \sqcap \exists \mathsf{atTime.Elapsed})) \sqcap \exists \mathsf{evBefore.}(\mathsf{Content\text{-}1} \sqcap \exists \mathsf{evBefore.Deadline\text{-}1}) \sqsubseteq \mathsf{Fulfilled}$$

$$\{\mathsf{obl\text{-}1}\} \sqcap \mathsf{Activated} \sqcap \neg \mathsf{KFulfilled} \sqcap \exists \mathsf{evBefore.}(\mathsf{Deadline\text{-}1} \sqcap \exists\ \mathsf{atTime.Elapsed}) \sqsubseteq \mathsf{Violated}$$

4 Demonstrative Prototype

In this section we describe the demonstrative prototype that we have implemented for testing the various layers of communication conventions for OIS proposed in this paper. Due to the openness of the proposed framework we based our approach to message structure and terminology on Semantic Web Technologies that are W3C recommendations. Similarly we based our approach to connectivity and data format on principles and standards of Service Oriented Architecture (SOA); this because they address crucial low-level aspects of openness and interoperability [2], are sufficiently mature and relatively stable, and are already accepted and used by a large industrial community.

For realizing the *connectivity* layer we adopted current SOA standards: the *transport protocol* is HTTP, and the *message structure protocol* is SOAP[7](Simple Object Access Protocol). In the most popular implementation of the Service Oriented Architecture two distinct software applications play the role of *Service Requestor* and the role of *Service Provider*. This architecture can be adapted to our need of peer to peer (P2P) interactions by merging the two roles into one software application that plays both roles simultaneously.

[7] `http://www.w3.org/TR/2007/REC-soap12-part0-20070427/`

All the components of our open system possess a *listening point* to receive messages and a *talking point* to send messages. The listening point is represented as a *web-service service*[8]. As such it is exposed on the Internet via a contract defined in a WSDL (Web Service Definition Language) file. The talking point is a *web-service client*, which communicates in conformance with the previously mentioned contract. More precisely the contract defines an operation called send-message, which expects as input a message with the parameters described in Section 2, and the listening point that provides the service of delivering the message. The corresponding service contract contains a reference to the abstract syntax of the messages proposed in this paper, specified using an XML schema. Both the envelope and the payload of the messages are serialized as an XML document and they constitute the body of a SOAP message. The content part of the message is serialized in XML using the RDF/XML syntax of OWL. A crucial advantage of this approach is the provision of a human-readable communication contract that can be easily handled with the support of runtime frameworks coming along with web-service technology, such as for example Apache CXF. Hence anyone can easily generate the infrastructure to handle the transmission of a message abiding to the exposed messaging protocol and adapt it to his needs, in order to participate in the OIS. Effort will only be required for the handling of ACL communication content.

Our approach assumes that each agent has a well-defined identity, which is kept constant across different interactions. This justifies the assumption that every agent has a unique *agent identifier*. The fact that it is unique implies the existence of a service for the registration of unique names. We do not assume that such an identifier is initially known to all possible partners: this implies the need of *agent directories*. Those services plus the description of the services provided by the participants in SOA are provided by the UDDI (Universal Description Discovery and Integration) component. In our open interaction system those services (plus other services in support of the communication layer) are provided by an *intermediary* that should not be understood as an agent (in that it has no autonomy in the choice of goals or strategies), but rather as a component of the OIS whose function is to enable and support openness. The intermediary interacts with all participating agents through the communication interface and may provide the following functions in support of the communication among agents: (i) checking messages for compliance at the levels of syntax, terminology, and preconditions; (ii) forwarding messages to receivers, provided that they are correct at all these levels (otherwise an error message will be sent to the sender); (iii) keeping track of the semantics of messages and monitoring the evolution of the state of the system (for example by checking if obligations are fulfilled and applying sanctions if they are violated).

[8] This name is due to the fact that we consider "web-service" as the name of a technology, that we differentiate from the formal concept of service and client as understood in SOC/SOA; in principle this can be provided on the web using web-service technology or something else.

To run the experiment, we have implemented the intermediary in Java and two dummy participant agents in JADE (Java Agent DEvelopment Framework), but in principle a participant agent can be any software able to comply with the communication conventions and with the web-service contract. The Java components of the implemented framework use the JENA[9] API for accessing and modifying the ontologies and the Pellet OWL 2 DL reasoner[10] for reasoning on the ontologies.

5 Discussion and Conclusions

The contribution of this paper regards mainly the formalization and implementation of a proposal for using OWL 2 DL as content language of an ACL and for expressing the commitment-based semantics [4] of the whole message.

The advantages of using Semantic Web languages, like OWL 2 DL, with respect to other formal languages proposed in other approaches, like FIPA SL or FIPA KIF[11], or formalisms like the Event Calculus [5], [13] are mainly due to the fact that: (i) Semantic Web languages are international standards, and therefore it is possible to realize systems able to reuse existing ontologies (like for example the FOAF ontology), and to use numerous existing tools for programming, editing, validating, and reasoning on ontologies; (ii) OWL 2 DL is a decidable fragment of FOL, for which several reasoners are available; (iii) ontologies from different sources can be merged by taking the union of their axioms (or using ontology alignment mechanisms when the different ontologies are not immediately compatible), making it possible for an agent to participate to different interactions using one knowledge base; (iv) the same ontologies can be adopted for defining the concepts to be used in the content language and in the *State Ontology*.

The idea of using Semantic Web languages for the specification of the content of communicative acts has been proposed for the first time by FIPA in 2001. The main differences between the FIPA's proposal and the one presented in this paper are that: (i) FIPA proposed to use RDF and RDF Schema for expressing the content of messages instead of using OWL 2 DL with the limitations discussed in Section 2.2; (ii) in FIPA's application-independent content language, fipa-rdf0, there is no definition of the concept of event, which is useful to specify conditions for the performance of actions, there is no formal definition of the notion of time, and actions are not related to their deadlines; (iii) FIPA's ACL semantics is only based on agents' beliefs and intentions and, differently from what we propose in this paper, it does not take into account the deontic consequences of the exchange of messages.

In [10] a proposal for using OWL DL as content language of FIPA-ACL is presented and the limits of FIPA SL and FIPA KIF as content languages are widely discussed. This work and our proposal have in common the use OWL DL

[9] `http://incubator.apache.org/jena/`

[10] `http://clarkparsia.com/pellet/`

[11] `http://www.fipa.org/repository/cls.php3`

for expressing the content of messages but in [10] the focus is on the semantics of the content part of messages, that is, the semantics of the OWL language. This approach is exemplified by the formalization of a query-ref message whose content is a referential expression formulated as an OWL DL class expression followed by an inform message whose content is a set of assertions in OWL DL. Differently in the ACL proposed in this paper we distinguish the semantics of the propositional content from the meaning of the whole message, which depends also from the message performative. We exemplified this approach with the formalization of a distinctive communicative act, the promise. In this type of acts the content of the message describes the action that has to be performed within a given deadline, and the meaning of the whole message is to commit the sender of the message to the actual performance of the action within the given deadline.

In [14] a proposal of using the Darpa Agent Markup Language (DAML) language for expressing the content of message is presented. The main concepts formalized in the content language are similar to the one presented in this paper, except for the use of a question part and a result part for queries. The main problem of the proposal of using DAML as content language is that it is not a current standard for representing knowledge in the Semantic Web.

Regarding our proposal of using Web Service technologies like WSDL, SOAP, and HTTP for message transport, our approach is similar to the one proposed in [8], where messages are sent between JADE platforms using SOAP over HTTP; in our proposal, however, the content of SOAP messages is described with an XML Schema instead of using JADE custom binary data format, a solution that better supports openness.

In this work we formalized the semantics of the whole message only for the promise communicative act; we plan to formalize the semantics of other types of communicative acts, like request, query-ref, inform, and declare by extending the approach proposed in this paper, in our future works.

References

1. Chopra, A.K., Artikis, A., Bentahar, J., Colombetti, M., Dignum, F., Fornara, N., Jones, A.J.I., Singh, M.P., Yolum, P.: Research directions in agent communication. ACM Transactions on Intelligent Systems and Technology (in press)
2. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, Upper Saddle River (2005)
3. Fornara, N.: Specifying and Monitoring Obligations in Open Multiagent Systems Using Semantic Web Technology. In: Elçi, A., Koné, M.T., Orgun, M.A. (eds.) Semantic Agent Systems. SCI, vol. 344, pp. 25–45. Springer, Heidelberg (2011)
4. Fornara, N., Colombetti, M.: Operational specification of a commitment-based agent communication language. In: Proceedings of the First International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2002, Bologna, Italy, July 15-19, pp. 536–542. ACM (2002)
5. Fornara, N., Colombetti, M.: Specifying Artificial Institutions in the Event Calculus. In: Dignum, V. (ed.) Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models. Information Science Reference, ch. XIV, pp. 335–366. IGI Global (2009)

6. Fornara, N., Colombetti, M.: Representation and monitoring of commitments and norms using OWL. AI Commun. 23(4), 341–356 (2010)
7. Fornara, N., Viganò, F., Verdicchio, M., Colombetti, M.: Artificial institutions: A model of institutional reality for open multiagent systems. Artificial Intelligence and Law 16(1), 89–105 (2008)
8. Greenwood, D., Lyell, M., Mallya, A., Suguri, H.: SOAP based Message Transport for the Jade Multiagent Platform. In: Industry Track Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2009, pp. 101–104. ACM (2009)
9. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
10. Schiemann, B., Schreiber, U.: OWL-DL as a FIPA-ACL content language. In: Proceedings of the Workshop on Formal Ontology for Communicating Agents, Malaga, Spain (2006)
11. Searle, J.R.: Speech Acts: An Essay in the Philosophy of Language. Cambridge University Press, Cambridge (1969)
12. Singh, M.P.: A Social Semantics for Agent Communication Languages. In: Dignum, F.P.M., Greaves, M. (eds.) Agent Communication. LNCS, vol. 1916, pp. 31–45. Springer, Heidelberg (2000)
13. Yolum, P., Singh, M.: Reasoning about commitment in the event calculus: An approach for specifying and executing protocols. Annals of Mathematics and Artificial Intelligence 42, 227–253 (2004)
14. Zou, Y., Finin, T., Peng, Y., Joshi, A., Cost, S.: Agent Communication in Daml World. In: Truszkowski, W., Hinchey, M., Rouff, C. (eds.) WRAC 2002. LNCS, vol. 2564, pp. 347–354. Springer, Heidelberg (2003)

Appendix A: OWL 2 DL

OWL 2 DL is a practical realization of a Description Logic known as $\mathcal{SROIQ}(\mathcal{D})$. It allows one to define *classes*, *properties*, and *individuals*. An OWL ontology consists of: a set of class axioms that specify logical relationships between classes, which constitutes the *Terminological Box* (*TBox*); a set of property axioms to specify logical relationships between properties, which constitutes a *Role Box* (*RBox*); and a collection of assertions that describe individuals, which constitutes an *Assertion Box* (*ABox*).

Classes are formal descriptions of sets of objects (taken from a nonempty universe), and individuals can be regarded as names of objects of the universe. A class is either a *basic class* (i.e., an atomic class name) or a *complex class* build through a number of available *constructors*. Properties can be either *object properties*, which represent binary relations between objects of the universe, or *data properties*, which represent binary relationships between objects and data values (taken from XML Schema datatypes).

Through *class axioms* one may specify that subclass ($\sqsubseteq$) or equivalence ($\equiv$) relationships hold between certain classes, and that certain classes are disjoint. In particular, class axioms allow one to specify the domain and range of a property p (p: $\mathsf{A} \rightarrow \mathsf{B}$ where class A is the domain and class B is the range), and that a property is functional or inverse functional. *Property axioms* allow one to specify that a given property (or chain of subproperties) is a subproperty of another property, that two properties are equivalent, or that a property is reflexive, irreflexive, symmetric, asymmetric, or transitive. Finally, *assertions* allow one to specify that an individual a belongs to a class C, $\mathsf{C(a)}$, that an individual a is or is not related to another individual b through an object property R, $\mathsf{R(a,b)}$ or $\neg\mathsf{R(a,b)}$, that an individual is or is not related to a data value through a data property, or that two individuals are equal or different.

Complex classes can be specified by using Boolean operations on classes: $\mathsf{C} \sqcup \mathsf{D}$ is the union of classes, $\mathsf{C} \sqcap \mathsf{D}$ is the intersection of classes, and $\neg\ \mathsf{C}$ is the complement of class C. Classes can be specified also through *property restrictions*: (i) $\exists\ \mathsf{R.C}$ denotes the set of all objects that are related through property R to some objects belonging to class C, at least one; if we want to specify to how many objects an object is related we should write: $\leq\mathsf{nR}$, $\geq\mathsf{nR}$, $=\mathsf{nR}$ where n is any natural number; (ii) $\forall\ \mathsf{R.C}$ denotes the set of all objects that are related through R only to objects belonging to class C; (iii) $\mathsf{R}\ni\mathsf{a}$ denotes the set of all objects that are related to a through R.

We use capital initials for classes, and lower case initials for properties and individuals. We assume that all individuals introduced in the *ABox* are asserted to be different individuals.