USE OF SEMANTIC TECHNOLOGY TO DESCRIBE AND REASON ABOUT COMMUNICATION PROTOCOLS

Miren I. Bagüés, Idoia Berges*, Jesús Bermúdez, Alfredo Goñi and Arantza Illarramendi †
University of the Basque Country, Donostia, Spain

Keywords: Protocols, Agent communication, Communication Acts.

Abstract: Nowadays there is a tendency to enhance the functionality of Information Systems by appropriate information agents. Those information agents communicate through communication acts expressed in an Agent Communication Language. Moreover, the aim is to achieve interoperation of those agents through standard communication protocols in a distributed environment such as that supported by the Semantic Web.
In this paper we present a proposal to describe those protocols using a Semantic Web language. Two are the main features of that proposal. On the one hand, the communication acts that appear in the communication protocols are described by terms belonging to a communication acts ontology called COMMONT. On the other hand, protocols are represented by state transition systems described using OWL-DL language. This type of description provides the means to reason about the communication protocols in such a way that several kinds of structural relationships can be detected, namely if a protocol is a *prefix*, a *suffix* or an *infix* of another protocol and that relationship taken in a sense of *equivalence* or *specialization*. Furthermore, equivalence and specialization relationships can also be detected for complete protocols. Those relationships are captured by subsumption of classes described with a Semantic Web language.

1 INTRODUCTION

NOWADAYS the need for interoperability among agent based information systems in order to interchange information and services is increasing. But it seems to be difficult to achieve it due to the heterogeneity in Agent Communication Languages (ACL).

Moreover, once the understanding of atomic communication acts at a semantic level is achieved, the next step is to deal with agent communication protocols. We consider a context where administrators of information systems first, select protocols from existing repositories of protocols and then customize them. Later, they assign those selected protocols to their software agents. Therefore, when agents from two different information systems want to interoperate between them, it is relevant to see if the protocol used by one of them has structural relationships with the protocol used by the other. We represent protocols by state transition systems and describe them using the OWL-DL language (Bechhofer et al., 2005). So, we can consider structural relationships in a sense of *equivalence* or *specialization* applicable to complete protocols or to parts of them (*prefix*, *suffix* and *infix*). In a sense, our approach creates a classification of protocols.

In relation to other related works, we can observe two general aspects: 1) many of them also use transition systems for representing communication protocols and, 2) they do not consider the structural relationships among protocols managed by our proposal. In the following we show the main differences among those works and our proposal.

The closer related work is (Mallya and Singh, 2007), where protocols are represented as transition systems and subsumption and equivalence of protocols are defined with respect to three state similarity funtions. We share some goals with that work, but the protocol description formalism used by them is not considered in the paper and there is no references to how protocol relationships are computed. In contrast, we describe protocols with a description logic language and protocol relationships are computed by

*The work of Idoia Berges is supported by a grant of the Basque Government.

†All authors are members of the Interoperable DataBases Group, online at http://siul02.si.ehu.es. This work is also supported by the University of the Basque Country, Diputación Foral de Gipuzkoa (cosupported by the European Social Fund) and the Spanish Ministry of Education and Science TIN2007-68091-C02-01.

a description logic reasoner.

The works of (Yolum and Singh, 2002) and (Fornara and Colombetti, 2003) are quite similar one to each other. Both capture the semantics of communication acts through agents' commitments and represent communication protocols using a set of rules that operate on these commitments. Moreover those rule sets can be compiled as finite state machines. Nevertheless, they do not consider the study of relationships between protocols. In addition, in (Desai et al., 2005), protocols are also represented with a set of rules with terms obtained from an ontology, but their main goal is protocol development and, in order to reason about protocol composition, they formalize protocols into the π-calculus. Then, equivalence through bisimulation is the only process relationship considered. Notice that in our approach, reasoning is made with protocols' own representation, without demanding another different formalism.

In the context of Web Services composition, (Berardi et al., 2005) leads to a description logic based specification of finite state machines that is very different in purpose to our approach. In their scenario, states and transitions descriptions are not prepared to be confronted in a comparison. In constrast, in our case, state and transition descriptions are carefully modelled as class descriptions such that subsumption relation captures structural relationhips.

In (Kagal and Finin, 2007) protocols are defined as a set of permissions and obligations of agents participating in the communication. They use an OWL ontology for defining the terms of the specification language, but their basic reasoning is made with an ad hoc reasoning engine. We share their main goal of defining protocols in a general framework that allows reutilization. Nevertheless, they do not consider relationships between protocols.

Finally, (d'Inverno et al., 1998) and (Mazouzi et al., 2002) use finite state machines and Petri nets, respectively, but without taking into account the meaning of the communication acts interchanged, neither considering relationships between protocols.

In the rest of this paper we present first the main features of a communication ontology called COMMONT. Then, in section 3 we show how we can describe protocols using OWL. Next, in section 4 we explain the way we deal with cycles in protocols. The structural relationships among protocols are presented in section 5. We finish with some conclusions in section 6.

2 COMMUNICATION ONTOLOGY

Among the different models proposed for representing protocols one which stands out is that of State Transition Systems (STS). In our proposal we use STS where each transition is labeled with a communication act class described in a communication ontology called COMMONT. Therefore, we present in the following the main features of that ontology.

The goal of COMMONT ontology is to favour the interoperation among agents belonging to different Information Systems. The leading categories of that ontology are: first, *communication acts* that are used for interaction by *actors* and that have different purposes and deal with different kinds of contents; and second, *contents* that are the sentences included in the communication acts.

The main design criteria adopted for the communication acts category of the COMMONT ontology is to follow the *speech acts* theory (Searle, 1969), a linguistic theory that is recognized as one of the main sources of inspiration for designing the most familiar standard agent communication languages. Following that theory, every communication act is the sender's expression of an attitude toward some possibly complex proposition. A sender performs a communication act which is expressed by a coded message and is directed to a receiver. Therefore, a communication act has two main components. First, the attitude of the sender which is called the *illocutionary force* (F), that expresses social interactions such as informing, requesting or promising, among others. And second, the *propositional content* (p) which is the subject of what the attitude is about. In COMMONT this *F(p)* framework is followed, and different kinds of illocutionary forces and contents leading to different classes of communication acts are supported.

COMMONT is divided into three interrelated layers: *upper*, *standards* and *applications*, that group communication acts at different levels of abstraction. Classes of the COMMONT ontology are described using the Web Ontology Language OWL (more specifically the OWL DL sublanguage).

In the upper layer, according to Searle's speech acts theory, five upper classes of communication acts corresponding to *Assertives*, *Directives*, *Commissives*, *Expressives* and *Declaratives* are specified. But also the top class `CommunicationAct`[3] is defined that represents the universal class of communication acts. Every particular communication act is an individual of that class. In COMMONT, components of a class

[3]`This type` style refer to terms specified in the ontology.

are represented by properties. The most immediate properties of `CommunicationAct` are the content and the actors who send and receive the communication act. There are some other properties related to the context of a communication act such as the conversation in which it is inserted or a link to the domain ontology that includes the terms used in the content.

A standards layer extends the upper layer of the ontology with specific terms that represent classes of communication acts of general purpose agent communication languages, like those from KQML or FIPA-ACL. With respect to FIPA-ACL, we can observe that it proposes four primitive communicative acts (FIPA, 2005): *Confirm*, *Disconfirm*, *Inform* and *Request*. The terms `FIPA-Confirm`, `FIPA-Disconfirm`, `FIPA-Inform` and `FIPA-Request` are used to respectively represent them as classes in COMMONT. Furthermore, the rest of the FIPA communicative acts are derived from those mentioned four primitives. Analogously, communication acts from KQML can be analyzed and the corresponding terms in COMMONT can be specified. It is of vital relevance for the interoperability aim to specify ontological relationships among classes of different standards.

Finally, it is often the case that every single information system uses a limited collection of communication acts that constitute its particular agent communication language. The applications layer reflects the terms describing communication acts used in such particular information systems. The applications layer of the COMMONT ontology provides a framework for the description of the nuances of such communication acts. Some of those communication acts can be defined as particularizations of existing classes in the standards layer and maybe some others as particularizations of upper layer classes.

3 PROTOCOL DESCRIPTIONS

In order to represent protocols using OWL-DL, we have defined three different classes: `Protocol`, `State`, and `Transition`, which respectively represent protocols, states and transitions in protocols.

We consider propositions that can change their truth values due to events. A state is described by the propositions that holds in it plus the transitions that can take place from it. Some states are declared final states. A transition is associated to the communication act that is sent and to the state reached with that transition. Only one communication act can be associated to a transition and only one state is reached by a transition. Then, with these descriptions, a protocol is

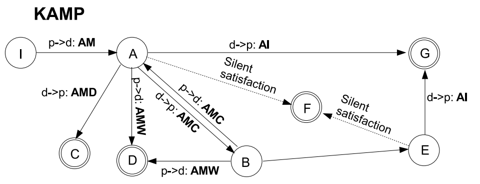

Figure 1: The KAMP protocol.

characterized by its initial state. An actual conversation following a protocol is an individual of the class `Protocol`.

Following are some OWL axioms presented in a logic notation[4] instead of the more verbose OWL/XML syntax.

$$
\begin{array}{rcl}
\texttt{Protocol} & \equiv & \exists\texttt{hasInitialState.State}\sqcap \\
 & & \forall\texttt{hasInitialState.State} \\
\texttt{State} & \equiv & \forall\texttt{hasTransition.Transition} \\
 & & \exists\texttt{hasPredicate.Proposition}\sqcap \\
 & & \forall\texttt{hasPredicate.Proposition} \\
\texttt{Transition} & \equiv & \texttt{=1 hasCommAct.CommunicationAct}\sqcap \\
 & & \texttt{=1.hasNextState.State} \\
\texttt{FinalState} & \sqsubseteq & \texttt{State} \\
\texttt{CommunicationAct} & \sqsubseteq & \forall\texttt{hasSender.Actor}\sqcap \\
 & & \forall\texttt{hasReceiver.Actor}\sqcap \\
 & & \forall\texttt{hasContent.Content}\sqcap \\
 & & \forall\texttt{hasSubject.Subject}
\end{array}
$$

Following, we introduce one example using the previous kind of description to represent a concrete protocol, in particular the KAMP protocol. This KAMP protocol is our specialization of the well-known protocol Request conversation policy of KAoS (Bradshaw et al., 1997)(KAoSAppointmentModifyProtocol(KAMP)) and its graphical representation can be seen in Fig. 1.

Two agents take part in our protocol: Patient, labelled with *p*, and Doctor, labelled with *d*. First, agent *p* requests agent *d* to modify the date of an appointment, using the communication act AppointmentModify, or shortly, *AM* (described in the COMMONT ontology). Then, several situations can arise: On the one hand, some communication acts can lead us to a final state. This happens when *d* agrees to modify the date (AppointmentInform, *AI*), when *d* declines to modify the date (AppointmentModifyDecline, *AMD*) or when *p* withdraws the request (AppointmentModifyWithdraw,*AMW*). On the other hand, both agents can keep countering one to another (AppointmentModifyCounter, *AMC*) until they reach an agreement in the date of the appointment.

[4]This notation is common in the Description Logics (DL) field. See (Baader et al., 2003) for a full explanation.

Following we present our initial description of KAMP. Notice that the complete description of a protocol comprises a set of definitions, although the protocol is represented by the class name KAMP. Moreover, the description of a protocol is directed by the graph that represents that protocol. Class names for states are prefixed by S and class names for transitions are prefixed by T and formed with the letters of source and target states.

$$
\begin{aligned}
\texttt{KAMP} &\equiv \texttt{Protocol} \sqcap \exists\texttt{hasInitialState.S-I} \\
\texttt{S-I} &\equiv \texttt{State} \sqcap \exists\texttt{hasTransition.T-IA} \sqcap \exists\texttt{hasPredicate.PI} \\
\texttt{S-A} &\equiv \texttt{State} \sqcap \exists\texttt{hasTransition.(T-AC} \sqcup \texttt{T-AD} \sqcup \texttt{T-AB} \sqcup \texttt{T-AG)} \\
&\quad \sqcap \exists\texttt{hasPredicate.PA} \\
\texttt{S-B} &\equiv \texttt{State} \sqcap \exists\texttt{hasTransition.(T-BA} \sqcup \texttt{T-BD} \sqcup \texttt{T-BE)} \\
&\quad \sqcap \exists\texttt{hasPredicate.PB} \\
\texttt{S-C} &\equiv \texttt{FinalState} \sqcap \exists\texttt{hasPredicate.PC} \\
\texttt{S-D} &\equiv \texttt{FinalState} \sqcap \exists\texttt{hasPredicate.PD} \\
\texttt{S-E} &\equiv \texttt{State} \sqcap \exists\texttt{hasTransition.T-EG} \sqcap \exists\texttt{hasPredicate.PE} \\
\texttt{S-F} &\equiv \texttt{FinalState} \sqcap \exists\texttt{hasPredicate.PF} \\
\texttt{S-G} &\equiv \texttt{FinalState} \sqcap \exists\texttt{hasPredicate.PG} \\
\texttt{T-IA} &\equiv \texttt{Transition} \sqcap \exists\texttt{hasCommAct.AppointmentModify} \\
&\quad \sqcap \exists\texttt{hasNextState.S-A} \\
\texttt{T-AC} &\equiv \texttt{Transition} \sqcap \exists\texttt{hasCommAct.AppointmentModifyDecline} \\
&\quad \sqcap \exists\texttt{hasNextState.S-C} \\
\texttt{T-AD} &\equiv \texttt{Transition} \sqcap \exists\texttt{hasCommAct.AppointmentModifyWithdraw} \\
&\quad \sqcap \exists\texttt{hasNextState.S-D} \\
\texttt{T-AB} &\equiv \texttt{Transition} \sqcap \exists\texttt{hasCommAct.AppointmentModifyCounter} \\
&\quad \sqcap \exists\texttt{hasNextState.S-B} \\
\texttt{T-AG} &\equiv \texttt{Transition} \sqcap \exists\texttt{hasCommAct.AppointmentInform} \\
&\quad \sqcap \exists\texttt{hasNextState.S-G} \\
\texttt{T-BA} &\equiv \texttt{Transition} \sqcap \exists\texttt{hasCommAct.AppointmentModifyCounter} \\
&\quad \sqcap \exists\texttt{hasNextState.S-A} \\
\texttt{T-BD} &\equiv \texttt{Transition} \sqcap \exists\texttt{hasCommAct.AppointmentModifyWithdraw} \\
&\quad \sqcap \exists\texttt{hasNextState.S-D} \\
\texttt{T-BE} &\equiv \texttt{Transition} \sqcap \exists\texttt{hasCommAct.AppointmentModifyAccept} \\
&\quad \sqcap \exists\texttt{hasNextState.S-E} \\
\texttt{T-EG} &\equiv \texttt{Transition} \sqcap \exists\texttt{hasCommAct.AppointmentInform} \sqcap \\
&\quad \exists\texttt{hasNextState.S-G}
\end{aligned}
$$

Notice that communication acts that appear in the transitions of the protocol are defined in the COMMONT ontology in the following way:

$$
\begin{aligned}
\texttt{AppointmentModify} &\equiv \texttt{Request} \\
&\quad \sqcap =1\ \texttt{hasContent.AppointmentOverwrite} \\
\texttt{AppointmentOverwrite} &\equiv \texttt{Overwrite} \\
&\quad \sqcap \exists\texttt{hasSubject.Appointment} \\
\texttt{AppointmentInform} &\equiv \texttt{Responsive} \\
&\quad \sqcap =1\ \texttt{hasContent.AppointmentInformation} \\
&\quad \sqcap \exists\texttt{inReplyTo.AppointmentModify} \\
\texttt{AppointmentInformation} &\equiv \texttt{Proposition} \\
&\quad \sqcap \exists\texttt{hasSubject.Appointment} \\
\texttt{AppointmentModifyCounter} &\equiv \texttt{Counter} \\
&\quad \sqcap =1\ \texttt{hasContent.AppointmentOverwrite} \\
&\quad \sqcap =1\ \texttt{inCounterTo.AppointmentOverwrite} \\
\texttt{AppointmentModifyAccept} &\equiv \texttt{Accept} \\
&\quad \sqcap =1\ \texttt{hasContent.AppointmentOverwrite} \\
\texttt{AppointmentModifyDecline} &\equiv \texttt{Decline} \\
&\quad \sqcap =1\ \texttt{hasContent.AppointmentOverwrite} \\
\texttt{AppointmentModifyWithdraw} &\equiv \texttt{Withdraw} \\
&\quad \sqcap =1\ \texttt{hasContent.AppointmentOverwrite}
\end{aligned}
$$

4 CYCLES: CAN WE MANAGE THEM?

As we said in section 3, our OWL-DL description of a protocol is directed by the graph that represents that protocol. Notice also that a protocol with cycles can be represented by different graphs that deploy the same runs of communication acts. Unfortunately, when state classes of protocols are involved in cycles our descriptions of them does not result adequate to capture equivalence relationships between states of different graphs allowing for the same set of runs beginning on that state (notice that the transitive clousure operator of regular languages is not supported by OWL-DL).

Fortunately, we have found a way to bypass this inconvenience assuming that cycles in protocols mean repetition of a deliberation process in such a way that a complete cycle execution returns to a state class equivalent to the initial class of the turn. Therefore, considering the task of comparing protocol structure in order to decide compliance of protocol runs, it is not important the number of times a cycle is walked around but only if it is possible to go for one turn. Consequently, we systematically modify the descriptions of protocols with cycles in order to achieve correct answers with respect to the structural relationships defined in section 5. It is worth mentioning that this modification is systematic and it is only carried out for the reasoning process.

Moreover, we also assume that, for every protocol run, each communication act involved in a cycle maintains invariant the class of its content. The repetition of deliberation assumed for every cycle in a run is represented by the fact that the content of an occurring communication act in a transition involved in a cycle, removes from the state the content of the previously occurrence of a communication act in the same transition. A formalization of those two assumptions is represented by the following rule.

RULE 1: *e(x) in a cycle $\wedge$ HoldsAt(P(i), t) $\wedge$ owner(P(i),O) $\wedge$ Happens(e(x), t) $\wedge$ Initiates(e(x), P(j), t) $\wedge$ owner(P(j),O) $\wedge$ i, j belong to the same class $\rightarrow$ Terminates(e(x), P(i), t)*

By the owner of a proposition we refer to the sender of the message whose content is that proposition itself. Rule 1 declares that when the owner of a proposition that is now holding causes an event that initiates holding of a new proposition that belongs to the same class than the previous one, and the event is in a cycle, the previous proposition ceases to hold.

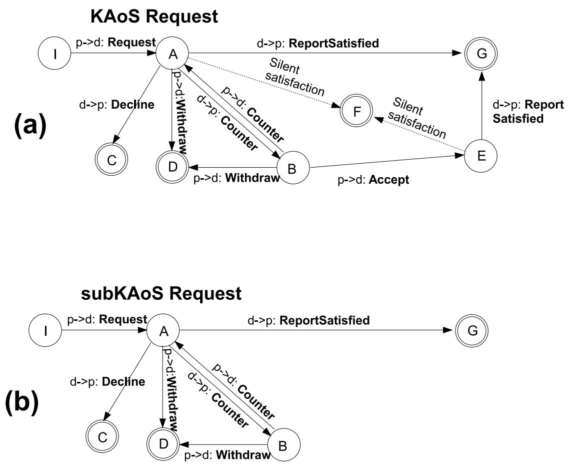

Figure 2: Equivalence of protocols.

5 PROTOCOL STRUCTURAL RELATIONSHIPS

When two agents of different systems want to intercommunicate using a particular protocol each, both protocols must match properly. Thanks to our protocol description, a reasoning process can decide if two protocols are equivalent, or if one is a specialization of the other, otherwise they are incompatible protocols. Moreover, we have developed a reasoning service that decides if a protocol is a prefix, a suffix or an infix of another protocol, and those relationships taken also in the sense of equivalence or specialization. Notice also that due to the fact that protocols are described as OWL-DL classes, structural relationships among protocols can also be defined in terms of DL semantics. We write $\vdash$A$\sqsubseteq$B (respectively $\vdash$A$\equiv$B) to mean that class A is subsumed by class B (respectively A is equivalent to B).

Definition 1. *Two protocols A and B are equivalent (A$\equiv_{tot}$B) if $\vdash$A$\equiv$B.*

By specialization of protocols we mean specialization of the classes of communication acts appearing in the protocol.

Definition 2. *Protocol A is a specialization of protocol B (A$\sqsubseteq_{tot}$B) if $\vdash$A$\sqsubseteq$B.*

For example, the KAMP protocol in Fig. 1 is a specialization of the KAoS Request protocol shown in Fig. 2a. Every communication act that appears in the transitions of the KAMP Protocol is a subclass of the communication act in the corresponding transition of the KAoS Request protocol.

In order to look for a prefix relationship, we can modify a protocol description by converting some states into final states. Let S be a state subclass, we denote *final*(S) to the expression resulting from adding by conjunction the class `FinalState` to the defining expression of S. For example, if `S-E` $\equiv$ `State` $\sqcap$ $\exists$`hasTransition.T-EG` then *final*(`S-E`) is the expression `FinalState`$\sqcap$ `State` $\sqcap$ $\exists$`hasTransition.T-EG` . Moreover, we can suppress some transitions by removing their axiom definitions and removing the corresponding expressions $\exists$`hasTransition.T` from the state definitions which include it. We denote $Prune[P/(S_1 \dots S_n),(T_1 \dots T_m)]$ to the new protocol described by the set of descriptions that results from replacing definition expressions of $S_1 \dots S_n$, respectively, by expresssions *final*(S_1)... *final*(S_n), and removing occurrences of transitions $T_1 \dots T_m$ in the set of descriptions of protocol P.

Definition 3. *Protocol A is a prefix of protocol B (A$\equiv_{pre}$B) if there exist states $S_1 \dots S_n$ and transitions $T_1 \dots T_m$ in the set of equations defining B such that $\vdash$A$\equiv$ Prune[B$/(S_1 \dots S_n),(T_1 \dots T_m)$].*

Analogously, protocol A is a specialization of a prefix of protocol B (A$\sqsubseteq_{pre}$B) if $\vdash$A$\sqsubseteq$ Prune[B$/(S_1 \dots S_n),(T_1 \dots T_m)$].

For example, the subKAoS Request protocol shown in Fig. 2(b) is a prefix of the KAoS Request protocol in Fig. 2(a) because it is equivalent to that protocol after removing two transitions: one from state A to state F and another from state B to state E.

With respect to suffix relationship, we modify a protocol description converting some states into initial states. Let $S_1 \dots S_n$ states in the definition of protocol P. We denote *ChangeInit*[$P/S_1 \dots S_n$] to the new protocol described by the set of descriptions that results from modifying the P protocol definition in the following way: adding by conjunction $\exists hasInitialState.S_i$ ($i = 1 \dots n$).

Definition 4. *Protocol A is a suffix of protocol B (A$\equiv_{suf}$B) if there exist states $S_1 \dots S_n$ in the set of equations defining B such that $\vdash$A$\equiv$ChangeInit[B$/S_1 \dots S_n$].*

Analogously, protocol A is a specialization of a suffix of protocol B (A$\sqsubseteq_{suf}$B) if $\vdash$A$\sqsubseteq$ChangeInit[B$/S_1 \dots S_n$].

We denote *Prot*[S] to a new protocol defined as *Prot*[S] $\equiv$ `Protocol` $\sqcap$ $\exists$`hasInitialState.S`.

Definition 5. *Protocol A is an infix of protocol B (A$\equiv_{inf}$B) if there exist a state S in the set of equations defining B such that A$\equiv_{pre}$Prot[S] and Prot[S]$\equiv_{suf}$B.*

Analogously, protocol A is a specialization of an infix of protocol B (A$\sqsubseteq_{inf}$B) if A$\sqsubseteq_{pre}$Prot[S] and Prot[S]$\sqsubseteq_{suf}$B.

6 CONCLUSIONS

In this paper we have presented a proposal to describe communication protocols, represented by STS, using the OWL-DL language. The main contribution of this proposal is the possibility that it brings to reason about communication protocols in such a way that different kinds of structural relationships -*equivalence* or *specialization*- between complete protocols or parts of them: *prefix*, *infix* and *suffix* can be detected. The formal definition of those structural relationships has been included in the paper within a justification of how cycles can be managed.

REFERENCES

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., editors (2003). *The Description Logic Handbook. Theory, Implementation and Applications.* Cambridge University Press.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuiness, D., Patel-Schneider, P., and Stein, L. (2005). *OWL Web Ontology Language Reference*. World Wide Web Consortium.

Berardi, D., Calvanese, D., Giacomo, G. D., Lenzerini, M., and Mecella, M. (2005). Automatic service composition based on behavioral descriptions. *Int. J. Cooperative Inf. Syst.*, 14(4):333–376.

Bradshaw, J. M., Dutfield, S., Benoit, P., and Woolley, J. D. (1997). Kaos: toward an industrial-strength open agent architecture. pages 375–418.

Desai, N., Mallya, A. U., Chopra, A. K., and Singh, M. P. (2005). Interaction protocols as design abstractions for business processes. *IEEE Trans. Softw. Eng.*, 31(12):1015–1027.

d'Inverno, M., Kinny, D., and Luck, M. (1998). Interaction protocols in agentis. In *In Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98)*, pages 261–268.

FIPA (2005). FIPA communicative act library specification. http://www.fipa.org/specs/fipa00037/SC00037J.html.

Fornara, N. and Colombetti, M. (2003). Defining interaction protocols using a commitment-based agent communication language. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 520–527, New York, NY, USA. ACM Press.

Kagal, L. and Finin, T. (2007). Modeling conversation policies using permissions and obligations. *Autonomous Agents and Multi-Agent Systems*, 14(2):187–206.

Mallya, A. U. and Singh, M. P. (2007). An algebra for commitment protocols. *Autonomous Agents and Multi-Agent Systems*, 14(2):143–163.

Mazouzi, H., Seghrouchni, A. E. F., and Haddad, S. (2002). Open protocol design for complex interactions in multi-agent systems. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 517–526, New York, NY, USA. ACM Press.

Searle, J. R. (1969). *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, New York.

Yolum, P. and Singh, M. P. (2002). Flexible protocol specification and execution: Applying event calculus planning using commitments. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 527–534. ACM Press.