Personalizing web services by reasoning about interaction protocols

Matteo Baldoni (baldoni@di.unito.it)
Cristina Baroglio (baroglio@di.unito.it)
Viviana Patti (patti@di.unito.it)
Dipartimento di Informatica
Università di Torino, C.so Svizzera 185, 10149, TORINO

Abstract

In this work, we argue the importance of including in the emerging standards for web service descriptions also the high-level communication protocol, used by a service to interact with its clients. We will motivate this claim by setting web services in a multi-agent framework, where web services are implemented as software agents that interact with other agents, the personal assistants, used to crawl the Internet trying to pursue specific goals. In our proposal, interaction is interpreted as the effect of action execution; we will show a typical situation in which the ability of reasoning about interactions and conversations is crucial for personalizing the service fruition.

Introduction

The capability of reasoning about conversations is useful in different application domains. Besides the often mentioned robotic applications, also the web world offers interesting scenarios. Recently a great attention has been posed on web services, on standard languages for web service specification, as DAML-S (DAML-S, 2003) and WSDL (WSDL, 2003), and on the connections between web services and agent systems (e.g. McIlraith & Son., 2001). In this line of work, we show how the capability of reasoning about agent interaction protocols and conversations allows agents to personalize service fruition, making the whole interaction closer to human expectations.

Consider, for instance, a personal assistant implemented as a rational agent, whose task is to find web services that match a user's specification, consisting of a description of the user's goals ("book two tickets for the movie Akira") and of additional conditions on how to perform the interaction ("Do not send my credit card number").

Communication among agents is commonly ruled by possibly non-deterministic interaction protocols, aimed at gathering/supplying the necessary information. The provider and the personal assistant can, then, be seen as the two actors of a conversation determined by the protocol. If the personal assistant could *reason about* the possible interactions ruled by such a protocol, it could *personalize* the interaction by choosing an execution path that match the user's requirements; in case this is not possible, it could at least decide to search for another service provider.

We have studied a solution based on techniques for reasoning about actions and change; in particular, we exploit the features of an agent logic programming language, DyLOG (Baldoni et. al, 2001; Patti, 2002), that is based on a modal logic of actions and mental attitudes and allows one to specify agent complex behaviors and to reason about them before acting. The action framework DyLOG is based on allows to deal with communicative behaviors, considering a conversation protocol as the (non-deterministic) specification of the complex communicative behavior of a single agent. Some characteristics of the DyLOG approach to communication, that make it particularly appealing for our purpose, is that DyLOG allow to model *single rational agents*, that decide how to behave by taking into account the outcomes of reasoning about interaction protocol executions, while most of the approaches in the logic-based communicative agents literature focus on modeling the interactive behavior of the overall multi-agent system and on proving universal properties of such interactions.

Issue and motivations

The world-wide web is a continuous source of inspiration and a test field for many Artificial Intelligence (AI) research areas. The huge amount of information urged the development of standard languages for representing *meta-data* and then *semantic knowledge*, which lead to the definition of languages, such as RDF (RDF, 1999), DAML+OIL (DAML+OIL, 2001) and OWL (OWL, 2003), founded on logic approaches to knowledge representation.

In the very same way in which these languages allow the representation of the semantics of a web page, very recently some attempt to standardize the *descriptions* of *web services*, which we can consider as dynamic entities with a behavior, motivated the design of the description languages DAML-S and WSDL. While WSDL is a standardization initiative of the commercial world, focussed on the issue of interoperability, DAML-S is concerned with providing a way to describe web services so that it is possible to *reason about them* (Bryson et. al., 2002). A service description is divided in three conceptual areas: the *profile*, used for the purpose of advertising the service and allow possible users to discover it, the *process model*, that describes

how it works, and the *grounding*, a lower-level information that describes how an agent can actually access the service. From an AI perspective, the process model is the most interesting component. DAML-S describes how a service works in a way inspired by the language GOLOG and its extensions (Levesque et. al, 1997, De Giacomo & Lesperance & Levesque, 2000), relying on the *action metaphor*: a service is viewed as an *action* (atomic or complex) with preconditions and effects, that can modify the state of the world and the state of agents that work in the world; the process model is a description of such an action. In this context, we can, then, design agents, which apply techniques for *reasoning about actions and change* to web service process models, for producing composite and customized services, for automatic web service discovery, and for automatic execution and monitoring.

In this work we claim the need of including in a DAML-S-like web service description also the *conversation protocol* followed by the web service, in order to achieve a new degree of *personalization* in the various tasks described above and, in particular, web service fruition. We set our work in a multi-agent framework in which the web service is an agent that communicates with other agents in a FIPA-like Action Communication Language (ACL). In this context, the *web service behavior* itself can be expressed as a conversation protocol, which describes the communications that can occur between the service and its client at a high-level (i.e. not at network-level).

Actually, in the literature about multi-agent systems, when a communication is enacted, agents exchange their communication protocol (Mamdani & Pitt, 2000). The advantage of protocol exchange is that agents can reason about the change caused by a conversation to their own belief state and make rational assumptions about the change caused to the *other* agent beliefs. Thus agents can decide, *before* starting the actual interaction, if that interaction respects given constraints.

Let us consider the software agent described in the introduction; in the following we will refer to it as *pa* (personal assistant). The task of the agent is to invoke a web service that must satisfy the user's requirements, not *only in the kind of service it provides* but also *in the kind of interaction taken*: the agent must avoid to communicate to the web service the user's credit card number. Suppose that two cinema booking services are available, called *click-ticket* and *all-cinema* respectively, that follow different interaction protocols, one permitting both to book a ticket to be paid later by cash (Figure 2, first graph) and to buy it by credit card (Figure 2, second graph), the other allowing only ticket purchase by credit card (Figure 3). Given the user's requests and knowledge about the interaction protocols, *pa* could choose *click-ticket* because its interaction protocol allows conversations in which the user's credit card number is not requested.

In order to perform this kind of reasoning, it is necessary to formalize *at a high-level* the communicative acts and the interaction protocols. A rational model of communicative acts requires the agent to make assumptions about its interlocutor's beliefs (Sadek & Bretier, 1997, Herzig & Longin, 1999); for instance, the agent communicates a piece of information if it believes the other agent to be ignorant about it. The solution that we propose exploits a modal logic approach to agent programming. In the following sections we will briefly describe the DyLOG language and, afterwards, we will use it to develop agent *pa*.

Modeling communicative agents in DyLOG

The DyLOG agent language is used to program rational agents by describing their behavior in terms of either atomic or complex actions. Intuitively, atomic actions are either *world* actions (affecting the world, we denote them by A) or *mental* actions, i.e. sensing or communicative actions which only affect the *agent beliefs* (we respectively denote them by C and S). For each atomic action a and agent ag_i we introduce the modalities $[a^{agi}]$ and $\langle a^{agi}\rangle$: $[a^{agi}]\alpha$ means that α holds after every execution of action a by agent ag_i; $\langle a^{agi}\rangle$ means that there is a possible execution of a (by ag_i) after which α holds. For each atomic action a in $\mathsf{A} \cup \mathsf{C}$ we also introduce a modality $Done(a^{agi})$ for expressing that a has been executed. $Done(a^{agi})\,\alpha$ is read "a has been executed by ag_i ; before its execution, α was true". The modality $\Box$ denotes formulas that hold in all the possible agent mental states. The DyLOG formalization of complex actions draws considerably from dynamic logic for the definition of action operators like sequence, test and non-deterministic choice but, differently than (Levesque et. al., 1997), we refer to a *Prolog-like* paradigm, in which procedures are defined by means of (possibly recursive) clauses. For each procedure p, the language contains also the universal and existential modalities $[p]$ and $\langle p\rangle$.

The *mental state* of an agent is described in terms of a consistent set of *belief formulas*. The belief state of a DyLOG includes *nested beliefs* for representing what *other* agents believe and reasoning on how *they* can be affected by communicative actions. We use the modal operator B^{agi} to model the beliefs of agent ag_i. The modality M^{agi} is defined as the dual of B^{agi} ($\mathsf{M}^{agi}\varphi \equiv \neg\mathsf{B}^{agi}\neg\varphi$). Intuitively $\mathsf{M}^{agi}\varphi$ means that ag_i considers φ possible. All the modalities of the language are normal; $\Box$ is reflexive and transitive, its interaction with action modalities is ruled by $\Box\,\varphi \supset [a^{agi}]\,\varphi$. The epistemic modality B^{agi} is serial, transitive and euclidean. The interaction of the $Done(a^{agi})$ modality with other

modalities is ruled by: $\varphi \supset [a^{agi}]Done(a^{agi})\,\varphi$ and $Done(a^{agj})\,\varphi \supset \mathsf{B}^{agi}Done(a^{agj})\,\varphi$ (awareness), with $ag_i = ag_j$ when a^{agi} is not a communicative act.

A non-monotonic solution to the persistency problem is given, which consists in maximizing persistency assumptions about fluents after the execution of actions, within an abductive framework.

In the line of (Baldoni et. al., 2001) the *behavior* of an agent ag_i can be specified by a *domain description*, which includes: (i) a specification of the agent *belief state*, (ii) action and precondition laws for describing the *atomic world actions* in terms of their preconditions and effects on the executor's mental state, (iii) sensing axioms for describing *atomic sensing actions*, (iv) procedure axioms for describing *complex behaviors*.

Belief state

An agent is an individual with a *subjective* point of view on a dynamic domain, represented by the contents of its *mental state*. Notice that in this perspective, we do not model the real world but only the *internal dynamics* of each agent in relation to the changes caused by actions. A mental state is a set of belief formulas (*belief state*), that represent what it believes about the world and about the other agents beliefs.

A belief state is a complete and consistent set of rank 1 and 2 belief fluents, where a *belief fluent F* is a belief formula $\mathsf{B}^{agi}L$ or its negation. L denotes a *belief argument*, i.e. a *fluent literal* (f or $\neg f$), a *done fluent* ($Done(a^{agi})\mathsf{T}$ or its negation), or a belief fluent of rank 1 ($\mathsf{B}l$ or $\neg\mathsf{B}l$). We use l for denoting attitude-free fluents, i.e. fluent literals or done fluents.

Consistency is guaranteed by the seriality of the B^{agi} modalities[1]. In essence a belief state provides, for each agent ag_i, a three-valued interpretation of all the possible belief arguments L: each L is either *true*, *false*, or *undefined* when both $\neg\mathsf{B}^{agi}L$ and $\neg\mathsf{B}^{agi}\neg L$ hold. We use $\mathsf{U}^{agi}L$ for expressing the ignorance of a^{agi} about L.

Primitive speech acts

World and sensing actions performed by an individual agent ag_i are considered atomic actions, described in terms of preconditions and effects on local actor's mental state. In particular they trigger a revision process on actor's beliefs. Also communication primitives are atomic actions, described in terms of preconditions and effects on the agent mental state. They have the form *speech_act(sender, receiver, l)*, where *sender* and *receiver* are agents and l is either a *fluent literal* or a *done fluent*. Following the standard rational model (Cohen and Levesque, 1990), primitive speech acts are viewed as special mental actions, affecting both the sender's and the receiver's mental state. In our model we focussed on the *internal representation*, that individual agents have of each speech act, by specifying ag_i 's belief changes both when it is the sender and when it is the receiver. Such a representation it is necessary for providing our agents with the capability to reason in advance about possible results of conversations arising with other agents.

Speech act specification is, then, twofold: one definition holds when the agent is the sender, the other when it is the receiver. In the first case, the precondition laws contain some *sincerity condition* that must hold in the agent mental state. When ag_i is the receiver, the action is considered always executable, since ag_i cannot influence the executability of other's actions.

Let us consider some primitive speech acts from the standard FIPA-ACL (FIPA, 2000), and let us see how we define their semantics within our framework:

a) $\Box(\mathsf{B}^{agi}l \wedge \mathsf{B}^{agi}\mathsf{U}^{agj}l \supset \langle\text{inform}(ag_i, ag_j, l)\rangle\mathsf{T})$
b) $\Box([\text{inform}(ag_i, ag_j, l)]\mathsf{M}^{agi}\mathsf{B}^{agj}l)$
c) $\Box(\mathsf{B}^{agi}\mathsf{B}^{agj}\text{authority}(ag_i, l) \supset [\text{inform}(ag_i, ag_j, l)]\mathsf{B}^{agi}\mathsf{B}^{agj}l)$
d) $\Box(\mathsf{T} \supset \langle\text{inform}(ag_j, ag_i, l)\rangle\mathsf{T})$
e) $\Box([\text{inform}(ag_j, ag_i, l)]\,\mathsf{B}^{agi}\mathsf{B}^{agj}l)$
f) $\Box(\mathsf{B}^{agi}\text{authority}(ag_j, l) \supset [\text{inform}(ag_j, ag_i, l)]\mathsf{B}^{agi}l)$
g) $\Box(\mathsf{M}^{agi}\text{authority}(ag_j, l) \supset [\text{inform}(ag_j, ag_i, l)]\mathsf{M}^{agi}l)$

By *inform(sender,receiver, l)* the sender informs the receiver about the matter l. To perform an *inform* act ag_i must believe l and must believes that the receiver ignores l (a). Moreover, when ag_i is the sender it thinks possible that the receiver will adopt its belief, although it cannot be certain -*autonomy assumption* (b)-. If it believes that ag_j considers it a *trusted authority* about l, it is confident that the receiver will adopt its belief (c). When ag_i is the receiver, it believes that l is believed by the sender ag_j (e), but it adopts l as an own belief only if it thinks ag_j is a trusted authority (f)-(g).

a) $\Box(\mathsf{U}^{agi}l \wedge\neg\, \mathsf{B}^{agi}\,\mathsf{U}^{agj}l \supset \langle\text{queryIf}(ag_i, ag_j, l)\rangle\mathsf{T})$
b) $\Box(\mathsf{T} \supset \langle\text{queryIf}(ag_j, ag_i, l)\rangle\mathsf{T})$
c) $\Box([\text{queryIf}(ag_j, ag_i, l)]\mathsf{B}^{agi}\mathsf{U}^{agj}l)$

By *queryIf(sender,receiver,l)* the sender asks the receiver if it believes l. To perform a *queryIf* act, ag_i must ignore l and it must believe that the receiver does not ignore l (a). When ag_i is the receiver after a *queryIf* act it will believe that the sender ignores l (c).

a) $\Box(\mathsf{U}^{agi}l \wedge \mathsf{B}^{agi}\text{Done}(\text{queryIf}(ag_j, ag_i, l))\mathsf{T} \supset \langle\text{refuse}(ag_i, ag_j, l)\rangle\mathsf{T})$

[1] A belief state is *not consistent* when it contains: a belief B^{agi} l and its negation, or the belief formulas $\mathsf{B}^{agj}\mathsf{B}^{agi}$ l and $\mathsf{B}^{agj}\mathsf{B}^{agi}$ $\neg l$, or the belief formulas $\mathsf{B}^{agj}\mathsf{B}^{agi}$ l and $\mathsf{B}^{agj}\neg\mathsf{B}^{agi}$ l.

b) $\Box$ (T $\supset$ <refuse(ag_j, ag_i, l)>T)
c) $\Box$ ([refuse(ag_j, ag_i, l)]$B^{agi}U^{agj}l$)

By *refuse (sender, receiver, l)* an agent refuses to give an information it was asked for. The refusal can be executed only if: the sender ignores l and it believes that the receiver previously queried it about l. After a refusal the receiver believes that the sender ignores l.

Conversation Protocols

Complex behaviors of an agents are expressed by means of procedure definitions, which allow to build complex actions for an agent ag_i on the basis of other complex actions, atomic actions and *test actions* on ag_i mental state (Baldoni et.al, 2001). Formally, a ag_i's complex action is defined by means of a set of *procedure axioms* of our modal logic having form $<p_0>$ φ ⊂ $<p_1>$ $<p_2>$ … $<p_n>$φ, where p_0 is a procedure name and the p_k are either procedure names, or ag_i 's atomic actions, or tests.

We suppose individual speech acts to take place in the context of predefined conversation protocols that specify communication patterns (Mamdani & Pitt, 2000). Such conversation protocol are also expressed by means of a set of procedure axioms, with the difference that the p_k's are only ag_i primitive speech acts or actions for getting information from another agent engaged in the conversation, that we interpret as *sensing actions* (Patti, 2001).

Each agent has a subjective view of the communication with other agents, for this reason in our formalization each protocol has as many procedural representations as the possible roles in the conversation.

As an example, let us see how to represent a simplified version of the FIPA *yes_no_query* protocol (FIPA, 2000). The protocol has two complementary views, one to be followed for making a query (*yesnoQ*) and one for responding (*yesnoI*):

<yesnoQ (Self, Other, Fluent)> φ ⊂
<queryIf (Self, Other, Fluent) ;
getasw (Self, Other, Fluent)> φ
[getasw (Self, Other, Fluent)] φ ≡
[inform(Other, Self, Fluent) ∪
inform(Other, Self, ¬ Fluent) ∪
refuse(Other, Self, Fluent)] φ

Intuitively, the right hand side of the definition of *getasw* represents all the possible answers expected by agent *Self* from agent *Other* about *Fluent*, in the context of a conversation ruled by the *yesnoQ* protocol.

<yesnoI(Self, Other, Fluent)>φ ⊂
<getstart (Self, Other, Fluent) ;
B^{Self}Fluent?; inform(Self, Other, Fluent)>φ
<yesnoI(Self, Other, Fluent)>φ ⊂
<getstart(Self, Other, Fluent) ;
B^{Self}¬Fluent?; inform(Self, Other, ¬Fluent)>φ
<yesnoI(Self, Other, Fluent)>φ ⊂
<getstart (Self, Other, Fluent) ;
U^{Self} Fluent? ; refuse(Self, Other, Fluent)>φ

The *yesnoI* protocol specifies the behavior of the agent *Self*, that waits a query from *Other*; afterwards, it replies according to its beliefs on the query subject. *getstart* is a sensing action specifying the message(s) the agent can get at that point of the conversation, and is ruled by the axiom: [getstart (Self, Other, Fluent)]φ ≡ [queryIf (Other, Self, Fluent)]φ

Reasoning about conversation protocols

Given a domain description, we can reason about it and formalize the *temporal projection* and the *planning* problem, by means of existential queries of the form:

$$<p_1> <p_2> \ldots <p_m>Fs$$

where *Fs* is a conjunction of belief fluents and each p_k is either be an (atomic or complex) action executed by a^{agi} or an external speech act in the communication kit. By the word *external* we denote a speech act in which our agent plays the role of the receiver.

By checking if a query of the kind above succeeds we can cope with the *planning problem*. In fact this corresponds to answering to the question "is there a sequence of actions, that is an execution of $p_1, \ldots, p_n$ leading to a state where the conjunction of belief fluents *Fs* holds for ag_i?". Such an execution trace is actually a *plan* to bring about *Fs* in the agent mental state. The procedure definition is a sort of plan schema and constrains the search space (procedural planning).

In presence of communication, the planning problem turns into the problem of reasoning about *conversation protocols*, where a *conversation* is a sequence of speech acts. This allows, for instance, an agent to *investigate* the possible changes produced by a specific conversation to its mental state or to understand if a conversation is an instance of a given protocol. In fact, when the *query* contains a communication protocol rather than some procedure for acting on the world, the planning process will find a specific interaction with other agents, which is an instance of the protocol and after which the desired condition *Fs* holds in our agent mental state.

Writing protocols as complex actions is the *key* to planning the interaction. Protocols include actions for getting information from another agent, interpreted as *sensing action;* since agents cannot read each other's mind, they cannot know in advance the messages that they will receive, so, the planning process must take into account all of the possible *alternatives*. If we did not have a description of the interaction schema, it

would be impossible to foresee them. The extracted plan will be *conditional*: for each information acquisition action it will contain as many branches as possible action outcomes. Each path in the resulting tree is a *linear plan* bringing about the desired condition *Fs*.

Let us now return to our leading example. The aim of the personal assistant *pa* is to look for a cinema booking service that satisfies the user's requests. The two web services, *click-ticket* and *all-cinema*, respectively follow the interaction protocols *get_ticket_1* and *get_ticket_2*, the difference between them being that the former permits both to book a ticket to be paid later by cash and to buy it by credit card, while the latter allows only the second option. Let us suppose that, as argued in our proposal, such protocols are part of the DAML-S descriptions of the two web services. Each protocol has two *complementary views*: the view of the *web service* and the view of the *client*, i.e. *pa*. Figures 2 and 3 report the view that *pa* (the client) has of the two protocols. We refer to them as *get_ticket_1*$_C$ and *get_ticket_2*$_C$; Figure 1 reports, as an example, the DyLOG clauses for specifying *get_ticket_1*$_C$. Finally, let us suppose that *pa* knows the credit card number (*cc_number*) of the user but it is requested not to use it; let us see how the agent reasons on the *way* in which conversations will be carried on.

(a) $\langle \text{get\_ticket\_1}_C(Self, WebS, Film)\rangle\varphi \subset$
$\langle \text{yes\_no\_query}_Q(Self, WebS, available(Film))$;
$\mathcal{B}^{Self} available(Film)?$; $\text{get\_info}(Self, WebS, cinema(C))$;
$\text{yes\_no\_query}_I(Self, WebS, pay\_by(credit\_card))$;
$\mathcal{B}^{Self} pay\_by(credit\_card)?$; $\text{inform}(Self, WebS, cc\_number)$;
$\text{get\_info}(Self, WebS, booked(Film))\rangle\varphi$

(b) $\langle \text{get\_ticket\_1}_C(Self, WebS, Film)\rangle\varphi \subset$
$\langle \text{yes\_no\_query}_Q(Self, WebS, available(Film))$;
$\mathcal{B}^{Self} available(Film)?$; $\text{get\_info}(Self, WebS, cinema(C))$;
$\text{yes\_no\_query}_I(Self, WebS, pay\_by(credit\_card))$;
$\neg\mathcal{B}^{Self} pay\_by(credit\_card)?$; $\text{get\_info}(Self, WebS, pay\_by(cash))$;
$\text{get\_info}(Self, WebS, booked(Film))\rangle\varphi$

(c) $\langle \text{get\_ticket\_1}_C(Self, WebS, Film)\rangle\varphi \subset$
$\langle \text{yes\_no\_query}_Q(Self, WebS, available(Film)); \neg\mathcal{B}^{Self} available(Film)?\rangle\varphi$

(d) $[\text{get\_info}(Self, WebS, Fluent)]\varphi \subset [\text{inform}(WebS, Self, Fluent)]\varphi$

Figure 1: DyLOG specification of *get_ticket_l*$_C$

Protocol *get_ticket_1*$_C$ works in the following way: the personal assistant (*Self*) is supposed to begin the interaction. After checking if the requested movie is available in some cinema, it waits (*get_info*) for an information from the provider (*WebS*) about which cinema shows it. Afterwards, the form of payment is defined: clause (a) schematizes the interaction that occurs when the tickets are paid by credit card (Figure 2, first AUML graph); (b) is selected when $\neg B^{Self}$ *pay_by(credit_card)* is contained in *pa* mental state (which is our case, Figure 2 second graph), leading to book a ticket to be paid by cash. In both cases a confirmation of the ticket booking is expected by *pa*. This is implemented by the action *get_info* here used to wait for an information from *WebS* (d). Clause (c) tackles the case in which the movie is not available.

The task of understanding if the web service *click_ticket* allows to satisfy all the user's requirements can be formalized by the query:
$<\text{get\_ticket\_1}_C(pa, click\_ticket, akira)>B^{pa}\neg B^{click\_ticket}$ *cc_number* $\wedge$ B^{pa}*booked*(*akira*)
which amounts to wonder if there is a possible execution of the protocol *get_ticket_1*$_C$ after which a ticket for the desired movie has been booked without informing the web service about the user's credit card number. Such an execution is a specific conversation between *pa* and *click_ticket* about the movie *akira*. Agent *pa* works on the behalf of a user, thus it knows the user's credit card number (B^{pa}*cc_number* is contained in its initial mental state) and the user's desire not to communicate it in the current transaction ($\neg B^{pa}$*pay_by(credit_card)*). It also believes to be an *authority* about the form of payment and about the user's credit card number (i.e. its information about these topics are reliable) and that *click_ticket* is an authority about cinema and tickets: B^{pa}*authority(pa,cc_number)* and B^{pa}*authority(click_ticket, booked(akira))* initially hold. The initial mental state will also include that *pa* believes that no ticket for *akira* has been booked yet, $B^{pa}\neg$*booked(akira)*, and some hypothesis on the interlocutor's mental state, e.g. the belief $B^{pa}\neg B^{click\_ticket}$*cc_number* (the *pa* believes that the web service does not already know the credit card number).

Suppose that a ticket is available; since *pa* mental state contains the belief $\neg B^{pa}$*pay_by(credit_card)*, when it reasons about the protocol execution, it concludes that the test in clause (a) on B^{pa}*pay_by(credit_card)* fails. Then, clause (b) is to be followed, leading *pa* to be informed that it booked a ticket, B^{pa} *booked(akira)*, which is supposed to be paid cash. No communication involves the belief $B^{pa}\neg B^{click\_ticket}$*cc_number*, which *is assumed to persist* from the initial state. Even when the ticket is not available, the interaction ends without consequences on the fluent $B^{pa}\neg B^{click\_ticket}$ *cc_number*. The conclusion of the reasoning process is that the agent finds an execution trace of *get_ticket_1*$_C$, which corresponds to a *personalized conditional dialogue plan* between itself and the provider *click_ticket*, that leads to satisfy the whole user's requirement of not giving the credit card number besides making a ticket reservation. By reasoning on the protocol followed by *all_cinema*, *pa* would instead conclude that there is no possible interaction that allows to satisfy the user's constraints. Therefore, it would ignore this provider.

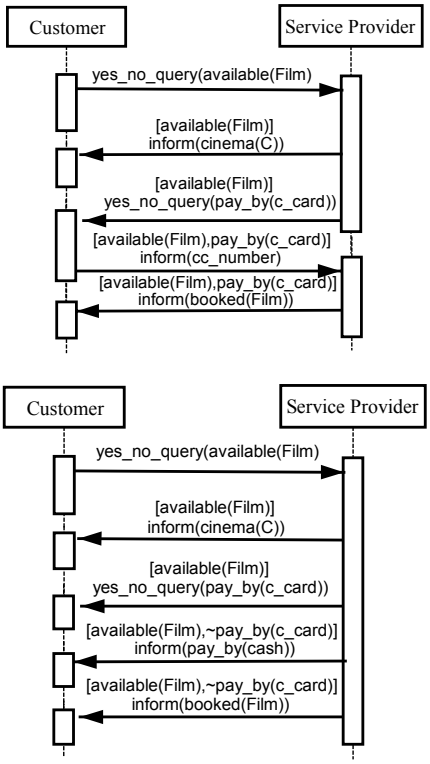

Figure 2: AUML graphs for *get_ticket_1*.

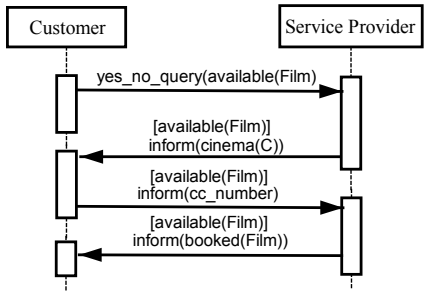

Figure 3: AUML graph for *get_ticket_2*.

Conclusions

We have shown the possible benefits of enriching the (DAML-S) web service description with the explicit high-level representation of the conversation protocol followed by the service. By reasoning about such protocols, agents can personalize the interaction by selecting an interaction course that satisfies user-given requirements. This process can be started before the actual interaction takes place and can be exploited for web service selection or search.

The idea of declaring the conversation protocol derives from our experience in the multi-agent research area, where agents commonly exchange communication protocols before interacting. Our work is set in a modal action logic framework. For agent specification, we used the logic language DyLOG, that includes a communication kit based on FIPA-like speech acts. We took a subjective representation of conversation protocols, where agents make rational assumptions and reason about the interlocutor's mental state.

References

Baldoni, M., Giordano, L., Martelli, A., and Patti, V. (2001). Reasoning about Complex Actions with Incomplete Knowledge: A Modal Approach. *Proc. of ICTCS* (pp. 405-425) LNCS series, vol. 2202. Springer.

Bretier, P., and Sadek, D. (1997). A rational agent as the kernel of a cooperative spoken dialogue system: implementing a logical theory of interaction. *Proc. of Intelligent Agents {III}, ECAI-96 Workshop on Agent Theories, Architectures, and Languages*. LNAI series, vol. 1193. Springer.

Bryson, J., Martin, D., McIlraith, S., and Stein, L. A (2002). Agent-Based Composite Services in DAML-S: The Behavior-Oriented Design of an Intelligent Semantic Web. Springer-Verlag.

Cohen, P.R. and Levesque, H, J. (1990) Rational interaction as the basis for communication. *Intentions in communication*, MIT press.

De Giacomo, G., Lesperance, Y., and Levesque, H. J. (2000). ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence, 121, 109-169.*

Levesque, H. J., Reiter, R., Lesperance, Y., Lin, F., and Scherl, R. B. (1997). GOLOG: A Logic Programming Language for Dynamic Domains. *J. of Logic Programming, 31, 59-83.*

Herzig, A., and Longin, D. (1999). Beliefs Dynamics in Cooperative Dialogues. *Proc. of AMSTELOGUE99.*

Mamdani, A., and Pitt, J. (2000). Communication Protocols in Multi-Agent Systems. Issues in Agent Communication (pp. 160-177). LNCS series, vol. 1916. Springer.

McIlraith, S. and Son, T. (2001). Adapting Golog for Programming the Semantic Web. *Proc. of the 5th Int. Symp. on Logical Formalization of Commonsense Reasoning* (pp. 195-2002)

Patti, V. (2002) Programming Rational Agents: a Modal Approach in a Logic Programming Setting, PhD Thesis, Dept of CS, Università di Torino.

DAML-S, http://www.daml.org/services/daml-s/0.9/, v.0.9, 2003

DAM+OIL, http://www.daml.org/2001/03/daml+oil-index.html, 2001.

FIPA, http://www.fipa.org/specifications/index.html, 2000.

RDF, http://www.w3c.org/TR/1999/REC-rdf-syntax-19990222/, 1999.

OWL, http://www.w3c.org/TR/owl-guide/, 2003.

WSDL, http://www.w3c.org/TR/2003/WD-wsdl12-20030303, v.1.2., 2003.