Reasoning about Communicating Agents in the Semantic Web

Matteo Baldoni[1], Cristina Baroglio[1], Laura Giordano[2], Alberto Martelli[1], and Viviana Patti[1]

[1] Dipartimento di Informatica, Università degli Studi di Torino, Torino, Italy
E-mail: {baldoni,baroglio,mrt,patti}@di.unito.it
[2] Dipartimento di Informatica, Università del Piemonte Orientale, Alessandria, Italy
E-mail: {laura}@mfn.unipmn.it

Abstract. In this article we interpret the Semantic Web and Web Service issues in the framework of multi-agent interoperating systems. We will advocate the application of results achieved in the research area of reasoning about actions and change by showing scenarios and techniques that could be applied.

1 Introduction

The fast diffusion of Internet and the World Wide Web has inspired new paradigms for the development of applications distributed over the network, leading to the concept of "web service". We can consider a web service as a program (software) or a device (hardware) accessible through a network, that can be invoked in an automatic way by programs or other web services. This perspective brings along many interesting issues: how to describe the function executed by a web service in a machine-interpretable way? How to advertise web services? How to choose among the providers of apparently identical services? All these questions and many others demand for the definition of tools and languages for handling semantic information, not only ontologies but also information about the functioning of services.

The standardization organizations developed a series of languages for representing the semantic contents of a resource accessible on the web, from the Resource Description Framework (RDF), to OWL and DAML+OIL for ontology description, to WSDL [6] and DAML-S [7] for web service semantic description. DAML-S inherited from the experience of the research community that studies agent systems and their logic formalizations and draws considerably from the action metaphor: a service can be 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. Such a semantic characterization of the service as an action is described in the so called DAML-S process model and can be used for accomplishing tasks like automatic identification of a service of interest, automatic invocation, automatic composition of services, and so forth.

In this line, one promising direction of research, that we mean to investigate, consists in exploiting results achieved by the community that studies *logic for AI* and, in particular, *reasoning about actions and change*. Indeed, the availability of semantic information about web resources enables the application of reasoning techniques, such as constraint reasoning, non-monotonic reasoning, and temporal reasoning. The main purpose of adopting reasoning techniques is to allow the design of flexible systems, that can adapt to different users and that are open to interact with one another in ways that cannot be fully foreseen at design time and, thus, require such systems to reason for taking autonomous decisions.

Some work in this direction has already been carried on in [19], where, within the context of the DAML-S project [5], the reasoning techniques supported by GOLOG, an agent language based on the situation calculus, are applied to produce composite and customized services. Actually, when a service is described in terms of the function that it executes, with its preconditions and effects, the use of *agents* that can reason about the consequences of its invocation is a natural choice: a rational agent is by definition characterized by a high-level of autonomy, it has an own internal state containing information about the world and about its goals, it can reason about how to behave for fulfilling them, it can react to alterations of the environment. Agents have also *social capabilities* that enable them to interact both with other agents or devices as well as with human beings. This leads us to claim that there is another fundamental behavior level, currently not addressed by the proposers of DAML-S, that should instead be seriously considered and explicitly incorporated in the high-level service description: the *interaction level*, concerning the *communicative behavior* of a web service, and more specifically the *interaction protocol* that it adopts for communicating with its clients or partners. Our proposal is set in a framework in which the web service is an agent that communicates with other agents in a FIPA-like Agent Communication Language (ACL) using predefined protocols. In this context, the communicative behavior of a service can be expressed as a conversation protocol in a logic language, at high (not at network) level. Having a logic specification of the protocol, it is possible to reason about the effects of engaging specific conversations, and to verify properties of the protocol.

In this paper we introduce two related approaches to reasoning about communicative actions by using as a running example a simple scenario of a web service. Both approaches are based on an action theory where communicative actions are formalized with a set of action and precondition laws. In Section 4 we show how the logic language DyLOG can reason about the changes caused by a communicative action to the beliefs of the involved agents, and how this can be exploited for realizing new forms of personalization in web service fruition. In Section 5 we present a more general action theory which allows to specify systems of communicating agents and to verify properties of such systems containing temporal constraints.

2 Communication between agents

Communication and dialogue have intensively been studied in the context of formal theories of agency [8]. In particular, a great deal of attention has been devoted to the definition of *standard agent communication languages* (ACL), such as FIPA and KQML. The crucial issue was to achieve interoperability in open agent systems, characterized by the interaction of heterogeneous agents, where it is fundamental to have a universally shared semantic.

Agent communication languages are complex structures because a communicative act must specify many kinds of information, such as its content and the kind of performative. The definition of formal semantics for individual communicative acts has been one of the major topics of research in this field. Most of the proposals are ultimately based on the philosophical theory of speech acts developed by Austin and Searle in the sixties. Following the basic insight of the speech act theory, communications are not just considered as transmitting information but as *actions* that, instead of modifying the external world, affect the mental states of the involved agents. As a consequence, individual speech act semantic has been given in terms of preconditions and effects on mental attitudes, as it is commonly done with action semantic, and standard techniques for reasoning about change have been exploited for proving conversation properties, planning communication with other agents and answer selection. In this line, many approaches in the literature are based on variant of modal logic, in which mental attitudes, such as beliefs, goals and intentions, as well as communicative acts are represented by modalities [4, 18, 9].

Only recently the attention has been moved to formalize those aspects of communication that are related to the conversational context in which communicative acts occur [21]. The formalization of conversation policies adds a higher semantic level, which improves the interoperability of the various components (often separately developed) and simplifies the verification of compliance to the desired standards. In the area of agent languages based on logic, some examples of definition of protocols for guiding the agent communicative behavior can be found in [26, 1]. By working at the level of protocols, agents can more easily be seen as individuals, developed independently, on different platforms and with different approaches, a very attractive view in the applicative field of web applications and web services. For all these reasons we focus on a semantics of communication that supports the specification and reasoning about single speech acts, as well as the specification and reasoning about speech acts in the context of a conversation protocol.

Instead of referring to a mentalistic approach as described above, some authors have proposed a social approach to agent communication [27, 14]. The mental approach is not well suited for the verification of an "open" multi-agent system, where the history of communications is observable, but the internal states of the single agents may not be observable. In contrast, in the social approach communicative actions affect the "social state" of the system, rather than the internal states of the agents. The social state records the social facts, like the *permissions* and the *commitments* of the agents, which are created and modified in the interactions among them. The dynamics of the system emerges from the interactions of the agents, which must respect these permissions and the commitments (if they are compliant with the protocol). The social approach provides a high level specification of the protocol, and does not require the rigid specification of all the allowed action sequences by means of finite state diagrams.

3 A simple scenario

In this section we will define a simple scenario aimed at showing the advantage of expressing and reasoning about the interaction protocol followed by a web service. Let us consider a software agent (we will refer to it as *pa*) whose task is to crawl the internet for executing specific requests of a given user; indeed, *pa* is a user personal assistant. Let us suppose that *pa* current task is to book a ticket at a cinema where a given movie is shown. In a web service context, it will have to look for a provider of a cinema booking service by consulting a registry, and interact with it accordingly, supplying the requested information. As a further condition, let us imagine that the user requested the personal assistant not to use his credit card number in the upcoming transaction. Suppose also that two cinema

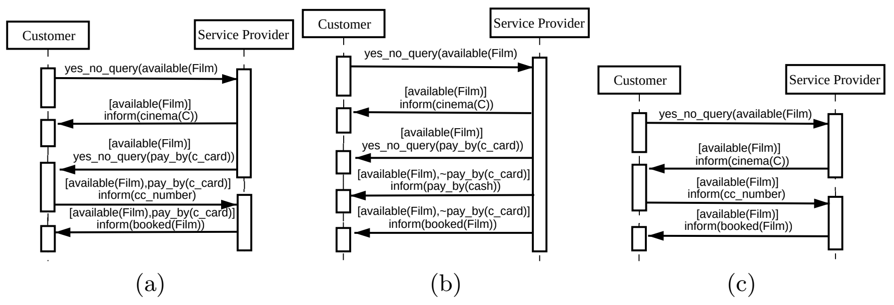

Fig. 1. The three AUML graphs [23] represent the communicative interactions occurring between the customer (*pa*) and the provider; (a) and (b) are followed by *click_ticket*, (c) is followed by *all_cinema*. Formulas among square brackets represent conditions on the execution of the speech act.

booking services are available, called *click_ticket* and *all_cinema* respectively, that apply two different interaction protocols, one permitting both to book a ticket to be paid later by cash (Fig. 1 (a)) and to buy it by credit card (Fig. 1 (b)), the other allowing only ticket purchase by credit card (Fig. 1 (c)). These descriptions would induce a human assistant to choose *click_ticket*, selecting the option to pay cash; this choice can be done because we can reason about the consequences of communicative acts and procedures.

4 Interaction protocols in DyLOG

In the Web service scenario, we are interested in formal languages that support reasoning techniques for proving existential properties of the kind "given a protocol and a set of desiderata, is there a specific conversation, respecting the protocol, that also satisfies the desiderata?". In different words, the scenario demands for some technique that allows the personalization of the interaction. We will show how reasoning methods supported by the agent language DyLOG can be exploited in order to obtain this kind of personalization.

DyLOG is a high-level logic programming language for modelling and programming rational agents [3, 24, 1]. It is based on a modal theory of actions and mental attitudes where modalities are used for representing actions as well as beliefs for modelling the agent's mental state. It accounts both for atomic and complex actions, or procedures. Atomic actions are either world actions, affecting the world, or mental actions, i.e. sensing and communicative actions which only affect the agent beliefs. Complex actions are defined through (possibly recursive) definitions, given by means of Prolog-like clauses and by making use of action operators like sequence, test and non-deterministic choice. The action theory allows to cope with the problem of reasoning about complex actions with incomplete knowledge and in particular to address the temporal projection and planning problem. Intuitively DyLOG allows to specify the behavior of a rational agent that reasons about its own behavior, chooses a course of actions conditioned on its mental state and can use sensors and communication for obtaining fresh knowledge. In this spirit it has already been used with success for agent programming, in implementing a web application where a *virtual tutor* helps students to build personalized study curricula [2], based on the description of courses viewed as actions (an application that bears many analogies with web service process model description and usage).

Let us recall how to specify and reason about communicative behaviors in DyLOG, by focussing on the web service application scenario depicted above. For a detailed description of the overall agent theory see [3, 1].

4.1 Specifying communicative behaviours in DyLOG

Let us start with FIPA-like *speech acts*. Following the mentalistic approach, in DyLOG they are considered atomic actions, described in terms of preconditions and effects on the agent mental state, having form speech_act(ag_i, ag_j, l), where ag_i (sender) and ag_j (receiver) are agents and l (a fluent) is the object of the communication. Since speech acts can be seen as mental actions, affecting both the sender's and the receiver's mental state, we have modelled them by generalizing non-communicative action definitions, so to allow also the representation of the effects of an action executed by some other agent on the current agent mental state, described by a consistent set of *belief fluents*. In fact in our formalization each agent has a twofold, personal representation of the speech act: one is to be used when it is the sender, the other when it is the receiver. Such a representation provides the capability of *reasoning about* conversation effects from the subjective point of view of the agent holding the representation. In the speech act specification that holds when the agent is the sender, the preconditions contain some *sincerity condition* that must hold in its mental state. When it is the receiver, instead, the action is always executable. Let us consider, as an example, a primitive speech act from the standard agent communication language FIPA-ACL, and let us define its semantics within the DyLOG framework: the *inform* speech act (more examples can be found in [24]).

a) $\Box(\mathcal{B}^{Self} l \wedge \mathcal{B}^{Self}\mathcal{U}^{Other} l \supset \langle \mathsf{inform}(Self, Other, l)\rangle \top)$
b) $\Box([\mathsf{inform}(Self, Other, l)]\mathcal{M}^{Self}\mathcal{B}^{Other} l)$
c) $\Box(\mathcal{B}^{Self}\mathcal{B}^{Other} authority(Self, l) \supset [\mathsf{inform}(Self, Other, l)]\mathcal{B}^{Self}\mathcal{B}^{Other} l)$
d) $\Box(\top \supset \langle \mathsf{inform}(Other, Self, l)\rangle \top)$
e) $\Box([\mathsf{inform}(Other, Self, l)]\mathcal{B}^{Self}\mathcal{B}^{Other} l)$
f) $\Box(\mathcal{B}^{Self} authority(Other, l) \supset [\mathsf{inform}(Other, Self, l)]\mathcal{B}^{Self} l)$

Clause (a) states that $Self$ will execute an inform act only if it believes l (we use the modal operator $\mathcal{B}^{ag_i}$ to model the beliefs of agent ag_i) and it believes that the receiver (*Other*) does not know l. It also considers possible that the receiver will adopt its belief (the modal operator $\mathcal{M}^{ag_i}$ is defined as the dual of $\mathcal{B}^{ag_i}$, intuitively $\mathcal{M}^{ag_i}\varphi$ means the ag_i considers φ possible), clause (b), although it cannot be certain about it -autonomy assumption-. If agent $Self$ believes to be considered a trusted *authority* about l by the receiver, it is also confident that *Other* will adopt its belief, clause (c). Instead, when $Self$ is the receiver, the effect of an inform act is that $Self$ will believe that l is believed by the sender (*Other*), clause (e), but $Self$ will adopt l as an own belief only if it thinks that *Other* is a trusted authority, clauses (f).

DyLOG agents can be provided with a set of *conversation protocols*, that build on individual speech acts and specify communication patterns guiding the agent communicative behavior during a protocol-oriented dialogue. Reception of a messages is modelled as a special kind of sensing action, what we call *get message actions*. Indeed from the agent perspective receiving a message corresponds to query for an external input, whose outcome is unpredictable. The main difference w.r.t. normal sensing actions is that *get message actions* are defined by means of speech acts performed by the interlocutor. Protocols are expressed by means of a collection of procedure axioms of the action logic, having form $\langle p_0\rangle\varphi \subset \langle p_1\rangle\langle p_2\rangle \ldots \langle p_n\rangle\varphi$, where p_0 is the procedure name the p_i's can be i's communicative acts or special sensing actions for the reception of message. Each agent has a subjective perception of the communication with other agents, for this reason each protocol has as many procedural representations as the possible roles in the conversation. Let us consider for instance the personal assistant introduced in Section 3, its aim is to look for a cinema booking service that satisfies the user's requests. The web service, *click_ticket* follow the interaction protocol get_ticket_1, that permits both to book a ticket to be paid later by cash and to buy it by credit card. Let us suppose that such protocol is a part of the DAML-S descriptions of *click_ticket*. Since the protocol is meant to allow the interaction of two agents, it has two complementary views: the view of the web service and the view of the client, i.e. *pa*. Intuitively, if one of the two agents plays the part of the sender of a piece of information, the other should play the part of the receiver. In the following we will report the view –written in DyLOG– that *pa* has of the protocol get_ticket_1. We refer to it as $\mathsf{get\_ticket\_1}_C$. Notice that it builds on primitive speech acts as well as on procedures for making a query ($\mathsf{yes\_no\_query}_Q$) or replying to a query ($\mathsf{yes\_no\_query}_I$) according to the FIPA Query Interaction protocol [9].

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

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

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

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

Protocol $\mathsf{get\_ticket\_1}_C$ 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 by the $\mathsf{yes\_no\_query}_Q$ protocol, it should wait for an information (get_info) from the provider ($WebS$) about which cinema shows it. Then the form of payment is defined: (a) defines the interaction that occurs when the tickets are paid by credit card (see Fig. 1(i)); (b) is selected when $\neg\mathcal{B}^{Self} pay\_by(credit\_card)$ is contained in *pa* mental state, leading to book a ticket to be paid by cash (see Fig. 1(ii)). In both cases a confirmation of the ticket booking is returned to the *pa*. Clause (c) tackles the case in which the movie is not available. Clause (d) describes get_info, which is a get_message action.

Given a set $\Pi_{\mathcal{C}}$ of simple action laws defining an agent ag_i's primitive speech acts, a set $\Pi_{\mathcal{S}get}$ of axioms for the reception of messages, and a set $\Pi_{\mathcal{CP}}$, of procedure axioms specifying a collection of conversation protocols, we denote by CKit^{ag_i} (the *communication kit* of a DyLOG agent ag_i), the triple $(\Pi_{\mathcal{C}}, \Pi_{\mathcal{CP}}, \Pi_{\mathcal{S}get})$. CKit^{ag_i} is a part of Π_{ag_i}, i.e. the domain description of the agent ag_i, including also S_0, i.e the initial set of ag_i's belief fluents, and eventually laws and axioms for specifying the agent non communicative behaviors.

4.2 Reasoning about the interaction with a Web service

Given a DyLOG domain description Π_{ag_i} containing a CKit^{ag_i} with the specifications of the interaction protocols and of the relevant speech acts, a *planning* activity can be triggered by *existential queries* of form $\langle p_1\rangle\langle p_2\rangle \ldots \langle p_m\rangle Fs$, where each p_k ($k = 1, \ldots, m$) may be an atomic or complex action (a primitive speech act or an interaction protocol), executed by our agent, or an external[3] speech act,

[3] By the word *external* we denote a speech act in which our agent plays the role of the receiver

that belongs to CKit^{ag_i}. Checking if the query succeeds corresponds to answering to the question "is there an execution of $p_1, \ldots, p_n$ leading to a state where the conjunction of belief fluents Fs holds for agent ag_i?". Such an execution is a plan to bring about Fs. The procedure definition constrains the search space. During the planning process $\mathsf{get\_message}$ actions are treated as sensing actions, whose outcome cannot be predicted before the actual execution: since agents cannot read each other's mind, they cannot know in advance the answers that they will receive. All of the possible alternatives are to be taken into account and, indeed, we can foresee them because of the existence of the protocol. Therefore, the extracted plan will be *conditional*, in the sense that for each $\mathsf{get\_message}$ and for each sensing action it will contain as many branches as possible action outcomes. Each path in the resulting tree is a linear plan that brings about Fs.

The problem that the personal assistant pa has in the Web service scenario outlined above can be naturally turned into a planning problem in presence of communication, as the one treated by DyLOG. In fact the question pa tries to answer is: "is there some possible conversation, that is an instance of the protocol followed by the Web service provider and satisfies all the conditions posed by the user (e.g. at the and of the interaction the service mustn't know the user's credit card number)?". In a way, pa wonders if it is possible to personalize the interaction with its interlocutor so to achieve certain goals. Let us take a DyLOG domain description containing the description of the $\mathsf{get\_ticket\_1}_C$ protocol reported above, suppose that pa knows the credit card number (cc_number) of the user but it is requested not to use it, and consider the query:

$$\langle \mathsf{get\_ticket\_1}_C(pa, click\_ticket, akira)\rangle \mathcal{B}^{pa}\neg\mathcal{B}^{click\_ticket} cc\_number$$

that amounts to determine if there is a conversation between pa and $click\_ticket$ about the movie $akira$, that is an instance of the conversation protocol $\mathsf{get\_ticket\_1}_C$, after which the service does not know the credit card number of the user. Agent pa works on the behalf of a user, thus it knows the user's credit card number ($B^{pa} cc\_number$) and his desire not to use it in the current transaction ($\neg\mathcal{B}^{pa} pay\_by(credit\_card)$). It also believes to be an authority about the form of payment and about the user's credit card number and that $click\_ticket$ is an authority about cinema and tickets. This is represented by the beliefs: $\mathcal{B}^{pa} authority(pa, cc\_number)$ and $\mathcal{B}^{pa} authority(click\_ticket, booked(akira))$. The initial mental state will also contain the fact that pa believes that no ticket for $akira$ has been booked yet, $\mathcal{B}^{pa}\neg booked(akira)$, and some hypothesis on the interlocutor's mental state, e.g. the belief fluent $\mathcal{B}^{pa}\neg\mathcal{B}^{click\_ticket} cc\_number$, meaning that the web service does not already know the credit card number. Suppose, now, that the ticket is available; since pa mental state contains the belief $\neg\mathcal{B}^{pa} pay\_by(credit\_card)$, when it reasons about the protocol execution, the test on $\mathcal{B}^{pa} pay\_by(credit\_card)$? fails. Then clause (b) is to be followed, leading pa to be informed that it booked a ticket, $\mathcal{B}^{pa} booked(akira)$, which is supposed to be paid cash. No communication involves the belief $\mathcal{B}^{pa}\neg\mathcal{B}^{click\_ticket} cc\_number$, which persists from the initial state. Even when the ticket is not available or the movie is not known by the provider, the interaction ends without consequences on the fluent $\mathcal{B}^{pa}\neg\mathcal{B}^{click\_ticket}cc\_number$. The briefly described reasoning process lead to find an execution trace of $\mathsf{get\_ticket\_1}_C$, which corresponds to a *personalized conditional dialogue plan* between pa and the provider $click\_ticket$, always leading to satisfy the user goal of not giving the credit card number.

5 Specifying and verifying systems of communicating agents

DyLOG is a sequential language which can describe the behavior of a single agent and prove *existential* properties, such as finding a sequence of actions achieving some goal. A more general problem is that of modelling systems of communicating agents, so as to be able to prove properties of the whole system. In this section we present a theory for reasoning about actions which allows to describe the behavior of a network of sequential agents which coordinate their activities by performing common actions together[12, 13]. This theory is based on the Product Version of Dynamic Linear Time Temporal Logic (denoted $DLTL^{\otimes}$) [16], a logic which extends LTL, the propositional linear time temporal logic, by strengthening the *until* operator by indexing it with the regular programs of dynamic logic. Regular programs are well suited to model both the agent behaviors and the communication protocols. Moreover, the formulas of the logic are decorated with the names of sequential agents, thus allowing to describe the behavior of a network of sequential agents which coordinate their activities by performing common actions together. Let us first give a quick overview of the logic.

5.1 The logic $DLTL$ and its product version

First we recall the syntax and semantics of DLTL as introduced in [17]. $DLTL$ is an extension of LTL in which the next state modality is labelled by actions and the until operator is indexed by programs in Propositional Dynamic Logic (PDL) [15].

Let Σ be a finite non-empty alphabet whose members are interpreted as actions. Let $Prg(\Sigma)$ be the set of programs on Σ, defined as regular expressions. A set of finite words, representing computation sequences, is associated with each program by the mapping $[[]] : Prg(\Sigma) \rightarrow 2^{\Sigma^*}$.

Let $\mathcal{P} = \{p_1, p_2, \ldots\}$ be a countable set of atomic propositions. The set of formulas of DLTL(Σ) is defined as follows:

$$\text{DLTL}(\Sigma) ::= p \mid \neg\alpha \mid \alpha \vee \beta \mid \alpha\mathcal{U}^{\pi}\beta$$

where $p \in \mathcal{P}$ and α, β range over DLTL(Σ), and π ranges over $Prg(\Sigma)$.

A model of DLTL(Σ) is a pair $M = (\sigma, V)$ where σ is an infinite sequence of actions and V is a valuation function. Given a model $M = (\sigma, V)$, a finite word $\tau \in prf(\sigma)$ (a finite prefix of σ), and a formula α, the satisfiability of a formula α at τ in M, written $M, \tau \models \alpha$, is defined as usual for the classical connectives. Moreover:

- $M, \tau \models \alpha \mathcal{U}^\pi \beta$ iff there exists $\tau' \in [[\pi]]$ such that $\tau\tau' \in prf(\sigma)$ and $M, \tau\tau' \models \beta$. Moreover, for every τ'' such that $\varepsilon \leq \tau'' < \tau'$, $M, \tau\tau'' \models \alpha$.

The formula $\alpha \mathcal{U}^\pi \beta$ is true at τ if "α until β" is true on a finite stretch of behaviour which is a computation sequence of the program π.

The derived modalities $\langle\pi\rangle$ and $[\pi]$ can be defined as follows: $\langle\pi\rangle\alpha \equiv \top \mathcal{U}^\pi \alpha$ and $[\pi]\alpha \equiv \neg\langle\pi\rangle\neg\alpha$. Furthermore $\bigcirc$ (next), $\Diamond$ and $\Box$ of LTL can be defined as follows: $\bigcirc\alpha \equiv \bigvee_{a \in \Sigma} \langle a \rangle \alpha$, $\quad \Diamond\alpha \equiv \top \mathcal{U}^{\Sigma^*} \alpha$, $\quad \Box \equiv \neg\Diamond\neg\alpha$.

Let us now recall the definition of $DLTL^\otimes$ from [16]. Let $Loc = \{1, \ldots, K\}$ be a set of *locations*, the names of the agents. A *distributed alphabet* $\tilde{\Sigma} = \{\Sigma_i\}_{i=1}^K$ is a family of (possibly non-disjoint) alphabets, where Σ_i is the set of actions which require the participation of agent i. If an action a belongs to Σ_i and to Σ_j, the two agents i and j will synchronize on this action. Let $\Sigma = \bigcup_{i=1}^K \Sigma_i$.

Atomic propositions are introduced in a local fashion, by introducing a non-empty set of atomic propositions $\mathcal{P}$. For each proposition $p \in \mathcal{P}$ and agent $i \in Loc$, p_i represents the "local" view of the proposition p at i, and is evaluated in the local state of agent i.

The formulas in $DLTL^\otimes(\tilde{\Sigma})$ are boolean combinations of formulas with the main constraint that no nesting of modalities $\mathcal{U}_i$ and $\mathcal{U}_j$ (for $i \neq j$) is allowed. A model of $DLTL^\otimes(\tilde{\Sigma})$ is a pair $M = (\sigma, V)$, where $\sigma \in \Sigma^\infty$ and $V = \{V_i\}_{i=1}^K$ is a family of functions V_i, where each V_i is the valuation function for agent i. The satisfiability of formulas in a model is defined as in DLTL, except that propositions are evaluated locally and the sequence of actions σ is projected on the alphabet of local actions of each agent.

5.2 Action theories and protocols

Given a set of communicating agents, each agent participating in an action execution has its own local description of the action determining the effects on its local state. The global state of the system can be regarded as a set of local states, one for each agent i. The *action laws* and *causal laws* of agent i describe how the local state of i changes when an action $a \in \Sigma_i$ is executed. The underlying model of communication is the synchronous one: the communication action $comm\_act(i, j, m)$ (message m is sent by agent i to agent j) is shared by agent i (the sender) and agent j (the receiver) and executed synchronously by them. Their local states are updated separately, according to their action specification. Though, for simplicity, we adopt the synchronous model, an asynchronous model can be easily obtained by explicitly modelling the communication channels among the agents as distinct locations.

A *protocol* defines the meaning of communicative actions involved in the conversation. In particular, by adopting a social approach, the protocol describes the effects of each action on the social state of the system. These effects, including the creation of new commitments, can be expressed by means of *action laws.* Moreover, the protocol establishes a set of preconditions on the executability of actions (*permissions*), which can be expressed by means of *precondition laws.*

Each agent has a local view of the social state and the execution of a communicative action can in general affect both the state of the sender and the state of the receiver. In particular, all agents can see the effects on the social state of the actions to which they participate.

For instance, in the example of Section 3 there are two agents, the personal assistant *pa* and the web service *ws* providing ticket booking. The conversation protocol for the two agents will be given through a set of action laws and constraints in the form of permissions or commitments. Since our theory does not allow to express a global states, the protocol will be projected on the local states of the participating agents. Observe that, since the two agents participate in all communicative actions, they have the same local view of the social state, and of the action laws and constraints of the protocol.

Let us assume that *pa* is the sender of the following actions[4] *queryIf(pa, ws, available(Film)), askBooking(pa, ws, Cinema), give_cc(N)*, whereas the actions whose sender is *ws* are *inform(ws, pa, at(Film, Cinema)), inform(ws, pa, ¬ available(Film)), makeBooking(ws, pa, Cinema), sendTicket(ws,pa)*. The effects of actions will be described by action laws such as (where $k = pa, ws$):

$$\Box_k([queryIf(pa, ws, available(Film))]_k asked(Film)$$
$$\Box_k([makeBooking(ws, pa, Cinema)]_k booked(Cinema)$$

where $asked(Film)$ and $booked(Cinema)$ are fluents of the social state.

Commitments can be effects of actions and will be represented by special fluents. They can be base-level commitments, of the form $C(ag_1, ag_2, action)$ (agent ag_1 is committed to agent ag_2 to execute the *action*), or they can be conditional commitments of the form $CC(ag_1, ag_2, p, action)$ (agent ag_1 is committed to agent ag_2 to execute *action*, if the condition p is brought about).

For instance, when the web service finds a cinema, it commits to make the booking, if the customer asks it. Furthermore it commits to send a ticket if the customer gives its credit card number.

$$\begin{aligned}&\Box_k([inform(ws, pa, at(Film, Cinema))]_k\\ &\qquad CC(ws, pa, askedBooking(Cinema), makeBooking(ws, pa, Cinema))\\ &\qquad \wedge CC(ws, pa, cc\_given, sendTicket(ws, pa))\end{aligned}$$

Some reasoning rules have to be defined for cancelling commitments when they have been fulfilled and for dealing with conditional commitments. For instance we can have the law (where $k = i, j$):

$$\Box_k((CC(i, j, p, a) \wedge \bigcirc_k p) \rightarrow \bigcirc_k(C(i, j, a) \wedge \neg CC(i, j, p, a)))$$

saying that a conditional commitment $CC(i, j, p, a)$ becomes a base-level commitment $C(i, j, a)$ when the condition p has been brought about. This law is a *causal law*.

The protocol can specify constraints (permissions) on the execution of actions by giving precondition to the actions. For instance *ws* will not send the ticket before the credit card number has been given:

[4] This formulation does not correspond exactly to the diagram in Fig. 1.

$\Box_k(\neg cc\_given \rightarrow [sendTicket(ws, pa)]_k \bot)$

meaning that $sendTicket(ws, pa)$ cannot be executed in those states in which $\neg cc\_given$ holds, i.e. cc_given is a precondition of the action.

An agent i *satisfies its commitments* when, for all commitments $C(i, j, a)$ in which agent i is the debtor, the formula:

$\Box_i(C(i, j, a) \rightarrow \Diamond_i \langle a \rangle_i \top)$

holds. Such a formula says that, when an agent is committed to execute action a, then it must eventually execute a[5].

Note that a protocol specified in this way is less rigid that the one given in Fig. 1, and can have different executions satisfying the action laws, preconditions and commitments. For instance the customer can leave the conversation before asking booking (the web service will have no base-level commitments to fulfill), or after asking booking and receiving confirmation, but before giving the credit card number (in this case the web service will only be committed to make booking, but not to send the ticket).

5.3 Reasoning about protocols

Given a protocol, we denote with $\mathcal{D}_i$ the *domain description* of agent i, i.e. its action laws and causal laws[6], with $Perm_i$ the set of precondition laws of the actions whose sender is i, and with Com_i the set of all temporal formulas, as the one above, describing the satisfaction of the commitments of agent i.

If we do not know the behavior of any agent, we can only reason on the protocol by proving some properties of it, by assuming that all agents respect their permissions and commitments. This can be formalized as a validity check of the formula:

$$\bigwedge_j (\mathcal{D}_i \wedge Perm_j \wedge Com_j) \rightarrow p$$

where j ranges over all agents.

Following [29] we might also extract from the protocol a plan, that is an execution of the protocol, satisfying some given properties. Planning can be formulated in our theory as a satisfiability problem., i.e. as the problem of finding a model having the plan as a finite prefix. Assume instead that we know the behavior of some agents. For instance we are given a program (regular expression) π which describes the behavior of the web service. In this case we would like to verify that *ws* always satisfies its social fact, i.e. its permissions and commitments. Since we don't know anything about the behavior of *pa*, we can only assume that it respects its social facts.

[5] Here we assume that an agent cannot change its mind about commitments. However the language allows to define actions for manipulating commitments, for instance for cancelling them, as in [29]

[6] Actually $\mathcal{D}_i$ must also model the frame problem. To deal with it we make use [12] of a completion construction which, given a domain description, introduces frame axioms for all the fluents in the style of the successor state axioms introduced by Reiter [25] in the context of the situation calculus.

If $Prog_{ws}$ is the domain description of the behavior of ws, the following formula:

$$(\mathcal{D}_{ws} \wedge Prog_{ws} \wedge \mathcal{D}_{pa} \wedge Perm_{pa} \wedge Com_{pa}) \rightarrow (Perm_{ws} \wedge Com_{ps})$$

is valid if in all the executions of the system, in which agent ws respects its specification $Prog_{ws}$, and pa (whose internal program is unknown) respects the protocol specification (including its permissions and commitments), the permissions and commitment of agent ws are also satisfied. In general it is possible to prove that an agent is compliant (respects its "social facts") under the assumption that all other agents in the protocol are compliant.

The above verification and satisfiability problems can be solved by extending the standard approach for verification and model-checking of Linear Time Temporal Logic, based on the use of Büchi automata. As described in [17], the satisfiability problem for DLTL can be solved in deterministic exponential time, as for LTL, by constructing for each formula $\alpha \in DLTL(\Sigma)$ a Büchi automaton $\mathcal{B}_\alpha$ such that the language of ω-words accepted by $\mathcal{B}_\alpha$ is non-empty if and only if α is satisfiable. This result has been extended in [16] to $DLTL^{\otimes}$. The verification of a formula $\alpha \rightarrow \beta$ can be carried out by constructing the two Büchi automata for α and the *negation* of β. If the two automata have a common execution sequence, this sequence provides a counterexample for $\alpha \rightarrow \beta$. Thus $\alpha \rightarrow \beta$ is valid if the language accepted by the product of the two automata is empty. The similarities between the verification approach for LTL and that for $DLTL^{\otimes}$ suggest the possibility of using techniques and tools which have been developed for LTL. For instance, it is possible to extend to $DLTL^{\otimes}$ the efficient tableau-based algorithm of [10] for constructing the automaton on the fly.

6 Conclusions

In this paper we have presented various approaches to reasoning about conversation protocols within the framework of logic-based agent languages. We have shown that a theory of communicative actions can be formulated in the DyLOG logical framework, so as to allow the modelling of software agents that can interact with one another by a speech act based communication mechanism. This framework allows an agent to reason about conversation protocols with other agents. We have also presented an action theory based on the logic $DLTL^{\otimes}$ which provides a unified framework for specifying and verifying systems of communicating agents: Programs are expressed as regular expressions, (communicative) actions can be specified by means of action and precondition laws, properties of social facts can be specified by means of causal laws and constraints, and temporal properties can be expressed by means of the *until* operator.

A related approach is that of ConGolog [11], an extended version of the language Golog, that incorporates a rich account of concurrency, in which complex actions (plans) can be formalized as Algol-like programs in the situation calculus. A substantial difference with ConGolog, apart from the different logical foundation, is that here we model agents with their own local states, while in

Congolog the agents share a common global environment and all the properties are referred to a global state.

Other related proposals for the specification and verification of systems of communicating agents, based on a mentalistic approach, are presented [20] and [28]. The goal of [20] is to extend model checking to make it applicable to multi-agent systems, where agents have BDI attitudes. This is achieved by using a new logic which is the composition of two logics, one formalizing temporal evolution and the other formalizing BDI attitudes. In [28] agents are written in MABLE, an imperative programming language, and have a mental state. MABLE systems may be augmented by the addition of formal claims about the system, expressed using a quantified, linear time temporal BDI logic. [29] presents a social approach based on event calculus to protocol specification and execution. A different approach to specification and verification of web services is presented in [22], which shows how to encode in a Petri Net formalism a service description given in DAML-S, providing decision procedures for web service simulation, verification and composition. Guerin's thesis [14] defines an agent communication framework which gives agent communication a grounded declarative semantics, and defines different languages for agent programming, for specifying agent communication and social facts, and for expressing temporal properties.

References

1. M. Baldoni, C. Baroglio, A. Martelli, and V. Patti. Reasoning about self and others: communicating agents in a modal action logic. In C. Blundo and C. Laneve, editors, *Proc. of ICTCS'2003*, volume 2841 of *LNCS*. Springer, 2003.
2. M. Baldoni, C. Baroglio, and V. Patti. Applying logic inference techniques for gaining flexibility and adaptivity in tutoring systems. In C. Stephanidis, editor, *Proc. of the HCII 2003*, Crete, 2003. Lawrence Erlbaum Associates.
3. M. Baldoni, L. Giordano, A. Martelli, and V. Patti. Reasoning about Complex Actions with Incomplete Knowledge: A Modal Approach. In *Proc. of ICTCS'2001*, volume 2202 of *LNCS*, pages 405–425. Springer, 2001.
4. P. Bretier and D. Sadek. A rational agent as the kernel of a cooperative spoken dialogue system: implementing a logical theory of interaction. In *Intelligent Agents III*, volume 1193 of *LNAI*. Springer-Verlag, 1997.
5. J. Bryson, D. Martin, S. McIlraith, and L. A. Stein. Agent-based composite services in DAML-S: The behavior-oriented design of an intelligent semantic web, 2002.
6. R. Chinnici, M. Gudgin, J. J. Moreau, and S. Weerawarana. Web Services Description Language (WSDL) version 1.2, 2003. Working Draft.
7. The DAML-S coalition. DAML-S: Web service description for the semantic web. In *the 1st Int. Semantic Web Conference (ISWC)*, Sardinia, Italy, 2002.
8. F. Dignum and M. Greaves. Issues in agent communication. In *Issues in Agent Communication*, volume 1916 of *LNCS*, pages 1–16. Springer, 2000.
9. FIPA. FIPA 2000. Technical report, FIPA (Foundation for Intelligent Physical Agents), November 2000.
10. R. Gerth, D. Peled, M.Y.Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proc. 15th Work. Protocol Specification, Testing and Verification*, Warsaw, 1995. North Holland.

11. G. De Giacomo, Y. Lespèrance, and H.J. Levesque. Congolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121:109–169, 2000.
12. L. Giordano, A. Martelli, and C. Schwind. Reasoning about actions in dynamic linear time temporal logic. In *J. of the IGPL*, Vol. 9, No. 2, pp. 289-303, March 2001.
13. L. Giordano, A. Martelli, and C. Schwind. Specifying and Verifying Systems of Communicating Agents in a Temporal Action Logic. In A. Cappelli and F. Turini, editors, *Proc. of the 8th Conf. of AI*IA*, volume 2829 of *LNAI*. Springer, 2003.
14. F. Guerin. *Specifying Agent Communication Languages*. Phd thesis, Imperial College, London, April 2002.
15. D. Harel. First order dynamic logic. In D. Gabbay and F. Guenthner, editors, *Extensions of Classical Logic, Handbook of Philosophical Logic*, volume II, pages 497–604. D. Reidel, 1984.
16. J.G. Henriksen and P.S. Thiagarajan. A product version of dynamic linear time temporal logic. In *Proc. of CONCUR'97*, 1997.
17. J.G. Henriksen and P.S. Thiagarajan. Dynamic linear time temporal logic. *Annals of Pure and Applied logic*, 96(1-3):187–207, 1999.
18. A. Herzig and D. Longin. Beliefs dynamics in cooperative dialogues. In *Proc. of AMSTELOGUE 99*, 1999.
19. S.A. Mc Ilraith, T.C Son, and H. Zeng. Semantic web services. *IEEE Intelligent Systems*, pages 46–53, 2001.
20. F. Giunchiglia M. Benerecetti and L. Serafini. Model checking multiagent systems. *Journal of Logic and Computation*, 8(3):401–423, 1998.
21. A. Mamdani and J. Pitt. Communication protocols in multi-agent systems: A development method and reference architecture. In *Issues in Agent Communication*, volume 1916 of *LNCS*, pages 160–177. Springer, 2000.
22. S. Narayanan and S. A. McIlraith. Simulation, verification and automated composition of web services. In *Proc. of the Eleventh International World Wide Web Conference, WWW-11*, May 2002.
23. James H. Odell, H. Van Dyke Parunak, and Bernhard Bauer. Representing agent interaction protocols in UML. In *Agent-Oriented Software Engineering*, pages 121–140. Springer, 2001. http://www.fipa.org/docs/input/f-in-00077/.
24. V. Patti. *Programming Rational Agents: a Modal Approach in a Logic Programming Setting*. PhD thesis, Dipartimento di Informatica, Università degli Studi di Torino, Italy, 2002. Available at `http://www.di.unito.it/~patti/`.
25. R. Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. *Artificial Intelligence and Mathematical Theory of Computation*, pages 359–380, 1991.
26. F. Sadri, F. Toni, and P. Torroni. Dialogues for Negotiation: Agent Varieties and Dialogue Sequences. In *Proc. of ATAL'01*, Seattle, WA, 2001.
27. M. P. Singh. A social semantics for agent communication languages. In *Proc. of IJCAI-98 Workshop on Agent Communication Languages*, Berlin, 2000. Springer.
28. M. Wooldridge, M. Fisher, M.P. Huget, and S. Parsons. Model checking multi-agent systems with mable. In *Proc. of AAMAS'02*, pages 952–959, Bologna, 2002.
29. P. Yolum and M.P. Singh. Flexible protocol specification and execution: Applying event calculus planning using commitments. In *Proc. of AAMAS'02*, pages 527–534, Bologna, Italy, 2002.