Reasoning about Conversation Protocols in a Logic-Based Agent Language

Matteo Baldoni, Cristina Baroglio, Alberto Martelli, and Viviana Patti

Dipartimento di Informatica, Università degli Studi di Torino
C.so Svizzera 185, I-10149 Torino, Italy
{baldoni,baroglio,mrt,patti}@di.unito.it

Abstract. We present an approach to reasoning about conversations within the framework of a logic-based agent language. Our agent theory is based on a modal logic of actions and beliefs and permits the representation of communicative acts and conversation protocols, allowing agents to reason about them before their execution. The work is framed in a world wide web context, in which we show how reasoning about the interaction with web service providers can be exploited for personalizing the service fruition.

1 Introduction

In this work we face the problem of allowing a rational agent, which is situated in a multi-agent framework, to reason about the conversations that it could carry on with other agents as a means of pursuing some goal of its own interest.

The capability of reasoning about conversations is, indeed, extremely useful in different application domains. Besides the often mentioned robotic applications, where, for instance, groups of agents must trade how to cooperate in order to perform some task, also the web world offers many interesting scenarios. Let us consider, as an example, a software agent, which is a user personal assistant. The task that it performs is to search the web in order to find *web services*, according to the user's specification. The scientific literature offers some standard languages for describing web services (such as DAML-S [5] and WSDL [4]) and there already are systems for finding web services that perform tasks of interest (e.g. the UDDI framework, where the WSDL standard is exploited). Currently, these languages base the description of a service on the lists of the inputs and of the outputs respectively required and returned by it. Let us suppose that you asked the agent to "book two tickets at a cinema where they show Akira" and that it should "not give the credit card number". The agent should match the information that you supplied to the inputs and outputs of the available services and select one of them that satisfies the requests. If we take a closer look to the two conditions, however, we can see that while the first identifies the *kind* of service and the *goal* to achieve, the second condition constrains the way in which the *interaction* between the service provider and the personal assistant should be carried on. The personal assistant, however, has no way for knowing

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 300–311, 2003.

how its interlocutor will carry on the interaction because the service description does not contain any behavioral information, as noticed by McIlraith et al. [3]. In our work we focus on a specific kind of behavioral information, the *interaction protocol* followed by the provider. Whether or not a provider is a software agent, for communicating with its clients, it will follow some possibly non-deterministic protocol, aimed at getting/supplying all the necessary information. The cinema booking service and our personal assistant can, then, be seen as the two actors of a conversation, determined by the protocol imposed by the provider. If the personal assistant could *reason about* the possible interactions outcoming from such a conversation protocol, it could either verify if the interaction may be *personalized* by following an execution path that satisfies all the user's requirements, or, when this is not possible, it could search for another provider.

We faced the problem of describing and reasoning about conversation protocols in an *agent logic programming* setting by extending the modal action and belief framework of the language DyLOG, introduced in [2], so to deal also with *communicative behaviors*. In particular, we model how an agent can reason about the interaction that it is going to enact with another agent, with the aim of proving if there is a possible execution of the communication protocol, after which a set of beliefs of interest (goal) will be true in the agent mental state. Such a form of reasoning implies making assumptions about the mental state of the *other* agent, the one ours wishes to interact with. We considered a conversation protocol as (possibly non-deterministic) procedure that specifies the complex communicative behavior of a *single* agent, based upon simpler FIPA-like communicative acts, where the agent can either play the part of the sender or of the receiver of a message.

2 The Agent Language

DyLOG, introduced in [2], is a logic programming language for specifying and reasoning about the behavior of rational agents. It is based on a modal logic for reasoning about *actions* and *beliefs* and permits to define both complex actions and sensing actions for information gathering. Agents programmed in DyLOG can choose a course of actions, conditioned by their beliefs about the world and about other agents, and can use sensors for acquiring knowledge. Originally, DyLOG did not encompass communicative acts, but since in a *multi-agent environment* conversations with other agents can be a means of pursuing some goal of interest, we aim at integrating in the language a *communication kit*, that includes both primitive speech acts and pre-defined conversation protocols, that the agents can use to communicate with one another. Our action logic accounts both for atomic and complex actions, or procedures. Atomic actions are either world actions, which affect the world, or mental actions, i.e. sensing or communicative actions which only affect the mental state of an agent by modifying its beliefs. The set of atomic action consists of the set $\mathcal{A}$ of the world actions, the set $\mathcal{C}$ of communicative acts, and the set $\mathcal{S}$ of sensing actions. For each atomic action a and agent ag_i we introduce the modalities $[a^{ag_i}]$ and $\langle a^{ag_i} \rangle$. $[a^{ag_i}]\alpha$ means that α

holds after every execution of action a by agent ag_i; $\langle a^{ag_i} \rangle \alpha$ means that there is a possible execution of a (by ag_i) after which α holds. For each atomic action a in $\mathcal{A} \cup \mathcal{C}$ we also introduce a modality $Done(a^{ag_i})$ for expressing that a has been executed. $Done(a^{ag_i})\alpha$ is read "a has been executed by ag_i; before its execution, α was true"[1]. The modality $\Box$ denotes those formulas that hold in all possible agent mental states. Our formalization of complex actions draws considerably from dynamic logic for the definition of action operators like sequence, test and non-deterministic choice. However, differently than [10], we refer to a *Prolog-like* paradigm: procedures are defined by means of (possibly recursive) Prolog-like clauses. For each procedure p, the language contains also the modalities $[p]$ and $\langle p \rangle$ (universal and existential modalities respectively).

The mental state of an agent is described in terms of a consistent set of *belief fluents*. We enriched the belief state of a DyLOG agent by allowing also nested beliefs, for representing what other agents believe and reasoning on how they can be affected by communicative actions. We use the modal operator $\mathcal{B}^{ag_i}$ to model the beliefs of agent ag_i. The modality $\mathcal{M}^{ag_i}$ is defined as the dual of $\mathcal{B}^{ag_i}$ ($\mathcal{M}^{ag_i}\varphi \equiv \neg \mathcal{B}^{ag_i} \neg \varphi$). Intuitively $\mathcal{M}^{ag_i}\varphi$ means that ag_i consider φ possible.

All the modalities of the language are normal, $\Box$ is reflexive and transitive; its interaction with action modalities is ruled by $\Box\varphi \supset [a^{ag_i}]\varphi$. The epistemic modality $\mathcal{B}^{ag_i}$ is serial, transitive and euclidean. The interaction of $Done(a^{ag_i})$ with other modalities is ruled by: $\varphi \supset [a^{ag_i}]Done(a^{ag_i})\varphi$ and $Done(a^{ag_j})\varphi \supset \mathcal{B}^{ag_i} Done(a^{ag_j})\varphi$ (awareness), with $ag_i = ag_j$ when $a^{ag_i} \notin \mathcal{C}$. A non-monotonic solution to the persistency problem is given, which consists in maximizing persistency assumptions about fluents after the execution of action sequences, in the context of an abductive characterization. Such a solution is a generalization of the one in [2] for dealing with persistency of *nested* beliefs [13].

Beliefs. Our agents are modelled as individuals, each having its *subjective* point of view; we do not model the real world, the only relevant characterization of the states concerning the internal dynamics of each agent: its *mental state*, modelled as a set of beliefs. A *belief state* contains what ag_i (dis)believes about the world and about the other agents. Agents do not control *other* agents beliefs: when an agent ag_j informs ag_i about l, it cannot be sure that ag_i will believe l; this will happen only if ag_i believes ag_j to be an authority about l. We denote by l either a fluent literal (f or $\neg f$), or a *done fluent*, i.e. a modal atom $Done(a^{ag_i})\top$ or its negation. The belief state includes both a (possibly negated) epistemic fluent $\mathcal{B}^{ag_i} l$ for each literal, and, for each agent ag_j, a (possibly negated) epistemic fluent for each of its beliefs ($\mathcal{B}^{ag_i}\mathcal{B}^{ag_j} l$); ag_j may also be equal to ag_i. To guarantee the *consistency* of the belief state, we impose the modality $\mathcal{B}^{ag_i}$ to be *serial*. In the single agent case [2], seriality guarantees that it is not possible that both $\mathcal{B}^{ag_i} l$ and $\mathcal{B}^{ag_i} \neg l$ hold in the same state. In a multi-agent setting, when we deal with nested beliefs, seriality also guarantees that no agent ascribes inconsistent beliefs to other agents; so for each couple of agents ag_i, ag_j and for each belief formula $\mathcal{B}^{ag_i} l$: neither it is possible that $\mathcal{B}^{ag_j}\mathcal{B}^{ag_i} l$ and $\mathcal{B}^{ag_j}\mathcal{B}^{ag_i} \neg l$ nor that $\mathcal{B}^{ag_j}\mathcal{B}^{ag_i} l$

[1] $Done(a^{ag_i})\top$ is read "the action a has been executed by agent ag_i".

and $\mathcal{B}^{ag_j}\neg\mathcal{B}^{ag_i}l$ hold in the same state. From the seriality of the $\mathcal{B}^{ag_i}$ operators, the general formula schema for the rank 2 beliefs $\mathcal{B}^{ag_i}\mathcal{B}^{ag_j}\neg\varphi \supset \neg\mathcal{B}^{ag_i}\mathcal{B}^{ag_j}\varphi$ holds in our logic for any two agents ag_i and ag_j. This property guarantees that when an inconsistency arises "locally" in the beliefs ascribed from ag_i to other agents, then the beliefs of ag_i itself will be inconsistent. In essence a belief state is a complete and consistent set of rank 1 and 2 epistemic fluents. It provides, for each agent ag_i, a three-valued interpretation of all the possible belief arguments, i.e. fluent literals, done fluents and epistemic fluents of rank 1: each belief argument L is either *true*, *false*, or *undefined* when both $\neg\mathcal{B}^{ag_i}L$ and $\neg\mathcal{B}^{ag_i}\neg L$ hold. In the last case we use $\mathcal{U}^{ag_i}L$ for expressing the ignorance of ag_i about L.

Behavior. In the line of [2] the *behavior* of an agent ag_i can be specified by a domain description, which includes, besides a specification of the agent current beliefs: (1) action and precondition laws for describing the *atomic world actions* in terms of their preconditions and effects on the executor's mental state; (2) sensing axioms for describing *atomic sensing actions*; (3) procedure axioms for describing *complex behaviors*.

World actions are described by their preconditions and effects on the actor's mental state; they trigger a revision process on the actor's beliefs. Formally, *action laws* describe the conditional effects on ag_i's belief state of an atomic action $a \in \mathcal{A}$, executed by ag_i itself. They have the form:

$$\Box(\mathcal{B}^{ag_i}L_1 \wedge \ldots \wedge \mathcal{B}^{ag_i}L_n \supset [a^{ag_i}]\mathcal{B}^{ag_i}L_0) \quad (1)$$

$$\Box(\mathcal{M}^{ag_i}L_1 \wedge \ldots \wedge \mathcal{M}^{ag_i}L_n \supset [a^{ag_i}]\mathcal{M}^{ag_i}L_0) \quad (2)$$

Law (1) states that if ag_i believes the preconditions to an action a in a certain epistemic state, after a execution, ag_i will also believe the action's effects. (2) states that when the preconditions of a are unknown to ag_i, after the execution of a, ag_i will consider unknown also its effects[2].

Precondition laws specify mental conditions that make an action in $\mathcal{A} \cup \mathcal{C}$ executable in a state. They have form:

$$\Box(\mathcal{B}^{ag_i}L_1 \wedge \ldots \wedge \mathcal{B}^{ag_i}L_n \supset \langle a^{ag_i}\rangle\top) \quad (3)$$

The meaning is trivial: ag_i can execute a when the precondition fluents of a are in ag_i's belief state.

Sensing Actions produce knowledge about fluents; they are defined as non-deterministic actions, with unpredictable outcome, formally modelled by a set of *sensing axioms*. If we associate to each sensing action s a set $dom(s)$ of literals (domain), when ag_i executes s, it will know which of such literals is true:

$$[s]\varphi \equiv [\bigcup_{l \in dom(s)} s^{\mathcal{B}^{ag_i}l}]\varphi \quad (4)$$

$\cup$ is the choice operator of dynamic logic and $s^{\mathcal{B}^{ag_i}l}$, for each $l \in dom(s)$, is an *ad hoc* primitive action, that probes one of the possible outcomes of the sensing.

[2] Laws of form (2) allow actions with non-deterministic effects, that may cause a *loss* of knowledge, to be specified.

Notice that the binary sensing action on a literal l is a special case of sensing where the associated finite set is $\{l, \neg l\}$.

Complex actions We specify agent complex behaviors by means of *procedure definitions*, built upon other actions. Formally, a complex action is defined by means of a collection of *inclusion axiom schema* of our modal logic, of form:

$$\langle p_0 \rangle \varphi \subset \langle p_1; p_2; \ldots; p_m \rangle \varphi \tag{5}$$

p_0 is a procedure name and the p_i's $(i = 1, \ldots, m)$ are either procedure names, atomic actions, or test actions; the operator ";" is the sequencing operator of dynamic logic. Procedure definitions may be recursive and procedure clauses can be executed in a goal directed way, similarly to standard logic programs.

3 Communication

The integration of a *communication theory* in the general agent theory came straightforwardly by adding further axioms and laws to ag_i's domain description. In this section we enrich the specification of an agent behavior with a *communication kit*, that includes a predefined set of *primitive speech acts*, a set of special *get message actions*, and a set of *conversation protocols*.

Speech Acts Communication primitives are treated as atomic actions and described in terms of preconditions and effects on the agent mental state. Each communicative primitive action has the form $\mathsf{speech\_act}(sender, receiver, l)$, where *sender* and *receiver* are agents and l is the propositional content of the speech act[3]. As a difference with world actions, in the case of communication primitives, the sender aims at modifying the mental state of the receiver, possibly inducing a cooperative behavior. Such actions can be seen as special mental actions, affecting both the sender's and the receiver's mental state. In our model we focused on the *internal representation*, that each agent has of each speech act, by specifying ag_i's belief changes both when ag_i is the sender and when it is the receiver. Such a representation is necessary for providing the capability of *reasoning about* conversation effects. Hence, speech acts are modelled by generalizing the action and precondition laws of form (1), (2), and (3), so to allow the representation of the effects of communications performed by other agents on our agent mental state. Then, our speech act specification is twofold: one definition holds when the agent is the executor (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 it is the receiver, the action is considered *always* executable. Let us define the primitive speech acts we will use, inform, queryIf, and refuseInform; a wider set can be found in [13].

[3] For the sake of simplicity we allow as propositional contents only attitude-free fluents, i.e fluent literals or done fluents.

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

Clause (a) states that an inform act can be executed when the sender believes l and believes that the receiver does not know l. 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 to be a trusted *authority* about l, it is confident that the receiver will adopt its belief, (c). When ag_i is the receiver of an inform act, it will believe that l is believed by the sender ag_j (e), but it will adopt l as itw own belief only if it recognizes ag_j as a trusted authority, (f)-(g).

$\mathsf{queryIf}(sender, receiver, l)$
a) $\Box(\mathcal{U}^{ag_i} l \wedge \neg\mathcal{B}^{ag_i}\mathcal{U}^{ag_j} l \supset \langle \mathsf{queryIf}(ag_i, ag_j, l)\rangle \top)$
b) $\Box(\top \supset \langle \mathsf{queryIf}(ag_j, ag_i, l)\rangle \top)$
c) $\Box([\mathsf{queryIf}(ag_j, ag_i, l)]\mathcal{B}^{ag_i}\mathcal{U}^{ag_j} l)$

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

$\mathsf{refuseInform}(sender, receiver, l)$
a) $\Box(\mathcal{U}^{ag_i} l \wedge \mathcal{B}^{ag_i} Done(\mathsf{queryIf}(ag_j, ag_i, l))\top \supset \langle \mathsf{refuseInform}(ag_i, ag_j, l)\rangle \top)$
b) $\Box(\top \supset \langle \mathsf{refuseInform}(ag_j, ag_i, l)\rangle \top)$
c) $\Box([\mathsf{refuseInform}(ag_j, ag_i, l)]\mathcal{B}^{ag_i}\mathcal{U}^{ag_j} l)$

By refuseInform an agent refuses to give an information it was asked for. The refusal can be executed only if: the sender is ignorant about l and it believes that the receiver previously queried it about l. After a refusal the receiver believes the sender to be ignorant about l.

Get Message Actions are used for *receiving* messages from other agents; they can be viewed as queries for an external input, whose outcome is unpredictable. We model them as a special kind of sensing actions, the main difference being that they are defined by means of speech acts. Formally, we represent the fact that ag_i expects a communication from ag_j, by means of a get_message action, given by an axiom schema of form:

$$[\mathsf{get\_message}(ag_i, ag_j, l)]\varphi \equiv [\bigcup_{\mathsf{speech\_act} \in \mathcal{C}_{\mathsf{get\_message}}} \mathsf{speech\_act}(ag_j, ag_i, l)]\varphi \qquad (6)$$

having as defining primitive actions a finite set of speech acts $\mathcal{C}_{\mathsf{get\_message}} \subseteq \mathcal{C}$. $\mathcal{C}_{\mathsf{get\_message}}$ is the set of the possible communications that ag_i expects from ag_j in

the context of a given conversation. Then, we do not associate to a get_message action a domain of mental fluents, but we calculate the information obtained by looking at the effects of the speech acts in $\mathcal{C}_{\mathsf{get\_message}}$ on ag_i's mental state.

Conversation protocols Often primitive communication acts occur in the context of predefined conversation schemas [11] that specify communication patterns. To cope with these aspects we provide agents with conversation protocols. Each agent has a subjective perception of on-going communications, for this reason protocols have as many procedural representations as possible roles in the conversation. Let us consider, for instance the yes_no_query protocol reported hereafter, a simplified version of the FIPA Query Interaction Protocol [8]. The protocol has two representations, one to be followed for making a query ($\mathsf{yes\_no\_query}_Q$) and a complementary one for responding ($\mathsf{yes\_no\_query}_I$):

$$\langle \mathsf{yes\_no\_query}_Q(Self, Other, Fluent)\rangle\varphi \subset \langle \mathsf{queryIf}(Self, Other, Fluent); \mathsf{get\_answer}(Self, Other, Fluent)\rangle\varphi$$

$$[\mathsf{get\_answer}(Self, Other, Fluent)]\varphi \equiv [\mathsf{inform}(Other, Self, Fluent) \cup \mathsf{inform}(Other, Self, \neg Fluent) \cup \mathsf{refuseInform}(Other, Self, Fluent)]\varphi$$

Intuitively, the right hand side of get_answer represents all the possible answers expected by agent $Self$ from agent $Other$ about $Fluent$, in the context of a conversation ruled by the $\mathsf{yes\_no\_query}_Q$ protocol.

$$\langle \mathsf{yes\_no\_query}_I(Self, Other, Fluent)\rangle\varphi \subset \langle \mathsf{get\_start}(Self, Other, Fluent); \mathcal{B}^{Self} Fluent?; \mathsf{inform}(Self, Other, Fluent)\rangle\varphi$$
$$\langle \mathsf{yes\_no\_query}_I(Self, Other, Fluent)\rangle\varphi \subset \langle \mathsf{get\_start}(Self, Other, Fluent); \mathcal{B}^{Self} \neg Fluent?; \mathsf{inform}(Self, Other, \neg Fluent)\rangle\varphi$$
$$\langle \mathsf{yes\_no\_query}_I(Self, Other, Fluent)\rangle\varphi \subset \langle \mathsf{get\_start}(Self, Other, Fluent); \mathcal{U}^{Self} Fluent?; \mathsf{refuseInform}(Self, Other, Fluent)\rangle\varphi$$

Intuitively the $\mathsf{yes\_no\_query}_I$ 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. get_start is a get_message action ruled by the following axiom:

$$[\mathsf{get\_start}(Self, Other, Fluent)]\varphi \equiv [\mathsf{queryIf}(Other, Self, Fluent)]\varphi$$

We define the *communication kit* of an agent ag_i, CKit^{ag_i}, as the triple ($\Pi_{\mathcal{C}}$, $\Pi_{\mathcal{CP}}$, $\Pi_{\mathcal{S}get}$), where $\Pi_{\mathcal{C}}$ is the set of simple action laws defining ag_i's primitive communicative speech acts, $\Pi_{\mathcal{S}get}$ is a set of axioms for ag_i's get_message actions and $\Pi_{\mathcal{CP}}$ is the set of procedure axioms specifying ag_i's conversation protocols. In this extension of the DyLOG language, the *Domain Description* for agent ag_i is a triple ($\Pi, \mathsf{CKit}^{ag_i}, S_0$), where CKit^{ag_i} is ag_i communication kit, S_0 is the initial set of ag_i's belief fluents, and Π is a tuple ($\Pi_{\mathcal{A}}, \Pi_{\mathcal{S}}, \Pi_{\mathcal{P}}$), where $\Pi_{\mathcal{A}}$ is the set of ag_i's action and precondition laws for world actions, $\Pi_{\mathcal{S}}$ is a set of axioms for ag_i's sensing actions, $\Pi_{\mathcal{P}}$ a set of axioms that define complex actions.

4 Reasoning about Conversations

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

$$\langle p_1 \rangle \langle p_2 \rangle \dots \langle p_m \rangle Fs \tag{7}$$

Each p_k, $k = 1, \dots, m$ in (7) may either be an (atomic or complex) action executed by ag_i or an external speech act, that belongs to CKit^{ag_i} (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 form (7) succeeds we can cope with the planning problem. In fact this corresponds to answering the question "is there an execution trace of p_1, ..., p_n leading to a state where the conjunction of belief fluents Fs holds for ag_i?". Such an execution trace is a plan to bring about Fs. The procedure definition constrains the search space. Notice that when all the p_k in the query are atomic actions that belong to $\mathcal{A} \cup \mathcal{C}$, by checking if the query succeeds, we cope also with the temporal projection problem: "does Fs hold for ag_i, after the execution of the action sequence $a_1, \dots, a_m$?".

In presence of communication, the planning and the temporal projection problems turn respectively into the problem of reasoning about *conversation protocols* and reasoning about simple *conversations*, where a conversation is a sequence of speech acts. This allows, for instance, an agent to *investigate* the possible changes to its mental state, produced by a specific conversation, or if a conversation is an instance of some predefined protocol [7].

In the case of planning, the action sequence will contain both actions in which the agent is the sender and actions in which it is the receiver. In this process we treat get_message actions as sensing actions, whose outcome cannot be known at planning time. Since agents cannot read each other's mind, they cannot know in advance the answers that they will receive. For this reason all of the possible alternatives are to be taken into account (and we can foresee them because of the existence of the protocol). Therefore, the extracted plan will be *conditional*, meaning that for each 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 the desired condition Fs. More formally, a conditional plan σ is either:

- a action sequence $a_1; \dots; a_m$, with $m \geq 0$;
- if $a_1; \dots; a_m$ is an action sequence ($m \geq 0$), $s \in \mathcal{S}$ is a sensing action, and $\sigma_1, \dots, \sigma_t$ are conditional plans then $a_1; \dots; a_m; s; ((\mathcal{B}^{ag_i} l_1?); \sigma_1 \cup \dots \cup (\mathcal{B}^{ag_i} l_t?); \sigma_t)$, where $l_1, \dots, l_t \in dom(s)$;
- if $a_1; \dots; a_m$ is an action sequence ($m \geq 0$), $g \in \mathcal{S}$ is a get_message action, and $\sigma_1, \dots, \sigma_t$ are conditional plans then $a_1; \dots; a_k; g; ((\mathcal{B}^{ag_i} Done(c_1) \top?); \sigma_1 \cup \dots \cup (\mathcal{B}^{ag_i} Done(c_t) \top?); \sigma_t)$, where $c_1, \dots, c_t \in \mathcal{C}_g$.

The proof procedure is a natural evolution of the work in [2], and is described in [1]; it is goal-directed and based on negation as failure (NAF). NAF is used

to deal with the persistency problem to verify that the complement of a mental fluent is not true in the state resulting from an action execution; while in the modal theory we adopted an abductive characterization [13]. The proof procedure allows agents to find *linear* and *conditional* plans for achieving a goal from an incompletely specified initial state. The soundness can be proved under the assumption of e-consistency, i.e. for any action the set of its effects is consistent [6]. Moreover, the extracted plans always lead to a state in which the desired condition *Fs* holds, for all the possible results of the sensing actions.

5 An Example in the Web Services Scenario

Let us consider the personal assistant (pa) mentioned in the introduction, its aim is to search a *cinema booking service* that satisfies the user's requests. Suppose that two services are available, *click_ticket* and *all_cinema*, that they follow two different interaction protocols (get_ticket_1 and get_ticket_2), see Fig. 1, and that such protocols are part of the web service descriptions. The difference between them is 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 ticket purchase by credit card. Let us consider the DyLOG specifications of the two protocols, that describe the interaction of pa with the web services from the *customer perspective*: $\mathsf{get\_ticket\_1}_C$ and $\mathsf{get\_ticket\_2}_C$. We will suppose that pa knows the credit card number (cc_number) of the user but that it is requested not to use it. Let us see how pa reasons on the *way* in which conversations will be carried on.

(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 waits for an information (get_info) from the provider ($WebS$) about which cinema shows it. Then the form of payment is defined: (a) encodes the interaction occuring when the tickets are paid by credit card (see Fig. 1); (b) is selected when $\neg\mathcal{B}^{Self} pay\_by(credit\_card)$

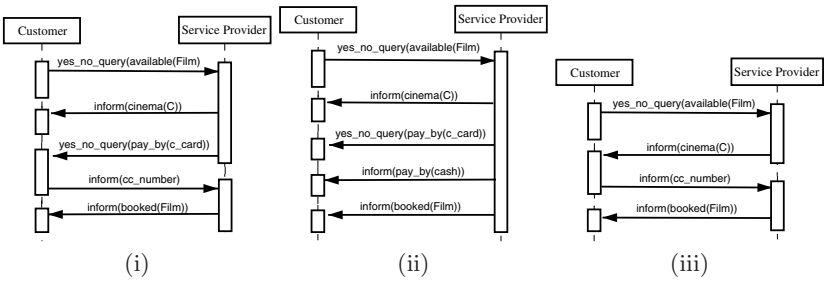

Fig. 1. The three AUML graphs [12] represent the communicative interactions occurring between the customer (pa) and the provider.

is contained in pa mental state (our case), leading to book a ticket to be paid cash. In both cases a confirmation of the ticket booking is returned to pa. Clause (c) tackles the case in which the movie is not available; (d) describes get_info, which is a get_message action. Let us now 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 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 his 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 $click\_ticket$ mental state, e.g. the belief fluent $\mathcal{B}^{pa} \neg B^{click\_ticket} cc\_number$, 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 $\mathcal{B}^{pa} \neg B^{click\_ticket} cc\_number$, 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 such a fluent. After the briefly described reasoning process, the agent finds an execution trace of $\mathsf{get\_ticket\_1}_C$, which corresponds to a *personalized conditional dialogue plan* between itself and the provider $click\_ticket$, always leading to satisfy the user goal of not giving the credit card number:

$\mathsf{queryIf}(pa, click\_ticket, akira)$;
  $((\mathcal{B}^{pa} Done(\mathsf{inform}(click\_ticket, pa, akira))\top?)$;
    $\mathsf{get\_info}(pa, click\_ticket, cinema(C))$;
    $(\mathcal{B}^{pa} Done(\mathsf{inform}(click\_ticket, pa, cinema(C)))\top?)$;

$\mathsf{get\_info}(pa, click\_ticket, pay\_by(credit\_card)));$
$(\mathcal{B}^{pa}\, Done(\mathsf{queryIf}(click\_ticket, pa, pay\_by(credit\_card)))\top?);$
$\mathsf{inform}(pa, click\_ticket, \neg pay\_by(credit\_card));$
$\mathsf{get\_info}(pa, click\_ticket, pay\_by(cash));$
$(\mathcal{B}^{pa}\, Done(\mathsf{inform}(click\_ticket, pa, pay\_by\_cash))\top?);$
$\mathsf{get\_info}(pa, click\_cinema, booked(akira));$
$(\mathcal{B}^{pa}\, Done(\mathsf{inform}(click\_ticket, pa, booked(akira)))\top?) \cup$
$(\mathcal{B}^{pa}\, Done(\mathsf{inform}(click\_ticket, pa, \neg akira))\top?) \cup$
$(\mathcal{B}^{pa}\, Done(\mathsf{refuseInform}(click\_ticket, pa, akira))\top?))$

Let us now considering $\mathsf{get\_ticket\_2}_C$, the protocol followed by *all_cinema*:

(e) $\langle \mathsf{get\_ticket\_2}_C(Self, WebS, Film)\rangle\varphi \subset$
$\langle \mathsf{yes\_no\_query}_Q(Self, WebS, available(Film));$
$\mathcal{B}^{Self} available(Film)?\ ;\ \mathsf{get\_info}(Self, WebS, cinema(C));$
$\mathsf{inform}(Self, WebS, cc\_number);$
$\mathsf{get\_info}(Self, WebS, booked(cinema\_ticket, Film))\rangle\varphi$

(f) $\langle \mathsf{get\_ticket\_2}_C(Self, WebS, Film)\rangle\varphi \subset$
$\langle \mathsf{yes\_no\_query}_Q(Self, WebS, available(Film)); \neg\mathcal{B}^{Self} available(Film)?\rangle\varphi$

We can easily see that in this case, the query $\langle \mathsf{get\_ticket\_2}_C(pa,\ all\_cinema,\ akira)\rangle\ \mathcal{B}^{pa}\neg\mathcal{B}^{all\_cinema} cc\_number$ fails because, the protocol allows only one interaction sequence, containing the action $\mathsf{inform}(pa,\ all\_cinema,\ cc\_number)$ that causes the mental state of *pa* to contain $\mathcal{B}^{pa}\mathcal{B}^{all\_cinema}\ cc\_number$.

6 Conclusions

We have presented an approach to reasoning about conversation protocols within the framework of a logic-based agent language. We have studied how to embed a theory of communicative actions in the DyLOG logical framework [2] so to allow the modelling of software agents that, when situated in a multi-agent environment, can interact with one another by a speech act based communication mechanism. In the extended language agents have their own local beliefs on the world and on the other agents mental state. The semantics of communicative acts is described in terms of their effects on the mental state of both the sender and the recipient. In the line of [11], we used conversation protocols to provide our agents decision procedures for suitably responding to communications. We took a subjective representation of conversation protocols, by making hypothetical assumptions on the other's answers. As a consequence protocols have been easily integrated with other policies defining the agent behavior, being both represented as procedures specified in DyLOG. Notice that, since we are only interested in reasoning about the local mental state dynamics, our approach differs from other logic-based approaches to communication in multi-agent systems, as the ones taken in [15, 9], where communicative acts affect the global state of a multi-agent system and the target is to verify global properties of the overall multi-agent system execution. Instead, our focus on the internal specification of interaction protocols for planning dialogue moves is closer to the one taken in [14], where

negotiation protocols, expressed by sets of dialogue constraints, are included in the agent program and used for triggering dialogues that achieve goals. However such an approach leads to different results and, being somehow focused on goal achievement in a multi-agent environment with limited resources, is not aimed at implementing the reasoning about conversations we focused on.

References

[1] M. Baldoni, C. Baroglio, A. Martelli, and V. Patti. Reasoning about self and others: communicating agents in a modal action logic. In R. Gorrieri, C. Blundo, and C. Laneve, editors, *Proc. of ICTCS'2003*, LNCS, Bologna, Italy, 2003. Springer. To appear. 307

[2] 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. 301, 302, 303, 307, 310

[3] 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. 301

[4] R. Chinnici, M. Gudgin, J. J. Moreau, and S. Weerawarana. Web Services Description Language (WSDL) version 1.2, 2003. Working Draft. 300

[5] The DAML-S coalition. DAML-S: Web service description for the semantic web. In *the 1st Int. Semantic Web Conference (ISWC)*, Sardinia, Italy, 2002. 300

[6] M. Denecker and D. De Schreye. Representing Incomplete Knowledge in Abduction Logic Programming. In *Proc. of ILPS '93*, Vancouver, 1993. The MIT Press. 308

[7] F. Dignum and M. Greaves. Issues in agent communication. In *Issues in Agent Communication*, volume 1916 of *LNCS*, pages 1–16. Springer, 2000. 307

[8] FIPA. Fipa 2000, fipa Query Interaction Protocol Specification. Technical report, FIPA (Foundation for Intelligent Physical Agents), November 2000. 306

[9] L. Giordano, A. Martelli, and C. Schwind. Specifying and Verifying Systems of Communicating Agents in a Temporal Action Logic. In *Proc. of the 8th Conf. of AI*IA*, LNAI. Springer, 2003. 310

[10] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. GOLOG: A Logic Programming Language for Dynamic Domains. *J. of Logic Programming*, 31:59–83, 1997. 302

[11] 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. 306, 310

[12] 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/. 309

[13] 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/`. 302, 304, 308

[14] F. Sadri, F. Toni, and P. Torroni. Dialogues for Negotiation: Agent Varieties and Dialogue Sequences. In *Proc. of ATAL'01*, Seattle, WA, 2001. 310

[15] S. Shapiro, Y. Lespéance, and H. J. Levesque. Specifying communicative multi-agent systems. In *Agents and Multi-Agent Systems - Formalisms, Methodologies, and Applications*, volume 1441 of *LNAI*, pages 1–14. Springer-Verlag, 1998. 310