A Plan Based Agent Architecture for Interpreting Natural Language Dialogue

LILIANA ARDISSONO, GUIDO BOELLA AND LEONARDO LESMO
Dipartimento di Informatica, Università di Torino,
Corso Svizzera n.185, 10149 Torino, Italy.
email: {liliana, guido, lesmo}@di.unito.it
http://www.di.unito.it/{~liliana,~guido,~lesmo}

This work is going to appear in the
International Journal of Human-Computer Studies, ACADEMIC PRESS
(http://www.academicpress.com/www/journal/hc.htm)

Abstract

This paper describes a plan-based agent architecture for modeling NL cooperative dialogue; in particular, the paper focuses on the interpretation dialogue and explanation of its coherence by means of the recognition of the speakers' underlying intentions. The approach we propose makes it possible to analyze and explain in a uniform way several apparently unrelated linguistic phenomena, which have been often studied separately and treated via ad-hoc methods in the models of dialogue presented in the literature. Our model of linguistic interaction is based on the idea that dialogue can be seen as any other interaction among agents: therefore, domain-level and linguistic actions are treated in a similar way.

Our agent architecture is based on a two-level representation of the knowledge about acting: at the metalevel, the Agent Modeling plans describe the recipes for plan formation and execution (they are a declarative representation of a reactive planner); at the object level, the domain and communicative actions are defined. The Agent Modeling plans are used to identify the goals underlying the actions performed by an observed agent; the recognized plans constitute the dialogue context, where the intentions of all participants are stored in a structured way, in order to be used in the interpretation of the subsequent dialogue turns.

Keywords: Dialogue Processing, Plan and Goal Recognition, Agent Modeling, Natural Language Interpretation.

1 Introduction

This paper presents a computational model of dialogue that conceives dialogue as an interaction among autonomous agents. Each agent has his own beliefs, goals, and intentions and collaborates with his interlocutors: the cooperation implies that, at each step of the interaction, the agents share some goals and act to reach them; moreover, the presence of such shared goals is what supports the coherence of dialogue. During the dialogue, each agent must be able to recognize the explicit goals set forth by his partners, and to discover some of their implicit intentions. Then, the agent can balance what he has understood about the other agents against his own goals, in order to choose his next step in the interaction.

The computational framework we adopt is based on plans and models the activity of an agent as plan formation and execution; in order to behave properly, the agent needs knowledge about the actions that can be brought about in the domain of interest and knowledge about the linguistic actions that can be executed to interact with other agents. Moreover, he needs knowledge about the way a plan can be built, out of the available actions, and executed to reach a given goal. Although two substantially different types of knowledge are involved (knowledge about domain/linguistic actions and knowledge about planning), we represent them uniformly: in both cases, plan recipes express the relevant information.

The choice of modeling agent behavior in a plan-based, declarative way is due to the basic assumption that the knowledge structures underlying linguistic behavior are common to the two phases of language understanding and language production. So, an agent must resort to the same piece of knowledge both to understand, for instance, a polite request ("Saresti così gentile da aprire la finestra?", i.e. "Would you be so kind as to open the window?")[1] and to produce the same request in appropriate circumstances. Note that also the plan formation activity must be properly recognized by a hearer, as is shown by the classical case of applicability condition check ("Is the library open?", Allen (1983)). Therefore, also knowledge about how to build plans is necessary for both the phases of understanding and producing behavior.

A consequence of the choice of assimilating dialogue as general cooperative interaction is that we model dialogue avoiding specific knowledge about the admissible sequences of turns. This is accomplished by treating the turns as instances of the possible continuations of an interaction: a turn is coherent with respect to the previous part of the dialogue if it can be explained as the means for achieving one of the goals that have been set forth (implicitly or explicitly) in the dialogue and which are still pending.

The solution given in most dialogue systems is to have the list of possible turn sequences (e.g. question-answer and request-refusal) by means, for example, of a dialogue grammar. However, the specifications of all the possible continuations of a dialogue are

[1] Our prototype has been developed for the Italian language: it is based on the GULL syntactic and semantic interpreter (Lesmo & Torasso 1985, Lesmo & Terenziani 1988, Di Eugenio & Lesmo 1987, Ardissono *et al.* 1991), that takes as input Italian NL sentences. However, this paper focuses on the analysis of dialogue behavior, which is largely independent of the syntactic and semantic properties of a given language; therefore, from now on, all the examples will be reported in English for the reader's convenience.

open-ended, too rigid, and lack a common justification. An approach that allows an explanation for an answer to a question, or a refusal to a request, on the basis of deeper unifying principles of rational behavior, as the ones embodied in our model, overcomes all these problems and, again, assimilates linguistic behavior as a general agent activity.

We will show how our goal-based model of dialogue can explain adjacency pairs, insertion sequences, presequences, grounding processes and various other phenomena, like acknowledgements, overanswering, and subdialogues due to problems in the interpretation process; this model also sheds light on the phenomenon of repairing to misunderstandings.

Our architecture includes two levels of plans: a metalevel, describing recipes for plan formation and execution, and an object level, where domain and linguistic actions are defined. All the actions contribute to changing the state of the world: the domain-level actions are used to modify the surrounding physical situation, the linguistic actions are used to modify the beliefs of the partners, while plan-management recipes involve a modification of the agents' own intentions about what to do next. So, in modeling dialogue, turns can be interpreted as actions having the same status as the domain actions performed to achieve specific world states. We call the metalevel "Agent Modeling" (AM) level, and the recipes therein AM plans; the Agent Modeling plans have the object-level actions as their arguments.

In a sense, the Agent Modeling plans are the planner of our system: they embody the strategies that lead an agent to adopt a particular line of behavior, i.e. to choose a given sequence of (domain and/or linguistic) actions to reach his goal(s). Following the ideas developed in the research about agents after the first definitions of intention (Bratman *et al.* 1988), the Agent Modeling level describes behavior in terms of a "reactive planner" (like those in Georgeff & Ingrand (1989) and Firby (1987)): in order to satisfy a goal, the agent chooses a more or less complex action, checks its preconditions, executes a suitable recipe down to the elementary actions; it also monitors whether the action succeeds and it replans any action whose execution fails.[2]

Since the agent is embedded into a community of agents, he can ask for cooperation, as well as offer his own collaboration to his partners: therefore, we also model the collaborative behavior described in Cohen & Levesque (1991), as notifying the partners about how the execution of an action proceeds.

The main difference with respect to other planners is that we chose to describe the planner in a plan-based declarative language. The plan-based architecture allows us to apply the plan recognition algorithms developed so far (Carberry 1990b, Lambert 1993) in order to recognize and trace both the object-level plans carried on by the agent, and

[2] The term "reactive planner" is used here to denote reactive deliberative systems, also defined as "practical reasoning systems" (Jennings *et al.* 1998); on the contrary, we are not referring to the stimulus-driven reactive agents like the "Subsumption architecture" described in Brooks (1991).

his planning behavior; this approach enables us to identify the planning phase, the monitoring activity, etc. performed by the agent through the observation of the object-level (domain and linguistic) actions he is doing. The Agent Modeling plans are the glue that keeps together the various object-level actions by specifying how the turns of the dialogue contribute to the higher goals shared by the agents.

During the interpretation process, the recognized instances of plans are recorded to contribute to the formation of the agent's internal representation of the ongoing dialogue. The agent also maintains the instances of his own Agent Modeling plans (i.e. the AM actions used to plan his behavior), as well as the object-level actions considered in these plans. In this way, the plan-based representation allows the agent to maintain in the context the trace of his own intentional activity, which can be exploited for understanding if and how his partners contribute to his own goals.

The declarative representation of the planning knowledge enables the system to model in the same way the two interacting agents. This is in contrast with the traditional view, where it is assumed that the user and the system play asymmetrical roles in dialogue. In particular, in such a view the observed agent (the system's "user") is supposed to devise and execute his own plans to reach his own goals. So, the interaction can be continued coherently, on the basis of the recognition of the user's plans. This view seems unbalanced: it can be applied correctly just in case there is a fixed role for each participant: the user asks and the system helps. Although this assumption is largely correct, it poses some limits to the generality of the model: in an interaction of this fixed type, different models have to be defined for the two roles. In our approach, where a uniform "agent" model is defined, the "function" of the turns of both speakers can be specified in the same way: they can be a continuation of a previous activity, or they can be linguistic acts, performed to satisfy an overt or implicit desire of the partner, and this is usually, but not necessarily, the system's role.

To summarize, the idea is to treat linguistic and domain-level actions uniformly, and to feed them to a reactive (collaborative) planner represented declaratively, in order to get a more finely-grained dialogue model. Moreover, if we enable an agent to recognize his partner's behavior in terms of his planning activity, we achieve a better understanding of his dialogue contributions, where his domain and linguistic actions are explained uniformly in terms of the same goal-directed behavior.

The paper is organized as follows: section 2 discusses how the work done in other research areas supports the idea that dialogue phenomena depend upon the speakers' intentions; then, in section 3 we sketch informally the main ideas underlying our work and in section 4 we present a variety of phenomena which can be explained by our solution. Sections 5 and 6 present the plan libraries used by the system and the interpretation process, respectively. Finally, a related work section and the conclusions close the paper.

2 The intention-based approach to dialogue modeling

It is widely accepted that dialogue depends on the intentions of the agents engaging in an interaction. This idea has been developed in Linguistics, Philosophy and Artificial Intelligence.

Among the first supporters of this conception of dialogue, Levinson (1981) highlights that the relevant replies to a turn depend on the background activity of the interactants. The main argument he advances is that after a question we can find not only answers, but also some other questions about the relevance of the original question, the negation of its presuppositions, or a proposal for solving the higher-level goals of the speaker who asked the question. Moreover, he notices the problem of replies concerning the perlocutionary effects of a speech act, which are, by definition, loosely related to the speaker's linguistic activity.

Also Searle (1992) agrees with this conception and advances his own explanation: he suggests that the difficulty in constraining the space of the possible replies to a speech act is due to the fact that the relevance of a contribution cannot depend only on the previous speech acts: in fact, illocutionary acts are defined in terms of goals (e.g. a request is an attempt to make the hearer perform an action) and there is no intrinsic goal in conversation.

In the AI community, dialogue has been modeled in some works by considering only the relation among speech acts and relying on the observable regularities in human dialogue: for instance, see the finite-state grammar model of Winograd & Flores (1986). However, the role of intentions and planning in dialogue has been recognized by several researchers, such as Allen & Perrault (1980) and Hobbs & Evans (1980). Also Cohen (1994) challenges the idea that dialogue is a rule-based behavior: he claims that "no definitions are given for the transition types, no reasons are provided for the observed transitions between states and no explanation is given of why or how a particular subgraph constitutes a dialogue unit" (Smith & Cohen (1996), page 29). In Smith & Cohen (1996) the semantics of speech acts is specified in terms of the activity of forming and maintaining teams aimed at achieving a speaker's goals. In this view, a sequence of speech acts is admissible when the speakers' contributions aim at satisfying some goal prescribed by the team activity established previously. In turn, team activity is defined in terms of mutual beliefs about goals, and weak mutual goals (Cohen & Levesque 1991): an agent involved in the collaboration should inform the partner that the goal has been achieved, is unachievable, or irrelevant, as soon as he becomes aware of this fact. Given this definition of joint intention, Smith & Cohen (1996) model speech acts as "attempts" to form a team for pursuing a goal with the minimal acceptable result of making mutually believed the intention to form the team.[3]

Also Grosz & Sidner (1990) support the role of intentions in dialogue modeling: in particular, in a development of their work, Lochbaum proposes a dialogue conception where the agent's intentions guide his behavior and the "partner recognition of those intentions aids in the latter's understanding of utterances" (Lochbaum (1994), page 1). In fact, an agent engages in dialogues and subdialogues for reasons deriving from the mental state requirements of action and collaboration. Lochbaum (1994) and Lochbaum (1995) provide a computational model of dialogue based on the Shared Plans operator defined in (Grosz & Sidner 1990, Grosz & Kraus 1996); Lochbaum's model aims at recognizing the intentional structure of dialogue and at using it in discourse processing.

Moreover, the notion of intention has been exploited to model human-computer dialogue in Bretier & Sadek (1997) and Goerz & Ludwig (1998): both works describe (belief, desire, intention) agent architectures for modeling dialogue as rational agent interaction.

The intention-based nature of dialogue is indirectly supported also by some of the "grammar-based" approaches to dialogue processing: for example, Stein & Maier (1994) explain the behavior of their finite state dialogue automata in terms of agents' expectations or goal satisfaction. Moreover, they superimpose on the automaton a formalism based on the Rhetorical Structure Theory (Mann & Thompson 1987) to explain the relations between speech acts; however, in Moore & Paris (1993), the rhetorical relations have been explained in terms of a speaker's goals directed towards the receiver's understanding. Note also that Smith & Cohen (1996) aim at proving that the structure of such dialogue grammars follows from the logical relationships arising from the definition of communicative actions in terms of team formation.

Other intention-based approaches to dialogue modeling have discussed and analyzed the importance of considering also the agents' metalevel activity for modeling their interaction with other agents (Litman & Allen 1987, Ramshaw 1991, Carberry *et al.* 1992): this metalevel activity is important to understand behaviors like plan exploration.

3 An architecture based on plans

In our architecture, the actions which an agent can execute are of three types and, correspondingly, are defined as belonging to three different sets (libraries): the Agent Modeling (AM) library, the Speech Act (SA) library, and the Domain library (D).
While the AM library has a truly distinguished role (as outlined in Section 1, it is a "metalevel"), the SA library and the Domain library are kept apart just because the former is

[3] This is a minimal success condition because believing that the speaker wants the hearer to act is only a *motivation* for the hearer to do it.

domain-independent, while the contents of the latter depend on the application domain.

Currently, the system does not include a generation module, so it is clearer to consider the implemented prototype as an external observer: it takes as input from the keyboard the utterances of both speakers and produces the representation of a dialogue context containing the interpretations of turns, together with the coherence relations among them.[4] In the following description, we assume that the input sentences are produced either by the user (U) or by a consultant (C), while the system observes the exchange.

When a sentence is given as input, the system realizes that a speech act has been done. The Agent Modeling library specifies that an action is usually carried out as a step of a more general plan; therefore, a more general linguistic action, having the observed speech act as a step, is looked for. This process is iterated upwards, under the control of the AM plans; at a certain point, by analyzing the metalevel knowledge, the system realizes that a top-level speech act (e.g. a request) can be related to a domain-level action: the upward search is switched from the Speech Act library to the Domain library, but nothing else changes. Again, the AM library is inspected to determine the activities concerning plan formation and execution: so, it is possible to decide whether the recognized speech act is related to plan formation (e.g. the possible ways to carry out an action are being inspected) or plan execution (e.g. a request to register for an exam has been put forward).

When the subsequent dialogue turns are received, they are interpreted in exactly the same way. During this analysis, instances of actions in the plan libraries, including AM actions, speech acts, and domain-level actions, are recognized and maintained in a separate structure, to record a complete picture of the observed linguistic event. This structure, an extension of the Context Models described in Carberry (1990b), is the detailed representation of the speakers' activity, as recognized by the system: it specifies what they are doing and why, according to the system's comprehension.

Figure 1 sketches the representation of a turn in the pictorial form that will be used throughout the paper: in the figure, the box labeled with AM1 represents the structure containing the metalevel actions recognized while interpreting the turn; the boxes labeled as D1 and SA1 represent the structures containing the recognized Domain-level actions and speech acts. The action instances occurring in the interpretation structures are usually represented by specifying their names, the values to which their arguments are bound (such values may be actions in the case of AM actions) and possibly their preconditions (see the arcs labeled with "prec"), constraints ("constr") and effects ("eff"). The dotted arrows relating AM actions to object-level actions (i.e. domain-level actions and speech acts) represent the links between a metalevel action instance and its

[4] This approach has clear limitations: for example, it imposes the underlying assumption that no misunderstandings occur among the speakers. Nevertheless, this is a temporary, simplifying solution, to be maintained until the generation phase will be implemented in our system.

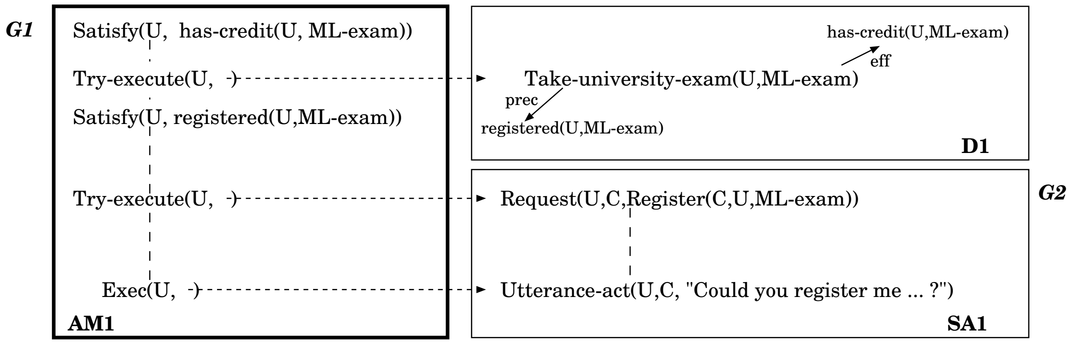

G1
Satisfy(U, has-credit(U, ML-exam))
Try-execute(U, -)
Satisfy(U, registered(U,ML-exam))
Try-execute(U, -)
Exec(U, -)
AM1
has-credit(U,ML-exam)
eff
Take-university-exam(U,ML-exam)
prec
registered(U,ML-exam)
D1
G2
Request(U,C,Register(C,U,ML-exam))
Utterance-act(U,C, "Could you register me ... ?")
SA1

Figure 1. Sketch of an turn representation structure.

arguments: for instance, the bottom "Exec" action in AM1 has "U" as its first argument and "Utterance-act(U, C, "Could you register me ...?")" as its second argument. Instead, the dashed lines between actions contained in the same box relate complex action instances to their substeps, omitting intermediate steps, which are not shown because only the relevant portion of the interpretation structures is provided.

During the interpretation of a dialogue, the system maintains two models containing the interpretation of the turns by the interactants: the user's (agent) model (*UAM*) and the consultant's one (*CAM*). When a turn is interpreted, the system expands the speaker's model with the turn representation. Then, in order to represent explicitly the coherence of the dialogue, the system tries to connect the turn to some contextual goal; this operation succeeds only if the main goal $G1$ (see Figure 1) which explains the reason why the turn has been produced also appears in the *UAM* or in the *CAM*. For example, given a turn of the consultant C, there are two possibilities:

- If G appears in the *UAM*, then he has decided to cooperate with U and has produced a turn to satisfy G.
- Otherwise, G is contained in the *CAM*, so C is continuing some of his own activities, possibly because there are no other goals belonging to U to be satisfied.

In the second case, the plan structures contained in the *CAM* and representing C's activity are extended with the model of the last turn; this integration is the classical "focusing" step introduced in Carberry (1990b). The search for this goal G is carried out by the system; if it succeeds, a link is established between the turn representation and the *UAM* or the *CAM*; this link relates the two occurrences of the top-level goal G.

Note that G does not necessarily need to be satisfied by C: G is the top goal of the turn representation because C's reply has been produced while attempting to satisfy G; the only requirement is that C's reply is "in some way" connected (i.e. contributes) to G, where the possible ways are defined by the system's knowledge about plans, as encoded in the Agent Modeling library. For example, after a request by a speaker U, a clarification requested by C to U doesn't satisfy U's goal, but it is a coherent reply, because it can be explained as an attempt to satisfy one of the preconditions required to fulfill U's desires, perhaps a precondition on C side.

This sequence of steps can be exemplified by the following exchange. Figure 2 shows very sketchily the evolution of the Agent Modeling models, in order to point out the interactions between them. We report the complete representation of this dialogue model in Section 6. In the following paragraphs, we provide the details required to understand the role of the different substructures and how are they actually built and linked together.

Example 3.1:

U-1: Could you register me for the ML exam?
C-1: What is your name?
U-2: Mario Rossi.
C-2: Ok; you are registered!
U-3: Can you tell me the exact date of the exam?

The analysis of the user's first turn (U-1) leads the system to initialize a *UAM* including a speech act model (*SA1*). The initialization is driven by the interpretation process under the control of the Agent Modeling plans; they enable the system to determine that an indirect speech act was performed with the goal of expressing a request. This request concerns the domain action "register for an exam" (the goal that the partner registers U is *G2* in Figures 1 and 2), that probably makes true a precondition of the action "take an exam" (contained in the structure *D1*, with the goal "having a credit for the exam", *G1*): the AM plans specify that a first step in executing an action is to check its preconditions and, if needed, to make them true.[5]

In Step 2, the system observes C-1 and initializes the consultant's Agent Model (*CAM*). The turn is interpreted as a question that aims at satisfying the precondition "know name" of the requested action "register for an exam". So, the action "Register" is replicated in the *CAM* (*D2* with effect *G2′* in Figure 2), with a suitable link connecting *G2* and *G2′* (*L1*). This turn initiates a subdialogue and advances the new subgoal "know name", denoted as *G3* in the *CAM.*

In Step 3, U-2 is observed. Here, there are two alternatives: either the user U is continuing his activity, or he is trying to satisfy one of the consultant's goals. Here, U is contributing to *G3*, because providing the requested information makes it shared among the interactants. A new structure (*AM3, linked to SA3*) is inserted in the *UAM* with the top goal "know name" (*G3′*).

[5] In this case, the action is "Take-University-exam"; since the user knows that he is not already registered, the check does not cause the execution of any observable actions.

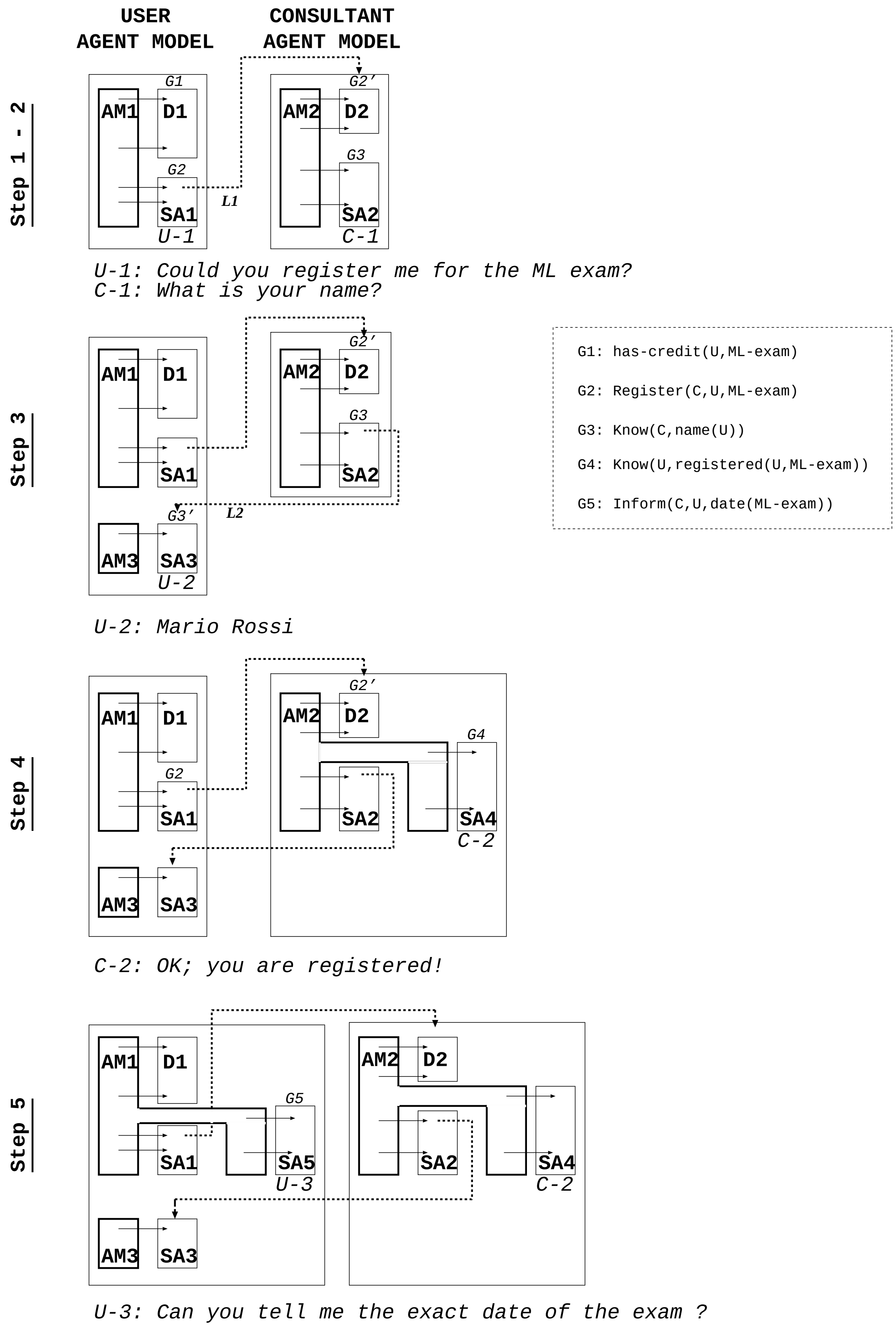


Figure 2. Evolution of *UAM* and *CAM* models during the interpretation of Example 3.1.

In Step 4, C-2 signals the user U that the requested action has been executed successfully. C-2 satisfies *G4*, a classic "notification" goal (Cohen & Levesque 1991); the actions described in the Agent Modeling library prescribe that, after the execution of an object-level action, an agent acquires the goal to notify his partner. A new speech act representation (*SA4*) is inserted in the *CAM* to account for the linguistic interpretation of C-2, and it is related to *AM2* as the last step of the Agent Modeling action describing how to perform the object-level actions.

In the last step, U-3 is a standard continuation step: the user has accepted the completion of the previous subtask and proceeds with his overall plan, by checking another precondition of the domain-level action in focus (achieved by pursuing the goal $G5$, which is linked to *AM1*).

As stressed before, the architecture outlined in this section is the one that underlies the current implementation. To let the system really interact with the user coherently, we have to exploit the Agent Modeling plans for satisfying the goal that the system adopts after it has interpreted the user's turn, so that the system takes the role of the consultant C. Given the ability of the system to generate sentences, the instances of the executed metalevel and object-level plans can then be inserted in the *CAM*: in this way, the system has a model of its own activity, against which it interprets the user's turns in order to understand how they contribute to it.

It should be observed that the relationships existing among the various substructures present in the agent models (AMs) reflect the intentional/attentional structure of the dialogue as defined in Lochbaum (1994) after the work described in Grosz & Sidner (1986). In developing these dialogue structures, we focus on an analysis of the content of utterances, and have not yet exploited cue words. It is clear that exploiting both sources of information would make our system more robust.

The approach we have adopted requires that, given a set of dialogue phenomena to model, the intentional aspects of agent behavior necessary to explain them are represented in our Agent Modeling plans. In this way, the recognition algorithm (and eventually the generation one) can be reduced to a device which works on complex and hierarchical plan operators, but needs no other knowledge about the agent's goal-directed behavior.

Currently, we have explored a number of phenomena deriving from the acquisition of a single high-level goal; on the other hand, we have not yet considered the problems deriving from the acquisition of multiple high-level goals, and the cases where agents carry on multiple plans in parallel. The next section describes a set of phenomena which are explained by the dialogue model we have adopted.

4 The coherence of an interaction

In our work, we aimed at analyzing dialogue coherence on the basis of general principles, which can be exploited in order to model different phenomena uniformly. In Castelfranchi & Parisi (1980) and Castelfranchi & Falcone (1997), some principles were defined in order to model the acquisition of intentions among cooperating agents. We found that the cooperation notion introduced in those works was particularly useful for defining and modeling dialogue coherence in a framework of agent interaction, where domain-level actions, as well as speech acts, can be performed. Therefore, we have exploited that notion in our computational model of dialogue, which has been defined under the hypothesis that speakers are cooperating agents, and are capable of means-ends reasoning to reach the (joint) goals acquired while interacting with their interlocutors. In the following discussion, we will see that the exploitation of Castelfranchi, Parisi and Falcone's cooperation notion within our framework allows us to model uniformly the production of domain-level actions and speech acts, including communicative actions usually classified as meta-communication (e.g. grounding acts).

Our framework models a goal-based notion of coherence in an interaction, where the decision about which sequences of actions are possible is based on the existence of a relation between the goal underlying the last turn and some previous open contextual goal, belonging to the speaker or to his partner. If the goal belongs to the partner, the speaker may have identified it because it was communicated explicitly by the partner (by means of an illocutionary act), or may have inferred it although it was not stated overtly (Allen & Perrault 1980, Allen 1983)).

Following the notion of cooperation in terms of goal delegation and goal adoption defined in Castelfranchi & Parisi (1980) and Castelfranchi & Falcone (1997), we consider an utterance coherent with the previous context if its receiver A can interpret it as the means of the speaker B to achieve a goal g such that:

1. *Goal adherence:* g is a goal that A has expressed explicitly that he wants B to achieve; in other words, B has adopted exactly one of the goals he has been delegated by A to achieve. For example, in:

   A: I need a book. Where is the library?
   B: It is over there.

   B's response contributes to the satisfaction of A's overt goal of knowing where the library is.

2. *Goal adoption:* g is a goal that B has inferred A is going to aim at; A has not delegated B explicitly to achieve g, but B infers it by reasoning about A's plans. For instance, in:

A: I need a book. Where is the library?

B: The library is closed.

B provides *A* with extra-helpful information which satisfies his future goal of checking whether the library is open.[6]

3. *Plan continuation:* *g* contributes to a plan that *B* is carrying on. This relation covers the typical cases where the agent continues a domain-level (or communicative) plan (Carberry *et al.* 1992); moreover, it covers the continuation of Agent Modeling plans, as in the following example:

   B: Where is the library?

   A: It's over there.

   B: Is it open today?

   We assume that an agent continues the execution of his own plans only when there is no other local goal of the partner to be achieved.

These three relations are defined to model cooperative dialogues. However, as shown in Castelfranchi (1992), the set of coherence relations can be extended straightforwardly to consider also the interactions among conflicting agents, who perform actions to contrast the other agents' goals.

In the following, we will describe these three relations in detail; we will show how different dialogue phenomena can be understood as instances of such relations. Our approach is based on the recognition of the intentions underlying an agent's observed actions and on the ability to appropriately relate such intentions to the interactants' pending contextual goals.

4.1 Goal adherence

The goal adherence relation models every contribution that derives from an agent's intention to cooperate with another agent to reach the goals communicated explicitly by him.[7] It covers the direct correspondences between speech acts, like the adjacency pairs studied by conversationalists (e.g. *question-answer* and *request-accomplishment*, see Sacks *et al.* (1974)); it also covers other more complex phenomena, among which are the occurrence of subdialogues after a question (e.g. the insertion sequences: *question-question-answer-answer,* see Schegloff (1972)) and some notifications. Consider the following example:

[6] Note that our goal adoption relation is more restrictive than that of Castelfranchi & Falcone (1997): it corresponds to their literal conservative adoption.

[7] Note that the goal pursued by the interlocutor is inferred from the speaker's utterances in both the goal adherence and adoption cases. However, the difference is that the goal adherence relation covers the situations where the interlocutor infers that the speaker has explicitly communicated that he wants to delegate him to achieve the goal.

Example 4.1.1

A: Could you give me a pass for the CS laboratories?

B: a) Are you enrolled in a program of the CS Department?

B: b) *(he checks on the computer whether A is in the enrollment list)*

B: c) Can you show me your enrollment card?

B: d) Here you are *(he gives A the pass)*

B: e) No, you are not enrolled in any program of the CS Department.

Here, A requests the clerk B to give him a pass (he does so indirectly, by means of a surface question); 4.1.1.(a - e) is a list of alternative replies of B to the request. B's reactions derive from his intention of executing the requested action, but they are intrinsically different: 4.1.1.a is a question; 4.1.1.b is a domain-level action; 4.1.1.c is a request (expressed as a surface question); finally, 4.1.1.(d,e) are assertions playing opposite roles. Nevertheless, under the hypothesis that the interlocutors are cooperating, it is possible to recognize the inner rationality of each reaction, and to explain how it contributes coherently to the interaction. The explanation is provided by analyzing the turns from the point of view of agent behavior, as described in our Agent Modeling plans. All the descriptions below are based on the (cooperation) hypothesis that B is trying to satisfy A's overt goal to be given a pass:

4.1.1.a: the Agent Modeling plans prescribe that, to execute an action, an agent (B) must first check its preconditions; as specified in the Domain library, "Give-pass" has the precondition that the addressee is enrolled in a program of the CS Department; therefore, the question "Are you enrolled ... ?" can be interpreted as a way to check this precondition.

4.1.1.b: as above, but the "check precondition" activity is carried out by executing a domain-level action ("reading the enrollment list") instead of a linguistic action.

4.1.1.c: again, an attempt to check a precondition; the domain knowledge must specify that somebody can read from the enrollment card which program a student is enrolled in, and that before reading something you have to see it. Moreover, showing y to x has the effect that x sees y. The Agent Modeling plans describe how these domain-level relations can be used for checking the mentioned precondition.

4.1.1.d: this behavior follows from the conditional goal of "notifying" the successful execution of a plan, after its completion; the notification task is the last step of the Agent Modeling plan for achieving goals. If the object-level plan succeeds and it is executed on behalf of another agent, that agent should be informed about the success.

4.1.1.e: the converse of 4.1.1.d: during the execution of the requested action, B has discovered that a constraint of the action "Give-pass" is false and, since he is collaborating with A, he informs the partner that the action cannot be executed.

The goal adherence relation also models acknowledgements to information acts, repetitions, and other grounding behaviors (Traum & Hinkelman 1992): we interpret the illocutionary act of "informing" as expressing the speaker's communicative intention that the hearer update his beliefs with the new information. Therefore, the hearer's subsequent reactions can be interpreted as positive, or negative, notifications regarding the updating action. Consider the following example:

Example 4.1.2

A: Dr. Smith is in office number 214.
B: a): mhm
B: b): Oh
B: c): 214.

Here, B recognizes A's goal of making mutually believed the information conveyed in his statement; so, similar to what happens for domain actions (see turn C-2 in Example 3.1), he notifies A that he has updated own his beliefs. The goal of letting A know that B now believes the information makes B plan a suitable action for grounding the dialogue, depending on the context where the "updating" action occurs: for example, differently from the unmarked 4.1.2.a, the reaction "oh" in 4.1.2.b is used when B previously believed that Dr. Smith was not in that office (see Heritage (1984)); finally, in 4.1.2.c, B performs a grounding act by repeating the main information conveyed in A's sentence (see Clark & Schaefer (1989)).[8]

4.2 Goal adoption

Goal adoption models spontaneous cooperative behavior; it covers many dialogue phenomena, among which: extra-helpful behavior, the displays of behavioral and conversational cooperation among agents, the occurrence of presequences, subdialogues, notifications and acknowledgments for grounding purposes; this relation is also related to repairs of misunderstandings. We have treated these phenomena in Ardissono *et al.* (1997) and Ardissono *et al.* (1998).

First, we consider overanswering, a phenomenon widely studied in the plan recognition literature about information-seeking dialogues: see Allen (1983) and Carberry (1990b). In overanswering, the receiver of a question volunteers some extra-helpful information which was not asked explicitly by the information seeker. The rationale behind this fact is that, by reasoning about the partner's plans, the speaker infers that the partner will also aim

[8] Many subdialogues addressing the believability of statements can result from the failure of the "Update" action; for example, the execution of "Update" can detect the presence of a previous belief incompatible with the new information (Lambert 1993):
A: CS144 is taught by Prof. Smith
B: But wasn't Prof. Smith on sabbatical?

at other goals (not only concerning knowledge) because they are crucial to his plans. For instance, compare the two alternative responses in the following example:

Example 4.2.1

A: Could you give me a pass for the CS laboratories?

B: a): Sorry, we have finished the passes for today. Ask your tutor for a written permission to enter the laboratory.

B: b): Sorry, today the laboratories are closed.

In 4.2.1.a, the clerk (B) notifies the student that it is impossible to perform the requested action (similar to 4.1.1.e); then, he suggests an alternative plan to satisfy A's goal of entering the labs. Note that B performs a request deriving from his adoption of A's higher-level goal to enter the labs, and therefore, to have a permission for that. On the other hand, B's request does not derive from the goal expressed explicitly by A, which is that B provide him with a pass.

In 4.2.1.b, B's reply does not satisfy A's request of being given a pass, nor his goal of having it, but that information is useful to A: B has understood A's plan to work in the labs and he adopts the goal that A knows that they are closed; in fact, it is possible to work in the labs only when they are open, and A will have to check this constraint before performing the action.

As pointed out in Airenti *et al.* (1993), the cooperation among agents can be clearly divided into behavioral and conversational: an agent might refuse his cooperation at the domain level, but he may cooperate at the shared conversational activity. In particular, goal adoption can explain the turns where an agent states his willingness to cooperate at the behavioral level, as well as refusals. For example:

Example 4.2.2

A: Could you give me a pass for the CS laboratories?

B: a): Here you are *(he gives A the pass)*.

B: b): Ok, just one moment.

B: c): Sorry, but I am busy now.

In 4.2.2.a, B directly executes the action and notifies A about his success.

In 4.2.2.b, B anticipates A's intention to know whether B is going to perform the requested action by letting him know that he's just about to do it. In fact, when an agent tries to make somebody else intend to perform an action, the agent's success is not guaranteed: as for any other action, he may fail (Smith & Cohen 1996). This uncertainty causes the agent to check whether the intended effect of his attempt has been achieved. In this specific example, B's utterance provides A with the feedback he needs before A solicits him.

On the contrary, in 4.2.2.c, B refuses to cooperate at the domain level (i.e. to perform the requested action), but he cooperates conversationally by telling A that he does not intend to help him.

A particular role is played by the goal adherence and adoption relations when the object-level plan in execution is a linguistic action: as for domain-level actions, the constraints of linguistic actions must be checked, by means of other speech acts. This provides an explanation of the phenomenon known in conversational analysis as *presequences* (Schegloff 1979, Levinson 1981). Various conditions are monitored in the preparatory activity before the execution of a linguistic action:

- The speaker monitors the receiver's attention, which is a precondition of the action of uttering a sentence with a communicative goal. In the following dialogue, an agent A tries to attract his partner's attention; in 4.2.3.a, A checks whether the precondition of the uttering action holds, while in 4.2.3.b he makes it true by calling his partner:

  Example 4.2.3
  A: a) Are you listening to me?
  A: b) John!
  B: Yes!
  A: Can you give me a pencil?

- In the case of requests, a speaker has to check that the preconditions of the requested action hold; in the following dialogue, A anticipates his request for a pass by asking B whether he has any:

  Example 4.2.4
  A: Do you have any passes for the laboratories?
  B: Yes.
  A: May I have one?
  B: Here you are.

- In the case of actions of informing, the speaker has to check whether the hearer already knows the information he wants to communicate to him (this is a constraint of the action of informing):

  Example 4.2.5
  A: Do you know where I've been yesterday?
  B: No.
  A: I've been to Liza's party.

If the above dialogues represent typical presequence patterns, in collaborative interactions the receivers of a turn initiating a presequence often anticipate their partners, so producing more compact dialogues. This behavior is only possible if agents perform a plan recognition activity to infer their partners' goals and adopt them. Consider the following examples:

Example 4.2.6

A: John!

B: Yes, tell me.

Example 4.2.7

A: Do you have any passes for the laboratories?

B: a) Here you are *(B gives A a pass)*

B: b) Do you want one?

Example 4.2.8

A: Do you know where I've been yesterday?

B: No, where?

In 4.2.6 and 4.2.8, B invites the first speaker to talk; in 4.2.7.a, he anticipates A's expression of his intention to have a pass by satisfying it directly; finally, in 4.2.7.b, B anticipates A by offering him the pass.

The goal adoption relation also explains subdialogues, notifications and acknowledgements occurring when the receiver of an utterance interprets it. In fact, if a speaker A performs an utterance act, he has the goal that the receiver B interprets it. Therefore, if B is cooperative, he adopts the interpretation of the utterance as a joint goal, and performs an "Interpret" action. This goal involves the identification of the syntactic and semantic interpretation of the utterance, and also of its illocutionary force; moreover, since the utterance represents a turn of a dialogue, also the coherence relation with the previous intentional context has to be determined. So, the coherence of subdialogues concerning interpretation problems and the notification of the successful execution of the interpretation action are explained by the goal adoption relation, too. These notifications can be performed as in 4.2.9.d, or by means of backchannels,[9] as in 4.2.9.e (see Schegloff (1982)).

Example 4.2.9

A: The professor teaching the AI course is on sabbatical.

B: a): What does sabbatical mean?

B: b): The professor of the advanced course?

B: c): Why are you telling me?

B: d): Ok, I've understood.

[9] Backchannels are also used for notifying the speaker that the receiver is paying attention to him: in such a case, they are explained by a goal adoption relation satisfying this specific precondition of uttering a sentence; compare this fact with the discussion about Example 4.2.3.

B: e): mhm

The "Interpret" action has the precondition that the receiver knows the meaning of all the words occurring in the sentence and he can resolve the references unambiguously (see Heeman & Hirst (1995)). In 4.2.9.a above, *B* tries to satisfy the first precondition by asking *A* what is the meaning of "sabbatical"; instead, in 4.2.9.b, *B* tries to disambiguate the reference of the phrase "AI course". These replies initiate two subdialogues and they resemble turn C-1 of Example 3.1. Also 4.2.9.c initiates a subdialogue aiming at understanding the reason behind *A*'s statement.

4.3 Plan continuation

This relation explains the coherence of a turn representing the continuation of an agent's course of action, which may concern a metalevel or an object-level plan.

The ordinary continuation of a plan, as analyzed in most plan recognition frameworks, corresponds to the sequential execution of a domain-level recipe: for example, a student who wants to have credit for an exam first does the laboratory work and then he takes the exam. On the other hand, Example 3.1 provides two instances of plan continuation at the Agent Modeling level: the user's turns U-1 and U-3 contribute to the satisfaction of the applicability conditions of the "Take-university-exam" action; the consultant's turn C-1 aims at learning the values of the arguments of the requested action; C-2 represents a notification of the successful execution of the action.

The continuation of an agent's metalevel activity is useful for explaining the occurrence of requests for grounding a dialogue: for instance, the grounding actions of Example 4.1.2 can be solicited by the speaker if he wants to know whether his partner has understood or not; see the example from Traum & Hinkelman (1992):[10]

Example 4.3.1

(A and B discuss about a plan to move some orange tanks from town to town)

A: so there's an engine at Avon, right?

B: right

Finally, plan continuation can be performed for linguistic actions, too: although in the paper we have considered "single-step" linguistic actions, our speech act library also includes recipes for complex linguistic actions, which are composed of more than one linguistic action. We have introduced these complex speech acts to model structured discourse, where a speaker may utter a sequence of speech acts related in an intentional structure for rhetorical purposes (Maybury 1993, Moore & Pollack 1992). For example,

[10] As noted by a reviewer, in Traum and Hinkelman's example, only this interpretation of "right?" is considered; the phrase also can be interpreted as an expression of doubt about the truth of the proposition, a phenomenon we do not currently consider.

in order to motivate a request, a speaker may select a recipe composed of an assertion about the underlying domain task and the real request:

Example 4.3.2

A: I have to work in the labs. May I have a pass to enter them?

A turn composed of such a sequence of speech acts must be interpreted by identifying a single discourse plan which explains both utterances. This interpretation process is the same as the one performed to recognize a sequence of actions belonging to a single domain-level plan.

5 Representation of actions

The domain, linguistic and agent modeling actions belong to three separate plan libraries, which store the pre-compiled recipes for executing complex actions.

The *Domain plan library* describes the recipes for reaching the domain goals in a restricted domain; we chose the University domain as a testing domain; see Ardissono *et al.* (1993) and Ardissono & Sestero (1996).

The *Speech Act library* describes the speech acts (and the discourse plans) with particular attention to the direct and indirect approaches to communication and politeness techniques (Ardissono *et al.* 1995a, Ardissono *et al.* 1995b).

The *Agent Modeling library* describes a reactive planner used for modeling agent behavior and will be described in detail later on.[11]

We are also working on the definition of some object-level (domain-independent) actions representing "internal" activities of agents: among them, there are the "Update" action, that represents the updating of an agent's beliefs, and the "Interpret" action, that represents the interpretation of the turns of a dialogue.

All the libraries are organized on the basis of a Decomposition and a Generalization Hierarchy (Kautz 1991, Weida & Litman 1992). The first hierarchy specifies how a complex action can be performed by executing simpler actions. The Generalization Hierarchy supports feature inheritance among actions: this hierarchy allows the representation, at a higher level of abstraction, of generic actions associated with different behaviors, depending on the context where they are executed. For example, the generic action that represents taking an exam can be specialized into an action for taking a theoretical or an experimental exam, which have different recipes: for instance, the latter contains the step of doing laboratory work.

Actions are represented by means of the same formalism in the three plan libraries, so that the same procedures can be used to work and reason about them. There are four

[11] In Ardissono *et al.* (1996), a preliminary version of the metalevel plans was introduced and it was referred to as "Problem-Solving" library.

1) A generic action from the Domain library (Take-university-exam):

name:	**Take-university-exam**
roles:	((agt, a, person)(obj, ex, exam))
constraints:	enrolled(a, Turin-university)
preconditions:	registered(a, ex)
effects:	has-credit(a, ex)
specific-actions:	((**Take-expe-exam** (a, ex) where experimental(ex)) (**Take-theor-exam** (a, ex) where theoretical(ex)))

2) A simple action from the Speech Act library (Surface-imperative):

name:	**Surface-imperative**
roles:	((spkr, s, person)(hear, h, person)(obj, content, action))
body:	(()(block (**Locutionary-act** (s, h, content)) where (imperative(content) $\wedge$ action(content) $\wedge$ agent(h, content))))
is-substep-of:	((Direct-request, 1))

3) A complex action from the Agent Modeling library (Try-execute):

name:	**Try-execute**
roles:	((agt, a, person)(source, s, person)(action, act, action))
body:	(()(block (**Check-constr** (a, s, act)) (**Verify-prec** (a, s, act)) (**Perform-action** (a, s, act))))
is-substep-of:	((Self-act, 0.3)(Perform-body-step, 0.7))

FIGURE 3. Representation of some of the actions of the libraries.

action types, depending on their position in the hierarchy:

- *Complex actions*: a complex action A includes a decomposition formed by some substeps B_1, B_2, ..., B_n, where $n > 1$. There is no action C more specific than A.
- *Simple actions*: a simple action A includes a decomposition formed by a single substep D. In Pollack's terms, D *generates* A (Pollack 1990). There is no action C more specific than A.
- *Generic actions*: for each generic action A there are some actions C_1, C_2, ..., C_n (with $n \geq 2$) such that A is more generic than C_i for $1 \leq i \leq n$. Generic actions have no decompositions.
- *Elementary actions*: they have no decompositions and no specializations.

The definitions above are rather standard. The only thing to notice is that we decided to keep apart the Decomposition and Generalization hierarchies: decomposition and specializations cannot be specified for the same action; moreover, decompositions are not inherited. Basically, we adopted this restriction in order to simplify the management of feature inheritance in our framework.[12]

In any of the three plan libraries, the representation of actions includes the following features:

- the action name;
- the roles (participants) of the action, characterized by:
  - a role name;[13]
  - an internal variable associated with the role;
  - the type restriction on the role fillers;
- the type declaration of other variables occurring in the action (if any);
- the applicability conditions:
  - *Wh-restrictions*: they constrain the relations holding among the role fillers. For instance, the domain-level restriction "has-office(teacher office)" constrains the two variables to take values such that *"office"* denotes the office of *"teacher"*.
  - *Constraints*: they are conditions that specify in which contexts the action can be executed; like the wh-restrictions, constraints cannot be made true if they do not hold; i.e. they cannot be adopted as subgoals by an agent.
  - *Preconditions*: they are conditions of actions like constraints, but if the agent believes they are false, he can adopt them as subgoals and try to satisfy them.
- the effects of the action, i.e. the conditions that may become true after it is executed successfully. If an action execution fails, its effects do not affect the world state.
- the output variables, bound as a side effect of the execution of the action. They are representation-internal: their main purpose is to assign values to variables, in order to enable the execution of a subsequent step of the plan.
- the more generic action, i.e. the name of the immediate ancestor of the action in the Generalization Hierarchy.
- the more specific actions: i.e. the list of the actions more specific than the action described (in the Generalization Hierarchy); each more specific action is characterized by a condition distinguishing it from its alternatives.

[12] See (Weida & Litman 1992, Devanbu & Litman 1996) for an action representation language which takes into account inheritance of recipes, and other features, like temporal constraints.

[13] The role names have no particular impact within the representation of the action, but they are fundamental for establishing the correspondence between actions and linguistic expressions: role names must correspond exactly to the θ-roles of the verb (where such a verb exists).

- the decomposition: the decomposition (body) of an action contains a number of steps, representing occurrences of simpler actions, which have to be executed in order to perform the main action. In the decomposition, the parameters of the steps can be shared with those of the main action, or they can be strictly related to the action parameters by explicit wh-restrictions. The variables shared between an action and its steps are unified when the action instance is created; the others are bound to values which satisfy the wh-restrictions where they occur.

  The body of a complex action may consist of a linear sequence of steps, or of a cyclic block; in this case, we can have either an iterative execution of the action body ("for each" loop) with a loop variable which takes its value from a given range; or, we can have a "while" loop, that terminates when the loop predicate becomes false.[14]

  In each step of the recipe, there may occur further restrictions relating the step parameters with those of the action whose decomposition is described.

  In the decomposition of an action, there can be some conditional steps, which are executed only when their execution condition holds. The execution condition of a step "s" is reported in the "if" subfield of the occurrence of "s" in the body of the main action.

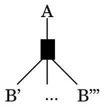
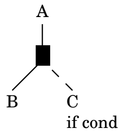
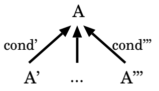
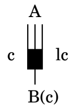
- the "is substep of" list: this is a list of <action-name, probability> pairs which express the probability that the described action occurs as a step of a higher-level action, denoted as *"action-name"*.

Figures 3 and 9 show the schemata of some actions in the representation formalism of our system. Figures 5, 6, 7 and 8 show some portions of the plan libraries in a pictorial and simplified form: the graphical notation is described in Figure 4. In the figures, the capitalized names are used to denote constants. Note that, for space reasons, the role labels have been omitted in all the figures representing the plan libraries and the interpretation of utterances.

5.1 The Domain plan library

This plan library describes the knowledge which guides an agent in reaching his domain goals, like earning a degree in Computer Science, taking the exams, and so on. It contains the precompiled recipes describing the typical well-formed plans to achieve some domain goal (currently, it includes about 100 actions); since the structure of this plan library and its content are very similar to other well-known solutions, like Carberry (1990b), we will just sketch the recipe for taking an exam. Figure 5 shows pictorially the plan for the

[14] Providing a formal definition of our representation of actions is future work. See DeGiacomo & Lenzerini (1995) for a definition of a logic of actions which includes features like conditional actions and repetition of actions.

	Action A has as a decomposition the plan composed of the actions B', ..., B'''.
	A's plan has conditional step C which must be executed only when condition cond is true
	A has some more specific actions, A',..., A''' which can be executed if, respectively, cond', ..., cond''' are true
	"For Each": to complete action A, B must be repeated for each element c of the list lc

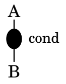
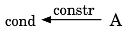
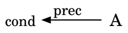
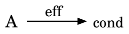
	"While": to complete A, B must be repeated until cond is false
	"Constraint": if cond is false, A cannot be executed
	"Precondition": if cond is false, A cannot be executed, but the agent may act to make cond true
	"Effects": cond becomes true after a successful execution of A
A [x <-]	As a side effect of the execution of A, the variable x is assigned a value

FIGURE 4. The symbols used in the figures of actions.

"Take-university-exam" action, which is specified into two alternative actions, according to the type of the exam to be taken; for the real representation of this action, see the first action schema in Figure 3.

1. "Take-expe-exam" describes the recipe for taking an experimental exam: first, the agent must do laboratory work ("Do-lab-work"); then, he must take an oral exam ("Take-oral-exam"). In order to do the laboratory work, he must enter the labs ("Enter") and implement a prototype ("Write-program"). The "Enter" action has the constraint that the labs are open and the precondition that the agent has the pass for accessing them; the action also has the effect that, after having performed it successfully, the agent is at the labs.

2. 'Take-theor-exam" describes the recipe for taking a theoretical exam. In that case, the agent must write a report on a topic of interest for the exam ("Write-report"); then, he must take the written and the oral parts of the exam ("Take-written-exam" and "Take-oral-exam").

5.2 THE SPEECH ACT LIBRARY:

In this library, about 50 linguistic actions are represented, like in the Domain plan library, as actions which an agent tries to perform in order to change the world state: in this case, the hearer's beliefs and intentions. We model communication by introducing an action hierarchy where the higher-level actions regard the speakers' activity aimed at making interlocutors acquire intentions; the lower-level actions describe the means which can be

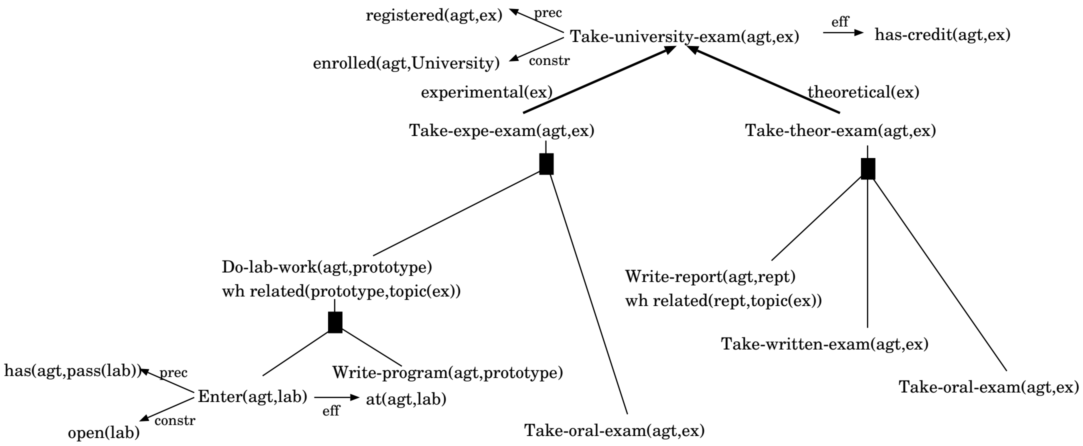


FIGURE 5. The “Take-university-exam” recipe.

adopted in order to perform the higher-level actions, developing them down to the surface forms which can be used in utterances.

- At the top level of the hierarchy there is the “Get-to-do” action, which describes the agent’s behavior to induce the hearer to act in a certain way (Smith & Cohen 1996). “Get-to-do” has the effect that the hearer intends to act as specified in its “*content*” parameter. Note that we model standard declarative (information-providing) sentences as expressions of the intention that the partner believe something; more precisely, as an intention that he perform an “Update” action on his own beliefs; therefore, “Update” can be the argument of “Get-to-do”, like any other object-level action: in this case the “Get-to-do” will be decomposed into an action of informing.
  “Get-to-do” has some more specific actions, representing alternative ways to induce people to act: they are associated with different motivations for acting (e.g. bare cooperativity, fear) and they are not necessarily linguistic. Figure 6 shows only one more specific action, “Cooperative-get-to-do”, which relies on the hearer’s willingness to adopt the speaker’s intention: this restriction is specified in the condition “cooperative(hear, spkr, content)” that distinguishes “Cooperative-get-to-do” from the other alternative actions more specific than “Get-to-do”.

- The actions more specific than “Get-to-do” are performed by means of one or more illocutionary acts. By definition, an illocutionary act establishes a minimal effect: making the interactants share that the speaker has the communicative intention that he intends that the hearer intends to act in a certain way. For example, “Cooperative-get-to-do” is performed by means of a request to execute an action. “Request” has the effect that the interlocutors mutually believe that the speaker has the communicative intention that the speaker intends that the hearer intends

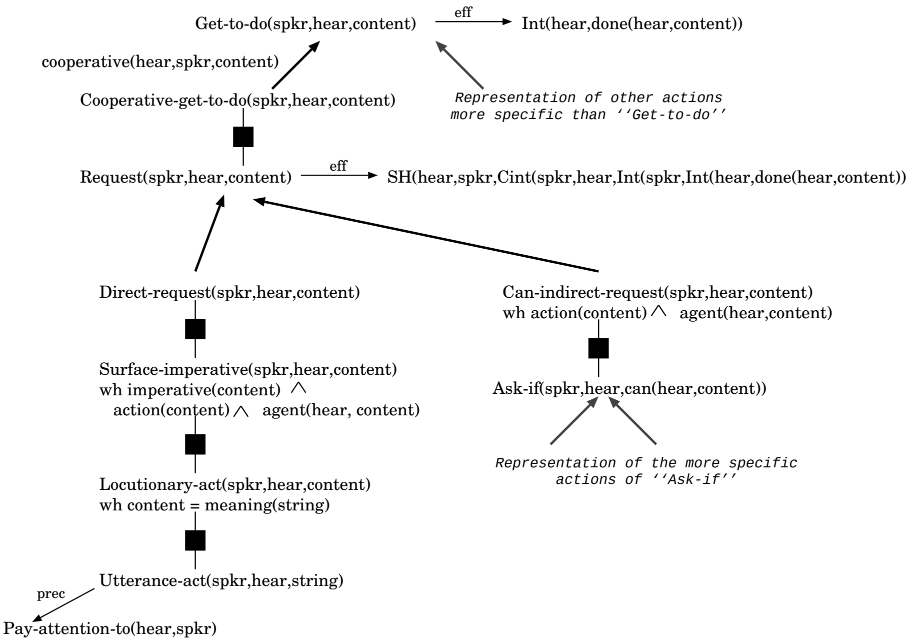

Get-to-do(spkr,hear,content)
eff
Int(hear,done(hear,content))
cooperative(hear,spkr,content)
Cooperative-get-to-do(spkr,hear,content)
Representation of other actions more specific than ''Get-to-do''
Request(spkr,hear,content)
eff
SH(hear,spkr,Cint(spkr,hear,Int(spkr,Int(hear,done(hear,content))
Direct-request(spkr,hear,content)
Can-indirect-request(spkr,hear,content)
wh action(content) ∧ agent(hear,content)
Surface-imperative(spkr,hear,content)
wh imperative(content) ∧
action(content) ∧ agent(hear, content)
Ask-if(spkr,hear,can(hear,content))
Representation of the more specific actions of ''Ask-if''
Locutionary-act(spkr,hear,content)
wh content = meaning(string)
Utterance-act(spkr,hear,string)
prec
Pay-attention-to(hear,spkr)

FIGURE 6. A fragment of the Speech Act library representing requests.

to perform the action described by "*content*".[15]

- The illocutionary acts are specialized into the means for performing them, which include indirect forms: most illocutionary acts can be expressed via different surface forms, according to the adopted politeness strategy (Ardissono *et al.* 1995a). Figure 6 shows a direct and an indirect way of performing requests:
  - "Direct-request" describes the case where the speaker does not bother about the hearer's face (Brown & Levinson 1987); its direct decomposition is a surface speech act in imperative mood.
  - "Can-indirect-request" is more polite: it is executed by questioning the hearer about his capability of doing the requested action, as in "Could you open the door?": this is represented by the occurrence of "Ask-if" in its decomposition: note that the third argument of "Ask-if" is "can(hear, content)", while it was "*content*" in "Can-indirect-request".

[15] Consider the effect of the "Request" action in Figure 6. The communicative intention operator, "Cint", was introduced in Airenti *et al.* (1993) to model the Gricean notion of communication (Grice 1957): $Cint_{xy}\ p \equiv Int_x\ SH_{yx}\ (p \wedge Cint_{xy}\ p)$.
In the formula, "SH" is the modal operator of mutual belief: $SH_{xy}\ q \equiv Bel_x\ (q \wedge SH_{yx}\ q)$. If an agent x has the communicative intention p towards another agent y, then x has the intention that it is a common belief among the two that p holds, and that x had that communicative intention towards y. So, x wants it mutually believed that p, and that she wanted to communicate it.
The "done" operator in the formula has two parameters: an agent and an action performed by the agent; the operator maps the action on the world state after the action has been executed successfully.

Actually, "Ask-if" is also the way of asking real questions; so, it also appears in the decomposition of the "Obtain-info" action, the illocutionary act used to acquire information. Moreover, "Ask-if" has different more specific actions for asking polite questions. In this way, progressively more polite speech acts can be obtained; for example, consider the request "I would like to ask you if you could lend me some money", expressed by means of an assertion that performs a polite question.

- All the surface speech acts are generated by the "Locutionary-act" action: as described in Austin (1962), this is the action of communicating a propositional content. The surface speech acts are distinguished from one another by the illocutionary force indicator devices occurring in their propositional content; these devices are typically the sentence mode (e.g. declarative, interrogative), the verb mood and the occurrence of modals. For example, as shown in Figure 6, the "Surface-imperative" act is generated by a "Locutionary-act" if the syntactic/semantic representation of the propositional content contains the feature "imperative mood" and concerns an action having the hearer as its agent: "imperative(content) $\wedge$ action(content) $\wedge$ agent(hear, content)" in the wh-restriction.[16]

- In turn, "Locutionary-act" is generated by an "Utterance-act", where the input sentence, which is contained in the parameter "*string*" (that is bound to a text) has "*content*" as its meaning: see the restriction "content = meaning(string)".

Note that the "meaning" predicate has a dual role in the interpretation and generation task; in the interpretation phase, as in the current implementation, this predicate is satisfied by invoking the parser and the semantic interpreter (Di Eugenio & Lesmo 1987); so, its "*string*" parameter is bound to the input string, while "*content*" becomes bound after the analysis. Dually, in a generation process, "*content*" should be bound to the semantic representation of the locutionary act and the "meaning" predicate would produce a call to a surface linguistic generator, which would bind "*string*" to the result of the generation task.

5.3 The agent modeling library

The Agent Modeling actions represent the problem solving recipes used by an agent to plan and execute actions for reaching his goals. In the plan recognition perspective, such recipes are used to interpret an observed behavior of an agent, under the assumption that all agents act following the same recipes for plan formation and execution.

[16] In the figure only the link with the surface imperative is represented, but "Locutionary-act" occurs with different restrictions in the decomposition of other actions: "Surface-assertion", "Surface-yn-question" and "Surface-wh-question".

The AM plans are precompiled and contain about 50 actions. We have defined the Agent Modeling plans as recipes describing the behavior of the modeled agent in a complete way; i.e. they are the only knowledge source describing the system's goal-directed behavior: in this way, we can recognize the intentions underlying many dialogue phenomena which occur between agents who cooperate at a single high-level goal; at the same time, however, we avoid a further level of knowledge necessary for building the Agent Modeling recipes themselves.[17]

The fixed nature of our Agent Modeling plans requires that they take into account all the possible outcomes of a given behavior; therefore, the AM plans may contain conditional steps; moreover, many AM actions have several more specific actions, each one suited to a different execution context.

The Agent Modeling plans defined in the library have an homogeneous structure: any step of the plan "enables" a subsequent one, although this is not made explicit in the representation; e.g. the step of executing a plan comes after the one for finding an executable plan. For this reason, although these plans are precompiled, they are not just scripts describing typical behavior patterns.

In principle, we could have justified the relation between the various steps by defining their effects and preconditions (as in fact happens in the D and SA libraries); for instance, referring to the steps of "Satisfy" in Figure 7, "Find-plans" could have been defined as having the effect "have(a, plans)" and "Cycle-do-action" as having the precondition "have(a, plans)". However, this information cannot be used by the system, since, differently from the object-level actions, the Agent Modeling actions do not undergo a standard "planning and execution" process. In fact, since AM plans define the planner, this could have led to an infinite recursion, if applied to themselves. So the enablement information is left implicit, but it stands at the basis of the AM plans that have been devised.

The main Agent Modeling actions are "Satisfy", which describes the whole activity of an agent who reasons and acts to achieve an object-level goal, and "Try-execute", which describes the execution of a selected action.

The "Satisfy" action.

The "Satisfy" action, shown in Figure 7, represents the behavior of an agent working to achieve a goal. "Satisfy" models a fanatical behavior, in fact an agent who adopts a goal "g" tries to make true the desired state of affairs until he comes to believe that he has succeeded, or that it is impossible to achieve the goal (Rao & Georgeff 1992).

[17] Our system does not deal with agent behavior caused by the presence of multiple high-level intentions: e.g. we do not model agents who carry on different unrelated plans in parallel, abandon some of these plans, change their high-level commitments; these events are related to phenomena like interruptions and temporary topic changes, typically signaled by the occurrence of cue words and other linguistics markers which help to identify changes in the agents' intentions.

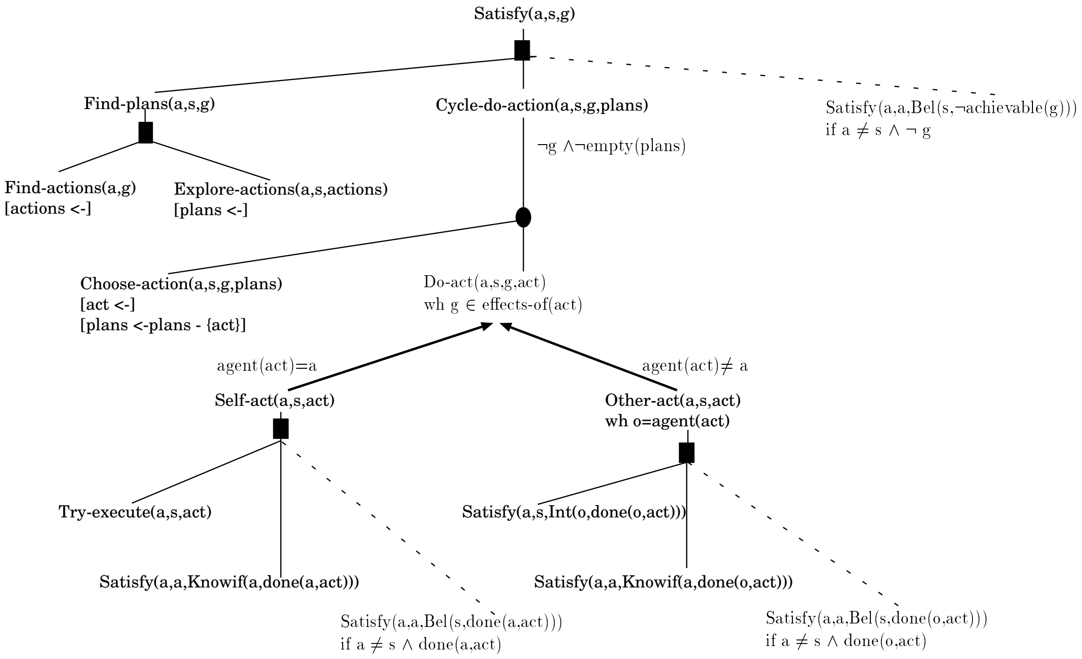


FIGURE 7. The “Satisfy” agent modeling action.

While the first parameter of “Satisfy”, “*a*”, represents the modeled agent, and the third one the goal “*g*” to be achieved, the second parameter, “*s*”, is used to express the fact that the agent “*a*” is committed to a joint intention with “*s*”; as discussed in Cohen & Levesque (1991), when an agent “*a*” pursues a goal for the sake of a “source” “*s*”, he adopts some supplementary goals, in order to keep “*s*” informed about the execution of the action. When he is acting for his own sake, the “*s*” parameter is bound to the same value as “*a*”. The “source” parameter occurs in most of the Agent Modeling actions to model this form of cooperation. “Satisfy” has three steps:

1. “Find-plans” consists of:

   a) Picking out the domain or linguistic actions having the desired goal “*g*” as an effect (“Find-actions”);[18]

   b) Exploring these actions in order to obtain a set of (possibly partial) plans for “*g*” (“Explore-actions”). In the exploration task, which recalls the exploration strategies described in Ramshaw (1991) and Carberry *et al.* (1992), the feasibility of the actions is checked and they are partially expanded, if they are complex; moreover, the preferred plans are selected.

2. “Cycle-do-action” describes the execution of the selected plans, until either the de-

[18] Some of the actions appearing as elementary in the AM library (e.g. like “Find-actions”, “Choose-action”, etc.) could be expanded by means of complex subplans; see Sestero (1998).

sired state of affairs "*g*" is reached, or all the alternatives have failed.[19] At each iteration, one of the candidate actions is selected in the "Choose-action" step; this step, which is currently present in a simplified form, also should contain the check that the action is compatible with the other goals of the agent (Bratman *et al.* 1988). After the selection of "*act*", the recipe continues with its execution: "Do-act" describes this execution. Note that the wh-restriction "g $\in$ effects-of(act)" is of paramount importance during the plan-recognition activity. Two continuations may follow, depending on which agent has been selected by "*a*" as the agent of the action "*act*" to be performed:

a) If the agent of "*act*" is "*a*" himself, "*a*" performs "Self-act", by means of "Try-execute".

b) Otherwise, "*a*" performs "Other-act": "*a*" acquires the AM goal to obtain that another agent "*o*" intends that the selected action is executed (by "*o*" himself): "Satisfy(a, s, Int(o, done(o, act)))".[20] In order to satisfy this goal, the agent "*a*" can choose to perform a suitable linguistic action more specific than "Get-to-do", aiming at inducing his partner to act. These possibly complex communicative acts are subject to the same planning and (reactive) executing process as the domain-level actions, by means of "Try-execute".

In both cases, after the execution of the action, which may have been performed by the agent "*a*" or by his partner, "*a*" checks whether the action was successful or not: see the two occurrences of "Satisfy" in the body of "Self-act" and "Other-act". In fact, the successful execution of an action does not necessarily imply that its effects hold, so that its result must be monitored. As an example, when an agent takes an exam, if he gets a negative mark he does not get the credit for that exam: for this reason, before considering his plan completed, he still needs to check the effect of the "Take-university-exam" action. In particular, if "*a*" has asked his partner to execute an action, the execution of this step can lead him to adopt linguistic behavior like soliciting feedback from the partner. This is one of the reactive aspects of the planner, that must recover from a failure in action execution by replanning its behavior. Also other parts of the Agent Modeling library contain actions for checking whether the object-level actions have been performed successfully.

The third (conditional) step occurring in "Self-act" and "Other-act" describes one of the subsidiary goals due to the presence of a joint intention between "*a*" and "*s*": if "*a*" has successfully executed the action on behalf of "*s*" (e.g. "done(a, act)" in

[19] "Cycle-do-action" is represented in detail in Figure 9, where the cycle decomposition is denoted by the "cycle-block" label, followed by the loop condition.

[20] For the meaning of the "done" operator, see footnote 15.

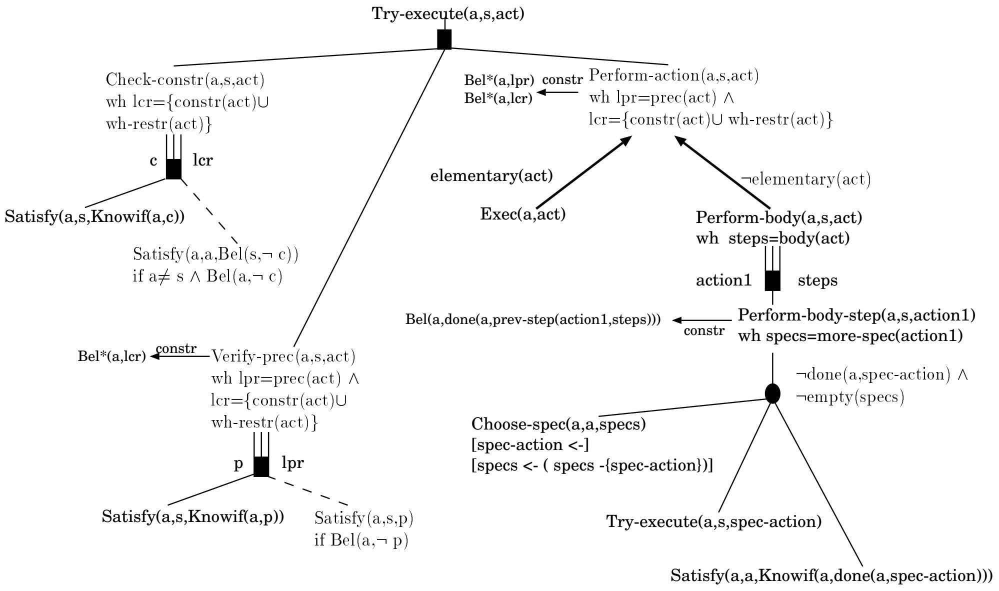


FIGURE 8. The "Try-execute" agent modeling action.

the figure), he acquires the goal to let the partner know about the success. This goal may lead "*a*" to a further planning and to the execution of a suitable action in order to inform the partner accordingly.

3. The third step of "Satisfy" models another subsidiary commitment induced by a joint intention between "*a*" and "*s*": if all the actions for the initial goal "*g*" have failed to reach their effects and the agent "*a*" shares "*g*" with a partner "*s*", then he will execute a "Satisfy" action on the goal of letting "*s*" believe that "*g*" cannot be achieved: "Satisfy(a, a, Bel(s, ¬achievable(g)))".

The "Try-execute" action.

The "Try-execute" action, shown in Figure 8, describes the whole process of executing an object-level action (denoted as "*act*" in the figure) to which may be associated a partial plan previously built in the exploration phase ("Find-plans"); this process consists of checking the prerequisites for its object action and performing the action steps, if there are any.

1. First, the agent checks whether the constraints of "*act*" are true ("Check-constr(a, s, act)"), which is done via a sequence of "Satisfy(a, s, Knowif(...))" actions. If any of these conditions is discovered to be false, the agent possibly notifies his partner about the failure and gives up the execution of the "Try-execute" plan; in fact, the constraint "Bel*(a, lcr)"[21] of the subsequent Agent Modeling step of "Try-execute" ("Verify-prec") is true only if the agent believes that the constraints of the

[21] The "Bel*" operator means that "*a*" believes that each of the conditions in the "*lcr*" list is true.

Cycle-do-action:

```
name:              Cycle-do-action
roles:             ((agent, a, person)(source, s, person)(goal, g, state)(plans, p, plan-list))
body:              (((action act))
                    (cycle-block (¬ g ∧ ¬ empty(p))
                     (Choose-action (a, s, g, p))
                     (Do-act (a, s, g, act)
                       wh g ∈ effects-of(act))
is-substep-of:     ((Satisfy, 1))
```

Check-constr:

```
name:              Check-constr
roles:             ((agent, a, person)(source, s, person)(action, act, action))
var-types:         ((condition-list lcr))
wh-restrictions:   lcr = constraints-of(act) ∪ wh-restrictions-of(act)
body:              (((condition c))
                    (iter-block c lcr
                     (Satisfy (a, s, Knowif(a, c)))
                     (if (a ≠ s) ∧ (Bel(a, ¬c)))
                       Satisfy (a, a, (Bel(s, ¬c)))))
is-substep-of:     ((Explore-actions, 0.3)(Try-execute, 0.7))
```

FIGURE 9. Detailed representation of Agent Modeling actions.

object-level action are true; as already discussed, "*a*" knows the truth values of the conditions in "*lcr*" because he has previously checked them, when he has performed "Check-constr".[22]

2. The preconditions of an action can be adopted as subgoals by the agent; so, the "Verify-prec" plan checks the truth value of these conditions and also tries to satisfy them, if they are false (see the second step of "Verify-prec"). Differently from "Check-constr", the goal of informing a partner that a precondition cannot be obtained comes indirectly from the fact that the goal of making the condition true is shared with him: in Figure 8, "Verify-prec" shares the "source" parameter with its conditional step "Satisfy(a, s, p)".

   Checking preconditions and making them true is another reactive aspect of the Agent Modeling plans and enables the planner to repair to the changes of the environment around him by replanning the agent's subgoals.

3. "Perform-action" models the execution of actions: if its object "*act*" is elementary, it is executed directly ("Exec(a, act)"); otherwise, "Try-execute" is called recursively on its steps, by means of the "Perform-body" action. The execution of the body of an object-level action stops and fails if there is a failure in the execution of any of its steps.[23] When a failure occurs after a portion of the body of the action

[22] Although in Figure 7 the "Explore-actions" plan is not shown, "Check-constr" may occur as a step of "Explore-actions", too: see Figure 9.

[23] In Figure 8 (see the constraint of "Perform-body-step"), the "prev-step(action1, steps)" function

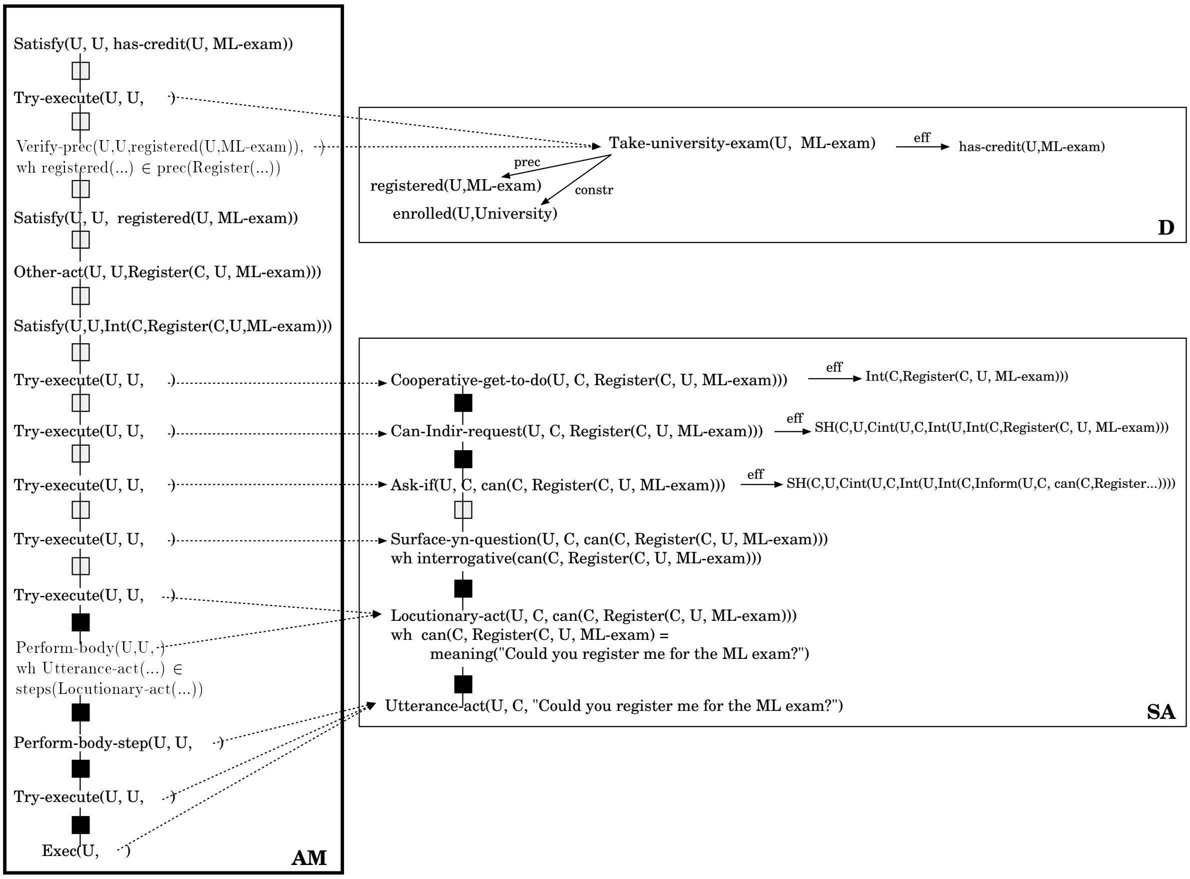

Satisfy(U, U, has-credit(U, ML-exam))
Try-execute(U, U,)
Verify-prec(U,U,registered(U,ML-exam)),)
wh registered(...) ∈ prec(Register(...))
Satisfy(U, U, registered(U, ML-exam))
Other-act(U, U,Register(C, U, ML-exam)))
Satisfy(U,U,Int(C,Register(C,U,ML-exam)))
Try-execute(U, U,)
Try-execute(U, U,)
Try-execute(U, U,)
Try-execute(U, U,)
Try-execute(U, U,)
Perform-body(U,U,)
wh Utterance-act(...) ∈
steps(Locutionary-act(...))
Perform-body-step(U, U,)
Try-execute(U, U,)
Exec(U,)
AM
Take-university-exam(U, ML-exam)
eff
has-credit(U,ML-exam)
prec
registered(U,ML-exam)
constr
enrolled(U,University)
D
Cooperative-get-to-do(U, C, Register(C, U, ML-exam)))
eff
Int(C,Register(C, U, ML-exam)))
Can-Indir-request(U, C, Register(C, U, ML-exam)))
eff
SH(C,U,Cint(U,C,Int(U,Int(C,Register(C, U, ML-exam)))
Ask-if(U, C, can(C, Register(C, U, ML-exam)))
eff
SH(C,U,Cint(U,C,Int(U,Int(C,Inform(U,C, can(C,Register...))))
Surface-yn-question(U, C, can(C, Register(C, U, ML-exam)))
wh interrogative(can(C, Register(C, U, ML-exam)))
Locutionary-act(U, C, can(C, Register(C, U, ML-exam)))
wh can(C, Register(C, U, ML-exam) =
meaning("Could you register me for the ML exam?")
Utterance-act(U, C, "Could you register me for the ML exam?")
SA

FIGURE 10. Local interpretation of the turn: "Could you register me for the ML exam?"

has been performed, only the effects of the steps successfully executed hold. "Perform-body-step" manages the execution of a step: each alternative way to execute the step is tried (if there are any), until the step has been completed, or no alternatives exist. In particular, if the step is a generic action, one of its more specific actions is selected for execution ("Choose-spec"); then, the selected action is performed by means of a "Try-execute"; finally, the agent satisfies the goal of knowing whether the action was indeed successful.

6 The interpretation process

As already mentioned, our current prototype is limited to the interpretation task; for this reason, the system always acts as the receiver of an utterance, switching its role at every turn; moreover, the system maintains the interpretation of the turns of both speakers within a single overall context of the dialogue.

The system takes as input, in textual form, each utterance and interprets it from the perspective of the current receiver; it also tries to relate the turn to the previous turns,

selects the step before "*action*1" in the recipe for "*act*".

in order to see if the utterance is a coherent continuation of the dialogue. Meanwhile, the system updates the representation of the dialogue context by adding to it the interpretation of the new turn, together with the hypothesized coherence relation which links the turn to the previous ones (see section 4).

The input of a sentence is represented in the action formalism as an instance of the Agent Modeling action "Exec", having as argument an instance of the SA (speech act) action "Utterance-act" which, in turn, has as argument the input text: for instance, "Exec(U, Utterance-act(U, C, "Could you register me for the ML exam?"))".

The interpretation process has the goal to find a connection with the representation of the previous part of the dialogue (that we call Dialogue Model - DM), making explicit the coherence relation. However, this connection is not direct, but involves some inferences on the observed "Exec" action, in order to identify the "local" goals that led the agent to produce the utterance. So, what is actually connected to the existing Dialogue Model is not the observed "Exec", but a more complex structure representing the result of these inferences. Since this structure is built by a process called Upward Expansion (see the next section), we will call it Upward Expanded Structure (UES).

For example, consider Figure 10, which shows the interpretation of the sentence "Could you register me for the ML exam?", after having established the top-level local goal that the speaker has credit for an exam.[24] This goal is represented by the "Satisfy(U, U, has-credit(U, ML-exam))" action instance at the top of the Agent Modeling branch, which is linked to the observed "Exec" via a chain of decomposition links.[25] The figure shows that the instances of the Agent Modeling actions have object-level action instances among their arguments. In turn, the object action instances are themselves related by decomposition links, so that the turn interpretations will be represented as two-level structures: the AM boxes contain chains of Agent Modeling action instances, while the D (domain) and SA (speech acts) boxes contain chains of instances of object-level actions. The dotted arrows from the AM box to the object-level boxes link the Agent Modeling action instances to their object-level arguments; note that more than one AM action instance may be related to the same object-level action instance. Although we usually represent the connections between an Agent Modeling action and its object-level argument as a link between two structures, in Figures 10 and 18, because of typographical reasons, we have in some cases included the referenced object-level action in the related AM action.

The Dialogue Model (DM) is a sequence of turn interpretations linked by coherence relations, where the root of each turn interpretation is related to some action occurring in

[24] In this figure, the grey boxes in the decomposition links are a shorthand: in fact, we have shown only the occurrences of the Agent Modeling actions most relevant in this example. The missing actions can be reconstructed using the action libraries described in Section 5.

[25] In the "Satisfy" action, the "source" parameter is bound to the speaker U of the utterance, because we assume that he is trying to reach a goal of himself.

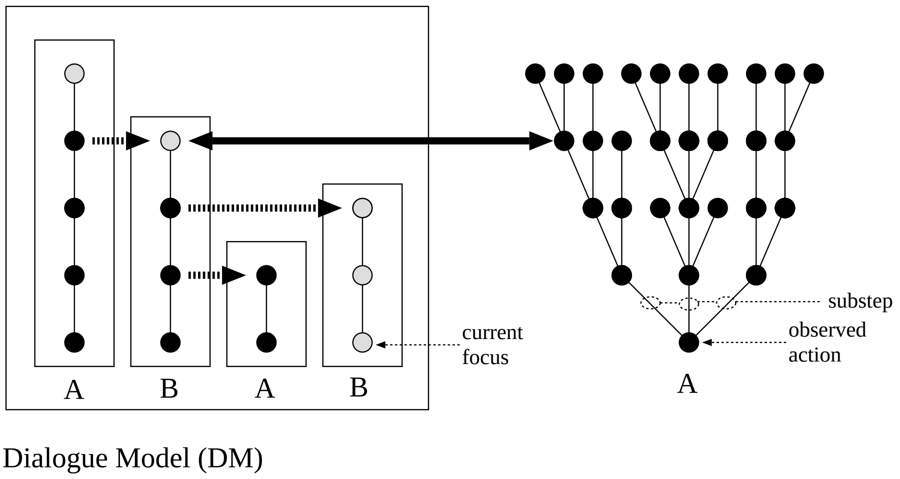


FIGURE 11. The problem of searching for coherence.

the interpretation of a previous turn.[26] In the next sections we describe how this complex structure is built.

6.1 Basic Algorithm

The algorithm for extending the dialogue model with the new input aims at guaranteeing that the coherence of the interaction is maintained, where the concept of coherence has been discussed in the previous sections. However, it is clear that the search space is huge since, in principle, any action for which the recognized one can act as a substep (even at many levels of depth) could be the right one for searching a connection among the actions in the dialogue model. The problem is described intuitively in Figure 11, where the box on the left is the Dialogue Model built until the previous turn, with various sub-boxes which represent the turn interpretations (see Figure 2). The reversed tree on the right (which could actually be a graph) is the set of the actions for which the action (presumably a Surface Speech Act) observed in the last turn can act as a substep. The thick arrow is the connection to be found.
Traditionally, the problem has been split into two parts (see Carberry (1990b)):

1. The generation of the structure appearing on the right in Figure 11. In fact, what is initially available is just the bottom node, *"Observed Action"*, and the structure must be generated incrementally by traversing the libraries. This phase has been called *Upward Expansion*.

[26] Much work has been reported in the literature to define how a dialogue can be segmented in turns, and which are the boundaries among turns. In our framework, a turn corresponds to the sequence of sentences typed in by the user as a unique input stream. However, if more than one sentence is present in the stream, the interpretation is performed separately for each sentence. To simplify the following discussion, when we refer to interpretation of a turn we intend the sequential interpretation of each single utterance in the turn.

```
The interpretation algorithm

  1. Set Dialogue-Node to the Current Focus of the Dialogue model

     /* ———————— Loop: Focusing ———————— */

  2. Find any connection between Observed-Action and Dialogue-Node

       2.1 If just one connection is found, establish it as the proper connection, and extend the
           Dialogue Model accordingly

       2.2 If more than one connection is found, select the best one, establish the selected one as
           the proper connection, and extend the Dialogue Model accordingly

       2.3 If none is found, set Dialogue-Node to the parent of the current Dialogue-Node

     2.3.1 If Dialogue-Node is now void (i.e. we were already on the root of the Dialogue Model) then
           fail.

     2.3.2 Otherwise, go to step 2

     /* —————————- End of Loop —————————- */
```

FIGURE 12. Algorithm for the interpretation of a turn.

2. The search for a connection between any newly generated node (action) of such a structure and any action in the Dialogue Model. This phase has been called *Focusing*.

In our approach, the picture is made somewhat more complex by the fact that there are three different libraries to take into account, and among them the Agent Modeling library has a distinguished role. Before entering the details, the algorithm can be roughly described as in Figure 12, where the terms in boldface will be discussed in more depth.

The algorithm starts from the literal interpretation of the last turn (*"Observed Action"*) and from the literal interpretation of the previous turn (*"Dialogue-Node"*, which is the value of the *"Current Focus"* of the Dialogue Model); then, it tries to find a path ("Find any connection") between *"Observed Action"* and *"Dialogue-Node"*. The search is carried out by looking for all the parents (and then, if needed, the ancestors) of *"Observed Action"*, and seeing if they match the current focus. Since the search is computationally expensive, it is partially supported by an (off-line) precompilation of the libraries, which builds an index of the paths existing between pairs of action schemas. In case no path is found, so that no connection can be set up, then *"Dialogue-Node"* is set to the parent of the *"Current Focus"* and the search is restarted. This process is iterated until a connection is found (the coherence has been established) or the root of the dialogue model is reached (the system judges the dialogue incoherent, according to its knowledge).

Given an instantiated AM action *CurrAct*, let *currAct* be its action schema.

1. Let *genAct∗* be the set of action schemas more generic than *currAct* (including *currAct* itself).
2. Let *parentActs* be the set of all action schemas for which one of the elements in *genAct∗* is a substep.
3. For each *upAct* ∈ *parentActs* create an instance *UpAct*, by binding its parameters.

FIGURE 13. The Upward Expansion algorithm.

6.2 DETAILS

In this section, we aim at making clear the meaning of the boldface words appearing in the sketch of the interpretation algorithm shown in Figure 12. Since the major complexities, with respect to other approaches, are due to the presence of the Agent Modeling actions, we must make more precise their role in the algorithm. This can be obtained by starting the description from the way the Upward Expansion phase is carried out.

6.2.1 Nodes reachable via (inverse) decomposition links from a given node (Upward Expansion)

Although we tried to make the basic ideas clear by keeping Figure 11 as simple as possible, we must note that the result of "observing an action" is not only an instantiation of a node in the Domain (or Speech Act) library, but it also includes an instantiation of an "Exec" action, coming from the Agent Modeling library, as shown in Figure 14.a.

Starting from this structure, we must obtain the Upward Expanded Structure shown in Figure 10. This is obtained via the Upward Expansion algorithm in Figure 13: given an action *CurrAct* (a member of *Current-Actions* in the Interpretation Algorithm in Figure 12) to be expanded, the Upward Expansion algorithm determines the actions for which *CurrAct* is a possible substep. Two points have to be taken into account: the first one concerns the existence of the Generalization Hierarchy (which is handled in Step 1), while the second one is more complex, and concerns the instantiation of the actions. In fact, the algorithm first finds the action schemas that, in the library, have *CurrAct* as a step ("*parentActs*"), and then binds the parameters of these schemas in the proper way.

The binding is obtained by taking into account the bindings of the arguments of the original *CurrAct*, the wh-restrictions associated with "*currAct*" in the body of "*upAct*",[27] and two global variables bound to the current speaker and to his interlocutor. Note that different sources contribute to the binding of the parameters of the newly instantiated actions: while the binding for the parameters shared by *CurrAct* and *UpAct* is immediately available, for the other parameters we must take into account the wh-restrictions

[27] We are referring to the wh-restrictions which relate the variables of the steps of an action; see, for instance, "Cycle-do-action" in Figure 9.

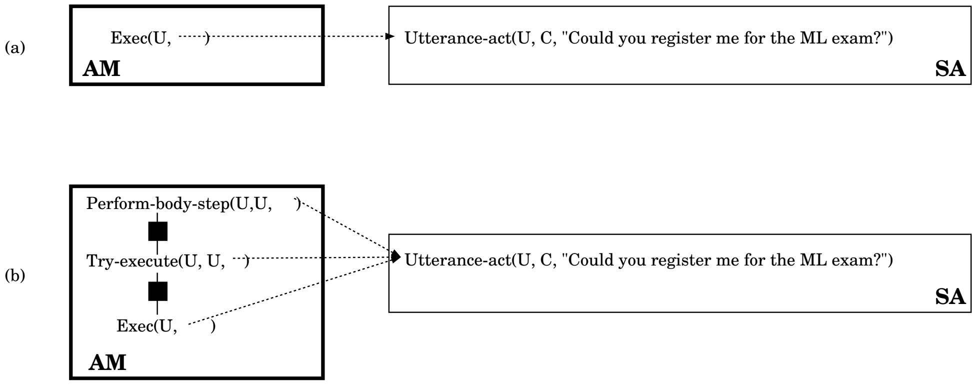

(a)
Exec(U,)
AM
Utterance-act(U, C, "Could you register me for the ML exam?")
SA
(b)
Perform-body-step(U,U,)
Try-execute(U, U,)
Exec(U,)
AM
Utterance-act(U, C, "Could you register me for the ML exam?")
SA

FIGURE 14. Two steps of the interpretation of a turn.

present in the body of *UpAct*. In fact, these restrictions specify the relations that must exist between the parameters of *UpAct* and the variables occurring in its body.

Figure 14.a shows the structure given as input to the first step of Upward Expansion. As stated in the algorithm, the Agent Modeling actions undergo the Upward Expansion, since it is the AM library that specifies how the various object-level actions can be linked together. Since "Exec" occurs as a substep only in the place shown in Figure 8, i.e. to accomplish a "Try-execute" (where the abstraction to "Perform-action", more generic than "Try-execute", is accounted for in Step 1 of the Upward Expansion algorithm), "*parentActs*" includes the single action schema "Try-execute". In this case, there are no problems with the parameters in the instantiation, since the three parameters of "Try-execute" are "*a*" (the same agent of the "Exec"), "*s*" (the current interlocutor), and "*act*" (which is bound to the same string already available in the "Exec" instantiation).

A more intricate situation arises after a further step of expansion, whose result is shown in Figure 15. In fact, here, the expansion of the Agent Modeling structures must be followed by a parallel expansion of the structure of the object-level actions. This is a consequence of the binding process on the AM action parameters, and is guided by the evaluation of the wh-restrictions present in the body of the new Agent Modeling actions to be attached.

"Perform-body-step" (which appears in Figure 14.b as the top Agent Modeling node, i.e. *CurrAct* in the algorithm) is a step of "Perform-body"; however, the wh-restriction in the decomposition of "Perform-body" only says that the third parameter of each instance of "Perform-body-step" must be equal to one of the steps of the object-level action which the "*act*" parameter of "Perform-body" is bound to ("steps = body(act)", and "action1 $\in$ steps"). In our example, the loop variable "*action*1" is bound to the "Utterance-act" of the observed utterance. This means that "Utterance-act" must occur in the body of some other action, which will play the role of third parameter in the "Perform-body" to be instantiated. In order to find a connection, we must exploit the object-level libraries. The

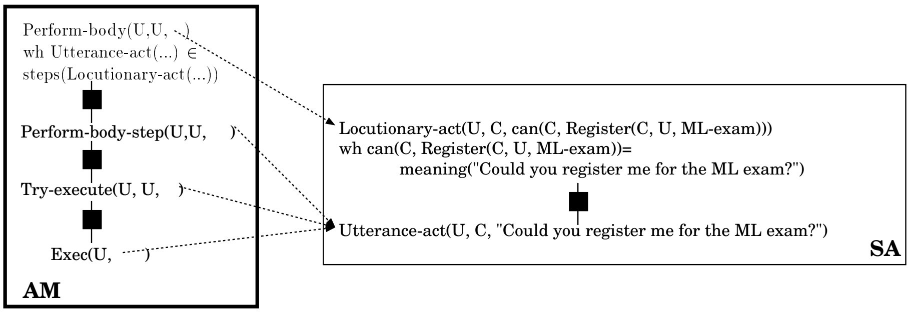

Perform-body(U,U, ..)
wh Utterance-act(...) ∈
steps(Locutionary-act(...))
Perform-body-step(U,U,)
Try-execute(U, U,)
Exec(U,)
AM
Locutionary-act(U, C, can(C, Register(C, U, ML-exam)))
wh can(C, Register(C, U, ML-exam))=
meaning("Could you register me for the ML exam?")
Utterance-act(U, C, "Could you register me for the ML exam?")
SA

Figure 15. A further expansion of the structure in Figure 14.b.

Speech Act library, to which "Utterance-act" belongs, unambiguously tells us that the super-action is "Locutionary-act"; the wh-restriction in it ("content = meaning(string)"), specifies that its third parameter must be bound to the "meaning" of the input utterance, so that the parser and the semantic interpreter are activated, producing the representation in Figure 15.[28]

So, the process of evaluating the action conditions in order to bind the parameters has two important side-effects: the parallel expansion of the object-level and metalevel interpretation structures and the call of the syntactic and semantic interpreter. A third noticeable result is that the evaluation process performed in Step 2 of the Upward Expansion algorithm can determine the relations between conditions and actions having them as preconditions or constraints: this task corresponds to the action identification phase described in Carberry (1990b). In fact, let's suppose that an inferred Agent Modeling action be "Satisfy(U, U, condition)"; one of the actions where "Satisfy" occurs as a substep is "Verify-prec" (see Figure 8). In particular, if we assume that "*condition*" is not of the form "Knowif(a,p)", only the second occurrence of "Satisfy" can match the input.[29] Now, the standard Upward Expansion step tries to instantiate "Verify-prec", which is done according to the wh-restrictions of "Verify-prec": these restrictions specify that "*condition*" must be one of the preconditions of some to be determined object-level action ("lpr = prec(act)", and "p ∈ lpr"); this is exactly what happens in Carberry (1990b) when the action in focus is identified from the attempt to make true one of its preconditions.

For example, consider "Satisfy(U, U, registered(U, ML-exam))" in Figure 10: "registered(U, ML-exam)" is a precondition of the domain-level action instance "Take-university-exam(U, ML-exam)". So, the Upward Expansion causes the instantiation of the Agent Modeling action "Verify-prec(U, U, Take-university-exam(U, ML-exam))", and, possibly, in the next step upward, of "Try-execute(U, U, Take-university-exam(U, ML-exam))",

[28] In the figures we have reduced the syntactic and semantic representation of the input sentences to logical forms for the reader's convenience without reporting details about syntactic information (such as, for example, the presence of question marks, conditional mood and so on).

[29] This optional occurrence concerns the possibility that if the agent believes that "*condition*" is not true, he tries to make it true.

which is the required inference.

Finally, we observe that during the Upward Expansion, the actions need not be fully instantiated; some variables can remain unbound: they will get their value when the connection with the Dialogue Model is found.

6.2.2 *Parent of a node in the Dialogue Model*

This is a rather simple point, since it's only a matter of climbing up the right frontier of the Dialogue Model (the grey circles in Figure 11), in order to take into account all the tasks that have not been completed yet. Remember, however, that the path that is followed is always the one of the Agent Modeling actions, so that in some cases, the father of a node must be found not via a decomposition link, but via a link connecting an action of agent A with an action of agent B (the dotted arcs in Figure 11). Because of these interconnections, the actions found on the path can pertain to either agent, so that the agents' goals are taken into account in an interleaved way, depending on the place where the process is positioned at a given time. Consequently, the possibilities of Plan Continuation or Goal Adherence/Adoption arise automatically.

6.2.3 *Connections*

As stated in the two previous sections, the Agent Modeling actions are the backbone of both the Dialogue Model and Upward Expanded Structure (UES). So, when we say that a connection has to be found between the Dialogue Model and the (upward expanded) interpretation of the last turn, we mean that a connection must be found between "an AM node" in the dialogue model and "an AM node" in the UES.

So, let's suppose the we have to check whether there is any connection between the AM action A_{new} observed or inferred from the last turn, and the AM action A_{old} existing in the Dialogue Model. Clearly, we are not talking of any connection, since any connection is what needs to be found by the whole process. So we are looking for some kind of more direct connection. In the following subsections we will discuss:

- Which kind of basic connections are looked for.
- How the search process takes place.
- How the presence of the AM library affects the processing

- *Basic connection types*

1. The most direct type of connection is the one where there is a decomposition chain between A_{old} and A_{new}; i.e A_{new} is a substep of A_{old} at any level of depth, as shown in Figure 16.1 (Connection type 1).

In order for this kind of connection to apply, we assume that, although A_{old} and A_{new} are Agent Modeling actions, there is a corresponding decomposition chain between their arguments, i.e. the involved object-level actions.

2. In order to increase the power of the system, we also accept more complex connections. The first of them is shown in Figure 16.2.
   The basic structure is analogous to the one of Figure 16.1, and in fact from the Agent Modeling perspective there is no substantial difference. However, we accept here that the argument (the object-level action) of A_{old} is not connected via a decomposition chain to the argument of A_{new}; so, we have to infer the argument of A_{old-x} before the connection can be found. The reason for introducing this facility is that the actions appearing in the Agent Modeling library can be complex, i.e. can have decompositions composed of more than one step. In some cases, the AM decomposition step can refer to different object-level actions which are not connected via a decomposition link, or to different conditions related to an object-level action. An example of this second case is the definition of "Other-act": the second step of this action consists of a checking of the successful execution of the domain-level action "*act*", by means of "Satisfy(a, a, Knowif(a, done(o, act)))" action. Of course, the notification action is not in the decomposition of the domain-level action "*act*". So, if we want to find a connection between A_{old} (the occurrence of "Other-act") and A_{new} we cannot rely only on the links between actions in the object-level libraries, but we must make a different move on the AM library in order to find A_{old-x} and then let the search be based on it.
   We have an example of this case in dialogue 4.3.1, where an agent A, after an utterance, makes a request in order to know whether his partner has understood or not ("so there's an engine at Avon, right?"). The action of requesting a notification is not in the body of the "Update" object-level action, which is the argument of "Other-act" ("Other-act(A, A, Update(B, ...))"). So, in order to connect A_{new} ("right?") the system must first expand the "Other-act" action and reach A_{old-x} ("Satisfy(A, A, Knowif(A, done(B, Update(B, ...))))") and then it can search for a connection between A_{new} and A_{old-x}.

3. A third case concerns the direct interplay of the roles of the two agents, as shown in Figure 16.3.
   Here, there is a new link, shown in the figure as a thick arrow. It is the link representing the fact that an agent is meeting a goal of his partner. In this case, we have the inverse problem as in the previous case: there is no difficulty at the object-level, since the action executed by the first agent is a substep of the action requested by the partner (a standard decomposition chain), but from the Agent

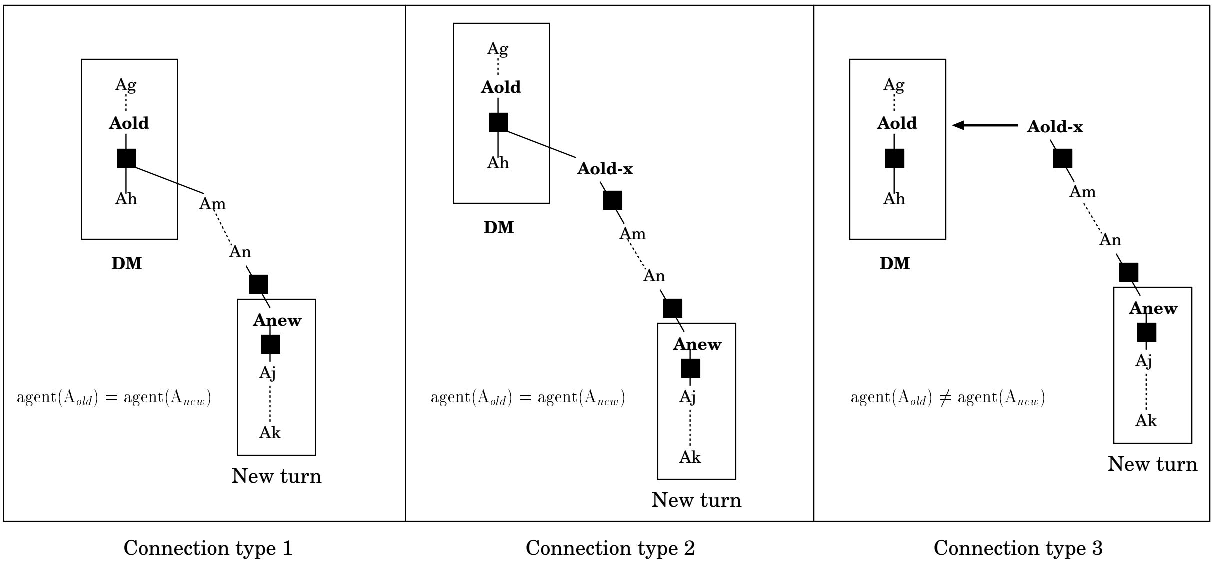


FIGURE 16. Three connection types.

Modeling point of view the actions A_{old} and A_{old-x} must be included in two different subspaces, each concerning one of the two agents.

- *Search for a connection*

Although a connection has to be found between two Agent Modeling nodes, it is clear that the AM library does not provide us with much information about the possible relations between actions. So, only the object-level libraries can guide the search. For instance, if we know that, in the Upward Expanded Structure, A_{new} is "Try-execute(U, U, act1)" while, in the Dialogue Model, A_{old} is "Try-execute(U, U, act2)", no connection between the two actions can be imagined, unless we know which are "*act*1" and "*act*2". But even looking for an object-level path is by itself a computationally expensive process. This problem can be mitigated by a precompilation step, where all the paths are pre-computed by inspecting the (object-level) libraries. We say that the problem is just mitigated, but not solved, because paths can only be used to exclude the possibility of a connection; but if a path is found, it must still be checked in order to verify that the actual parameters (of the object-level actions) satisfy the constraints expressed in the actions along the path.

So, the basic knowledge sources for finding a connection are two sets of compiled paths, the first one concerning the Speech Act library, and the second one concerning the Domain Library. For instance, in the first set all the possible paths are stored connecting a "Request" action to a "Surface-Question", while in the second one we can find all the paths connecting "Take-university-exam" to "Enter", or to "has(agt,pass(lab))" (see Figure 5). Note that in the last case we have a path between an action and a condition; they are also stored in the compiled knowledge base. All these paths are pre-computed starting from the defined libraries.

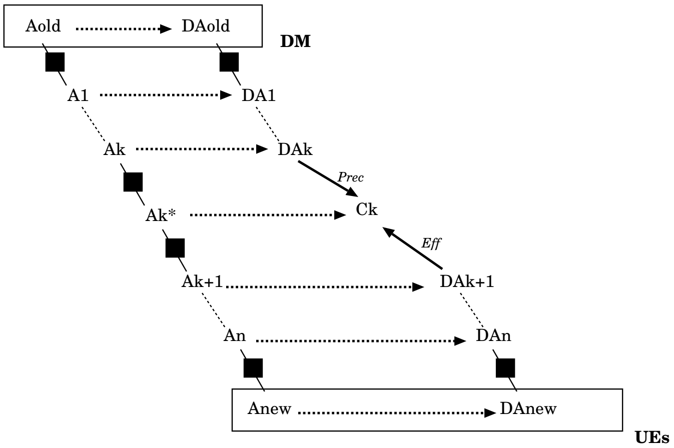


Figure 17.

Although the pre-compiled paths suggest the connections described in the previous subsection, we have extended the compilation mechanism to account for precondition-effect relations. This is described in Figure 17. Differently from the previous figures, we have made explicit here the relations between the Agent Modeling actions and the object-level actions. It is possible to see that the AM path is a standard decomposition path, while the object-level one includes a precondition-effect link between the object-level action "DA_k" and the object-level action "DA_{k+1}" (via the condition "C_k"). Of course, the Agent Modeling action "A_{k*}" must be an instance of "Satisfy(...)", in order to make explicit the fact that the AM plan has been developed considering the need to achieve the condition "C_k".

If the planning activity of the system were based exclusively on stored recipes, the need to account for effect-precondition connections would not arise. But we admit that in some cases the user enters a real planning behavior;[30] in such a case, a sequence of actions is not justified on the basis of an existing recipe, but on the more general basis that, in order to perform an action, the agent has to make its preconditions true, and so he has to execute some action to make them true. The path shown in Figure 17 accounts for this situation, from the perspective of the recognition activity. An observed action is accepted as coherent with the previous interaction if it aims at making true a condition enabling the execution of a subsequent action that is part of the previously recognized plans.

It must be observed that the various complex connections described in this subsection and in the previous one are "orthogonal", in the sense that they may combine with each other freely. For instance, we can have a path including an effect-precondition connection (Figure 17) aimed at satisfying a goal of another agent (see Figure 16.3) not yet explicitly present in the Dialogue Model (Figure 16.2).

[30] See Young *et al.* (1994) for a planner which interleaves causal planning and action decomposition.

However, since the precompilation of paths is made inside the libraries, the paths do not mix domain-level and linguistic-level actions. So, assuming that the executed action is the utterance act UA_{new} (but nothing changes if it is non-linguistic), and that the current focus is a previous utterance act UA_{focus}, then a linguistic path is searched, which connects UA_{new} and UA_{focus}. If the path is not found, then UA_{new} is "upward expanded", and the search is repeated, using the compiled paths, starting from its parent(s).

If the search succeeds within the linguistic library, we obtain something which may be seen as a purely linguistic behavior. Examples of this situation are direct answers to questions. For instance, the connection between "what is your name?" and "Mario Rossi" in Example 3.1 (see Figure 18) is found at this level. In fact, it is an instance of a connection of type 3 (see Figure 16.3) where A_{old} is "Obtain-info(C,U,name(U,name))", A_{new} is "Utterance-act(U,C,"Mario Rossi")" and A_{old-x} is the inferred goal of "Obtain-info" (i.e. "Inform(U,C,name(U,name))", not shown in Figure 18).

But if no such connection can be found, then the Upward expansion reaches the top of the Speech Act library and enters the domain library. In the current implementation, we have assumed that no path can be found between an action in the domain library and a (focused) action in the SA library. So, the focus is moved up to the nearest domain node in the DM, and the search continues on the basis of the same algorithm, but applied now to domain actions.

- *The role of the AM library*

Much of what has been described above can be accomplished without adding the complexities of the AM library. Consider, for instance Figure 14.b. According to the described algorithm, there is no way to find a connection to

"Try-execute(U,U,Utterance-act(U,C,'Could you register ...')"

unless you can find exactly the same connection to

"Exec(U,Utterance-act(U,C,'Could you register ...')"

or to

"Perform-body-step(U,U,Utterance-act(U,C,'Could you register ...')"

So why are three different nodes useful in the AM part of the discourse model? The relevant cases depend on the fact that the AM library models the reactive behavior of agents who know that actions may fail. In particular, the recipe of the "Satisfy" action (see Figure 7) includes a "Cycle-do-action" that is repeated until a plan for achieving the goal g has succeeded (see the $\neg g$ conjunct in the loop condition). The achievement of g is tested in the second step of both "Self-act" and "Other-act". What is implicit in the figure is that when "Knowif(a,done(agent,act))" (where *agent* can be a or o) is satisfied, the (possible) falsity of "done(agent,act)" is assumed to imply that g is false, so that the upper loop enters a new cycle, in order to apply a different recipe for achieving g.

At the AM level, all these actions appear explicitly, so that "Satisfy(a,a, Knowif (a,done(agent,act)))" is a node different from "Try-execute(a,s,act)", even if they refer to the same domain action *act*. This enables the system to keep apart the meta-level action of verifying the success of an action from its execution. The recognition of the actual situation is accomplished by the algorithm described above on the basis of the recipe associated with the *Knowif* goal. In the same way, the AM level enables the system to observe and recognize the activity of forming plans ("Find-plans"), of notifying a partner about the success of an action (the optional "Satisfy(a,a,Bel(s,done(a,act)))"), and so on[31]. An example of the structure resulting in these cases appears at the bottom of Figure 18. In this example, the final "OK" has been interpreted as an anticipation by C of the (presumed) next U's subgoal of knowing if C got the name, i.e. if the associated "Inform" linguistic action succeeded.

6.2.4 Best paths

In some cases, a new turn can be connected to the previous context in multiple ways; this fact means that there are different possible reasons why the new input is coherent with the discourse. In these cases, there are two alternatives: either one keeps all the hypotheses, by duplicating the Dialogue Model as many times as there are possible connections, or one makes a choice, preserving only the "most reasonable" connection and discarding all the other ones. The first alternative is computationally unfeasible, because an unmanageable number of different contexts should be maintained. The second alternative is more practicable, but it requires some means to choose the best hypothesis.

Various methods have been proposed in the literature to cope with this problem. For example, the model described in Carberry (1990a) and the one in (Bauer 1995) are based on the Dempster-Shafer theory of evidence, whose main advantage with respect to classical probability theory is its ability to keep apart uncertainty and ignorance. Although both approaches are interesting in providing the basis for modeling the process of choice, it can be observed that Carberry (1990a) allows just for local decisions, without describing a method for combining evidence coming from different sources. Instead, Bauer shows how such a combination can be carried out, but his method is not linked to any architecture for dialogue modeling. Furthermore, both approaches are concerned only partially with the basic difficulty of Dempster-Shafer's theory, i.e. the difficulty of assessing the Basic Probability Assignment (see however Bauer (1996)).

We have not faced the problem in depth, so we decided to approach this particular aspect of the overall architecture in the simplest way. This choice is also due to the fact

[31] The current content of the AM library is suitable for recognition (the task currently addressed), but some extra machinery would be required for using it in generation. In particular, it specifies that each executed action must be followed by a check about its success (i.e. that its expected effects actually hold). This is clearly unreasonable for many actions, such as the linguistic ones

that our architecture makes the problem a bit harder: in fact, we have to take into account two sources of ambiguity: the first one is the well known ambiguity inherent to Domain Knowledge (I can want to go to the library either to get a book or to get a certificate that I am an authorized visitor of the library), or to Speech Act Knowledge (I can ask you if you can raise your hand either as a request of doing so or as an inspection about your current abilities - if I am a physician visiting you). But ambiguity arises also from the Agent Modeling Library; in particular, any action concerning a precondition can pertain to the planning phase or the execution phase (I can ask you where is the library for two reasons: either because I have already decided that I want to borrow a book and I need to know about its location - I'm just continuing the execution of a plan - or because I am considering different possibilities - the first one is to borrow the book from the library, but I'm ready to find an alternative plan if the library is too far away).

Our main choice has been to interrupt the process of finding a connection as soon as a solution has been found. This means that the most local coherent interpretation of the last turn is privileged (i.e. the one nearest to the current focus). However, even with these hard-wired preferences, some ambiguity remains. In fact, different paths could link the same action of the Upward Expanded Structure to the same action of the Dialogue Model. These paths are related to the ambiguities discussed in the previous paragraph. In these cases, we resort to a simple probabilistic evaluation. First, we must specify which is the "most local coherent interpretation of the last turn" (MLCI). We define the set of MLCI's, as the set of instances of "Satisfy" reachable from the *"Observed Action"* without traversing, during the Upward Expansion process, any other "Satisfy".

Notice that the locality is not evaluated as the number of steps required to reach a "Satisfy", but the measure unit is the number of "Satisfy" encountered. However, the various "Satisfy" in the MLCI have to be weighted against each other. This would involve a combination of the evidence of the different steps of Upward Expansion. In order to avoid that, we assumed that each Upward Expansion step can take advantage of a local "probability" of the possible alternatives, so that a depth first search can be adopted.

Let's suppose that A_{new} is Upward Expanded, so that A_{new_1}, A_{new_2}, ..., A_{new_n} are reached, with probabilities p_1, p_2 , ..., p_n . If p_k is the maximum of all p_i (1≤i≤n), then A_{new_k} is used for the next Upward Expansion step. When the reached action is an instance of "Satisfy", then a connection is looked for. If it is found, then the process is completed, otherwise, a backtrack occurs to the last choice point (most recent branching point), and another expansion is tried, choosing the hypothesis with highest probability, among the remaining ones. Of course, according to the basic algorithm, the search for a connection path from any "Satisfy" identified as described above starts from the current focus, so that the most local contextual connection is the first one found.

However, there still remains a possible ambiguity: multiple paths can lead from a given “Satisfy” in the Upward Expanded Structure to the same action in the Dialogue Model. This means that the attempt of an agent to make true a given condition (i.e. the argument of “Satisfy(...)”) can be coherently interpreted in different ways. For instance, the attempt to satisfy a “Knowif(a, condition)” can be justified as a precondition check during plan construction or as the check done after each action in a reactive planner (assuming that the previous context has not already determined if the agent is in the phase of planning or execution). In these cases we simply multiply the probabilities of the single elements of a path, in order to obtain its overall score. This very rough approach is justified at least in two respects:

1. The product operator decreases the overall score for long paths; if, in general, this is a drawback, since it does not provide a good balance among the alternatives, in our case this is acceptable, because it seems reasonable to favor the shorter paths.

2. The product is a correct operation to get the probability of a joint event AB, just in case A and B are independent. Otherwise, you have to use conditional probabilities. However, the probabilities stored in our plan libraries are conditional probabilities. In fact, if the action A is declared in the library to be a possible step of A_1, A_2, ..., A_n, with probabilities p_1, p_2, ..., p_n, what is said is that p(A_i/A)=p_i, and not that the a-priori probability of A_i is p_i. Of course, this does not generalize to paths longer than two steps, so that we must accept approximate values in these cases.
   Apart from the last observation, there also is the problem of determining the correct values for the probabilities, a problem which is common to most approaches, and which we have not faced: we assigned the values according to our intuition just for testing the model.

6.2.5 Extending the Dialogue Model

When a connection is found, and it has been chosen as the best one, then it has already been checked that all constraints on the parameters of the action instances are respected. This check is carried out by building an instantiation of the whole path. This means that the real path is now available as a side-effect of the process of consistency verification. During this process, the variables that were left unbound in the Upward Expansion get their values on the basis of the contextual link. Nothing more is required, except for moving the instantiated path inside the Dialogue Model. This step involves:
- the update of an index that records which actions are part of the Dialogue Model.
- the update of the *“Current Focus”* of the Dialogue Model, which is set to the observed action (“Exec”) of the new turn of the dialogue.

6.3 The first turn: Upward Expansion without Focusing

When the first input is analyzed, the Dialogue Model is empty, so that the interpretation process cannot be guided by it. However, the mechanism of Upward Expansion already provides the guidelines for facing the problem. In fact, the *"Observed Action"* can be Upward Expanded exactly as explained in Section 6.2.4, i.e. via a shaded depth-first search driven by the conditional probabilities that an action serves as a step for a possible super-action; the only problem is when to stop the search. We believe that this is a rather hard problem in a general Knowledge Base of actions, but within a restricted domain there exists some top-level goal (e.g. graduating) where the upward movement is stopped necessarily. So, we let the Upward Expansion to reach the top of the Domain Hierarchy, and the obtained Upward Expanded Structure becomes the initial Dialogue Model.

6.4 The system implementation

The three plan libraries and the interpretation algorithm described in the previous sections has been fully implemented; our system is written in Lucid Common LISP and runs under the Unix operating system.

While, in order to identify the linguistic phenomena described in this paper, we have analyzed some corpora of task-oriented dialogues,[32] we have not tested our system on such corpora: the main reason is that, to understand such dialogues, we should have introduced in the system a huge amount of knowledge concerning other domains. In order to avoid this effort, we have preferred to select out of them a set of significant examples, containing the linguistic phenomena of interest to us, and adapt them to our University domain; in this way, we could feed the examples to our system. We have paid special attention to include in the dialogue set all the alternative user reactions to dialogue turns which we have discussed in Section 4. In this way, we could check that the system is able to recognize the intentions underlying dialogue, highlighting the fact that, although different continuations of a dialogue are possible, many of them can be reduced to the same set of high-level intentions.

Our system can correctly interpret almost all the examples reported in Section 4; the only exceptions are:

- the instances of "offers", which we have reported as interesting linguistic examples, but we do not handle in our dialogue model (e.g. consider Example 4.2.7.b). This limitation is due to the absence in the Speech Act library of an action suitable for modeling this behavior. In particular, it is not clear whether offers can be modeled

[32] For instance, we have analyzed some extracts from the "Trains" corpus (Gross *et al.* 1993); an airlines corpus (Transcripts 1992) and some Italian corpora (de Mauro *et al.* 1993, Gavioli & Mansfield 1990).

as specializations of "Get-to-do" (as it is currently the case for all the speech acts), it requires the introduction of a more general top-level action in the Speech Act library.

- the examples where agents suggest alternative ways of action to their interlocutors (e.g. consider the second part of Example 4.2.1.a). In particular, in our framework, an agent can identify a plan to suggest to his partner by entering the "Find-actions" and "Choose-action" steps of the "Satisfy" Agent Modeling action. However, suggestions may occur in various situations: our model covers the cases where the agent is required to suggest a way to achieve a goal; on the other hand, it currently does not cover the autonomous acquisition of the goal to monitor the actions performed by another agent and possibly suggest a better way to achieve his goals. The treatment of such behaviors requires that supplementary goals of the interactants are modeled.

Although in this paper we have focused on speech acts, it must be noted that our system can receive as input both NL (Italian) sentences, and observations relative to the execution of domain-level actions: these action instances can be introduced from the keyboard in the internal representation used in the system (e.g. "Register(C, U, ML-exam)"). So, the system also processes examples like 4.1.1.b: *"the consultant checks on the computer whether the user is in the enrollment list"* (page 13).[33]

Figure 18 shows the interpretation context which our system builds for the dialogue:

Example 3.1:

U: "Could you register me for the ML exam?"
C: "What's your name?"
U: "Mario Rossi"
C: "OK, you are registered."

Since most of the interpretation details have already been discussed in the previous sections, we will not repeat them here. It is only important to note that the notification ("OK") by agent C is interpreted as "Inform(C, U, done(C, act))", where *act* is a free variable which has been bound after the focusing step, i.e. after the system finds that the most local pending goal was "Satisfy(U, C, Knowif(U, done(C, Update(C, name(U, "Mario Rossi")))))".

[33] When the system receives an action instance as input, it processes it in the same way as it happens to an "Utterance-act".

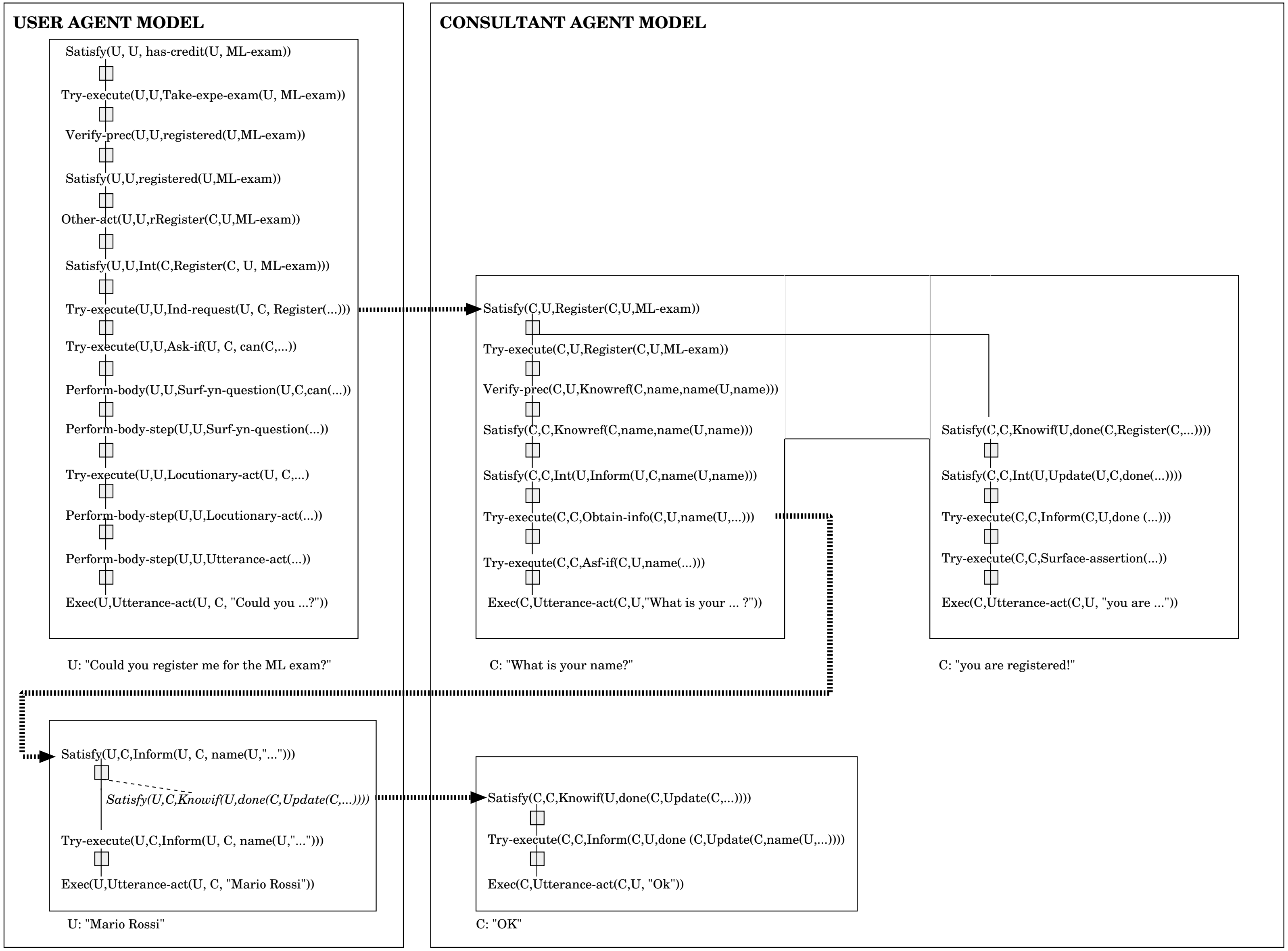

USER AGENT MODEL
Satisfy(U, U, has-credit(U, ML-exam))
Try-execute(U,U,Take-expe-exam(U, ML-exam))
Verify-prec(U,U,registered(U,ML-exam))
Satisfy(U,U,registered(U,ML-exam))
Other-act(U,U,rRegister(C,U,ML-exam))
Satisfy(U,U,Int(C,Register(C, U, ML-exam)))
Try-execute(U,U,Ind-request(U, C, Register(...)))
Try-execute(U,U,Ask-if(U, C, can(C,...))
Perform-body(U,U,Surf-yn-question(U,C,can(...))
Perform-body-step(U,U,Surf-yn-question(...))
Try-execute(U,U,Locutionary-act(U, C,...)
Perform-body-step(U,U,Locutionary-act(...))
Perform-body-step(U,U,Utterance-act(...))
Exec(U,Utterance-act(U, C, "Could you ...?"))
U: "Could you register me for the ML exam?"
Satisfy(U,C,Inform(U, C, name(U,"...")))
Satisfy(U,C,Knowif(U,done(C,Update(C,...))))
Try-execute(U,C,Inform(U, C, name(U,"...")))
Exec(U,Utterance-act(U, C, "Mario Rossi"))
U: "Mario Rossi"
CONSULTANT AGENT MODEL
Satisfy(C,U,Register(C,U,ML-exam))
Try-execute(C,U,Register(C,U,ML-exam))
Verify-prec(C,U,Knowref(C,name,name(U,name)))
Satisfy(C,C,Knowref(C,name,name(U,name)))
Satisfy(C,C,Int(U,Inform(U,C,name(U,name)))
Try-execute(C,C,Obtain-info(C,U,name(U,...)))
Try-execute(C,C,Asf-if(C,U,name(...)))
Exec(C,Utterance-act(C,U,"What is your ... ?"))
C: "What is your name?"
Satisfy(C,C,Knowif(U,done(C,Register(C,...))))
Satisfy(C,C,Int(U,Update(U,C,done(...))))
Try-execute(C,C,Inform(C,U,done (...)))
Try-execute(C,C,Surface-assertion(...))
Exec(C,Utterance-act(C,U, "you are ..."))
C: "you are registered!"
Satisfy(C,C,Knowif(U,done(C,Update(C,...))))
Try-execute(C,C,Inform(C,U,done (C,Update(C,name(U,...))))
Exec(C,Utterance-act(C,U, "Ok"))
C: "OK"

FIGURE 18. Interpretation context for Example 3.1.

7 Related work

In several models of dialogue, the coherence of an agent's behavior has been related to a continuation of domain-level activity by introducing heuristic rules associated with the most plausible shifts in the focus of attention of agents, during a typical (task-oriented) interaction (Carberry 1990b, Litman & Allen 1987). In particular, Carberry's model explains the coherence of an information-seeking dialogue with the reference, within all the turns, to the steps of a single, high-level domain plan of the information seeker, admitting smooth changes of the focus of attention. In our model, we refine this notion of coherence by identifying in which way the new actions in an interaction relate with the context: as in Carberry's approach, we identify the relation between the new goals and the ones already appearing in the context by looking for dependencies among goals. However, beyond the cases where an agent simply continues to execute one of his own plans, we treat homogeneously the turns where an agent adheres to the partner's overt goals or adopts one of the partner's identifiable (but not stated) goals.

The work of Litman & Allen (1987) introduces a metalevel of plan operators, called "Discourse plans" in order to model types of subdialogs that cannot be accounted for by considering only domain plans. In particular, the discourse plans specify how an utterance can relate to a discourse topic, where the discourse topic is represented by the object-level plans. The same plan recognition techniques are applied both to metalevel and to object-level plans and the discourse context is represented as a stack of plans and they follow the stack paradigm of discourse coherence.

Litman and Allen's goals are different from ours. They aim at modelling how people communicate with each other about plans. So, their discourse-level actions specify only how people talk about plans, and how the speech acts can be used to affect the performance of the plan. Consequently, they are not interested in specifying how the plan referred to in INTRODUCE-PLAN was built, or why an agent decided to insert a new step in a plan (CORRECT-PLAN). In particular, the plans do not say why, given that an agent intends to execute a plan, he decides to execute an INTRODUCE-PLAN: they aim at establishing the coherence of a given sequence of linguistic actions, but not at generating that sequence.

On the contrary, in our approach, speech acts are considered as a support for carrying out domain plans, not as means to talk about them. And our meta-level plans express coherence in terms of planning, not in terms of plan recognition: the behavior of an agent is coherent when it conforms to rational ways to plan and execute actions. So, a request is coherent just in case an agent could have planned it for achieving his goals; and what an agent can plan is specified in the AM library.

On the other hand, Litman & Allen (1987) anticipate our work in that some linguistic behaviors related to dialog coherence are modeled by the metalevel activity; in particular, the "Continue-plan" operator corresponds in our model to goal adherence and plan continuation, and "Track-plan" models acknowledgments after the execution of a domain action.

Another proposal strictly related to the one presented in this paper appears in Lambert (1993) and Carberry *et al.* (1993): however, our representation of speech acts is different in that we treat in a uniform way domain-level and linguistic actions. In their model, "the domain level represents the system's beliefs about the user's plan for achieving some goal in the application domain. [...] The discourse level captures the system's beliefs about each agent's communicative actions and how they contribute to one another" (Carberry *et al.* (1993), page 2). This leads them to build multiagent discourse plans, characterized by recipes where different speakers perform linguistic actions in an interleaved way. Instead, in our model only linguistic single-agent plans are considered. Since these plans justify the coherence of the dialogue, we do not need to specify separately the admissible sequences of turns.

As discussed in Section 2, Lochbaum (1994) and Cohen & Levesque (1991) include dialogue phenomena in the wider perspective of interaction; however, there seems to be some dishomogeneity in their treatment of linguistic actions. In particular, in Lochbaum's work, the role of speech acts is not always described as a collaborative activity: for example, the occurrence of notifications is modeled by means of specific steps of the main loops for modeling agent behavior and for recognizing the other agents' behavior.

Although Smith & Cohen (1996) explain notifications in a more general way, via the notion of Joint Intention, their treatment of speech acts still keeps apart linguistic actions from domain-level ones: in fact, they define speech acts in terms of attempts. They explain that a specific goal of a communicative act is to alter the receiver's mental state, in order to make him act in a desired way; but, of course, there is no certainty that the Speech Act will succeed, so the speech act is an attempt.

In our framework, we model the different goals pursued in the execution of a speech act with the "Get-to-do" and "Illocutionary-act" actions of our speech act library. In particular, while "Get-to-do" models the action which aims at making the partner intend to act, the illocutionary act establishes Smith and Cohen's "minimal success condition" (that the interactants share the belief that the speaker wants the hearer to act accordingly); finally, an illocutionary act has associated the means for performing it by means of a locutionary act. Given this structure of linguistic action, our solution enables us to reduce the planning and execution of linguistic actions to the same process of planning and executing any other actions; at the same time, the "attempt" nature of speech acts is modeled in that all these actions are performed by means of metalevel actions modeling a reactive planner. Note that the failure to persuade, by means of a given speech act, the hearer to perform an action, or to believe a proposition, corresponds to an unsuccessful execution of the speech act: in this case, some other action must be planned to recover from the failure. Our approach is able to manage the reported phenomena uniformly with respect to domain actions; moreover, it is consistent with the classical work on politeness (Brown & Levinson 1987), which opens the discussion on politeness by assuming that speakers are capable of doing means-end reasoning.

In contrast to the multi-level approach to dialogue processing, Grosz, Sidner and Lochbaum only use domain-level plans to model collaborative dialogue (Grosz & Sidner 1986, Lochbaum *et al.* 1990, Lochbaum 1991, Lochbaum 1995). We agree with Lochbaum that it is better not to introduce multiple plan types, but:
- Although we have three plan libraries, only the object-level and the metalevel plan libraries have functionally distinct roles.
- Our Agent Modeling plans aim at modeling intentional behavior in terms of a reactive planner model. So, they play a role similar to the Shared Plan operator in Lochbaum's framework. Although they may look more complex than the Shared Plan operator, they describe declaratively the whole knowledge for modeling behavior and they are the only responsibles for updating the agent's intentional state. On the contrary, in Lochbaum (1994), the Shared Plan operator represents a static notion of "having a plan"; so, the "augmentation" and "interpretation" processes are in charge of modifying an agent's intentional state (and of recognizing those changes, respectively). These processes work procedurally and model the occurrence of all the types of acknowledgements and disagreement statements.

Also Heeman & Hirst (1995) consider dialogue as a goal-directed collaborative activity among agents. They propose a plan-based model for the negotiation of the meaning of referring expressions; in their model, the speakers can perform the same actions, so that the same plan operators account for how utterances are generated and understood. Heeman and Hirst propose a two-tier framework where the plans for creating referring expressions are arguments of meta-actions which describe the process of establishing the meaning of such expressions and resolving interpretation problems by means of negotiation subdialogues.
Although we don't focus on referring expressions, in our framework we manage the difficulties in the interpretation of utterances (e.g. ambiguities and unknown meanings) as any other problem arising in the process of the execution of object-level actions under the control of the Agent Modeling plans; in particular, these plans establish the goal of reacting to possible failures in action execution, from which negotiation can be derived.

The feasibility of the approaches to dialogue based on intention recognition is challenged in McRoy & Hirst (1995): they claim that these models lead to an unconstrained search for higher-level intentions to explain a turn; therefore, they advance the idea of relying on linguistic expectations and linguistic intentions to limit the search space. The linguistic expectations are based on the idea that any speech act raises an expectation on how the partner will react to it; so, they capture the notion of adjacency pairs; the linguistic intentions are more general, but they are still strictly tied to conventional behavior. McRoy and Hirst exploit this solution in a model for detecting and repairing to misunderstandings in dialogue. In their system, the presence of a misunderstanding is hypothesized, when a deviance from the expected behavior occurs or when the beliefs of a speaker look contradictory. They can manage the defeasibility of these expectations because their system is based on a framework supporting default and abductive reasoning. Although McRoy and Hirst's approach is one of the most akin to ours, it can be noted that we do not require that linguistic expectations are represented as separate knowledge: they derive from the fact that linguistic actions are considered as means for making partners act. Therefore, expectations appear in the dialogue context as low-level goals inferred during the recognition of speech acts. When a new turn can be interpreted as aiming at the satisfaction of such low-level goals, there is no need to proceed in searching for higher-level goals that might explain it.

The finite-state automata approaches to dialogue have been usefully exploited in applicative domains. However, as Stein & Maier (1994) shows, these dialogue protocols alone are not sufficient to model dialogue: in fact, they introduce other devices, like dialogue scripts, to model the various phases of an information extraction dialogue, like the system's request to refine a user's query to constrain the search. In some way, their scripts merge object-level and problem-solving behavior and this fact makes them suited to model dialogues arising from fixed activities, but too specific to model general natural language interactions. Moreover scripts are not particularly suited to cope with mixed initiative dialogue, because they enforce the asymmetry between the roles of the system and of the user in human-computer dialogue.

In our framework, we model the double nature of collaboration pointed out in Airenti *et al.* (1993) and Traum & Allen (1994): it is not necessary that the partner cooperates at the high-level goal concerning the planning - executing activity on domain actions; in fact, he can be engaged in conversation simply by collaborating to a goal related with the planning - execution of linguistic actions, i.e. to the conversation itself. For example, the cooperation can be limited to informing the partner that the speaker does not intend to cooperate with him at the domain level. In this way, the two cooperation forms are explained by the same model for understanding dialogue.

It must be noted that the structure of the Agent Modeling plans allows us to describe in an explicit way the rationale behind the plan recognition heuristics introduced in Allen (1983) to identify the goals underlying an agent's behavior. Consider, for instance, the following rule:[34]

Precondition-Action rule:
SBAW(p) $\supset$ SBAW(act), if "*p*" is a precondition of "*act*"

In our model, we represent this inference by the fact that the "Try-execute" action has the "Verify-prec" step, where the preconditions of the action are checked and the agent may act to obtain them, if needed. So, if the system observes that the agent verifies a condition, it can hypothesize that the agent is trying to perform an action which has that condition as a precondition.

Allen's work also includes rules which have the opposite role with respect to the rules for goal recognition: e.g. the Action-Precondition rule (supporting inferences from the intention to perform an action to the intention that the action preconditions are satisfied) and the Action-Body rule. Similar inferences can be obtained by a generation process able to exploit our plan libraries to produce agent behavior.

8 Conclusions

We have described a model of linguistic interaction where dialogue is treated as a form of collaborative interaction among agents that behave directed by their beliefs, goals, and intentions.

We consider a dialogue coherent if any new turn contributes to some goal occurring in the previous part of the dialogue. In particular, the relation of a turn with the previous context can be of adherence (an agent tries to satisfy a goal explicitly set forth by one of his partners), adoption (an agent tries to satisfy a goal of a partner that the agent has inferred) or continuation (an agent proceeds in the attempt to reach his own goals).

In our framework, the representation of dialogues consists of a set of plan instances, which describe the activity of the interacting agents, connected via links that make explicit the relationships between the various turns.

In order to form plans, to recognize the plans being carried on by a partner, and to establish the proper coherence links, a set of known recipes is available. These recipes are grouped in three libraries: the Domain library, the Speech Act library, and the Agent Modeling library. In the paper, we argue that the introduction of the third library enables our model to account in a perspicuous way for a number of apparently unrelated linguistic phenomena that, in other approaches, require ad hoc treatment.

[34] In the formula, the SBAW notation means that the system believes that the agent A wants to obtain a condition, or to perform an action.

However, some assumptions limit the power of the model. The most important of them concerns the hypothesis that all agents behave according to plans known to all of them: an agent is able to understand (i.e. to realize the coherence of) a turn of a dialogue just if it results from the execution of one or more recipes known to him. Some flexibility is given by the fact that the Agent Modeling plans represent a reactive agent that can achieve action preconditions by replanning subgoals. However, some works, like Lesh & Etzioni (1995), have faced this problem in a more general way.

Moreover, while we currently represent interactions among agents who perform single-agent actions, we do not model more complex cases of cooperation, like multiagent plans. We hope this second restriction does not sound too limiting, since our work is focused on linguistic actions, which are typically single-agent (moreover, the problem of shared plans has already found satisfactory solutions as, for example, in Grosz & Kraus (1996)).

The system described in this paper is focused on the interpretation of other speakers' turns; therefore, we have not addressed some aspects concerning the generation of turns, like for example those considered (Moore 1995, Chu-Carroll & Carberry 1998). Most noticeably, in this work we do not distinguish goals from intentions; in fact, in the interpretation perspective, the system's role is just to recognize the intentions that lead the speakers to produce a turn; since the turns produced by the partner are the result of attempts to reach goals to which he has already committed, this problem is somewhat hidden.

In some more recent work, we have extended the model described in this paper to obtain a system that can recognize other agents' plans, as well as take the initiative to act on the basis of its own goals. In particular, in Ardissono *et al.* (1999b), we have considered the reasons underlying the selection of direct or indirect speech acts, on the basis of their characteristics, described in the Speech Act library. Morever, in Ardissono *et al.* (1999a), we have examined the motivations leading an agent to cooperate at the behavioral and conversational level, and we have provided an alternative solution to the obligation-based approaches to dialog (e.g. see Traum & Allen (1994)). The concept underlying both works is that the speaker's behavior is directed by the desire to respect social goals, like avoiding that the interlocutor is offended (see Brown & Levinson (1987)). Finally, the model of cooperation among agents has been refined in Boella *et al.* (1999), where a decision theoretic planning framework is exploited to allow an agent decide when it is worthwhile that he adopts some of the partner's goals and communicates with him to coordinate the shared activity.

Acknowledgments

We are grateful to the anonymous IJHCS reviewers who have helped us to improve substantially the first drafts of this paper, and to Sandra Carberry and Mathias Bauer, who have read

the paper carefully to provide further useful comments to it. We also thank Rossana Damiano for having contributed to the implementation of a portion of our prototype and for having provided useful comments on the paper.
This work was partially supported by MURST 60% and by the Italian National Research Council (CNR), projects "Pianificazione e riconoscimento di piani nella comunicazione" (Planning and plan recognition in communication) and "Conoscenza, Intenzioni e Comunicazione" (Knowledge, Intentions and Communication).

References

Airenti, G., Bara, B. & Colombetti, M. (1993). Conversational and behavior games in the pragmatics of discourse. *Cognitive Science*, 17:197–256.

Allen, J.F. & Perrault, C.R. (1980). Analyzing intention in utterances. *Artificial Intelligence*, 15:143–178.

Allen, J.F. (1983). Recognizing intentions from natural language utterances. In Brady, M. & Berwick, R.C., Eds. *Computational models of discourse*, pages 107–166. MIT Press.

Ardissono, L., Lesmo, L., Pogliano, P. & Terenziani, P. (1991). Representation of determiners in natural language. In *Proc. 12th IJCAI*, pages 997–1002, Sydney.

Ardissono, L., Lombardo, A. & Sestero, D. (1993). A flexible approach to cooperative response generation in information-seeking dialogues. In *Proc. 31st Annual Meeting ACL*, pages 274–276, Columbus.

Ardissono, L., Boella, G. & Lesmo, L. (1995a). Indirect speech acts and politeness: A computational approach. In *Proc. 17th Cognitive Science Conference*, pages 316–321, Pittsburgh.

Ardissono, L., Boella, G. & Lesmo, L. (1995b). Recognizing preliminary sentences in dialogue interpretation. In *Lecture Notes in Artificial Intelligence n. 992: Topics in Artificial Intelligence*, pages 139–144. Springer Verlag, Berlin.

Ardissono, L., Boella, G. & Lesmo, L. (1996). Recognition of problem-solving plans in dialogue interpretation. In *Proc. 5th Int. Conf. on User Modeling*, pages 195–197, Kailua-Kona, Hawaii.

Ardissono, L., Boella, G. & Damiano, R. (1997). A computational model of misunderstandings in agent communication. In *Lecture Notes in Artificial Intelligence n. 1321: Advances in Artificial Intelligence*, pages 48–59. Springer Verlag, Berlin.

Ardissono, L., Boella, G. & Damiano, R. (1998). A plan-based model of misunderstandings in cooperative dialogue. *International Journal of Human-Computer Studies*, 48:649–679.

Ardissono, L., Boella, G., Damiano, R. & Lesmo, L. (1999a). Conversational cooperation: the leading role of intentions. In *Proc. of the AMSTELOGUE'99 Workshop on Dialogue*, Amsterdam.

Ardissono, L., Boella, G. & Lesmo, L. (1999b). Politeness and speech acts. In *Proc. Workshop on Attitude, Personality and Emotions in User-Adapted Interaction*, pages 41–55, Banff, Canada.

Ardissono, L. & Sestero, D. (1996). Using dynamic user models in the recognition of the plans of the user. *User Modeling and User-Adapted Interaction*, 5(2):157–190.

Austin, J.L. (1962). *How to Do Things with Words.* Harvard University Press, Cambridge, Mass.

Bauer, M. (1995). A dempster-shafer approach to modeling agent preferences for plan recognition. *User Modeling and User-Adapted Interaction.*

Bauer, M. (1996). Acquisition of user preferences for plan recognition. In *Proc. 5th Int. Conf. on User Modeling*, pages 105–111, Kailua-Kona, Hawaii.

Boella, G., Damiano, R. & Lesmo, L. (1999). Cooperating to the group's utility. In *Proc. of The 6th Int. Workshop on Agent Theories, Architectures, and Languages (ATAL-99)*, Orlando, Florida.

Bratman, M.E., Israel, D.J. & Pollack, M.E. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355.

Bretier, P. & Sadek, D. (1997). A rational agent as the kernel of a cooperative spoken dialogue system: implementing a logical theory of interaction. In Muller, J.P., Wooldridge, M.J. & Jennings, N.R., Eds. *Lecture Notes in Artificial Intelligence n. 1193 Intelligent Agents III*, pages 189–203. Springer.

Brooks, R.A. (1991). Intelligence without reason. In *Proc. 12th IJCAI*, pages 589–595, Sydney.

Brown, P. & Levinson, S.C. (1987). *Politeness: some universals on language usage.* Cambridge University Press, Cambridge.

Carberry,S., Kazi, Z. & Lambert, L. (1992). Modeling discourse, problem-solving, and domain goals incrementally in task-oriented dialogue. In *Proc 3rd Int. Workshop on User Modeling*, pages 192–201, Wadern.

Carberry, S., Chu, J. & Lambert, L. (1993). User-system conflict in task-oriented consultation. In *Proc. IJCAI-93 Workshop: Computational models of conflict management in cooperative problem solving*, Chambery.

Carberry, S. (1990a). Incorporating default inferences into plan recognition. In *Proc. 8th Conf. AAAI*, pages 471–478, Boston.

Carberry, S. (1990b). *Plan Recognition in Natural Language Dialogue.* MIT Press.

Castelfranchi, C. & Falcone, R. (1997). From task delegation to role delegation. In Lenzerini, M., Ed. *LNAI 1321. AI*IA 97: Advances in Artificial Intelligence*, pages 278–289. Springer Verlag, Berlin.

Castelfranchi, C. & Parisi, D. (1980). *Linguaggio, conoscenze e scopi.* Il Mulino, Bologna.

Castelfranchi, C. (1992). No more cooperation, please! In search of the social structure of verbal interaction. In Slack, J., Ortony, A. & Stock, O., Eds. *Communication from an Artificial Intelligence perspective. Theoretical and Applied Issues.* Springer Verlag, Berlin.

Chu-Carroll, J. & Carberry, S. (1998). Collaborative response generation in planning dialogues. *Computational Linguistics*, 24(3):355–400.

Clark, H.H. & Schaefer, E.F. (1989). Contributing to discourse. *Cognitive Science*, 13:259–294.

Cohen, P.R. & Levesque, H.J. (1991). Confirmation and joint action. In *Proc. 12th IJCAI*, pages 951–957, Sydney.

Cohen, P.R. (1994). Models of dialogue. In Ishiguro, T., Ed. *Cognitive processing for voice and vision*, pages 245–274. Society of Industrial and Applied Mathematics.

de Mauro, T., Mancini, F., Vedovelli, M. & Voghera, M. (1993). *Lessico di Frequenza dell'Italiano Parlato.* ETASLIBRI.

DeGiacomo, G. & Lenzerini, M. (1995). Pdl-based framework for reasoning about actions. In *Lecture Notes in Artificial Intelligence n. 992: Topics in Artif icial Intelligence*, pages 103–114. Springer Verlag, Berlin.

Devanbu, P.T. & Litman, D.J. (1996). Taxonomic plan reasoning. *Artificial Intelligence*, 84:1–35.

Di Eugenio, B. & Lesmo, L. (1987). Representation and interpretation of determiners in natural language. In *Proc. 10th IJCAI*, pages 648–653, Milano.

Firby, R.J. (1987). An investigation into reactive planning in complex domains. In *Proc. 6th Conf. AAAI*, Seattle (WA).

Gavioli & Mansfield (1990). *The PIXI corpora: bookshop encounters in English and Italian.* CLUEB, Bologna, Italy.

Georgeff, M. & Ingrand, F.F. (1989). Decision-making in an embedded reasoning system. In *Proc. 11th IJCAI*, pages 972–978, Detroit, Michigan.

Goerz, G. & Ludwig, B. (1998). Modelling users, intentions and structure in spoken dialog. In *Proc. Second Workshop on Human-Computer Conversation*, Bellagio, Italy.

Grice, H.P. (1957). Meaning. *Philosophical Review*, 56:377–388.

Gross, D., Allen, J.F. & Traum, D.R. (1993). The TRAINS91 dialogues. Technical Report TN92-1, University of Rochester.

Grosz, B.J. & Kraus, S. (1996). Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357.

Grosz, B.J. & Sidner, C.L. (1986). Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12:175–204.

Grosz, B.J. & Sidner, C.L. (1990). Plans for discourse. In Cohen, P.R., Morgan, J. & Pollack, M.E., Eds. *Intentions in communication*, pages 417–443. MIT Press.

Heeman, P.A. & Hirst, G. (1995). Collaborating on referring expressions. *Computational Linguistics*, 21(3):353–382.

Heritage, J.C. (1984). A change-of-state token and aspects of its sequential placement. In Atkinson, J.M. & Heritage, J.C., Eds. *Structures of Social Action.* Cambridge University Press.

Hobbs, J.R. & Evans, D.A. (1980). Conversation as planned behavior. *Cognitive Science*, 4:349–377.

Jennings, N.R., Sycara, K.P. & Wooldridge, M. (1998). A roadmap of agent research and development. In *Autonomous Agents and Multi-agent Systems*, pages 275–306. Kluwer Academic Publishers, Boston.

Kautz, H. (1991). A formal theory of plan recognition and its implementation. In Brachman, R.J., Eds. *Reasoning About Plans*, pages 69–125. Morgan Kaufmann Publishers.

Lambert, L. (1993). *Recognizing Complex Discourse Acts: A Tripartite Plan-Based Model of Dialogue.* Ph.D. thesis, University of Delaware.

Lesh, N. & Etzioni, O. (1995). A sound and fast goal recognizer. In *Proc. 14th IJCAI*, pages 1704–1710, Montreal.

Lesmo, L. & Terenziani, P. (1988). Interpretation of noun-phrases in intensional contexts. In *Proc. Coling Budapest*, pages 378–383, Budapest.

Lesmo, L. & Torasso, P. (1985). Analysis of conjunctions in a rule based parser. In *Proc. 23rd Annual Meeting ACL*, pages 180–187, Chicago.

Levinson, S.C. (1981). The essential inadequacies of speech act models of dialogue. In Parret, M., Sbisá, M. & Verschueren, J., Eds. *Possibilities and Limitations of Pragmatics*, pages 473–492. Benjamins, Amsterdam.

Litman, D. & Allen, J.F. (1987). A plan recognition model for subdialogues in conversation. *Cognitive Science*, 11:163–200.

Lochbaum, K.E., Grosz, B.J. & Sidner, C.L. (1990). Models of plans to support communication: An initial report. In *Proc. 8th Conf. AAAI*, pages 485–490, Boston.

Lochbaum, K.E. (1991). An algorithm for plan recognition in collaborative discourse. In *Proc. 29th Annual Meeting of ACL*, pages 33–38, Berkeley,CA.

Lochbaum, K.E. (1994). *Using Collaborative Plans to Model the Intentional Structure of Discourse.* Ph.D. thesis, Harvard University.

Lochbaum, K.E. (1995). The use of knowledge preconditions in language processing. In *Proc. 14th IJCAI*, pages 1260–1265, Montreal.

Mann, W.C. & Thompson, S.A. (1987). Rhetorical structure theory: a theory of text organization. In Polanyi, L., Ed. *The Structure of Discourse.* Ablex Publishing Corporation.

Maybury, M.T. (1993). Communicative acts for generating natural language arguments. In *Proc. 11th Conf. AAAI*, pages 357–364, Washington DC.

McRoy, S.W. & Hirst, G. (1995). The repair of speech act misunderstandings by abductive inference. *Computational Linguistics*, 21(4):433–478.

Moore, J.D. & Paris, C.L. (1993). Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics*, 19(4):651–694.

Moore, J.D. & Pollack, M.E. (1992). A problem for RST: The need for multi-level discourse analysis. *Computational Linguistics*, 18(4):537–544.

Moore, J.D. (1995). *Participating in explanatory dialogues.* The MIT Press, Cambridge, MA.

Pollack, M.E. (1990). Plans as complex mental attitudes. In Cohen, P.R., Morgan, J. & Pollack, M.E., Eds.*Intentions in communication*, pages 77–103. MIT Press.

Ramshaw, L.A. (1991). A three-level model for plan exploration. In *Proc. 29th Annual Meeting of ACL*, pages 39–46, Berkeley,CA.

Rao, A. & Georgeff, M.P. (1992). An abstract architecture for rational agents. In Nebel, B., Rich, C., Swartout, Eds. *Principles of Knowledge Representation and Reasoning: Proc. 3rd Int. Conf. (KR:92)*, pages 439–449, Cambridge, MA.

Sacks, H., Schegloff, E.A. & Jefferson, G. (1974). A simplest systematics for the organization of turn-taking for conversation. *Language*, 50:696–735.

Schegloff, E.A. (1972). Sequencing in conversational opening. In Gumperz, J.J. & Hymes, D., Eds. *Directions in Sociolinguistics.* Rinehart and Winston, New York.

Schegloff, E.A. (1979). Identification and recognition in telephone conversation openings. In Psatas, G., Ed. *Everyday language: studies in ethnomethodology.* Irvington, New York.

Schegloff, E.A. (1982). Georgetown university roundtable on languages and linguistics. In Tannen, D., Ed. *Discourse as an interactional achievement: Some uses of 'uh huh' and other things that come between sentences.* Georgetown University Press, Washington, D.C.

Searle, J.R. (1992). *(On) Searle on Conversation.* Benjamins, Amsterdam.

Sestero, D. (1998). *Producing Responses in Goal-Endowed Dialogue Systems.* Ph.D. thesis, University of Torino.

Smith, I.A. & Cohen, P.R. (1996). Toward a semantics for an agent communications language based on speech-acts. In *Proc. 14th Conf. AAAI*, pages 24–31, Portland.

Stein, A. & Maier, E. (1994). Structuring collaborative information-seeking dialogues. *Knowledge-Based Systems*, 8(2-3):82–93.

SRI Transcripts (1992). Transcripts derived from audiotape conversations made at SRI international. prepared by Jacqueline Kowtko under the direction of Patti Price. Technical report, SRI, Menlo Park, CA.

Traum, D.R. & Allen, J.F. (1994). Discourse obligations in dialogue processing. In *Proc. 32nd Annual Meeting of ACL*, pages 1–8, Las Cruces, New Mexico.

Traum, D.R. & Hinkelman, E.A. (1992). Conversation acts in task-oriented spoken dialogue. *Computational Intelligence*, 8(3):575–599.

Weida, R. & Litman, D.J. (1992). Terminological reasoning with constraint networks and an application to plan recognition. In *Principles of Knowledge Representation and Reasoning: Proc. 3rd Int. Conf. (KR:92)*, pages 282–293, Cambridge, MA.

WINOGRAD, T. & FLORES, F. (1986). *Understanding Computers and Cognition.* Reading, MA: Addison-Wesley.

YOUNG, R.M., MOORE, J.D. & POLLACK, M.E. (1994). Towards a principled representation of discourse plans. In *Proceedings of the Sixteenth Conference of the Cognitive Science Society*, Atlanta, GA.