Dialogue systems and planning

Guy Camilleri

IRIT, CCI-CSC, Université Paul Sabatier,
118 route de Narbonne, 31062 Toulouse Cedex 4 France

Abstract. Planning processes are often used in dialogue systems to recognize the intentions conveyed in dialogue. The generation of utterances can also be achieved by a planning/execution mechanism. Some advantages of this kind of mechanim are: knowledge sharing, modular design, declarative description, etc.
In this paper, we present some planning mechanisms and the related models enabling the dialogue management (generation and understanding).

Introduction

In many works, a plan-based framework is used to represent goals (intentions) conveyed in dialogue. The philosophical studies of natural language proposed by Austin [1], Searle [2], Grice [3] and Bratman [4] constitute the theoretical bases of intentional plan based approaches of dialogue. These studies regard utterances as actions carried out by an agent (or speaker) in a particular environment. This association unifies the representation of actions to utterances, and consequently their handling mechanisms. Hence, utterances can be a part of a plan containing other utterances and/or actions. In this way, the dialogue can be modeled by an action plan. This kind of pragmatic context provides some agents goals hierarchically organized.

In this paper, we propose to describe the dialogue utterances generation and interpretation through plan-based models. In the earlier approaches, the interpretation mechanism builds a model (in the form of plans) describing the participant's intentions in dialogue. The generation process is rarely described. The intentions plan (built during interpretation) seems to be used as an input (data) by a utterances generator. Like Wilensky [5] we think, it is interesting to manage utterances generation through a plan-based model. Such approaches propose a plan model to guide the generation process. The main advantages of this modeling are:

- The relations between tasks (responsible of utterances generation), and tasks motivating their achievement, are described in an explicit way.
- The generation process is more modular (a declarative specification is used).
- This kind of description unifies the knowledge representations used by the interpretation and generation mechanisms. Hence, these mechanisms can share some models (knowledge) and processes.

The first section of this paper presents the main plan-based models used to manage dialogue. We discuss then the problems arising from the use of these models in utterances generation and we present new models enabling the integration of an exclusive plan management[1] of dialogue. We briefly discuss the advantages of such kind of modeling framework. Finally, this framework is illustrated by an example.

1 Plan-based models used in dialogue

Grosz and Kraus [6] have developed a formal model that specifies the beliefs and intentions that must be held by collaborative agents engaged in a collective activity. This model called Shared Plan (SP) is applied in a dialogue understanding context. The essential idea is that agents participating in a dialogue (discourse) have a collaborative behavior. The SP specifies all needed features to achieve a collective or individual goal. Agents produce utterances to reach a goal that is, they intend to satisfy all features expressed in the SP for this goal.

Lochbaum [7] has defined her interpretation process with the SP concept. The outcome of this process is represented by a graph structure, her Rgraph (or Recipe Graph). This graph represents all recognized goals participating to more abstract ones. In fact, this structure is a hierarchical tree of goals. Rgraph is a multi-agent plan containing detected goals of the dialogue participants.

Carberry, Lambert [8], Ardissono [9] use a Discourse Model (DM), Litman and Allen [10] a Plans Stack (PS), to capture the intentions of the dialogue participants. DM and PS are very similar to Lochbaum's Rgraph.

For the modeled agent (system), these dialogue representations seem to describe the system beliefs on user's intentions, and some local intentions of the system. The system has a description of the common activity (as Rgraph, DM, PS, etc). The part of the representation describing the system beliefs on the users intentions is often used as data (input) for utterances generation. The relations between interventions (actions) of the system agent and its partners are modeled through multi-agents plans.

2 Generating dialogue through plan-based models

We think that the understanding of intentions is not the recognition of some "common" plan, but in particular the understanding of the fact that the other partner may/must have a complementary view.

The use of Rgraph or DM or ... plan-based models in generation process poses some problems. These models describe the intentions (goals) of discourse participants. At least two agents interact in discourse, therefore these plan-based models are multi-agent plans. Unfortunately, multi-agent plans can not be performed. Generally, some tasks miss in multi-agent plans to be carried out by agents. For example, the multi-agent plan: `Obtain-Info-Ref(A,B,...)` by

[1] Interpretation and generation.

achieving `Ask-Ref(A,B,...)` and `Answer-Ref(B,A,...)` can not be performed in this form by `A`. To achieve his part of this plan, the `A` agent must wait the `B`'s answer. This waiting task is not described in this multi-agent plan. However, this task is required to the `Obtain-Info-Ref(A,B,...)` achievement.

More generally, an agent only achieves his actions[2]. Plans performed by an agent are thus mono-agent plans. Therefore, an executable plan-based model must be mono-agent. The interaction modeling in these mono-agent plans requires the utilization of synchronization tasks (as wait, etc). The previous multi-agent plan can be translated for A agent as follow: `Obtain-Info-Ref(A,B,...)` by `Ask-Ref(A,B,...)` and `Wait-Answer(A,B,...)`. The synchronization task `Wait-Answer(A,B,...)` replaces the task `Answer-Ref(B,A,...)` ascribed to B agent. This `Obtain-Info-Ref(A,B,...)` plan version is now achievable by A agent.

3 Modeling principle for dialogue generation and understanding

The key point of our modeling principle is: each knowledge is described from the point of view of the modeled agent. As discussed above, dialogue generation requires mono-agent plans.

3.1 Modeling primitives

The primitives proposed here are conceived to be handled by some classical mechanisms of planning, plan-based approaches of discourse and knowledge systems research areas. A task/method paradigm is chosen to describe our hierarchical models.

Task A task is an action represented as follows:

NAME	Task name
PAR	List of handled typed parameters
OBJECTIVE	Task goal expressed in state form
METHODS	list of methods achieving the task

The parameter list specifies the set of world objects handled by the task. All defined methods (or way to achieve) during the modeling phase are recorded in the method list of the considered task.

[2] Ascribed to him.

Method A method describes a way (at only one level of abstraction) to achieve a task.

HEADER	Task carried out
COND-APP	Applicability conditions
PREC	Preconditions set
EFFECTS	Effects set
CONTROL	Subtasks performance order
SUBTASKS	Subtask set

The action carried out by a method is indicated by the heading. Applicability conditions[3] are used to constrain the method instantiation. Preconditions and effects sets are handled in the usual way. The order of subtasks performance is described in the control field; subtasks are recorded in the subtasks set.

Terminal task A terminal task is directly executable. Its execution does not require the description of its decomposition.

Handling mechanisms Mechanisms applied in our plan-based models are planning (plan recognition[4]) processes and execution engines. Cond-app, Prec, Effects and Subtasks are used for planning and plan recognition purpose (see Camilleri [11, 12]).

Execution engines commonly used in task/method models can roughly be sketched by the following steps:

1. If the current task is a terminal task then execute it else
2. Select an applicable method (by checking Cond-app and Prec) to the current task.
3. Apply the selected method by performing the control field, which determinates the new task to be executed, and so on.

These execution engines can be applied on a task/method model to generate dialogue. The production of utterances results from the discourse tasks achievement (as `Ask-ref(...)` task of the figure 1), that is the application of an execution engine on discourse tasks.

3.2 Agent plan libraries

In plan-based approaches of discourse understanding, two kinds of knowledge play a crucial role: plan library, which enumerates the set of possible domain plans and user model describing agent's preferences. The former represents the knowledge allowing the achievement of domain goals, usually the latter describes the methods more likely employed by a particular agent (or group of agents).

The domain plan library appears in the majority of approaches as common ground knowledge. However, some possible differences of knowledge between

[3] as action parameters.

[4] Plan recognition is the process of infering (in term of plans) an agent's intention from his observed behavior.

agents are modeled through the methods and the goals preferences in user model. User model can also contain some plans, which are specific to a user. Usually, these particular plans correspond to erroneous (flawed) plans. In fact, all possible plans (methods) allowing the domain tasks achievement are described in domain plans. This library is filtered by user's preferences.

The plan recognition process interprets utterances from the plans contained in the common ground knowledge. The understanding process is guided by the user model, which possibly adapts plans to the modeled user.

In our plan-based framework, plans are used to generate and understand dialogues. Models are described in the task/method paradigm previously presented.

In our framework, the following categories of plan libraries are used:

- The *Domain plan library* represents all agent's goals (tasks) and methods typical to the application domain. This library specifies all way used by the agent to achieve the domain tasks.
- The *Agent plan library* describes the agent's behavior. The agent's behavior library is composed of discourse, activity, extension/execution and cooperative sub-libraries. These sub-libraries are independent of the application domain, and thus more generic. Discourse plans describe the different ways to communicate with others. The activity library represents agent life cycle. Extension/execution plans specify the planning/execution process. The cooperative plans define the distribution[5] and adapt plans to the current cooperative context.
- The *User plan library* describes the system's (or modeled agent) beliefs on its users. This library consists of two categories of plans: The system's beliefs on the methods employed by users to carry out tasks, and the system's beliefs on the system representation owned by users (image of oneself). The later category describes the methods whose users think the system uses.

4 Dialogue example

In this part, we illustrate our plan-based framework by the following dialogue adapted from Ardissono et al [9]. Agents participating to this dialogue are a system (noted S) and a user (U). The dialogue domain is the exam university registration. Students use this system to register themselves for exams.

U: Could you register me for the m10 exam ?
S: What's your name ?
U: Mario Rossi.
S: Ok, you are registered.

A part of S's plan-based models used in this example are mentioned in the figure 1. The system goal is to register the user `Register(S,U,ex)` by obtaining the user name (tasks `Request-ref(...)` and `Perceive-ref(...)`) , checking the user situation (`Check-student(...)`) and registering him/her (`Register--student(...)`). The system also knows the user registration way (cf. user plan

[5] The agents which perform tasks.

library). Only, the discourse and activity plan libraries are partly exposed, for brevity reasons the problem solving level (composed of extension/execution and cooperative plan libraries) is not presented here.

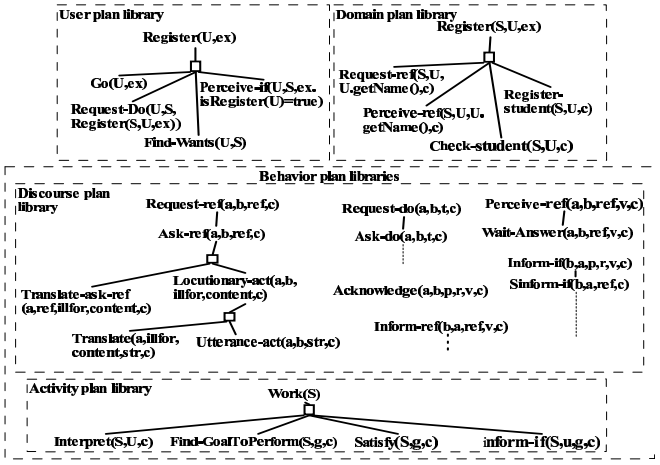

Fig. 1. Agent models

All plan libraries are mono-agent.

The plans in the figure 2 represent the various goals recognized and achieved by the agent S (system) during the proposed dialogue. The plan in the left side (boxed with wide dotted line) describes user's task recognized from his utterances and model. The plan sketched in the right side (boxed with small doted line) represents the system tasks plan recognized by user corresponding in this application to the system plan.

The utterance "Could you register me for the m10 exam ?" is translated by the surface act `Utterance-act(...)`. The plan recognition is applied to interpret (explain) this task from the `Register(...)` task. This process builds the plan in the top left corner (figure 2) by accomplishing the following reasoning: The user has performed the `Utterance-act(...)` to achieve `Locutionary-act(...)`, because the former task is only a subtask of the latter. The `Locutionary-act(...)` is a subtask of several methods, but the utterance form analysis indicates that this utterance is an `Ask-do(...)` act. The `Ask-do(...)` belongs to the `Request--do(...)` method which is a subtask of `Register(u1,m10)` task.

The generation of the "What's your name ?" utterance is provoked by the top right plan execution. The system performs the `Work(s)` task by choosing and executing the method { `Interpret(...), Find-GoalToPerform(...), ...`}, and so on. Utterance production is accomplished by the `Utterance-act(...)` achievement.

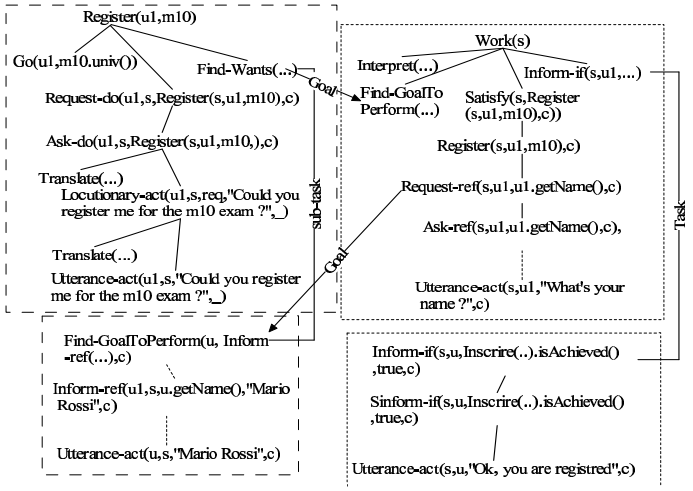

Fig. 2. Dialogue Plans

In the figure 2, the boxed plan corresponds to the path responsible of a speak turn. Subtask/task links mean that the bound tasks are either a subtask or the same task as the upper task. Goal edges specify that an effect of the original task indicates to the hearer that the speaker wants something, and the task in the target side describes the way whose the agent considers the goal[6].

All built plans are mono-agent plans. A direct link between a question (as "What's your name ?") and an answer ("Mario Rossi") does not exist. This link is accomplished through a goal adoption task `Find-GoalToPerform(...)`. We do not try to model the dialogue for itself but we describe the agent behavior, which entails the dialogue generation.

Conclusion and future work

Using plan-based model to generate utterances in dialogue enables an incremental system modeling. It is difficult in practice to completely model all behaviors of an intelligent system and its users in one step. We think that plan-based models appear to be a good way to represent agent's behavior. These models are expressed in a declarative way, which improve the model readability, modularity; they also facilitate the system maintenance, evolution, etc and enable an incremental design.

The dialogue generation and understanding is a huge problem, however we think that the representation of agent's behavior seems to be a good approach to manage dialogue[7]. The knowledge of the others points of view appears to be

[6] In this example, the agent adopts the goal.

[7] Enabling to handle a great number of dialogues.

important to design agents. Of course, all the knowledge constituting an agent is built (and inferred) from the agent's point of view. However, the agent can have some knowledge about points of view of the others. The key point of our modeling principle is that we represent the knowledge of an agent participating to (inside) the collective activity. We do not try to represent the dialogue, neither to describe the knowledge of all agents engaged in the activity. Only the necessary knowledge to the goals achievement of the modeled agent is regarded and established.

The agent plan library seems to be generic and reusable in several applications. Our aim is to design a set of agent plan libraries representing the potential agent behavior. This library set can constitute a knowledge base from which an agent library (for an application) can be constructed. Currently, we are focalized on the cooperation library and the features required to represent (and the way to design) a user model.

References

1. Austin, J.L.: How To Do Things With Words. Havard University Press, Cambridge Massachussetts (1962)
2. Searle, J.S.: Les Actes de Langage. Editions Hermann Paris (1972)
3. Grice, P.H.: Meaning. Philosophical Review **56** (1957) 377–388
4. Bratman, M.E.: Intentions in Communication. In: What is Intention? P r cohen, j l morgan, m e pollack edn. MIT Press, Cambridge MA (1990) 15–31
5. Wilensky, R.: Meta-planning: Representing and using knowledge about planning in problem solving and natural language understanding. Cognitive Science **5** (1981) 197–233
6. Grosz, B., Kraus, S.: Collaborative plans for complex group action. Artificial Intelligence **86** (1996) 269–357
7. Lochbaum, K.E.: A collaborative planning model of intentional structure. Computational Linguistics **24** (1998) 525–572
8. Carberry, S., Lambert, L.: A tripartite model of collaborative dialogue. Submitted to International Journal of Human-Computer Studies (??) ??
9. Ardissono, L., Boella, G., Lesmo, L.: A plan based agent architecture for interpreting natural language dialogue. appear in International Journal of Human Computer Studies, Academic Press **52** (2000) 583–635
10. Litman, D., Allen, J.: A plan recognition model for subdialogues in conversations. Cognitive Science **11** (1987) 163–200
11. Camilleri, G.: A generic formal plan recognition theory. In: IEEE International Conference on Information, Intelligence and Systems ICIIS'99. (1999) 540–547
12. Camilleri, G.: Une approche, basée sur les plans, de la communication dans les systèmes à base de connaissances coopératifs. PhD thesis, Université Paul Sabatier, IRIT (Institut de Recherche en Informatique de Toulouse) (2000)