Planning for Problem Formulation in Advice-Giving Dialogue

Paul Decitre, Thomas Grossi, Cléo Jullien, Jean-Philippe Solvay
Cap Sogeti Innovation
Centre de Recherche de Grenoble
Chemin du Vieux Chêne
38240 Meylan, France

Abstract

We distinguish three main, overlapping activities in an advice-giving dialogue: problem formulation, resolution, and explanation. This paper focuses on a problem formulation activity in a dialogue module which interacts on one side with an expert problem solver for financial investing and on the other side with a natural language front-end. Several strategies which reflect specific aspects of person-machine advice-giving dialogues are realized by incorporating planning at a high-level of dialogue.

Introduction

As performances and scope of intelligent systems increase and the interaction of a system with a user gains in complexity, it becomes desirable to provide an easy initial access to a system for the novice user. Natural language is a medium presumably known by most users. For the system however, it not only requires understanding natural language "utterances" (on a keyboard) but also recognizing the intentions behind these utterances. It leads to a full-fledged dialogue involving much reasoning at the pragmatic level of the communication process. The competence of most intelligent systems is usually bound to a restricted application domain and we can imagine that part of a dialogue is domain-dependent while another is domain-independent. Our efforts aim at designing a dialogue module making these different aspects explicit and interacting with other knowledge-based agents. This work contributes to Esprit Project 316 *Esteam*[1]: An architecture for distributed problem solving by cooperating data and knowledge bases. Advice-giving systems for financial investment have been chosen as a first testbed application. This paper describes preliminary research on the dialogue module of such a system and the resulting prototype.

An advice-giving dialogue comprises three main activities, which may overlap:

- *problem formulation*, where the various needs and capabilities of the user are elicited;
- *resolution*, in which a possible solution to the problem is determined;
- *explanation*, which aims at convincing the user that the solution is in fact what she/he needs.

Our work concentrates on a problem formulation activity in a dialogue module which cooperates with a problem solver and a natural language front-end. The problem solver selects adequate securities for basic investment situations of a private investor and is being developed as part of the same Esprit project [Bruffaerts 1986]. The natural language front-end based on functional grammars is the result of separate research at *Cap Sogeti Innovation* [Fimbel 1985, Lancel 1986].

Computational Aspects

Dialogue and communication theory are a broad field of studies drawing on several disciplines, among them philosophy, cognitive science, and artificial intelligence. The flurry of research devoted to these topics in recent years is largely enough to convince us we could not seriously hope to tackle the "general" problem. We have therefore limited our interest to person-machine advice-giving dialogue and we focus on two essential characteristics of this kind of dialogue:

- the system has intentions and extensive knowledge about the domain which are *a priori* unknown to the user;
- the user's intentions must be interpreted in terms of the system's abilities or inabilities.

We can see the first point as a manifestation of the "expertness" of the system, and the second as a manifestation of the "noviceness" of the user.

We briefly recall some other research issues connected to our work and then elaborate on the specific aspects of person-machine advice-giving dialogue.

Many efforts have been devoted to developing a general theory for speech act understanding [Searle 1969, Allen 1980, Cohen 1979]. Recognizing the illocutionary force of a speech act allows the system to reason about the intentions of the user and to behave accordingly. Most work in this field addresses only isolated speech acts or sometimes single utterances and is not concerned with a possible dialogue setting. Recent attempts, however, for reformulating speech act analysis inside a general theory of action [Cohen 1986] or for applying default reasoning to speech act understanding [Perrault 1986] may yet facilitate an extension to a whole dialogue. Another line of research has taken into account the dialogue dimension [Litman 1984, Wilensky 1984, Carberry 1986] and shown the strong interrelation between dialogue and plans but has

[1]The project is supported in part by the Commission of the European Communities

been mostly concerned with information-seeking dialogues in which the user seems to have an implicit knowledge of what the system can or cannot do. These dialogues often produce patterns of the type "question from the user requiring adequate answer from the system" and seldom consider a possible initiative on the system's behalf. Dialogue parsing is yet another approach which attempts to formalize the surface structure of a dialogue [Reichman 1984, Wachtel 1986]. It could however lead to some rigidity in the interaction between the user and the system; for instance, it may not provide the adequate primitive elements to detect and repair communicative failures in the dialogue. A possible way out would be to account for this surface structure of dialogue within a theory of pragmatics [Airenti 1986].

In person-machine advice-giving dialogue the challenge we face is how to make the expertise of the system accessible to the user in order to satisfy her/his needs: the "expertness" of the system and the "noviceness" of the user force a compromise between the system controlling the dialogue and the user expressing her/himself freely. We chose to rely on the system for conducting the dialogue without however ignoring the initiative of the user, which is to be examined within the intentional framework of the system. In the course of our research, we have derived a few general strategies which typify our approach.

- Whenever possible, the system should set a clear background to the conversation. This is particularly true of the beginning of a session, where the system should not leave the user in the dark but should at once define its own competence and suggest possible options to the user. This initial setting will reflect the global purpose of the dialogue and its expected unfolding.

- Each step of a dialogue takes place in a certain context. We must ensure a common perception of this context by the user and the system if we want a meaningful exchange between them.

- It is worth taking advantage of what the system can expect from the user when the latter "takes the floor" to guide the search of a correct interpretation and quickly decide the best-suited reaction. We should make these expectations of the system apparent in our model of dialogue. Nevertheless, we want the system to allow for user digressions, such as the introduction of a new topic or the correction of a previous statement. It is important to note that a sophisticated dialogue management which would allow the system to react adequately to this unexpected behavior of the user should not impair the straightforward and most probable reaction described just before. It should rather be called upon as a second best choice when the first one has failed, thus defining a preference hierarchy among possible reactions of the system. (In other words, first the clear-minded and obedient user, then the muddle-minded one!)

- The form of the interaction between a user and an advice-giver evolves with the experience they have of each other and the increase of their mutual knowledge, either in the course of a same session or through several sessions. The dialogue system should gradually lead the user toward a simpler and more efficient interface by suggesting the adequate jargon and steps which would allow the user to quicker and better formulate her/his problem [Slator 1986].

Description of the Prototype

• World

The World of our dialogue module consists of a set of objects among which several relations and inheritance mechanisms are defined. For instance, there are classical *is-a* links, *part-of* links (a cash-need is part of the investment plan) and *specification* links (an amount is "specified" by a number and a currency).

Parts of this semantic network are shared with other agents than the dialogue agent, or at least have the same representation in other agents. This is the case between the dialogue module and the problem solver for the problem formulation phase: a model of the expected problem is represented in the World.

For our application, the expected problem consists of an investment plan expressed in terms of the basic investment situations for which the problem solver is able to select the adequate securities. It may include an emergency-fund, *i.e.*, an amount of money which should be available at random time within a given delay, or a cash-need, *i.e.*, an amount of money which should be available at a given date. These financial objects in the problem model are related to objects describing goals and situations of the user's everyday world through *requirement* links. For instance, buying a car in five years may necessitate a cash-need, while covering unexpected expenses may ask for an emergency-fund. These *requirement* links will guide the recognition of the user plan when resolving references. Other domain knowledge for the problem formulation dialogue is encoded in terms of the problem model objects and includes preferred sequences for the interaction with the user and constraints on these objects.

For the dialogue module, the user is considered as another agent and her/his intentions and mental states are represented in terms of positions toward objects of the dialogue. Examples of such positions are 'user understands X', 'system wants to know the value of X', or 'user wants X to take a certain value'. We can view the objects and positions as representing respectively static and dynamic information in the system and allowing the exchange of information between agents.

• Focus-Stack and Agenda

We can characterize each step of the dialogue by a given attentional focus and a given task for the system. In our dialogue module these correspond respectively to a particular object — or set of objects — under discussion and to an action of the system.

During the dialogue, the focus of attention obviously evolves along a chronological dimension: one subject at a time. But a deeper analysis (cf., for example, [Grosz 1985]) reveals a layered structure . In the current prototype, these layers of foci come into play in refinement and digression. Refinement occurs when the treatment of a complex object is split into sub-dialogues about its parts: during such a sub-dialogue, the "parent" and "sibling" objects constitute background context layers. A typical digression takes place when the system suspends information-gathering to give an explanation and comes back to the suspended step of the dialogue. The system keeps track of the active layers of foci in the Focus-Stack.

The sequence of actions the system has currently planned to perform are stored in the Agenda.

• **Architecture**

The dialogue module contains four sub-modules: the INTERPRETER and the GENERATOR are in charge of relating logical form expressions of the natural language front-end to meanings about the World, the EXECUTOR carries out communicative-games for interacting with the user, and the REACTOR activates metaplans for updating the Agenda and the Focus-Stack.

The next sections of this paper investigates in greater detail how the metaplans and communicative-games model the possible actions and strategies of the system and enter into the dialogue planning process. An account on other aspects of this prototype may be found in a previous technical report [Decitre 1986].

High-Level Planning in the REACTOR

From the dialogue module's point of view, the entire conversation results from the goal, "Obtain an investment plan problem specification from the user". The goal is expanded according to the problem model into appropriate subgoals, which are pushed onto the Agenda for sequential execution. As each subgoal is considered, it may be further expanded as necessary. In other words, the decomposition of the communicative intentions (obtaining specifications) reflects the decomposition of the task intentions (investing). There exist two types of metaplans: the metaplans for expanding the Agenda and the metaplans for revising it according to some initiative from the user.

• **Expansion**

As an illustration of the first type of metaplans, let us consider what happens at the beginning of an advice-giving session. When the dialogue starts, the Agenda consists solely of a single action *treat(invest-plan)*. A *treat* action basically corresponds to a sequence of three steps: presentation of the object to the user, asking for values which specify this object, and finally asking for confirmation. But the expansion of *treat* actions can vary according to the type of their argument. For instance, an object may be either simple or complex, it may also be visible or transparent. A transparent object is part of the structure of the problem model but remains invisible to the user. This is the case for *partition(invest-plan)* which consists of the set of the parts of an investment plan, *i.e.*, {*emergency-fund*, *cash-need**, *long-term*}. These transparent objects attempt to model the differences which may exist between how the problem model is organized and how it may be perceived by the user. For a complex object, the expansion introduces treatments for the parts of the complex object, whereas simple objects have only specifications.

Let us just show how these expansion metaplans account for the first two of our general strategies.

The expansion of the initial goal *treat(invest-plan)* posts a *present(invest-plan)* onto the Agenda. The presentation of a complex object such as *invest-plan* reflects how it will be expanded, since the same source of information, *i.e.*, the problem model, is used for presentation and expansion, and thus provides a background setting for the dialogue. The order — in this case obligatory — in which the sub-objects of *invest-plan* are considered is: first, the *total-amount* for the plan; second, the *partition(invest-plan)*. The latter is a transparent object for which adequate presentation rules are defined: the presentation of a partition simply entails a presentation of all parts. The natural language front-end actually generates the following description:

> **system** - "Investment-plan: An investment plan is characterized by a total amount and is usually composed of an emergency-fund, one or several cash-needs and a long-term investment."

Update of the Focus-Stack is also governed by the expansion, and a layer containing all the objects introduced in this presentation is pushed onto the stack. The present example gives [*total-amount*, *emergency-fund*, *cash-need*, *long-term*]. We see again the effect of transparency: the parts themselves are directly pushed onto the stack and not the partition. This layer will constitute the backup layer of the Focus-Stack associated to the overall dialogue setting.

At this stage, the next action on the Agenda is *treat(total-amount)* which may be further expanded in *push-focus*, *ask-info-game*, *check-complete*, *pop-focus*. The ask-info-game is a communicative game which asks a question about the *total-amount* object:

> **system** - "What is the total amount of your plan of investment?"

and waits for the response of the user. The communicative game is designed to induce the user to specialize her/his focus of attention toward the refined context *total-amount*, and *push-focus(total-amount)* places this object on the Focus-Stack, updating it correspondingly.

• **Revision**

Our plan generation is simplified because the execution of one subgoal cannot invalidate another, so a constant monitoring of preconditions is obviated; but this is more than made up by the difficulty in accommodating possible changes to the plan necessitated by the user's input. The choice of a planning process which either expands or repairs an existing plan reflects our third strategy. Indeed, the natural expansion of a plan can be seen as corresponding to the expected behavior of the user and the revisions only happen when the user takes the initiative. In this approach, the reasoning which takes place when the user follows the expected course is reduced to its minimum and only digressions require extra efforts.

Interactions with the user are handled through communicative games and a special metaplan reacts when a communicative game appears on top of the Agenda. This metaplan triggers the execution of the game and analyzes the outcome of the execution to decide consequently the updates to the Agenda. If the game has completely succeeded, *i.e.* all responses of the user fit the expectations, the communicative game is simply removed from the Agenda and replaced by *ok-react* actions for each new position expressed by the user. Otherwise there exist unexpected responses and different actions are pushed onto the Agenda in such a way that the expected positions will be analyzed first by means of *ok-react* actions, then unexpected positions concerning the current focus and unexpected positions outside the current focus by means of *not-ok-react* actions. For all these *not-ok-react* actions, there are metaplans to consider the precise situation and to decide an appropriate reaction, with rearrangement and other modifications made as necessary to the Agenda of pending actions. Delaying the expansion of plans until it

becomes necessary to execute them facilitates taking into account the effect of the user's responses on goals not yet addressed, as in, for example, the verification of constraints which the various parts of the problem definition impose on one another, or in noticing that the value of a missing variable can be computed from the combination of other values the user has already given.

What sorts of snags can occur in a dialogue that might force the system to revise its plans? Our problem model provides certain relations which must hold between values provided by the user. The user might, however, give a value which is in conflict either with one of these constraints or with values previously given. We must point out the sticking-point and help the user resolve the conflict. The *verify-constraint* metaplan pushes a *meet-constraint-game* onto the Agenda. This game will present the local constraint which led to refusing the new position expressed by the user and the justifications which relate this local constraint to the global constraints of the problem model. Consider, for instance, a simple equality constraint between the total amount and the sum of the amounts of the parts. With a $20,000 *total-amount* and a $5,000 *amount* for the *emergency-fund*, a $16,000 assign-value position for the *amount* of the *cash-need* would bring

> **system** -"The amount of your cash-need should be less than or equal to $15,000 for consistency with the total amount."

We also have preferences (and sometimes obligations) in the ordering of the various points to be addressed during the conversation, but the user might not respect them. For instance, the user might at any moment decide to change subject, in which case we must consider the effects of the switch: if, for example, she/he asks to back up in the conversation to change something which was of necessity addressed before the current subject, this could force revision of all the values given since that point up to the present. Based on the following situations, we identify three classes of *change-subject* metaplans, which can trigger when the new position expressed by the user bears on a context which is not the current focus and modify accordingly the Agenda:

- the current focus must be treated before the new subject introduced by the user (according to sequencing policies in the problem model),
- the subject the user would like to examine has already been treated and a modification would have consequences on what has been discussed since,
- there is no sequencing difficulty.

If the user asks for explanation of some point which she/he doesn't understand, the system enters a digression in the dialogue, after which the original topic is resumed.

Low-Level Planning and the EXECUTOR

As discussed above, the decomposition of a plan often engenders the need for interaction with the user. This is done through the communicative games. Basically a communicative game aims at representing a pair of turns between the user and the system, *e.g.*, question/answer. (In fact, we also need to model one-turn games for the transitions between phases, *e.g.*, introduction/resumption of a new/old subject). Although we can never be sure the second turn will take place as desired, the interest of representing games is to provide local expectations for the interpretation of the response of the user. It should be noted that our intention in using these communicative games is not to impose a structure on the dialogue between the user and the system: these games correspond to an ideal dialogue in which the user would always respond as expected. The actual dialogue is a succession of communicative games which may fail, thereby reactivating the high-level planning process described in the previous section.

With each communicative game is associated an *out-meaning* which indicates the semantic content to be conveyed to the user when the game is executed. This *out-meaning* is expressed in the internal language of the dialogue module in which mostly appear objects of the problem model. Adequate references in logical form to these objects are provided by the GENERATOR of the dialogue module. The referring process utilizes:

- the semantic representation of the World;
- the Focus-Stack, especially the current focus which may be elliptically referred to;
- the conceptual state of the user.

This conceptual state is based on initial assumptions, *e.g.*, whether a concept is *a priori* familiar to the user, and on what has already transpired during the dialogue, *e.g.* whether a concept has already been explained, or how the user has previously referred to an object of the problem model. The GENERATOR takes this information to adapt its description and link unknown concepts to familiar ones. Thus the user progressively learns what the problem model consists of and how it relates to her/his familiar concepts: a simple but efficient approach to the evolving interaction between the user and the system held above as our fourth desirable strategy for person-machine advice-giving dialogues.

Symmetrically a communicative game is also characterized by an *in-expected* meaning which stands for the expected response of the user, usually in terms of positions on the current focus or on parts of the current focus. The user's sentence is put into logical form by the natural language front-end and possible meanings are proposed by the INTERPRETER. The latter has to determine which object of the problem model the description of the user could refer to. Each interpretation attempt is done within a context, that is a particular object which is the root of the search process. Interpretation is based on two search strategies: the first uses *specification* links, while the second uses *discriminant* properties and *requirement* links. Two types of reference can be recognized. Direct reference uses only the first strategy following the *specification* links starting from the context object and allows for elliptical answers to questions. Indirect reference uses successively both strategies: a search based on the *discriminant* properties determines candidate objects with a *requirement* link to the context object, then these candidates constitute the starting points for searching along *specification* links. The user does not have the same structured view of the financial world as the system do, and hence will not necessarily refer to things as we would like. The user will talk about "the car I want to buy in five years" which requires a cash-need. Interpretation attempts are ordered according to the stack of foci: the most salient focus (or layer of foci) is selected as context (or set of contexts), then the deeper foci are successively tried. The INTERPRETER only tries

a deeper focus if no interpretation has been found at a higher layer. Moreover, for each layer, the INTERPRETER tries to solve the direct reference before the indirect one and returns all possible interpretations within the first layer and type of reference which permitted to solve the reference. The structure of past foci partly reflects the evolution of our task structure [Grosz 1985] and allows the user to refer back to past segments of the dialogue. This structure is more supple than a mechanism which relies solely on unachieved goals because not only is the focus of a completed task not lost, but its location within this structure is influenced by the problem model in order to optimize subsequent recovery.

Additional knowledge is contained in game descriptions: a feature *in-react* complements *in-expected* in providing a set of game-specific rules for interpreting the literal meaning of the user's response returned by the INTERPRETER into its intended meaning within the particular game considered. A simple example consists of transformation rules for yes-ok/no answers depending on the game.

Conclusion

This work incorporates planning by the system at a high level of dialogue, and nevertheless leaves a great deal of initiative to the user. This flexibility is enhanced by the wide range of input styles which are allowed by the interpretation of input according to focus and indirect reference. At the moment we have a prototype of a dialogue module written in Prolog which implements general strategies for person-machine advice-giving dialogue. The natural-language front-end, written in C, has been interfaced with the prototype, but the generation side would require further investigation. Generalizing the planning component and integrating more sophisticated plan recognition techniques are some of the other issues addressed in a next prototype. Work is also under way to extend the concept base in our knowledge world to enrich the conversation with the user.

References

Airenti G., Bara B.G., and Colombetti M., "Cognitive Pragmatics," Research Report URIA 86-1, Unità di Ricerca di Intelligenza Artificiale, Università di Milano, 1986.

Allen J., "A Plan-Based Analysis of Indirect Speech Acts," *Journal of the Association of Computational Linguistics*, vol. 15, 1980.

Bruffaerts A., Henin E., and Marlair V., "An Expert System Prototype for Financial Counseling," Research Report 507, Philips Research Laboratory Brussels, 1986.

Carberry S., "User Models: the Problem of Disparity," *Proceedings of the XIth International Conference on Computational Linguistics*, pp. 29-34, Bonn (FR Germany), 1986.

Cohen P.R. and Perrault C.R., "Elements of a Plan-Based Theory of Speech Acts," *Cognitive Science*, no. 3, pp. 177-212, 1979.

Cohen P.R., "The Role of Speech Acts in Natural Language Understanding," Tutorials of the XIth International Conference on Computational Linguistics, Bonn (FR Germany), 1986.

Decitre P., Grossi T., Jullien C., and Solvay J.P., "A Summary Description of a Dialoguer Prototype," Technical Report CRG 86-1, Cap Sogeti Innovation, 1986.

Fimbel E., Groscot H., Lancel J.M., and Simonin N., "Using a Text Model for Analysis and Generation," *Proceedings of the Second Conference of the European Chapter of the Association for Computational Linguistics*, Geneva (Switzerland), 1985.

Grosz B.J., "Discourse Structure and the Proper Treatment of Interruptions," *Proceedings of the IXth IJCAI*, Los Angeles (USA), 1985.

Lancel J.M., Rousselot F., and Simonin N., "A Grammar Used for Parsing and Generation," *Proceedings of the XIth International Conference on Computational Linguistics*, pp. 536-539, Bonn (FR Germany), 1986.

Litman D.J. and Allen J.F., "A Plan Recognition Model for Subdialogues in Conversations," Technical Report 141, University of Rochester, 1984.

Perrault C.R., "An Application of Default Logic to Speech Act Theory," *Proceedings of the NATO Workshop on Structure of Multimodal Dialogues Including Voice*, Venaco (France), 1986.

Reichman R., "Extended Person-Machine Interface," *Artificial Intelligence*, vol. 22, pp. 157-218, 1984.

Searle J., *Speech Acts: An Essay in the Philosophy of Language*, Cambridge University Press, 1969.

Slator B.M., Anderson M.P., and Conley W., "Pygmalion at the Interface," *Communications of the ACM*, vol. 29 , no. 7 , pp. 599-604, 1986.

Wachtel T., "Pragmatic Sensitivity in NL Interfaces and the Structure of Conversation," *Proceedings of the XIth International Conference on Computational Linguistics*, pp. 35-41, Bonn (FR Germany), 1986.

Wilensky R., Arens Y., and Chin D., "Talking to UNIX in English: An Overview of UC," *Communications of the ACM*, vol. 27 , no. 6, pp. 574-593, 1984.