A Plan-Based Model for Response Generation in Collaborative Task-Oriented Dialogues*

Jennifer Chu-Carroll
Department of Computer Science
University of Delaware
Newark, DE 19716, USA
E-mail: jchu@cis.udel.edu

Sandra Carberry
Department of Computer Science
University of Delaware
Newark, DE 19716, USA
Visitor: Institute for Research in Cognitive Science
University of Pennsylvania
E-mail: carberry@cis.udel.edu

Abstract
This paper presents a plan-based architecture for response generation in collaborative consultation dialogues, with emphasis on cases in which the system (consultant) and user (executing agent) disagree. Our work contributes to an overall system for collaborative problem-solving by providing a plan-based framework that captures the Propose-Evaluate-Modify cycle of collaboration, and by allowing the system to initiate subdialogues to negotiate proposed additions to the shared plan and to provide support for its claims. In addition, our system handles in a unified manner the negotiation of proposed domain actions, proposed problem-solving actions, and beliefs proposed by discourse actions. Furthermore, it captures cooperative responses within the collaborative framework and accounts for why questions are sometimes never answered.

Introduction
In collaborative expert-consultation dialogues, two participants (executing agent and consultant) work together to construct a plan for achieving the executing agent's domain goal. The executing agent and the consultant bring to the plan construction task different knowledge about the domain and the desirable characteristics of the resulting domain plan. For example, the consultant presumably has more extensive and accurate domain knowledge than does the executing agent, but the executing agent has knowledge about his particular circumstances, intentions, and preferences that are either restrictions on or potential influencers (Bratman 1990) of the domain plan being constructed. In agreeing to collaborate on constructing the domain plan, the consultant assumes a stake in the quality of the resultant plan and in how the agents go about constructing it. For example, a consultant in a collaborative interaction must help the executing agent find the best strategy for constructing the domain plan, may initiate additions to the domain plan, and must negotiate with the executing agent when the latter's suggestions are not accepted (rather than merely agreeing to what the executing agent wants to do). Thus a collaborator is more than a cooperative respondent.

In this paper, we present a plan-based architecture for response generation in collaborative consultation dialogues, with emphasis on cases in which the system and the user disagree. The model treats utterances as proposals open for negotiation and only incorporates a proposal into the shared plan under construction if both agents believe the proposal to be appropriate. If the system does not accept a user proposal, the system attempts to modify it, and natural language utterances are generated as a part of this process. Since the system's utterances are also treated as proposals, a recursive negotiation process can ensue. This response generation architecture has been implemented in a prototype system for a university advisement domain.

Modeling Collaboration
In a collaborative planning process, conflicts in agents' beliefs must be resolved as soon as they arise in order to prevent the agents from constructing different plans. Hence, once a set of actions is proposed by an agent, the other agent must first evaluate the proposal based on his own private beliefs (Allen 1991) and determine whether or not to accept the proposal. If an agent detects any conflict which leads him to reject the proposal, he should attempt to modify the proposal to a form that will be accepted by both agents — to do otherwise is to fail in his responsibilities as a participant in collaborative problem-solving. Thus, we capture collaboration in a Propose-Evaluate-Modify cycle. This theory views the collaborative planning process as a sequence of proposals, evaluations, and modifications, which may result in a fully constructed shared plan agreed upon by both agents. Notice that this model is essentially a recursive one: the Modify action in itself contains a full collaborative process — an agent's proposal of a modification, the other agent's evaluation of the proposal, and potential modification to the modification.

We capture this theory in a plan-based system for response generation in collaborative task-oriented interactions. We assume that the current status of the interaction is represented by a tripartite dialogue model (Lambert & Carberry 1991) that captures intentions on three levels: domain, problem-solving, and discourse. The domain level contains the domain plan being constructed for later execution. The problem-solving level contains the agents' intentions about how to construct the domain plan, and the discourse level contains the communicative plan initiated to further their
joint problem-solving intentions.

Each utterance by a participant constitutes a proposal intended to affect the shared model of domain, problem-solving, and discourse intentions. For example, relating a user's query such as *Who is teaching AI?* to an existing tripartite model might require inferring a chain of domain actions that are not already part of the plan, including \( \text{Take-Course(} \text{User,} \text{AI)} \). These inferred actions explain why the user asked the question and are actions that the user is implicitly proposing be added to the plan. In order to capture the notion of proposals vs. shared plans in a collaborative planning process, we separate the dialogue model into an existing model, which consists of a shared plan agreed upon by both agents, and the proposed additions, which contain newly inferred actions. Furthermore, we augment Lambert’s plan recognition algorithm (Lambert & Carberry 1992) with a simplified version of Eller’s relaxation algorithm (Eller & Carberry 1992) to recognize ill-formed plans.

We adopt a plan-based mechanism because it is general and easily extendable, allows the same declarative knowledge about collaborative problem-solving to be used both in generation and understanding, and allows the recursive nature of our theory to be represented by recursive meta-plans. This paper focuses on one component of our model, the arbitrator, which performs the Evaluate and Modify actions in the Propose-Evaluate-Modify cycle of collaboration.

The Arbitration Process

A proposal consists of a chain of actions for addition to the shared plan. The arbitrator evaluates a proposal and determines whether or not to accept it, and if not, modifies the original proposal to a form that will potentially be accepted by both agents. The arbitrator has two subcomponents, the evaluator and the modifier, and has access to a library of generic recipes for performing actions.

The Evaluator

A collaborative agent, when presented a proposal, needs to decide whether or not he believes that the proposal will result in a valid plan and will produce a reasonably efficient way to achieve the high-level goal. Thus, the evaluator should check for two types of discrepancies in beliefs: one that causes the proposal to be viewed by the system as invalid (Pollack 1986), and one in which the system believes that a better alternative to the user’s proposal exists (Joshi, Webber, & Weischedel 1984; van Beek 1987). Based on this evaluation, the system determines whether it should accept the user’s proposal, causing the proposed actions to be incorporated into the existing model, or should reject the proposal, in which case a negotiation subdialogue will be initiated.

The processes for detecting conflicts and better alternatives start at the top-level proposed action, and are inter-

---

1 A recipe (Pollack 1986) is a template for performing an action. It encodes the preconditions for an action, the effects of an action, the subactions comprising the body of an action, etc.

2 Applicability conditions are conditions that must already be satisfied in order for an action to be reasonable to pursue, whereas an agent can try to achieve unsatisfied preconditions. Our evaluator considers a precondition satisfiable if there exists an action which achieves the precondition and whose applicability conditions are satisfied. Thus only a cursory evaluation of feasibility is pursued at this stage of the planning process, with further details considered as the plan is worked out in depth. This appears to reflect human interaction in naturally occurring dialogues.

3 Much of the information needed for the feasibility and well-formedness checks will be provided by the plan-recognition system that identified the actions comprising the proposal.
in which there is only one generic action (such as Take-Course) that achieves \( A_{t+1} \), but there are several possible instantiations of the parameters of the action (such as Take-Course(UserA,CS601) and Take-Course(UserA,CS621)).

**The Ranking Advisor**  The ranking advisor's task is to determine how best the parameters of an action can be instantiated, based on the user's preferences. For each object that can instantiate a parameter of an action (such as CS621 instantiating \(-\text{course}\) in Take-Course(UserA,\(-\text{course}\)), the **evaluator** provides the ranking advisor with the values of its attributes (e.g., Difficulty(CS621,difficult)) and the user's preferences for the values of these attributes (e.g., prefers(UserA, Difficulty(\(-\text{course}\),moderate), Take-Course, weak)).

Two factors should be considered when ranking the candidate instantiations: the **strength of the preference** and the **closeness of the match**. The strength of a preference\(^4\) indicates the weight that should be assigned to the preference. The closeness of the match (exact, strong, weak, or none) measures how well the actual and the preferred values of an attribute match. It is measured based on the **distance** between the two values where the unit of measurement differs depending on the type of the attribute. For example, for attributes with discrete values (difficulty of a course can be very-difficult, difficult, moderate, easy, or very-easy), the match between difficult and moderate will be **strong**, while that between difficult and easy will be **weak**. The closeness of the match must be modeled in order to capture the fact that if the user prefers difficult courses, a moderate course will be considered preferable to an easy one, even though neither of them exactly satisfies the user's preference.

For each candidate instantiation, the ranking advisor assigns numerical values to the strength of the preferences for the relevant attributes and computes the closeness of each match. A weight is computed for each candidate instantiation by summing the products of corresponding terms of the strength of a preference and the closeness of a match. The instantiation with the highest weight is considered the best instantiation for the action under consideration. Thus, the selection strategy employed by our ranking advisor corresponds to an **additive model** of human decision-making (Reed 1982).

**Example**  We demonstrate the ranking advisor by showing how two different instantiations, CS601 and CS621, of the Take-Course action are ranked. Figure 1 shows the relevant domain knowledge and user model information.

The ranking advisor matches the user's preferences against the domain knowledge for each of CS601 and CS621. The attributes that will be taken into account are the ones for which the user has indicated preferences. For each attribute, the advisor records the **strength of the preference** and the **closeness of the match** for each instantiation. For instance, in considering the attribute **workload**, the strength of the preference will be **low-moderate**, and the closeness of the match will be **strong** and **exact** for CS601 and CS621, respectively. Table 1 shows a summary of the strength of the preferences and the closeness of the matches for the relevant attributes for both instantiations. Numerical values are then assigned and used to calculate a final weight for each candidate. In this example, the normalized weight for CS601 is 43/48 and that for CS621 is 29/48; therefore, CS601 is considered a substantially better instantiation than CS621 for the Take-Course action for UserA.

**The Modifier**  The **modifier** is invoked when a proposal is rejected. Its task is to modify the proposal to a form that will potentially

---

\(^4\) We model six degrees each of positive and negative preferences based on the conversational circumstances and the semantic representation of the utterance used to express the preferences (Elzer, Chu, & Carberry 1994).
be accepted by both agents. The process is controlled by the Modify-Proposal action, which has four specializations: 1) Correct-Node, for when the proposal is infeasible, 2) Correct-Relation, for when the proposal is ill-formed, 3) Improve-Action, for when a better generic action is found, and 4) Improve-Parameter, for when a better instantiation of a parameter is found. Each specialization eventually decomposes into some primitive action which modifies the proposal. However, an agent will be considered uncooperative if he modifies a proposed shared plan without the collaborating agent’s consent; thus, the four specializations share a common precondition — that the discrepancies in beliefs must be squared away (Joshi 1982) before any modification can take place. It is the attempt to satisfy this precondition that causes the system to generate natural language utterances to accomplish the change in the user’s beliefs.

Figure 2 shows two problem-solving recipes, Correct-Relation and Modify-Relation, the latter being a subaction of the former. The applicability conditions of Correct-Relation indicate that it is applicable when the agents, $s_1$ and $s_2$, disagree on whether a particular relationship (such as contributes) holds between two actions ($nodel$ and $node2$) in the proposal. The applicability condition and precondition of Modify-Relation show that the action can only be performed if both $s_1$ and $s_2$ believe that the relationship $rel$ does not hold between $nodel$ and $node2$; in other words, the conflict between $s_1$ and $s_2$ must have been resolved. The attempt to satisfy this precondition causes the system to invoke discourse actions to modify the user’s beliefs, which can be viewed as initiating a negotiation subdialogue to resolve a conflict. If the user accepts the system’s beliefs, thus satisfying the precondition of Modify-Relation, the original dialogue model can be modified; however, if the user rejects the system’s beliefs, he will invoke the Modify-Proposal action to revise the system’s suggested modification of his original proposal.

In order to retain as much of the original proposal as possible when modifying a proposal, Modify-Relation has two specializations: Remove-Node and Alter-Node. The former is selected if the action itself is inappropriate, and will cause the action to be removed from the dialogue model. The latter is chosen if a parameter is inappropriately instantiated, in which case the action will remain in the dialogue model and the problematic parameter will be left uninstantiated.

**Example of Correcting an Invalid Proposal**

Suppose earlier dialogue suggests that the user has the goal of getting a Master’s degree in CS (Get-Masters(U,CS)). Figure 3 illustrates the dialogue model that would result from the following utterances.

(1) U: I want to satisfy my seminar course requirement.
(2) Who is teaching AI?

The evaluation process, which determines whether or not to accept the proposal, starts at the top-level proposed domain action, Satisfy-Seminar-Course(U,CS). Suppose the system believes that Satisfy-Seminar-Course(U,CS) contributes to Get-Masters(U,CS), that U can perform

**Table 2: Correct-Relation and Modify-Relation Recipes**

Satisfy-Seminar-Course(U,CS), and that there is no better alternative to the instantiation of Satisfy-Seminar-Course. The evaluator then checks its child action Take-Course(U,AI). The system’s recipe library indicates that Take-Course(U,AI) does not contribute to Satisfy-Seminar-Course(U,CS), since it believes that AI is not a seminar course, causing the proposal to be rejected.

The modifier performs the Modify-Proposal action, which selects as its specialization Correct-Relation, because the rejected proposal is ill-formed. Figure 4 shows the arbitration process and how Correct-Relation is expanded. Notice that the arbitration process (the problem-solving level in Figure 4) operates on the entire dialogue model in Figure 3, and therefore is represented as meta-level problem-solving actions. In order to satisfy the precondition of Correct-Relation, the system invokes the discourse action Inform as an attempt to change the user’s belief (in this case, to achieve believe(U, ¬holds(contributes, Take-Course(U,AI), Satisfy-Seminar-Course(U,CS)))). The Inform action further decomposes into two actions, one which tells the user of the belief, and one which provides support for the claim. This process will generate the following two utterances:

(3) S: Taking AI does not contribute to satisfying the seminar course requirement.
(4) AI is not a seminar course.

If the user accepts the system’s utterances, thus satisfying the precondition that the conflict be resolved, Modify-Relation can be performed and changes made to the dialogue model. In this example, the proposal is rejected due to an inappropriate instantiation of the parameter course; thus Modify-Relation will select Alter-Node as a specialization to replace all instances of AI in the dialogue model with
a variable. This variable can be re-instantiated by Insert-Correction, the second subaction of Correct-Relation.

Assuming that the system and the user encounter no further conflict in re-instantiating the variable, the arbitration process at the meta-level is completed and the original dialogue is returned to. The proposed additions now consist of actions agreed upon by both agents and will therefore be incorporated into the existing model. Notice that our model separates the negotiation subdialogue (captured at the meta level) from the original dialogue while allowing the same plan-based mechanism to be used at both levels. It also accounts for why the user’s original question about the instructor of AI is never answered—a conflict was detected that made the question superfluous. Thus certain situations in which questions fail to be answered can be accounted for by the collaborative process rather than being viewed as a violation of cooperative behavior.

Example of Suggesting Better Alternatives
Consider the following utterances, whose dialogue model has the same structure as that for utterances (1) and (2) (Figure 3).

(5) U: I want to satisfy my theory course requirement.
(6) Who is teaching CS621?

For space reasons, we skip ahead in the evaluation process to the optimality check for Take-Course(U,CS621). There are two instantiations of course that satisfy the constraints specified in the recipe for Satisfy-Theory-Course: CS601 and CS621. These are ranked by the ranking advisor based on the user’s preferences, summarized in Table 1, which suggests that CS601 is a substantially better alternative to CS621. Thus, Improve-Parameter is selected as a specialization of Modify-Proposal. Similar to the previous example, the Inform discourse action will be invoked as an attempt to resolve the discrepancies in beliefs between the two agents, which would lead to the generation of the following utterances:

(7) S: CS601 is a better alternative than CS621.
(8) CS601 meets at 2pm and involves formal languages and grammar.

Notice that utterance (8) provides supporting evidence for the claim in (7), and is obtained by comparing the sets of information used by the ranking advisor (Table 1) and selecting the features that contribute most to making CS601 preferable to CS621.

The Belief Level
We showed how our arbitrator detects and resolves conflicts at the domain level. Our goal, however, is to develop a mechanism that can handle negotiations at the domain, problem-solving, and discourse levels in a uniform fashion. The process can be successfully applied to the problem-solving level because both the domain and problem-solving levels represent actions that the agents propose to do (at a later point in time for the domain level and at the current time for the problem-solving level); however, the discourse level actions are actions that are currently being executed, instead of proposed for execution. This causes problems
for the modification process, as illustrated by the following example.

(9) U: I want to take AI.
(10) Dr. Brown is teaching AI,
(11) since he is a full professor.

Utterance (11) provides support for (10), which supports (9). However, if the system believes that whether one is a full professor has no relation to whether or not he teaches AI, the system and the user have a conflict as to whether (11) supports (10). Problems will arise if the system convinces the user that Dr. Brown teaches AI because that is his area of specialty, not because he is a full professor, and attempts to modify the dialogue model by replacing the Inform action that represents (11) with one that conveys specializes(Brown, AI). This modification is inappropriate because it indicates that the user informed the system that Dr. Brown specializes in AI, which never happened in the first place. Therefore, we argue that instead of applying the arbitration process to the discourse level, it should be applied to the beliefs proposed by the discourse actions.

In order to preserve the representation of the discourse level, and to handle the kind of conflict shown in the previous example, we expand the dialogue model to include a belief level. The belief level captures domain-related beliefs proposed by discourse actions as well as the relationship amongst them. For instance, an Inform action proposes a mutual belief (MD) of a proposition and an Obtain-Info-Ref action proposes that both agents come to know the referent (MkKnowRef) of a parameter. Thus, information captured at the belief level consists not of actions, as in the other three levels, but of beliefs that are to be achieved, and belief relationships, such as support, attack, etc.

**Discourse Level Example Revisited** Figure 5 outlines the dialogue model for utterances (9)-(11) with the additional belief level. Note that each Inform action at the discourse level proposes a mutual belief, and that supports relationships (inferred from Address-Acceptance) are proposed between the mutual beliefs.

The evaluation process starts at the proposed domain level. Suppose that the system believes that both Take-Course(U, AI) and Build-Plan(U, Take-Course(U, AI)) can be performed. However, an examination of the proposed belief level causes the proposal to be rejected because the system does not believe that Dr. Brown being a full professor supports the fact that he teaches AI. Thus, Correct-Proposal is selected as the specialization of Modify-Proposal in order to resolve the conflict regarding this supports relationship. Again in order to satisfy the precondition of modifying the proposal, the system invokes the Inform action which would generate the following utterance:

(12) S: Dr. Brown being a full professor does not provide support for him teaching AI.

Thus, with the addition of the belief level, the arbitrator is able to capture the process of evaluating and modifying proposals in a uniform fashion at the domain, problem solving, and belief levels. An additional advantage of the belief level is that it captures the beliefs conveyed by the discourse level, instead of how they are conveyed (by an Inform action, by expressing doubt, etc.).

**Related Work**

Allen (1991) proposed different plan modalities that capture the shared and individual beliefs during collaboration, and Grosz, Sidner and Lochbaum (Grosz & Sidner 1990; Lochbaum 1991) proposed a SharedPlan model for capturing intentions during a collaborative process. However, they do not address response generation during collaboration. Litman and Allen (1987) used discourse meta-plans to handle correction subdialogues. However, their Correct-Plan only addressed cases in which an agent adds a repair step to a pre-existing plan that does not execute as expected. Thus their meta-plans do not handle correction of proposed additions to the dialogue model, since this generally does not involve adding a step to the proposal. Furthermore, they were only concerned with understanding utterances, not with generating appropriate responses. Heeman and Hirst (1992) and Edmonds (1993) use meta plans to account for collaboration, but their mechanisms are limited to understanding and generating referring expressions. Although Heeman is extending his model to account for collaboration in task-oriented dialogues (Heeman 1993), his extension is limited to the recognition of actions in such dialogues. Guinn and Biermann (1993) developed a model of collaborative problem-solving which attempts to resolve conflicts between agents regarding the best path for achieving a goal. However, their work has concentrated on situations in which the user is trying to execute a task under the system's guidance rather than those where the system and user are collaboratively developing a plan for the user to execute at a later point in time.

Researchers have utilized plan-based mechanisms to generate natural language responses, including explana-


Conclusions and Future Work

This paper has presented a plan-based system that captures collaborative response generation in a *Propose-Evaluate-Modify* cycle. Our system can initiate subdialogues to negotiate implicitly proposed additions to the shared plan, can appropriately respond to user queries that are motivated by ill-formed or suboptimal solutions, and handles in a unified manner the negotiation of proposed domain actions, proposed problem-solving actions, and beliefs proposed by discourse actions. In addition, our system captures cooperative responses within an overall collaborative framework that allows for negotiation and accounts for why questions are sometimes never answered (even in the most cooperative of environments).

This response generation architecture has been implemented in a prototype system for a university advisement domain. The system is presented with the existing dialogue model and the actions proposed by the user’s new utterances. It then produces as output the logical form for the appropriate collaborative system response. In the future, we will extend our system to include various argumentation strategies (Sycara 1989; Quilici 1991; Maybury 1993) for supporting its claims.

Acknowledgments

The authors would like to thank Stephanie Elzer for her comments on earlier drafts of this paper.

References


