Planning Text about Plans Interactively *

Susan Haller
Computer Science and Engineering Department
University of Wisconsin – Parkside
Kenosha, Wisconsin 53141
haller@cs.uwp.edu
414-595-2343
FAX: 414-595-2114

Keywords: text planning, plan recognition

Abstract

In expository discourse, listeners ask questions in response to unsatisfactory explanations. A system that explains interactively must be able to recognize how to elaborate its answer to produce an explanation that meets a listener's informational needs. Sometimes, listeners ask questions that digress from the purpose of the discussion. Therefore, a system that provides interactive explanations must also be able to distinguish pertinent questions from questions that digress from the discussion purpose. It must also be able to recognize questions that are incoherent. This paper discusses the representation issues that are involved in building such a system.

The Interactive Discourse Planner (IDP) plans text to describe and/or justify a domain plan interactively. To perform these tasks, both the system's text plans and the domain plans are represented uniformly. The uniform representation of information makes it possible for IDP to access both domain and discourse information to recognize how to plan each response. The system's planning rules refer to information at both the domain- and discourse-level. Therefore, these levels of information are reasoned about together to plan additional discourse.

IDP uses questions from the user to recognize how to extend its own text plan in a way that both satisfies its listener and achieves the system's discourse goal. In the process of trying to recognize ways to expand its own text plan, IDP can detect three types of digressions from the discourse purpose that the user can initiate with a question. Two types of text plan representations that are used in the system make it possible for the system to distinguish these types of digressions. As a testbed, IDP plans text to discuss driving route plans.

1 Introduction

In expository discourse, listeners ask questions in response to unsatisfactory explanations. A system that explains interactively must be able to recognize how to elaborate its answer to produce an answer that meets a listener's informational needs. Sometimes listeners ask questions that digress from the purpose of

*I would like to thank Stuart C. Shapiro, and the members of the SNePS Research Group in the Computer Science Department at the State University of New York at Buffalo. Their advice and comments are reflected in the research that this paper describes.

the discussion. Therefore, a system that provides interactive explanations must also be able to distinguish pertinent questions from questions that digress. In addition, it must also be able to recognize questions that are incoherent. These types of questions require different treatment. Pertinent questions must be answered to achieve the discourse purpose. If the user asks a digressive question, the system may need to shift the focus of the discussion back to the purpose. Incoherent questions signal a more serious misunderstanding that requires clarification and repair.

The Interactive Discourse Planner (IDP) is designed to plan text to describe and/or justify a domain plan interactively. IDP uses questions from the user to recognize how to extend its own text plan in a way that both satisfies its listener and achieves the system's discourse goal. In the process of recognizing ways to expand its own text plan, IDP can detect three types of digressions that the user can initiate with a question. As a testbed, IDP plans text to discuss driving routes.

1.1 Problems with Explanation

Separate from NLG, explanation is an area of research associated with expert systems. These are automated reasoning systems that provide expert-quality recommendations in a domain. Expert systems also accept user queries about the reasoning behind those recommendations, and respond by generating explanations. Early attempts at explanation focussed on determining the types of queries that were possible and, for each type, writing rules to access appropriate information in the reasoning chain [Kukich, 1985]. These systems analyze and answer carefully formulated queries the same way each time. The explanations produced are stiff, and they appear to be closely mapped to the reasoning chain that produced the recommendation.

If the user is allowed to interact with the system, another problem that arises is that people often ask vaguely articulated questions like "Why?" and "What?". Under the assumption that the discussion is cooperative, one theory suggests that these kinds of questions are used by the listener to tell the speaker (quite efficiently) what parts of the explanation were not adequate so that appropriate elaborations can be made. The theory behind IDP is based on one mode of cooperative interaction in which the system is the primary speaker and the user is the primary listener. For describing and justifying plans, IDP treats the listener's questions as references to parts of the system's discourse plan that need to be expanded further to

satisfy the listener's informational needs.

1.2 Plan Explanation in IDP

1.2.1 The Discourse Goals

This work considers the situtation where the system is planning discourse about a domain plan. In this context, one or both of the following discourse goals can be active goals that the system is pursuing:

1. to have the user be able to execute the domain plan

2. to have the user adopt the domain plan

The first goal involves planning text to describe the domain plan in enough detail so that the listener can execute it. The second goal requires planning text to justify a recommended domain plan so that the listener will adopt it for use.

1.2.2 Plans as Conceptual Objects

To test my theory of interactive plan explanation, I have designed and implemented IDP to discuss driving routes interactively. In my theory, plans are conceptual objects that can be used as discourse entities. *Discourse entities* are any items that can be referred to in a conversation. They may be actual physical objects (e.g., the sun, my house), or they may be abstract objects (e.g., the weather, my health). *Plans* are often thought of as recipes for action. However, plans can also function as conceptual objects that become discourse entities. In the example,

> Instead of *painting the fence*, why don't you *stain it*?

the speaker invites the hearer to consider two alternative plans. My approach to domain plan explanation involves planning text by selecting content from two sources: the components of domain plans, and the propositions that are inferred in the process of reasoning about domain plans.

1.2.3 Domain-Independent Discourse Plans

In my theory, text planning is domain-independent because discourse plans are used time and again in different situations. Text plans are meta-level plans that select content from the components of a domain plan that is under discussion. Like domain plans, text plans are potential discourse entities. Therefore, I represent the system's text plans in the same formalism that is used for domain plans. The plan formalism is part of the SNePS semantic network knowledge representation and reasoning system [Shapiro, 1979; Shapiro and Rapaport, 1987; Shapiro, 1991; Shapiro and Rapaport, 1992].

1.3 Representing Text Plans and Domain Plans Uniformly

There are three reasons for integrating domain knowledge with text planning knowledge. First, my theory considers discourse planning[1] to be behavior that is part of a more general class of actions that can be used to fulfill goals. For example, a human agent can formulate a plan to mail a letter on her own, or she can plan discourse to request another agent to do it. The knowledge of both plans should be accessible to her so that she can choose between them. Secondly, in interactive explanations, the topic can shift to the explanation itself. To refocus discussions that digress from the discourse purpose or to repair explanations that were misunderstood, speakers often resort to explaining what they were attemping to do with their discourse plan. Thirdly, interactive explanation involves constantly accessing underlying knowledge to plan additional discourse. Since my system uses the same formalism for representing its text plans and domain plans, both are accessible for analyzing user feedback and for reasoning about how to extend the text plan further.

In theory, representing domain knowledge and language knowledge uniformly allows for planning and constraint propagation at all levels, from discourse/text planning down to lexical choice. Planning language at any level should be possible, and should take place when those decisions reflect the communicative goals of the system. These decisions having been made, a system should then pass its plan to a linguistic component for fleshing out. From the standpoint of the deep generation component, any decisions left for the linguistic component should be arbitrary. This research has resulted in the design and implementation of the first level of such a deep generation component. IDP plans at the discourse level to select content and text structure.

[1] I use the term *discourse planning* here to refer to text planning done by people.

I refer to this as *text planning*.

2 Some Examples

The theory that I have tested by implementing IDP leads to a unique combination of features. First, because the system represents its contributions to the discourse as a single, evolving text plan, IDP can generate concise answers to questions that are not cluttered with text repetition:[2]

```
U:    Should I take Maple or Sheridan to go to the Eastern-Hills Mall?
IDP:  take Maple.
U:    Why?
IDP:  taking Maple avoids heavy traffic.
U:    Why?
IDP:  since taking Maple there are fewer businesses than taking Sheridan,
```
(1)

IDP can also plan to include content that anticipates questions:

```
U:    Should I take Maple or Sheridan to go to the Eastern-Hills Mall?
IDP:  if now there was light traffic,
      you could take Sheridan however,
      now there is heavy traffic.
      you should take Maple.
      since taking Maple there are fewer businesses than taking Sheridan,
      taking Maple avoids heavy traffic.
```
(2)

and IDP can change how it plans in reaction to questions:

```
U:    Should I take Maple or Sheridan to go to the Eastern-Hills Mall?
IDP:  take Maple.
U:    Why?
IDP:  if now there was light traffic,
      you could take Sheridan however,
      now there is heavy traffic.
      since taking Maple there are fewer businesses than taking Sheridan,
      taking Maple avoids heavy traffic.
```
(3)

Secondly, IDP can recognize and execute a discourse plan that the listener's feedback suggests:

[2]The interactions that follow are actual demonstrations of the system.

```
U:    Should I take Maple or Sheridan to go to the Eastern-Hills Mall?
IDP:  take Maple.
U:    Why not go on Sheridan?                                          (4)
IDP:  you could take Sheridan however,
      taking Maple avoids heavy traffic.
```

In this demonstration, the user's response *Why not go on Sheridan?*, suggests that the text plan formulated in response should include information about the viability of an alternative route.

Thirdly, IDP can detect listener questions that initiate a digression, and it can use its text plan as text content to return the discussion to the original purpose:

```
U:    Should I take Maple or Sheridan to go to the Eastern-Hills Mall?
IDP:  take Maple.
U:    Why should I take Maple?
IDP:  taking Maple avoids heavy traffic.
U:    Why?                                                             (5)
IDP:  since taking Maple there are fewer businesses than taking Sheridan,
U:    Why is there heavy traffic now? *
IDP:  now is rush hour.
      as I was saying taking Maple is a plan for avoiding heavy traffic.
```

The asterisk (*) indicates where the system determines that the user's question digresses from the original discussion purpose.

3 Plan Recognition

3.1 Traditional Plan Recognition

Carberry notes that plan recognition research deals with bottom-up plan discussions [Carberry, 1990]. Bottom-up plan discussions are question-answering dialogues in which the user knows, or thinks she knows most of her domain plan. The system's task is to recognize the user's domain plan from her queries and thereby provide a helpful response. I will refer to this type of plan recognition as *traditional plan recognition.*

Traditional plan recognition systems like TRACK try to recognize an user's domain plan directly from the utterances that are used [Carberry, 1988]. As indicated below, the input is a representation of the surface-level utterance that the system processes to try and infer the user's intended domain plan.

user's utterance → user's intended domain plan

Researchers have also considered the fact that people use metaplans to inquire about their domain plans. This has led to the development of systems that use the user's initial utterance to try to recognize a metaplan that the user has executed to formulate her intended domain plan. The metaplan that these systems attribute to their user are either a discourse plan [Litman and Allen, 1987; Carberry, 1989]

user's utterances → user's executed discourse plan → user's intended domain plan

or a plan-construction plan [Ramshaw, 1989]

user's utterance → user's executed plan-construction plan → user's intended domain plan

More recently, Lambert has developed a plan recognition model under the assumption that a user uses a discourse plan to execute a plan-construction plan. The user executes the plan-construction plan to formulate her domain plan [Lambert, 1991]. Therefore, this system tries to infer the user's plans in that order.

user's utterance → user's executed discourse plan
→ user's executed plan-construction plan
→ user's intended domain-plan

In summary, traditional plan recognition systems use utterances to infer a user's activity, and then, to infer the domain plan that she intends to execute. The system has a passive role, and the appropriate context for analyzing the user's utterances are the metaplans she uses and the domain plans she may be considering.

3.2 Plan Recognition in IDP

Van Kuppevelt argues that when a speaker produces an unsatisfactory answer to a question, the questioner asks subquestions [van Kuppevelt, 1992]. The questioner may ask several subquestions to obtain the answer that he seeks. This process can continue recursively until the original, topic-constituting question is answered

to the questioner's satisfaction.

IDP generates plan descriptions and justifications interactively in top-down plan discussions. Carberry notes that in top-down plan discussions, the user communicates her domain plan up front because she knows little about how to pursue it and wants the system to tell her. In this context, the system, as the primary speaker, controls the discussion to describe and/or recommend a domain plan. However, the user may want additional information about one or more aspects of it. Therefore, the user, as the questioner, cooperates in the system's discourse plan by asking questions that indicate how she would like the system to plan the discourse to convey the additional information.

To make these indications, the user may ask vaguely articulated questions like *Why?*. If it is necessary, she may refer to aspects of the domain plan or to aspects of an alternative domain plan that she would like fleshed out, for example *Why not Sheridan Drive?*. In this context, the system's task is to use the user's utterance to identify an aspect of the user's intended domain plan that is being questioned. In the example above, the user is questioning the viability of an alternative route. Depending on domain information, that is, whether or not the route is just as good, at least viable, or not viable, the system must identify an appropriate discourse plan to use from a set of possible discourse plans that the user expects the system to use. This leads to the following chain of reasoning about plans.

user's utterance

→ user's intended (or considered) domain plan

→ a discourse plan that the user expects the system to execute

The theory behind IDP holds that for interactive generation, there is a plan recognition process that uses the user's utterance to recognize an aspect of the domain plans that are being considered that she does not know. To make it known, the system must recognize a text plan (TP) that the user intends for the system to use. In this interactive mode, the system has an active role, and the appropriate context includes the system's own TP and the domain plan that is under discussion.

4 Assumptions

4.1 The Explanation Assumptions

IDP works in a collaborative, interactive mode in which the system is the primary speaker, the user is the primary listener, and the system is the uncontested expert. The user makes the initial query. However, the system then has responsibility for seeing to it that the user gets the information that she needs.

The formulation of text plans and the processing procedures for this mode of interaction rely on two simplifying assumptions that explanation systems typically make. I refer to these as the *explanation assumptions* (EAs):

EA-1: The explanation facility's knowledge is correct.

EA-2: The listener automatically believes what the explanation facility informs him of.

EA-1 excuses IDP from learning or correcting its information based on interactions with users. Therefore, the IDP model is not concerned with reasoning about system beliefs in a context where a user presents contradictory ones. EA-2 is related to EA-1. It rules out argumentation. IDP assumes that the listener believes the system's beliefs when they are conveyed to him. This restricts IDP's DGs to those that have to do with the listener acquiring attitudes from the beliefs that he automatically acquires from the system. EA-2 is a strong assumption that is made because the focus of this research is not on belief. Given that the user believes what is said, she will often want to know the reasoning that leads to statements of fact.

The explanation assumptions are not appropriate for applications like cooperative work and tutoring. A cooperative work system uses interaction to carry out a task with the user. Such a system must be able to revise its knowledge about the domain to account for the occurrence of unexpected events as the task is carried out. Therefore, EA-1 is not a valid assumption. In contrast, tutoring systems can be assumed to contain correct information. However, a tutoring system must contend with a student's potential misconceptions, misunderstanding, and nonunderstanding. Therefore, EA-2 is not a valid assumption. However, the explanation assumptions are appropriate for those applications in which relatively simple factual information is disseminated. For example, people routinely ask for directions, travel advice, insurance claim information to name a few.

4.2 The Discourse Knowledge Assumptions

Grice's work on conversational implicatures is founded on the *cooperative principle* [Grice, 1957]. Grice examined the relationship of utterances to meaning and argued that the effect that a speaker intends for an utterance to have on a hearer, is the meaning of an utterance. The cooperative principle presupposes that discourse knowledge is common knowledge that conversational participants share. Starting with Grice's pragmatics, Thomason argues that at a given stage of a discourse, a hearer's task is to reconstruct the speaker's discourse plan [Thomason, 1990]. Often, this plan is part of a mutual plan for the discourse. To do this, Thomason contends that discourse must have a recognizable structure that allows both participants to reconstruct, remember, and anticipate the progress of a conversation. He refers to this as the *conversational record*.

Computational models have incorporated these ideas independently. In their model for plan recognition, Litman and Allen rely on the participants' knowledge of *commonsense* plans [Litman and Allen, 1990]. Commonsense knowledge refers to the intuitive beliefs and theories about the world that are used by artificial intelligence systems. Several systems have been designed that make tacit assumptions about the user's discourse knowledge. Litman and Allen's model for plan recognition requires that hearers be able to recognize a speaker's discourse plan when it is used [Litman and Allen, 1987]. Carberry's model uses the discourse expectations that an utterance invokes [Carberry, 1989]. Ramshaw posits that there are metaplan constructors that speakers use and hearers can recognize [Ramshaw, 1989]. Lambert's model relies on the existence of both metaplan constructors and discourse plan schemas to recognize domain plans [Lambert, 1991]. At a minimum, an implicit assumption that underlies all of this work is that discourse plan schemas are common knowledge.

In the IDP model, we make our assumptions about discourse knowledge explicit, stating them in terms of our ontology and the interactive mode that IDP operates in. These are called the *discourse knowledge assumptions* (DKAs).

DKA-1: Discourse knowledge is common knowledge.

DKA-2: When a discourse plan is used, the listener recognizes how the system's intent is being realized.

DKA-1 allows us to assume that IDP's text plan operators are knowledge that language-users have and assume that they share with other language users when they communicate. DKA-2 guarantees IDP that its listener recognizes the system's text plan as it is formulated and executed. It assumes that people perform the task that plan recognition systems seek to emulate. Although it is a strong assumption, it allows me to focus the current investigation on the problem presented by cooperative, task-oriented conversations where intentions are clear.

4.3 Direct vs. Indirect Feedback

In cooperative communication, a speaker is minimally entitled to the belief that his intent is understood. Moreover, by invoking the Gricean Maxim of Relation, the listener's feedback must somehow address that intent. Feedback can address the speaker's intent directly:

A: Take Maple.
B: OK. / No. (6)

In the above interaction, B recognizes that A's intent is to have B adopt the recommended plan. If B replies with "OK.", he acknowledges A's intent and he signals A that he has achieved what he intended. If B replies with "No.", he signals A that his attempt has failed.

Feedback may address the speaker's intent indirectly:

A: Take Maple.
B: Why? (7)

Like the "No"-reply, B's question signals A that A has not succeeded. However, it encourages A to provide an answer that will achieve his intention. Furthermore, by the Maxim of Quantity, people say as much as is required:

A: Take Maple.
B: Why not take Sheridan? (8)

In this version of the interaction, B's feedback indicates that he would like A's answer to include information

about another discourse entity.

4.4 The Feedback Assumptions

To develop an algorithm for analyzing the listener's responses, I use the following assumptions about the nature of the listener's feedback. These are the *feedback assumptions* (FAs):

FA-1: Feedback addresses the system's intent either directly or indirectly.

FA-2: Feedback in the form of questions indicates how the system should continue to plan to achieve its intent.

FA-3: The more information that feedback contains, the more that is needed for the system to recognize how to continue.

By FA-1, the IDP analyzer can use its own DG and DTP as the relevant discourse context. FA-2 guarantees that relevant questions are answerable by replanning the DTP, or extending it. FA-3 widens the scope of the search for an appropriate extension of the DTP when feedback includes additional information.

5 The Text Plans and Goals

5.1 The Discourse Goals

As the primary speaker, IDP is responsible for planning text to describe and/or justify a domain plan, and the user, who is the primary listener, is responsible for providing feedback that lets the system know how to expand its text plan in a way that is helpful. IDP's *intentions* are synonymous with the system's *discourse goals* (DGs). A DG is an unachieved goal that has to do with the attitude or abilities of the user.

With respect to plan discussions, IDP can have two kinds of DGs. The first DG has to do with having the user adopt a plan, ?p.

DO(user, adopt(?p))

The second DG is to have the user be able to execute a plan.

CANDO(user,?p)

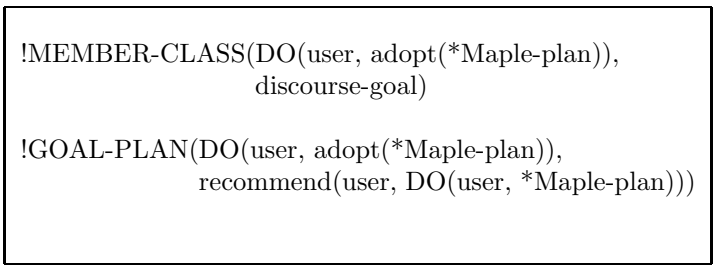

Figure 1: Knowledge Associated with a DG – An Example

5.2 Planning for Discourse Goals

The IDP model separates DGs from text plan (TP) effects. By EA-2, IDP asserts the effects of a TP as true when it executes the TP. This is consistent with the idea that any attempt to communicate has an effect, even if it fails to achieve the DG. In contrast, IDP does not automatically assert a DG when a plan to try to achieve it executes. A DG is *active* while it remains unasserted in the knowledge base. IDP analyzes feedback to determine if its TP has achieved the active DG for which it was planned.

Figure 1 gives an example of the representations that IDP associates with the DG of having the user adopt the Maple Road plan. The MEMBER-CLASS statement is used to assert meta-level knowledge that classifies the goal as an active DG. To associate a plan with the DG, IDP uses the GOAL-PLAN statement. In Figure 1, the GOAL-PLAN statement asserts that recommending to the listener that he do the Maple Road plan is a plan for the DG.

5.3 Content Goals

Using EA-1 and EA-2, goals that have to do with the user knowing a proposition are different from DGs. A *content goal* (CG) specifies a system goal to have the user know a proposition. Since user belief is automatic, and the effects of IDP's text plans are guaranteed, a CG is achieved immediately when IDP executes a text plan that has the CG as one of its effects.

Figure 2 gives representations that IDP uses for the CG: to have the user know that there are fewer businesses on the Maple Road route plan. The first asserted proposition states that having the user know that the Maple Road route plan has fewer businesses along it is a content goal. The second proposition states that the act of providing a circumstance under which taking Maple avoids heavy traffic has the effect

```
!MEMBER-CLASS(KNOW(user, FEWER-BUSINESSES(*Maple-plan, *Sheridan-plan)),
                    content-goal)

!ACT-EFFECT(circumstantiate(ACT-PLAN(avoid(heavy-traffic, *Maple-plan))),
                  KNOW(user, FEWER-BUSINESSES(*Maple-plan, *Sheridan-plan)))
```

Figure 2: Knowledge Associated with a CG – An Example

```
!FORALL-ANT-CQ({?g1, ?g2, ?p},
                       {GOAL-ACT(?g1),
                        ACT-PLAN(?g1,?p),
                        SECONDARY-GOAL-ACT(?g2),
                        ACT-PLAN(?g2, ?p)},

                       ACT-PLAN(motivate(user, DO(user, ?p)),
                                   snsequence(advise(user, DO(user, ?p)),
                                                 circumstantiate(ACT-PLAN(?g2, ?p)),
                                                 say(ACT-PLAN(?g2, ?p)),
                                                 restate(ACT-PLAN(?g2, ?p)))))
```

Figure 3: A TP-operator for Motivate

of letting the user know that there are fewer businesses on the Maple Road route plan.

5.4 Text Plan Operators

IDP's text plan operators (TP-operators) are based on Rhetorical Structure Theory (RST) [Mann and Thompson, 1987] and are written using the SNePS Actor planning formalism [Shapiro *et al.*, 1989]. In RST, each essential text message (called the *nucleus*) can be augmented with additional information (called the *satellite*) through a *rhetorical relation*. In the planning formalism, plan operators are written as rules that state what consequents can be deduced from a set of antecedents.

Figure 3 shows a TP-operator for the *motivate* act. Arguments that are enclosed in braces are unordered set arguments. In the formalism, an *act* decomposes into one or more structures of other acts called *plans*. IDP instantiates plans, preconditions, and effects for a given act by satisfying a rule's antecedents. These are the *constraints* on the plan, and the process of constraint satisfaction selects new content for the text. For TP-operators that are based on rhetorical relations, this new content is a satellite proposition that is appropriate to the rhetorical relation and the given nuclear proposition.

The TP-operator in Figure 3 asserts that if there is a domain goal-act ?g that is enacted by a plan ?p,

and a secondary goal-act ?g2 that is also enacted by plan ?p, then a plan for the act of motivating the user to do ?p is a sequence of up to four acts. First, advise the user to use plan ?p. Then, provide circumstantial information in support of the fact that ?p is a plan that achieves the secondary goal-act ?g2. State that ?p is a plan for achieving ?g2, and finally, restate that ?p is a plan for achieving ?g2.

5.5 The Kinds of Text Plans

IDP uses two types of *text plans* (TPs) separating those plans that address the system's DG directly, from those plans that provide additional information that augments the system's essential text message. The two kinds of TPs are *discourse text plans* (DTPs) and *content-selection text plans* (CTPs). The overarching plan is always a DTP. This is consistent with Moore and Pollack's contention that a speaker always structures information in a discourse with a high-level intention in mind [Moore and Pollack, 1992].

The division is based on a two-way division of the rhetorical relations that Mann and Thompson describe. According to Mann and Thompson, a *rhetorical relation* is a relation that is communicated by the speaker implicitly when he juxtapostions propositional content in his text plan. Each *presentational relation* relates pieces of text content for the purpose of increasing an inclination in the hearer. For example, in

> Take Maple. You'll avoid heavy traffic.

the speaker follows his advice with a fact that is understood by the hearer to be motivation for the advice. The hearer also understands that it is an attempt by the speaker to make the hearer accept the advice. In contrast, a *subject-matter* relation is used by a speaker with the intent of simply informing the hearer of the rhetorical relation that holds between two pieces of text. For example, in

> It is rush hour. There will be heavy traffic.

the speaker informs the hearer implicitly that the first clause is a circumstance behind the second statement.

In the IDP model, DTPs are used to attempt and reattempt the achievement of the system's discourse goals (DGs). Since these goals have to do with affecting the listener's attitudes and abilities towards domain plans, DTPs are based on speech acts and presentational rhetorical relations. The DTPs describe how to try to achieve discourse goals by selecting some text content. IDP can augment this essential content without

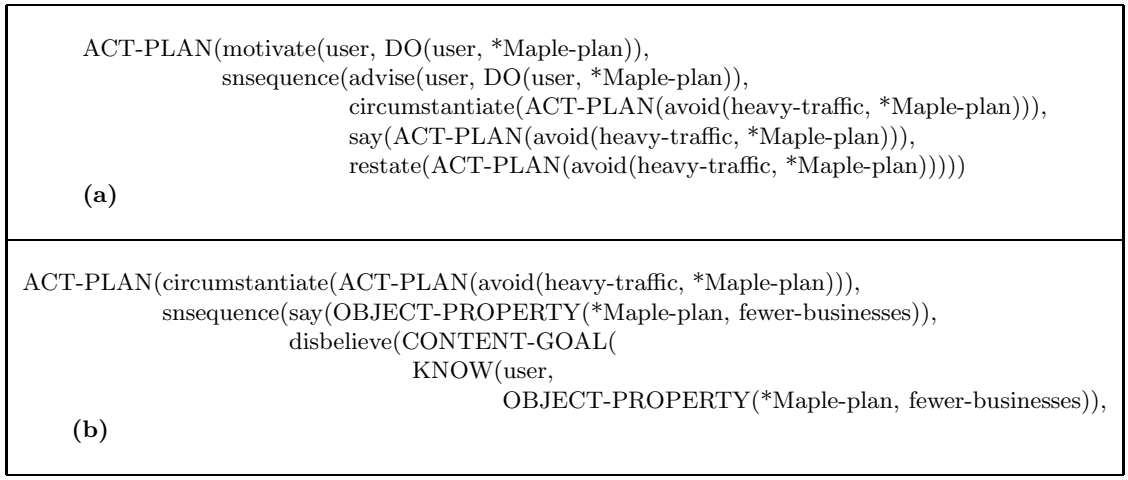

Figure 4: (a) A DTP for Motivate (b) A CTP for Circumstantiate

detracting from communicating its own intent by using one or more CTPs. Therefore, CTPs correspond to subject-matter rhetorical relations.

Figure 4(a) shows a DTP for *motivate* that IDP instantiates from the TP-operator given in Figure 3. The motivate act takes the listener and a nuclear clause (the listener taking the Maple Road route) as its arguments. The DTP for motivating the listener to take the Maple Road route sequences up to four other acts. Two of these acts are references to CTPs that are potential plan expansion points: *circumstantiate* and *restate*. A plan for circumstantiate is given in Figure 4(b). Since CTPs are not executed to affect the listener in any way other than to provide information, the listener is not an argument to acts for CTPs. IDP can only deduce a CTP for an act when there is an active CG that the plan satisfies. A constraint on all CTP-operators requires there to be an active CG to let the listener know the proposition that will be the satellite in a subject-matter rhetorical relation.

As shown in Figure 4(b) as the second step in executing the circumstantiate CTP, the CG is retracted (*disbelieved* by the system). Because this particular plan is deducible only when the CG exists, the SNePS Belief Revision component (SNeBR) [Martins and Shapiro, 1988] retracts the ACT-PLAN proposition from the knowledge base as part of the execution of the CTP. Unless there is an active CG the CTP cannot be deduced. This keeps the CTP from being expanded and used again even though it appears in the body of other DTPs like the one for the motivate act (Figure 4(a)).

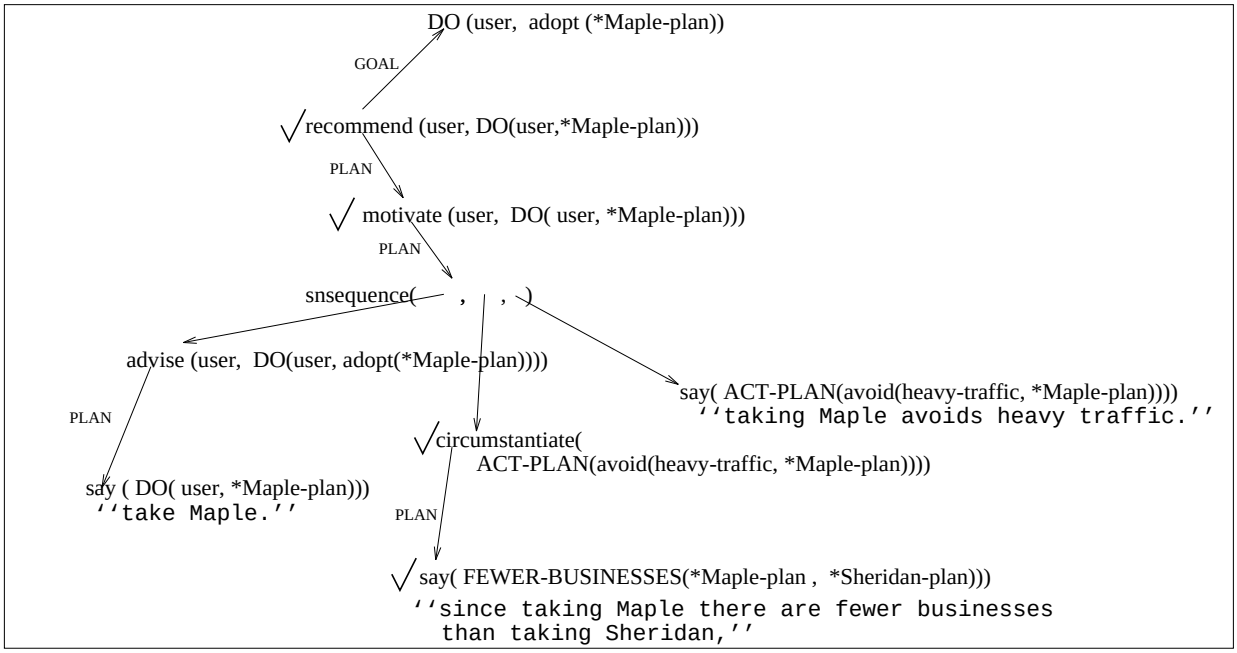

Figure 5: The TP for Before the Digressive Question in Demonstrations 1

5.6 The TP

Figure 5 shows IDP's TP after its third response in Demonstration 1. The TP has been formulated to achieve the DG of having the listener adopt the plan to take Maple Road. The high-level plan is a DTP which can decompose into other DTPs and CTPs. The TP always bottoms out in the primitive act, *say*. The argument to the say act is a text message which includes a proposition as the content to be expressed. In the TP, the checks ($\surd$) mark the active path. Note that the plan for motivating the listener has not been executed in the order indicated by the sequencing act *snsequence* (see Figure 4(a)). In particular, the second act expands to an optional CTP, which IDP does not use until it responds to `Why?` a second time. This happens because IDP replans and expands its TP in reaction to the listener's questions.

I have changed the semantics of the sequencing act to accommodate this behavior. If a plan for an act in the sequencing act cannot be inferred (as is the case with CTPs when there is no CG), the act is skipped over in the sequence. This is why *circumstantiate* does not execute the first time that the *motivate* act is expanded and planned. Acts in the sequence are also skipped if they have already been performed. This is why the system does not execute the *advise* act a second time again when the *motivate* act is expanded into a plan. Future work will involve representing this type of action control declaratively if experimentation

with the current implementation suggests that it is desirable.

6 The Discourse Context

To make the several sources of information that are needed for highly interactive explanations available, I represent them uniformly using the SNePS Semantic Network Processing and Reasoning System [Shapiro and Rapaport, 1987; Shapiro and Group, 1992]. Based on the SNePS Actor, IDP models a cognitive agent operating in a single-agent world [Kumar *et al.*, 1988].

IDP's DGs are unachieved system goals that have to do with the attitude or abilities of the user toward a domain plan. In the role of the primary speaker, IDP can post one or both of the following DGs:

1. to have the user adopt a domain plan
2. to have the user be able to execute a domain plan

IDP plans text to try to achieve its DG, and it interprets the user's feedback in the context of three types of knowledge that are all related to its TP:

1. the active path
2. growth points
3. the localized unknowns

Following Carberry, once the user knows the system's intention and how it has been realized, the user has expectations for what will follow [Carberry, 1989]. Motivated by Grice's Maxim of Relation [Grice, 1975], IDP analyzes questions using *growth points* on the *active path*. The active path marks the TPs that make up the most recent expansion of IDP's overall TP. Growth points are references to other TPs that are embedded in the body of each TP along the active path. Growth points suggest ways of adding content that augment the current TP [Hovy, 1990]. IDP also uses a set of propositions called the *localized unknowns*. The localized unknowns are propositions about domain plans and domain-related reasoning that, based on the user model, the user does not know. The localized unknowns are linked to the system's TP by the reasoning chains that were used to derive the TP.

```
PROCESS-DIRECT(fb)

If  is-acknowledgment(fb)
    then Let current-dg := find-most-recent-dg
         assert-in-kb(variable-value(current-dg))
         If   is-precondition(current-dg)
              then  Let dtp :=find-act-with-precond(current-dg)
                    continue-planning(dtp)
         Else terminate-planning
Else [fb is "No"] terminate-planning
```

Figure 6: Algorithm for Processing Feedback that Addresses the System's DG Directly

7 Processing Direct Feedback

Figure 6 gives an algorithmic description of how IDP's analyzer processes feedback that addresses the system's DG directly. If the feedback (**fb**) is an acknowledgment like *Ok* or *Right*, then the active path is traversed in the reverse direction of expansion until an active DG is found (**current-dg**) and asserted in the knowledge base. Furthermore, if this DG is a precondition to another DTP, then the planner-actor is re-invoked to continue planning that DTP. Otherwise, if there are no other active DGs, the analyzer terminates planning and the discussion ends.

If the feedback is *No*, the discussion is terminated also. Processing negative feedback involves argumentation. Argumentation involves sharing conversational control and reasoning about the user's beliefs. This is beyond the scope of the current investigation.

7.1 An Example

Figure 7, shows the system's TP for a simple interaction like the one in 6. In reaction to the initial query, IDP posts a DG to achieve a state where the user has adopted the plan to take Maple Road.

AGENT-ACT(user, adopt(*Maple-plan))

The full structure of IDP's domain plans is not presented here. The notation *Maple-plan represents that structure.

IDP plans the simple advice that it realizes as `take Maple.` As indicated, a DG can be accessed from a TP in the network through the GOAL-PLAN proposition. When the user replies `Ok.`, the analyzer starts

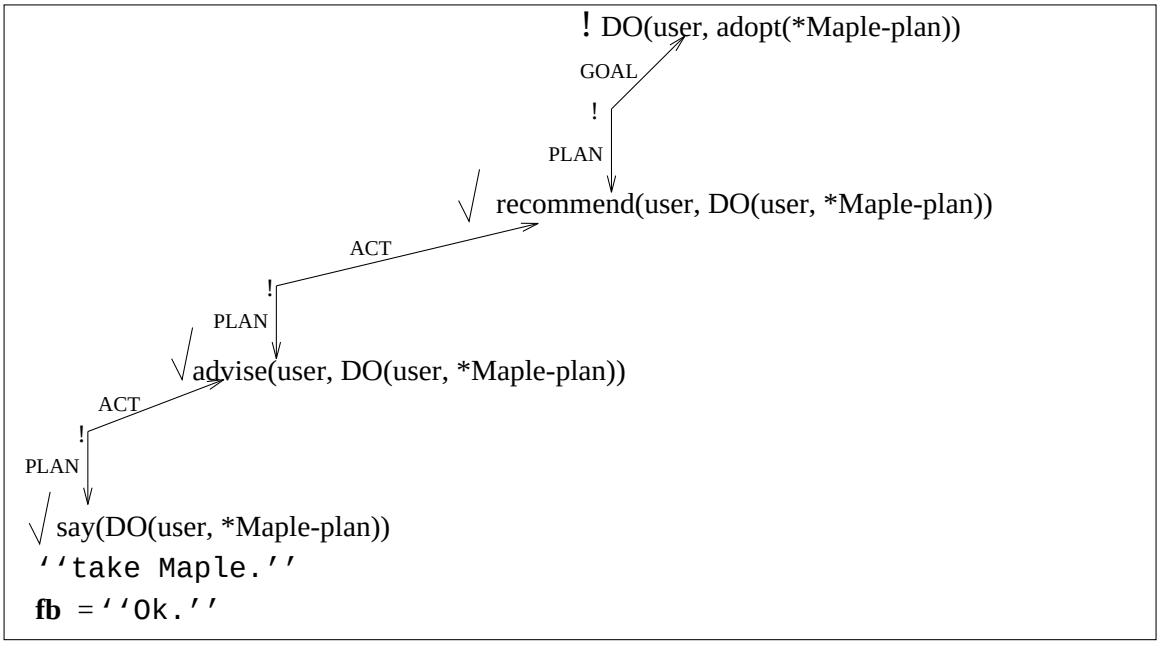

Figure 7: The TP After the Acknowledgment in Demonstration (1)

with the last act on the active path (the say act), and searches up the active path until it finds an active DG. In Figure 7 the only active DG is to have the user adopt the Maple Road plan. Therefore, the analyzer asserts the DG (with the !) and removes the active path. In the absence of any other active DGs, IDP terminates the interaction.

7.2 Declarative vs. Procedural Plans

A major issue in knowledge representation is the determination of what knowledge should be represented declaratively and what knowledge should be encoded in procedures. The IDP analyzer algorithms look very much like procedural plans. In particular in the algorithm above, asserting the DG when an acknowledgment is received (and thereby deactivating it) seems analogous to the step in each CTP where the CG is retracted (thereby deactivating it). However, the two types of goals are different. Recall that a CG has to do with the user knowing a proposition. By the second explanation assumption (EA-2), the listener automatically knows every proposition that the system conveys. Since the achievement of a CG is guaranteed when a CTP a CTP is executed, retracting the CG is stated declaratively as a step in the CTP that achieves it.

In contrast, recall that a DG has to do with influencing the attitudes and ablilities of the user. A DG is

not automatically achieved when a DTP that is planned for it executes. Instead, the system analyzes the user's feedback to see its DTP has met with success. A DG is achieved if the user acknowledges that she is satisfied with the system's explanation. However, if the user asks further questions, the system must replan its DTP or augment it with a CTP to try once more to achieve its DG. It is the system's attempts to achieve its current DG that drive the interaction. It is not known when, or if, the user will be satisfied. Therefore, the assertion/retraction of DGs is encoded procedurally in the algorithms responsible for analyzing the user's feedback, and replanning the DTP that is used to try to achieve a DG.

I have made every effort to represent IDP's planning knowledge declaratively so that it can be the content of plan discussions as well as executable code. It is possible to represent planning knowledge declaratively when the effects of plans are deterministic. However, strong assumptions (the Explanation Assumptions) have allowed me to make some text plans (namely, the CTPs) deterministic. To add to the system's capabilities, these assumptions would have to be lifted.

8 Processing Indirect Feedback

8.1 The Local Topic

In the IDP model, questions address the sytem's DG indirectly by making reference to what the system is doing to achieve its DG. Therefore, IDP uses feedback to try to recognize a TP to expand its current DTP. Following van Kuppevelt, the *local topic* is that which is questioned. IDP determines a local topic to search for a comment on it.

IDP processes questions in one of the following forms:

1. Why {not}?
2. Why {not} *plan*?
3. Why {not} *proposition*?

{not} indicates that the word "not" is an optional constituent of the input string. The parsing grammar requires these patterns literally. For example, to ask the system *Why does not taking Maple avoid heavy traffic?* the user must enter

```
Why taking Maple avoids heavy traffic.
```

The local topic is a domain act or proposition. When there is a simple why-question (1), the local topic is the last proposition that was expressed by the system with a say-act. If the question is in the form of 2 or 3, IDP makes the plan or proposition that is mentioned the local topic.

8.2 Planning Coherent Continuations

In the IDP model, to measure the coherence of I exploit the assumed mode of interaction, that is, that the system is controlling the conversation to achieve its discourse goal. In this context, when there are several text plans to choose from the best is the one that contributes most directly to the achievement of the system's goal. Discourse expectation constrains IDP's choices to the growth points on TPs along the active path. To select a TP-expansion, an important heuristic is the degree to which the proposed TP-expansion highlights the system's intent as realized by the DTP-level portion of its TP. Therefore, IDP considers the growth points for the DTPs on the active path in the following order:

1. DTPs that replan a DTP
2. CTPs that expand a DTP

IDP prefers DTPs that replan a DTP over CTPs that expand a DTP. This encodes a preference for plans that highlight the system's intent over plans that supply additional information.

8.3 The Algorithm

Figure 8 gives an algorithmic description of how IDP processes feedback that addresses the system's intent indirectly. IDP searches growth points that are DTPs (i.e. **type-of-growth-pt** = DTP). If an appropriate TP-expansion is not found, IDP searches growth points that are CTPs (i.e. **type-of-growth-pt** = CTP). If the user's feedback is "Why" or "Why not" then the local topic (**local-topic**) becomes the last proposition that IDP expressed. Otherwise, if the user asks why about an object or proposition that she mentions, that becomes the local topic. After collecting the localized unknowns, and initializing the new **tp-expansion** to nil, IDP starts with the last **dtp** on the active path.

While IDP has not found a **tp-expansion**, and a **dtp** exists, IDP collects the **growth-points** for the

```
PROCESS-INDIRECT(fb, type-of-growth-pt)

If      fb = "Why" {"not"}
        then Let local-topic := get-last-proposition-expressed
Elseif  fb = "Why" object-or-prop
        then Let local-topic := object-or-prop
Let localized-unknowns := get-localized-unknowns
Let tp-expansion := nil
Let dtp := get-last-dtp-on-active-path
Loop-while tp-expansion = nil and dtp
    Let growth-points := get-growth-pts(dtp, type-of-growth-pt)
    Loop-for-each growth-point in growth-points
        If lets-user-know(one-of(localized-unknowns, effect-of(growth-point)))
                and  nucleus-of(growth-point) = local-topic
          then Let tp-expansion := add(growth-point, tp-expansion)
    End-loop
    If more-than-one(tp-expansion)
      then Let tp-expansion = get-simplest(tp-expansion)
    If tp-expansion = nil
      then Let dtp = go-up-active-path-from-current(dtp)
End-loop
return(tp-expansion)
```

Figure 8: Algorithm for Processing Feedback that Addresses the System's DG Indirectly

current **dtp** of the type specified. If the growth point lets the user know one of the localized unknowns, and if its nucleus is the local topic, then the growth point is collected as a possible **tp-expansion**. After all of the growth points are tested, if there is more that one **tp-expansion**, the simplest plan expansion is returned. If no **tp-expansion** is found, IDP backs up the active path to test another DTP. If IDP finds a TP-expansion or processes the entire active path, the procedure terminates.

8.4 Examples

Figure 5 shows IDP's TP after Demonstration 1. In response to the user's first why-question, IDP replans the recommend DTP with the motivate DTP. When the user asks `Why?` a second time, IDP cannot find a way to replan its DTP. However, it does find a CTP, circumstantiate, that expands the motivate DTP. Note that the plan for motivate has not been executed in the order indicated by the sequencing act *snsequence*. In particular, the second act expands to the CTP circumstantiate, which IDP does not use until it responds to `Why?` a second time. In demonstration run (3), the user mentions an alternative plan. IDP uses the mentioned plan as the local topic, and identifies the DTP concede as the continuation that the user seeks.

9 Digressions

9.1 Characterizing Digressions from Discourse Purpose

Grosz and Sidner define a digression as a type of interruption [Grosz and Sidner, 1986]. Interruptions are utterances that don't fit in with the discourse. They note that interruptions take several forms ranging from ones that do not have any relevance to the ongoing discussion, to interruptions that are quite relevant. They define a *digression* as a discourse segment with a purpose that does not contribute to the purpose of the interrupted segment. It is linked to the current discourse segment only through a common discourse entity. Using their example, if while discussing Bill's role in company ABC, someone interrupts with *Speaking of Bill, that reminds me, he came to dinner last week.*, Bill, the discourse entity, remains salient. However, the discourse purpose changes along with the aspects of Bill that are relevant to achieving the new discourse purpose.

In the IDP model, a user-imposed digression occurs when the user asks a question that no longer addresses the system's purpose. The system's purpose (or intent) is expressed by its discourse goal and the text plan that it formulates and executes to try to achieve it. IDP detects a digression when the answer to a user's question cannot be incorporated into its existing text plan. IDP operates in a cooperative interactive mode in which the system is the primary speaker and the user is the primary listener. Henceforth, I refer to the user as the listener. In this interactive mode, IDP controls the discussion to make its own intentions clear and to try to achieve them.

9.2 A Demonstration

The following interaction demonstrates one of three types of digressions that IDP can detect and how the system responds to it. It is the simplest kind of digression that IDP can detect, and it occurs when the user asks a direct question about a proposition that was expressed earlier in the discourse.

```
PROCESS-DIGRESSION(fb)

   Let ctp =: IS-DIGRESSION(fb)
   If ctp
      Then save(*active-path*)
           schedule-plan-for(ctp)
           restore(*active-path*)
           Let last-dtp =: get-last-dtp
           Let last-say-act =: get-say-act-in(last-dtp)
           post-content-goal(KNOW(listener,
                                  DO(system, last-say-act))
           replan(last-dtp)
      Else say("Huh?")
```

Figure 9: Processing Digressions

```
  U:  Should I take Maple or Sheridan to go to the Eastern Hills Mall?
IDP:  if now there was light traffic,
      you could take Sheridan, however,
      now there is heavy traffic.
      you should take Maple.
      since taking Maple there are fewer businesses than taking Sheridan,
      taking Maple avoids heavy traffic.
  U:  Why is there heavy traffic now? *
IDP:  since now is rush hour,
      as I was saying taking Maple avoids heavy traffic.
```
(9)

As part of IDP's advice, it has expressed a proposition that lets the listener know circumstances that motivate the advice. In particular, IDP has told the listener that `now there is heavy traffic.` The system identifies the listener's question about this statement (marked with an asterisk) as a digression from its intent. Therefore, after answering the question, IDP shifts the discussion back to its recommendation.

9.3 Planning Text to Recover from a Digression

I do not have a theory of managing discourse initiative. Therefore, the IDP model does not address the question of how to decide if a digression should be allowed or for how long. When IDP detects that the listener's question is digressive, it answers the question, and continues to try to expand its TP.

Figure 9 gives an algorithmic description of how IDP processes digressions. The analyzer checks the listener's feedback (**fb**) to see if it is a digressive question (IS-DIGRESSION). If it is, a CTP (**ctp**) is returned that IDP uses to answer the question. The analyzer saves the active path (***active-path***) and invokes IDP's planner-actor to expand and execute the CTP. Next, the active path is restored and the last

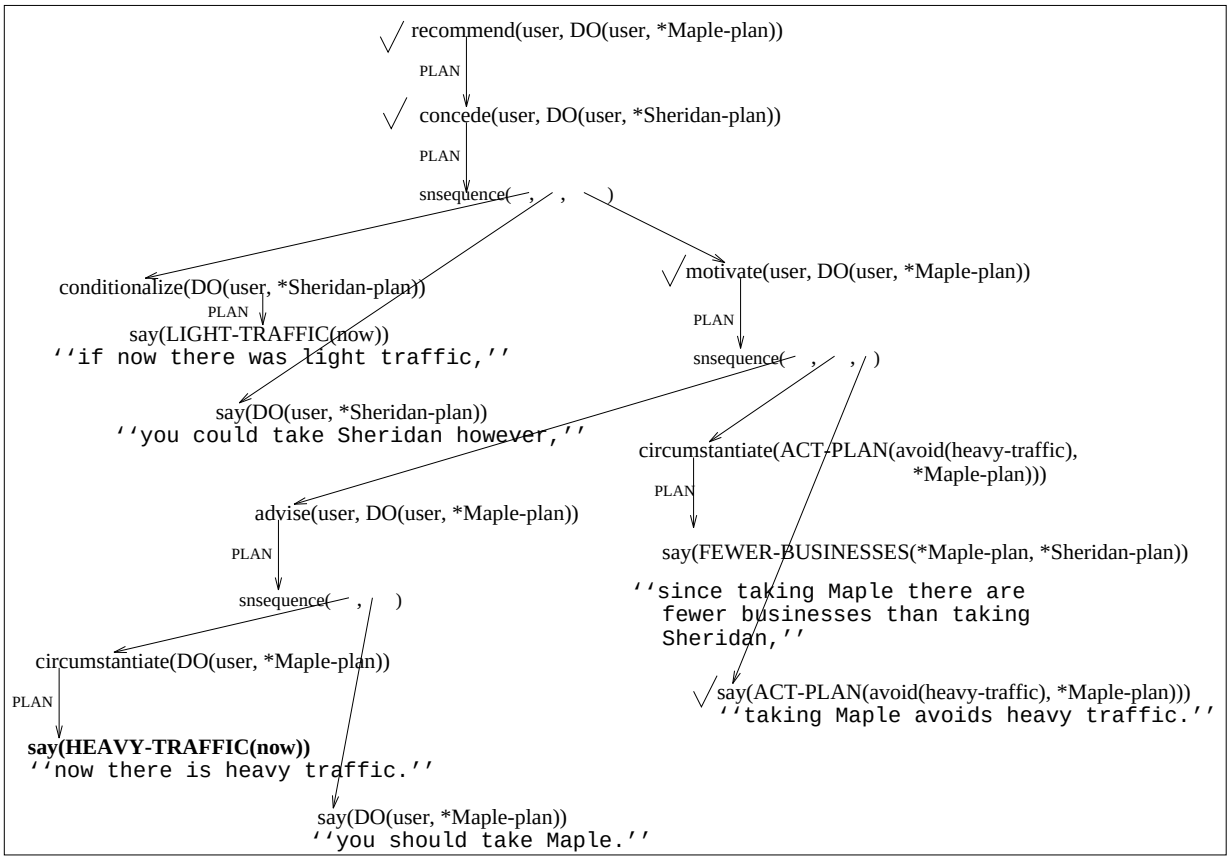

Figure 10: The System's TP Just Before the Digressive Question – Demonstration 9

DTP on the active path retrieved (**last-dtp**). In addition, the analyzer retrieves the last say act in the last DTP (**last-say-act**). This act is the most direct and immediate representation of the system's intention before the digression. The analyzer posts a CG for the listener to know what IDP was saying before the digression, and then it invokes the planner-actor to replan the last DTP.

9.4 An Example of Recovering from a Digression

Sidner notes that discourse markers are used by speakers to tell listeners that the next utterance conveys a new intention [Sidner, 1985]. The new intention could also be a return to an old one. To signal a return from a digression, Grosz and Sidner note that speakers use discourse markers like *anyways...* or *as I was saying....* As illustrated by the last line of Demonstration 9, one payoff of using a uniform representation for text plans and domain plans is that IDP can use its own TP as content to generate a metacomment.

To illustrate digression processing, consider Demonstration 9 and the system's TP for it (Figure 10) just

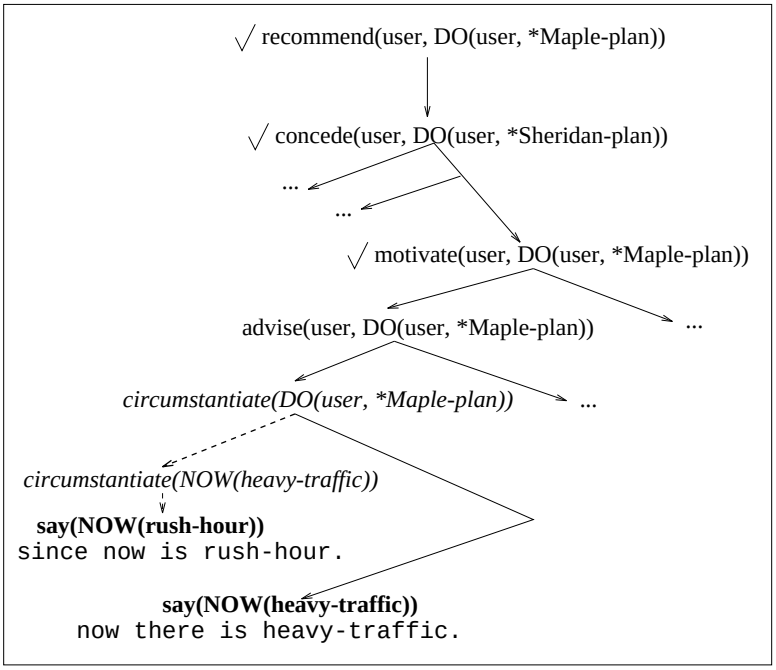

Figure 11: The TP to Answer the Digressive Question – Demonstration 9

before the listener asks the digressive question

`Why is there heavy traffic now?`

Given this feedback, the digression test succeeds and returns a CTP that can be used to answer the question:

circumstantiate(OBJ-PROP(now, heavy-traffic))

The proposition states that now there is heavy traffic. The analyzer saves the active path in Figure 10, and invokes the planner-actor to expand the CTP to answer the listener's question. Figure 11 gives this plan, a plan which results in the system's response

`since now is rush hour,`

After answering the digressive question, the active path is restored to what is shown in Figure 10. Then the analyzer accesses it to retrieve the last DTP on it,

motivate(listener, DO(listener, *Maple-plan))

Each DTP contains a say act that is used to express some minimal content thereby conveying a presentational rhetorical relation. The say act in the motivate DTP is

```
FORALL-IF-THEN(?n,
                    DONE(system, say(?n))
                    RESTATEMENT(?n,
                                        DONE(system, say(?n))))
```

Figure 12: A Rule for Deducing the RESTATEMENT Rhetorical Relation

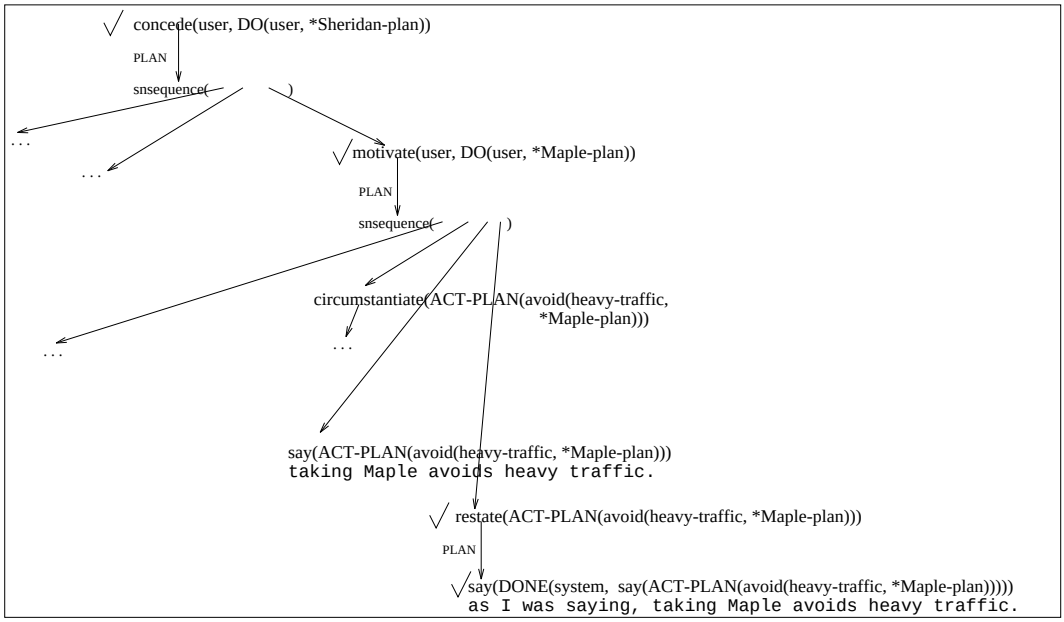

Figure 13: The System's Plan for Returning from the Digression

say(ACT-PLAN(avoid(heavy-traffic), *Maple-plan))

As part of recording the system's TP, the planner-actor has asserted in the knowledge base that the system has performed the above say act.

DONE(system,

say(ACT-PLAN(avoid(heavy-traffic), *Maple-plan)))

The analyzer makes the listener knowing the above proposition a CG. Each DTP contains a particular CTP growth point that allows the DTP to be expanded to restate the nuclear content of the DTP. This is the CTP *restate*. The above proposition satisfies a constraint on the restate CTP-operator that requires there to be an active CG corresponding to the effect of a CTP (Section 5.5).

The other constraint that must be satisfied on the restate CTP-operator is domain-related. IDP must

deduce the RESTATEMENT rhetorical relation between the given nuclear proposition and some new satellite proposition. A satellite restates a nucleus when the nucleus has been said before. Once IDP has said a proposition (and recorded that it has done so), it can deduce the RESTATEMENT relation using the rule given in Figure 12. The rule states that for all nuclear propositions ?n, if the system has said ?n, then the RESTATEMENT relation holds between ?n and the proposition stating that the system has said ?n.

When the planner-actor is invoked to replan the motivate DTP, IDP uses the rule and its recorded knowledge that is has said that taking Maple avoids heavy traffic to deduce the RESTATEMENT relation. Now both constraints on the CTP-operator for restate are satisfiable. Figure 13 shows how IDP returns to expanding its TP (Figure 10) by expanding the restate CTP to expand the motivate DTP.

10 Summary

I have designed and implemented a text planner to justify domain plans interactively. A speaker uses plan justification to have a listener adopt a recommended plan; he uses plan description to enable a listener to execute a plan. The text plan that IDP incrementally formulates and executes to justify its recommendation is represented uniformly in the same knowledge base with the domain plans that are under discussion. In this way, the text plan and the domain plans are both accessible for analyzing the listener's feedback. IDP can interpret vaguely articulated feedback, generate concise replies and metacomments, and detect feedback that initiates digressions.

IDP works in a collaborative, interactive mode in which the system is the primary speaker, the user is the primary listener, and the system is the uncontested expert. The formulation of text plans and the processing procedures for this mode of interaction rely on the explanation assumptions and the feedback assumptions. In the computational model, the discourse context is the system's text plan and the domain plans that the listener could be considering. The system's task is to recognize which one of its own text plans can be used to answer the listener's question.

11 Contributions

11.1 Planning Discourse about Plans

IDP was developed to plan interactive discourse *about* plans. This is an important area of explanation research that has not been explored. Paris notes that expert systems typically contain information about problem-solving techniques in a particular domain, but they do not have any information about why the problem-solving rules they contain achieve their goals. Therefore, these systems cannot justify their behavior [Paris, 1991]. Solutions to this problem lie in the development of models that integrate inference, planning, and acting [Kumar and Shapiro, 1993] and models that use plan execution to invoke specialized hybrid reasoners [Campbell, 1994]. Because plans link specific problem-solving techniques to domain goals, they are an important domain-independent representation that explanation facilities must be able to talk about.

11.2 Collaborative Models of Discourse

A viable model of interactive discourse must explain how interlocutors share conversational control to construct the discourse, and recognize the contributions of others to it. This work has focussed on one mode of collaboration. In this collaborative mode, the system is responsible for planning text and the listener is responsible for providing feedback that allows the system to recognize a way to extend the discussion in a way that is helpful. One insight that this work contributes is that attempts at top-down designs of "the collaborative discourse model" [Sidner, 1993] may be misguided. A better, bottom-up approach may be to identify the various modes of collaboration that communicators use, and build models for each one. Using this approach, we can make assumptions that lead to simpler, computationally tractable systems that provide for efficient and effective interaction.

11.3 Using a Uniform Representation

Researchers have argued that a viable architecture for generation must be able to incorporate constraints from several sources and allow for flexible control so that decision-making can occur when most appropriate. IDP represents the first step in designing and implementing a plan-based generation model with a uniform

representation that allows for the possibility of planning and constraint propagation at all levels of the generation process. For interactive text planning, the uniform representation of all information gives IDP concurrent access to its text plan, the domain plans, and the system's reasoning. The uniform representation also allows the system to generate metacomments.

11.4 Integrating Planning with Plan Recognition

The IDP model also contributes to our understanding of the relationship between text planning and plan recognition. Traditional plan recognition systems use utterances to infer a user's activity and then to infer the domain plan that he intends to execute. The system has a passive role, and the appropriate context for analyzing the user's utterances are the metaplans she uses and the domain plans she may be considering.

The uniform representation of plans in IDP allows for integration of planning with plan recognition. The theory behind IPD holds that for interactive generation, there is a plan recognition process that uses the listener's utterance to recognize an aspect of the domain plan that she does not know. To make it known, the system must recognize a text plan that the user intends for the system to use. In this interactive mode, the system has an active role, and the appropriate context includes the system's own TP and the domain plan that is under discussion.

12 Future Work

12.1 Plan Description

The implemented portion of IDP plans text to justify plan advice by informing the user of the reasoning behind a plan. The system's discourse goal is to have the user adopt the recommended plan. Text plan operators must be added so that IDP can generate texts interactively that describe a plan until a listener indicates that she can use it. Mellish and Evans [Mellish and Evans, 1989] and McKeown [McKeown, 1990] have worked on plan descriptions. More recently, Vander Linden has considered generating instruction texts using RST relations [Vander Linden, 1993]. However, all of this work has focussed on noninteractive plan descriptions. An area of future research is to see if the IDP model is robust enough to generate

interactive domain plan descriptions by simply adding the text plan operators that Vander Linden has found in instruction texts.

12.2 System-imposed Topic Shifts

Currently, IDP detects topic shifts from the user's questions. Some of IDP's text plans have preconditions that the system should be able to satisfy by initiating a topic shift. For example, if the user indicates that she does not know a plan well enough to use it, then IDP must shift the topic of discussion from recommending a plan to describing it. The current IDP analyzer model must be extended to allow for this kind of system-imposed topic shift.

12.3 Discourse Plan Explanation

Litman and Allen [Litman and Allen, 1987] have shown that clarification dialogues often require a speaker to refer to what she is doing in her communicative plan. Since IDP's text plan is represented uniformly with the domain plan that is under discussion, another objective is to extend the set of text plan operators so that the system can engage in full-blown, meta-level discussions of its own discourse plan. It remains to be seen how a clarification plan relates to the text plan that it is about.

12.4 Changing Collaborative Modes

In this work, I have focussed on one mode of collaboration in which the system is the primary speaker and the user is the primary listener. A collaborative mode is defined by what the various participants have responsibility for in that mode. For a given mode, if we define the computational system's responsibilities, and delegate other responsibilities to the user, we can make appropriate assumptions and identify the knowledge requirements. The point is that when the system's responsibility shifts, so does the knowledge that is used as the context for analyzing user contributions.

Defining collaborative modes in terms of system responsibility determines which assumptions are appropriate, and what knowledge is relevant. This leaves open the question of how responsibility shifts, that is, *how* and *why* the system moves from one mode to another, and how this should be represented. With respect

to *how*, one possibility is that the knowledge requirements for each mode are really highlighted aspects of an overarching model. As the system moves from one collaborative mode to another, the overarching model would need to be updated. IDP represents a bottom-up attempt at building such a model. The most challenging possibility for future research would be to try to map the IDP discourse model onto representations for an overarching model of the discourse collaboration.

References

[Campbell, 1994] A. Campbell. A semantic network interface for hybrid reasoning. In T. Wilmarth, editor, *Proceedings of the Ninth Annual University at Buffalo Graduate Conference on Computer Science*, 1994.

[Carberry, 1988] S. Carberry. Modeling the user's plans and goals. *Computational Linguistics*, 14(3), 1988.

[Carberry, 1989] S. Carberry. A pragmatics-based approach to ellipsis resolution. *Computational Linguistics*, 15(4), 1989.

[Carberry, 1990] S. Carberry. *Plan Recognition in Natural Language Dialogue*. MIT Press, 1990.

[Grice, 1957] H. P. Grice. Meaning. *Philisophical Review*, 66, 1957.

[Grice, 1975] H. P. Grice. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and Semantics 3: Speech Acts*. Academic Press, New York, 1975.

[Grosz and Sidner, 1986] B. J. Grosz and C. L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linquistics*, 12, 1986.

[Hovy, 1990] E. H. Hovy. Unresolved issues in paragraph planning. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*. Academic Press, 1990.

[Kukich, 1985] K. Kukich. Explanation structures in xsel. In *Proceedings of the Annual Meeting of the Asscociation for Computational Linguistics*, 1985.

[Kumar and Shapiro, 1993] Deepak Kumar and Stuart C. Shapiro. Deductive efficiency, belief revision and acting. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 5(2&3):167–177, April–September 1993.

[Kumar *et al.*, 1988] D. Kumar, S. Ali, and S. C. Shapiro. Discussing, using and recognizing plans in SNePS preliminary report - SNACTor: An acting system. In *Proceedings of the Seventh Biennial Convention of South East Asia Regional Confederation*. Tata McGraw-Hill, 1988.

[Lambert, 1991] L. Lambert. Tripartite plan-based model of dialogue. In *Proceedings of the Association for Computational Linguistics*, 1991.

[Litman and Allen, 1987] D. Litman and J. Allen. A plan recognition model for subdialogues in conversations. *Cognitive Science*, 11:163–200, 1987.

[Litman and Allen, 1990] D. Litman and J. Allen. Discourse processing and commonsense plans. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.

[Mann and Thompson, 1987] W. C. Mann and S. A. Thompson. Rhetorical structure theory: A theory of text organization. Technical report, Information Sciences Institute, 1987.

[Martins and Shapiro, 1988] J. P. Martins and S. C. Shapiro. A model for belief revision. *Artificial Intelligence*, 35(1), 1988.

[McKeown, 1990] K. McKeown. Natural language generation in comet. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation*. Academic Press, 1990.

[Mellish and Evans, 1989] C. Mellish and R. Evans. Natural language generation from plans. *Computational Linguistics*, 15(4), 1989.

[Moore and Pollack, 1992] J. D. Moore and M. E. Pollack. A problem for RST: The need for multi-level discourse analysis. *Computational Linguistics*, 18(4), 1992. discussion.

[Paris, 1991] C. Paris. Generation and explanation. In C. Paris, W. Swartout, and W. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Kluwer Academic Publishers, 1991.

[Ramshaw, 1989] L. A. Ramshaw. A metaplan model for problem-solving discourse. In *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*, 1989.

[Shapiro and Group, 1992] S. C. Shapiro and The SNePS Implementation Group. *SNePS-2.1 User's Manual*. Department of Computer Science, SUNY at Buffalo, 1992.

[Shapiro and Rapaport, 1987] S. C. Shapiro and W. J. Rapaport. SNePS considered as a fully intensional propositional semantic network. In N. Cercone and G. McCalla, editors, *The Knowledge Frontier*. Springer–Verlag, New York, 1987.

[Shapiro and Rapaport, 1992] S. C. Shapiro and W. J. Rapaport. The SNePS family. *Computers & Mathematics with Applications*, 23(2–5), 1992.

[Shapiro *et al.*, 1989] S. C. Shapiro, D. Kumar, and S. Ali. A propositional network approach to plans and plan recognition. In *Proceedings of the 1988 Workshop on Plan Recognition*, Los Altos, CA, 1989. Morgan Kaufmann.

[Shapiro, 1979] S. C. Shapiro. The SNePS semantic network processing system. In N. V. Findler, editor, *Associative Networks: The Representation and Use of Knowledge by Computers*. Academic Press, New York, 1979.

[Shapiro, 1991] S. C. Shapiro. Case studies of SNePS. *SIGART Bulletin*, 2(3), 1991.

[Sidner, 1985] C. Sidner. Plan parsing for intended response recognition in discourse. *Computational Intelligence*, 1, 1985.

[Sidner, 1993] C. Sidner. The role of negotiation in collaborative activity. Technical Report FS93-05, AAAI, October 1993.

[Thomason, 1990] R. H. Thomason. Accomodation, meaning, and implicature: Interdisciplinary foundations for pragmatics. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication*. MIT Press, 1990.

[van Kuppevelt, 1992] J. van Kuppevelt. About a uniform conception of S- and D- topics. In *Proceedings of the Prague Conference on Functional Approaches to Language Description*, Prague, 1992.

[Vander Linden, 1993] K. Vander Linden. Rhetorical relations in instructional text generation. In *Workshop on Intentionality and Structure in Discourse Relations – Proceedings of a Workshop Sponsored by the Special Interest Group on Generation of the Association for Computational Linguistics*, Columbus, Ohio, 1993.