Recognizing Digressive Questions During Interactive Generation *

Susan M. Haller
Department of Computer Science and Engineering
University of Wisconsin – Parkside
Kenosha, Wisconsin 53141
haller@cs.buffalo.edu

Abstract

In expository discourse, people sometimes ask questions that digress from the purpose of the discussion. A system that provides interactive explanations and advice must be able to distinguish pertinent questions from questions that digress. 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 original 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. As a testbed, IDP plans text to discuss driving routes. 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.

1 Characterizing Digressions

Grosz and Sidner define a digression as a type of interruption [4]. An interruption is a discourse segment with a purpose that does not contribute to the achievement of the current discourse purpose. They describe three kinds. A true interruption has a discourse purpose that is unrelated to the interrupted discourse segment. For example, if a speaker says

"John came by and dropped off the groceries *Stop that you kids.* and I put them away after he left."

the italized portion of text is a true interruption.

Speakers use a second type of interruption, the *flashback* or *filling in missing places* to bring objects and propositions into the discussion that aid in comprehension of the current discourse segment. This type of interruption provides background knowledge. However, it does not contribute directly to the current discourse purpose. For example in

"OK. Now how do I say that Bill is *Whoops I forgot about ABC. I need an individual concept for the company ABC...* Now back to Bill. How do I say that Bill is an employee of ABC?"

the speaker sets aside her current purpose to discuss a prerequisite that should have been introduced earlier.

We are concerned with a third type of interruption that Grosz and Sidner describe as a *digression.* A digression contains a reference to some entity that is salient in both the interruption and the interrupted segment. The digression's purpose is not unrelated to the purpose of the interrupted segment (as in a true interruption), but neither is it a prerequisite (as in a flashback). As an example of a digression, Grosz and Sidner note that if while discussing Bill's role in company ABC, a conversational participant interrupts with

Speaking of Bill, that reminds me, he came to dinner last week,

Bill remains salient, but the discourse purpose changes.

In the IDP model, the system's purpose (or intent) is expressed by its discourse goal and the text plan that the system formulates and executes to try to achieve it. IDP detects a digression when the user asks a question about a discourse entity or a proposition that is part of the system's text plan, and the answer to the question cannot be incorporated into the 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.

2 Digression Types

The first and simplest kind of digression that IDP can detect is when the listener asks a direct question about a proposition that was expressed by the system earlier in the discourse. The following interaction demonstrates this kind of digression and how the system responds to it.[1]

*I would like to thank my advisor, 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.

[1]All demonstrations given in courier font are actual interactions with IDP.

Demonstration 1:

```
  L:  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.
  L:  Why is there heavy traffic now? *
IDP:  since now is rush hour,
      as I was saying taking Maple avoids
      heavy traffic.
```

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 the discussion purpose. Therefore, after answering the question, IDP shifts the discussion back to its recommendation.

A second kind of digression that IDP can detect is when the listener questions a proposition that was never expressed, but that can be inferred from a proposition that has been presented. The following demonstration illustrates this digression type.

Demonstration 2:

```
  L:  Should I take Maple or Sheridan to go to
      the Eastern Hills Mall?
IDP:  take Maple.
  L:  Why take Maple?
IDP:  taking Maple avoids heavy traffic.
  L:  Why?
IDP:  since taking Maple there are fewer
      businesses than taking Sheridan.
  L:  Why is there heavy traffic now? *
IDP:  since now is rush hour,
      as I was saying taking Maple avoids
      heavy traffic.
```

IDP uses its text plan to detect that the listener's third question, unlike the listener's first two questions, initiates a digression. This is notwithstanding the fact that the system never expressed the proposition that the listener questions. The listener's digressive question relies on an inference that she made from the proposition that IDP conveyed in its second response.

IDP does not analyze every question as a digression.

Demonstration 2a:

```
  L:  Should I take Maple or Sheridan to go to
      the Eastern Hills Mall?
IDP:  take Maple.
  L:  Why is there heavy-traffic now? *
IDP   Huh?
```

In this example, IDP fails to make any coherent connection between the question and the text plan that they system has used so far. Note that the answer to this question does not fall outside the system's expertise. In Demonstration 2, IDP analyzed this question as a digression, and responded to it.

A third type of digression that the system detects is when the listener's line of questioning keeps referring to the last proposition that was expressed. This eventually leads to an explanation that does not address IDP's original purpose. The following interaction illustrates digressions of this kind.

Demonstration 3:

```
  L:  Should I take Maple or Sheridan to go to
      the Eastern Hills Mall?
IDP:  take Maple.
  L:  Why?
IDP:  taking Maple avoids heavy traffic.
  L:  Why?
IDP:  taking Maple there are fewer
      businesses than taking Sheridan.
  L:  Why? *
IDP:  since taking Maple is a newer route
      than taking Sheridan.
      anyways, taking Maple avoids heavy
      traffic.
```

Unlike the first two questions, the listener's third why-question requires an answer that cannot be incorporated into the IDP's text plan, a plan for trying to convince the listener to use the Maple Road route. Therefore, the system detects a digression.

3 Related Work

Moore and Swartout note that prior to their work on the EES Text Planner, little work had been done on using a system's goals and plans as a model of the discourse and its purpose [11, 12]. The EES Text Planner records its discourse goals and the plans that it formulates to achieve them in a *dialogue history*. When the user asks a follow-up question, the system uses the goal structure of its text plan, assumptions that have been made during generation, and alternative plans that it did not use to disambiguate the question and select a perspective for the answer to it. Then the system formulates the appropriate discourse goal and passes it to the text planner to formulate a response.

The dialogue history is a stack of text plans from previous exchanges. Each text plan in the stack has a pointer to a goal (called the *global context*) that the text plan has been constructed to achieve. However, the stack ordering is the only relationship among the text plans and their respective goals. There is no representation of the overarching goal for the interaction nor how the plan for each exchange contributes to it. Therefore, the system cannot detect when the user's follow-up question results in planning for a discourse goal that does not contribute to the original discussion purpose.

The IDP model treats the system's text plan as a richer source of information. To produce answers to follow-up questions, IDP replans or expands a single text plan that has an overarching discourse goal. In this respect, IDP is similar to the Explanatory Discourse Generator (EDGE) [2]. EDGE plans tutorials by formulating a single text plan in advance and then executing it incrementally. EDGE uses feedback from the listener to monitor the success of its plan thereby making decisions to prune or extend parts of it that have not been used yet.

Planning text ahead is appropriate when a system plans tutorials. In this situation the system has a structure of preset goals for the interaction. In contrast, IDP is designed for domains where there is no agenda for what the listener will come to know as a result of the interaction. IDP plans text to give advice and answer questions until the listener is satisfied. Therefore, like the EES Text Planner, IDP plans only in reaction to follow-up questions. However, to keep track of where the interaction is going, each plan that IDP constructs extends a single text plan that is formulated to achieve an overarching discourse goal.

4 Text Plan Operators

4.1 The Planning Formalism

IDP's text plan operators (TP-operators) are based on Rhetorical Structure Theory (RST) [9] and are written using the SNePS Actor planning formalism [16]. In RST, each essential text message (called the *nucleus*) can be augmented with additional information (called the *satellite*) through a *rhetorical relation.* In the SNePS Actor planning formalism, plan operators are written as rules that state what consequent propositions can be deduced from a set of antecedents.

Figure 1 shows a TP-operator for the *motivate* act. [2] In the formalism, an *act* decomposes into one or more structures of other acts called *plans.* IDP deduces propositions that state how acts decompose into plans (ACT-PLAN case frame) 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 same constraints are used to write rules for deducing the preconditions of the motivate act (ACT-PRECONDITION case frame), the effects of the act (ACT-EFFECT case frame), and the goals that the act (viewed as a high-level plan) can be used to achieve (GOAL-PLAN case frame).

The TP-operator in Figure 1 is an asserted rule (indicated by !) stating that for all (FORALL) ?g1, ?g2, and ?p, if the antecedents (ANT) can be satisfied, then the consequent proposition (CQ) can be deduced. The TP-operators are domain-independent because the constraints on them are stated in terms of the planning formalism. Hence, IDP's TP-operators are for producing text about a set of domain plans that are under discussion.

In Figure 1, the constraints are that ?g1 must be a goal-act, act ?g1 must be enacted by a plan ?p, ?g2 must be a secondary goal-act, and act ?g2 must also be enacted by plan ?p. If an instantiation of ?g1, ?g2, and ?p satisfies the constraints, then the system can deduce a plan for the act of motivating the user to do ?p. This plan is a sequence (snsequence) of four other acts. The snsequence act is one of six *control acts* that are used to structure several acts into a plan for an act.

4.2 The Kinds of Text Plans

IDP uses two types of *text plans* (TPs) separating those plans that address the system's discourse goal (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 an high-level intention in mind [14].

```
!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 1: A TP-operator for Motivate

This division is based on a two-way division of the rhetorical relations that Mann and Thompson describe and that has been used by Haller [5, 6] and Moore and Paris [13] to design systems with two types of text plans. Each *presentational relation* relates two text spans for the purpose of increasing an inclination in the hearer. In contrast, a *subject-matter* relation is used with the intent of informing the hearer of the rhetorical relation itself. 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 DGs by selecting some minimal text content. IDP can augment this content without detracting from its own intent by using one or more CTPs. Therefore, CTPs correspond to subject-matter rhetorical relations.

Figure 2(a) shows a DTP for *motivate* that IDP instantiates from the TP-operator given in Figure 1.[3] The motivate act takes the listener and a nuclear clause (the listener DOing the *Maple-plan) as its arguments. The DTP for motivating the listener to take the Maple Road route sequences up to four other acts. The first act in the sequence, *advise*, is another DTP that must be expanded if the system has not already used it. The third act, *say,* is the only primitive act. When the system executes it, a text message is sent to IDP's surface generation grammar. The second and fourth acts, *circumstantiate* and *restate*, are references to CTPs that are potential plan *growth points.* Growth points are references to CTPs 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 [7].

A plan for circumstantiate is given in Figure 2(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 *content-goal* (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 2(b) as the second step in executing

[2]Arguments enclosed in braces,{...}, are unordered set arguments.

[3]*Maple-plan represents a plan structure stated in terms of going and turning acts that is not shown here.

```
!ACT-PLAN(motivate(listener,
                   DO(listener,*Maple-plan)),
          snsequence(
            advise(listener,
                   DO(listener,*Maple-plan)),
            circumstantiate(ACT-PLAN(avoid(heavy-traffic,
                                           *Maple-plan))),
            say(ACT-PLAN(avoid(heavy-traffic,
                               *Maple-plan))),
            restate(ACT-PLAN(avoid(heavy-traffic,
                                   *Maple-plan)))))
(a)
```

```
!ACT-PLAN(circumstantiate(ACT-PLAN(avoid(heavy-traffic,
                                         *Maple-plan))),
          snsequence(
            say(OBJ-PROP(*Maple-plan,
                         fewer-businesses)),
            disbelieve(CONTENT-GOAL(
                         KNOW(listener,
                         OBJ-PROP(*Maple-plan,
                                  fewer-businesses)),
(b)
```

Figure 2: (a) A DTP for Motivate (b) A CTP for Circumstantiate

the circumstantiate CTP, the CG is retracted (*disbelieved* by the system). Because the ACT-PLAN proposition is deducible only when the CG exists, the SNePS Belief Revision component (SNeBR) [10] 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 even though it appears in the body of DTPs like the one for the motivate act (Figure 2(a)).

5 The Analyzer

5.1 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 [15]. IDP's DGs are unachieved system goals that have to do with the attitude or abilities of the listener 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 listener adopt a domain plan
2. to have the listener be able to execute a domain plan

IDP plans text to try to achieve its DG, and it interprets the listener'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 listener knows the system's intention and how it has been realized, she has expectations for what will follow [1]. Motivated by Grice's Maxim of Relation [3], IDP analyzes questions using growth points on the *active path*. The active path marks the TP that make up the most recent expansion of IDP's overall TP. 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 model of the listener, the listener does not know. The localized unknowns are linked to the system's TP by the reasoning chains that the system used to derive it TP.

5.2 The Local Topic

In the IDP model, questions address the system's DG indirectly by making reference to what the system is doing to achieve its DG. Therefore, IDP's Analyzer uses feedback to try to recognize a TP to use to expand or replan its current TP. Following van Kuppevelt, the *local topic* is that which is questioned, and the *comment* on the local topic is an answer to the question [18]. For each question, IDP's Analyzer determines the local topic and then searches for a way to expand its TP to include a comment on it.

The Analyzer 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 local topic is a domain plan or a 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.

5.3 A Measure of Coherence

From the generation perspective, a conversation is a communicative process that the system controls for its own purpose. In this context, the appropriate measure of coherence is the degree to which a new TP-expansion contributes to the achievement of the system's goal. Discourse expectation constrains IDP's choices to the growth points in the DTPs along the active path. An important heuristic in selecting a TP-expansion is the degree to which the proposed 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.

In the first phase, IDP analyzes why-questions in relation to its own intent as represented by the DTPs along the active path. The Analyzer starts with the most recently executed DTP (the last DTP on the active path), and the localized unknowns associated with it. It tries to find another way to expand a DTP along the active path that lets the listener know a localized unknown. The Analyzer backs up the active path testing each DTP in turn. If this fails, in the second phase, the Analyzer considers augmenting the existing DTP at the informational level. This level is reflected in the CTPs. The Analyzer examines the most focussed DTP on the active path to see if it can be expanded with a CTP to let the user know a localized unknown. A detailed account of this type of processing is presented in [5].

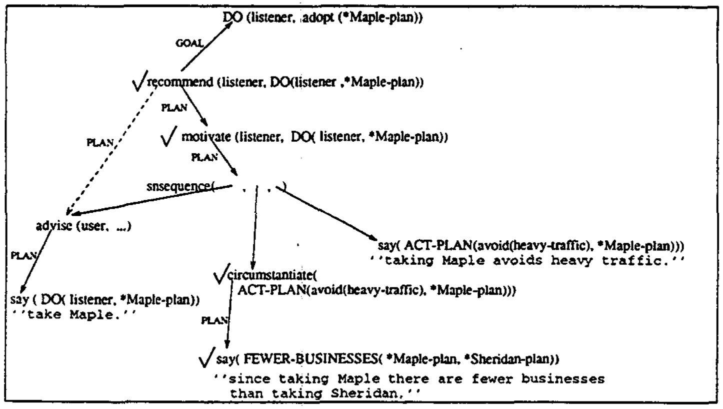
DO (listener, adopt (*Maple-plan))
GOAL
√recommend (listener, DO(listener ,*Maple-plan))
PLAN
PLAN
√ motivate (listener, DO(listener, *Maple-plan))
PLAN
snsequence(, ,)
advise (user, ...)
PLAN
say (DO(listener, *Maple-plan))
''take Maple.''
say(ACT-PLAN(avoid(heavy-traffic), *Maple-plan)))
''taking Maple avoids heavy traffic.''
√circumstantiate(
ACT-PLAN(avoid(heavy-traffic), *Maple-plan)))
PLAN
√say(FEWER-BUSINESSES(*Maple-plan, *Sheridan-plan))
''since taking Maple there are fewer businesses
than taking Sheridan.''

Figure 3: The TP for Before the Digressive Question in Demonstrations 2 and 3

5.4 An Example TP

Figure 3 shows IDP's TP after its third response in Demonstrations 2 and 3. 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 (√) 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 2(a)). In particular, to make the initial recommendation the planner selected a simple plan (*advise*, indicated by a dashed line) over a more complex plan (*motivate*). In response to the listener's first why-question, IDP replanned the recommendation with the more complex motivate act. Since its recorded TP indicated that it had just used the advise act, the only act in the plan for the motivate act that must be executed is the *say* act. This act conveys the satellite proposition that counts as the motivation for the advice.

The second act in the plan for motivate, *circumstantiate* expands to an optional CTP. IDP does not expand and use this plan until it responds to `Why?` a second time. This happens because IDP replans and expands its TP in reaction to the listener's questions.

6 Detecting Digressive Questions

If the Analyzer fails to recognize a way to replan or expand its own TP from the listener's feedback (Section 5.3), it tests the feedback to see if is a digressive question. Figure 4 gives the algorithm that the Analyzer uses for the IS-DIGRESSION test. The Analyzer tests the feedback (**fb**) in two stages. First, the Analyzer tests to see if the feedback has the correct form and content for one of three kinds of digressive questions that the system can detect. If one of the three tests succeeds, it will return a proposition that is the topic (**topic**) for the digression. The topic is the proposition that the listener questions in her digressive question.

In the second stage, if there is a topic, the Analyzer collects CTPs (**ctps**) that use the topic as a nuclear proposi-

```
IS-DIGRESSION(fb)

   Let topic := IS-DIGRESSION-TYPE-1(fb)
                or
                IS-DIGRESSION-TYPE-2(fb)
                or
                IS-DIGRESSION-TYPE-3(fb)
   If topic
       Then Let ctps =: get-ctps-with-nucleus(topic)
            Let related-reasoning =: get-reasoning(2,topic)
            Loop-for-each ctp in ctps
               If listener-knows(effect-of(ctp))
                 or
                 not(member(effect-of(ctp), related-reasoning))
                Then Let ctps := remove-from(ctps, ctp)
            End-loop
   If more-than-one(ctps)
       Then Let ctp-to-expand =: get-simplest-plan(ctps)
       Else Let ctp-to-expand =: ctps
   If ctp-expansion
     Then  post-content-goal(KNOW(listener,
                                   effect-of(ctp-to-expand)))
           return(ctp-to-expand)
     Else return('nil)
```

Figure 4: Detecting Digressive Questions from the Listener's Feedback

```
IS-DIGRESSION-TYPE-1(fb)

   If  fb = "Why" {"not"} proposition
       and
       DONE(system, say(proposition))
   Then return(proposition)
   Else return('nil)
```

Figure 5: Test 1: For Detecting Digressive Questions Based on What the System Said Previously

tion. It also collects domain reasoning that the system has performed that is related to the topic (**related-reasoning**). The Analyzer tests each CTP and removes any CTPs that have the effect of letting the listener know a proposition that she knows already. It also removes a CTP if its effect is not one of the related reasoning propositions. If there is still more than one candidate CTP, the Analyzer picks the simplest (**ctp-to-expand**). If a CTP has been found that meets all of the requirements, the Analyzer posts its effect as a CG and returns it for the Planner-Actor to expand to answer the question. If there is no such CTP, then the digression test fails.

6.1 Detecting Digressive Questions About What has been Said

The simplest kind of digression that IDP can detect is when the listener asks a direct question about a proposition that was expressed as part of the system's TP. The Analyzer uses the first digression test, IS-DIGRESSION-TYPE-1, that is described in Figure 5. If the feedback (**fb**) is a question in the form "Why" {"not"} **proposition** and the system has said the proposition as part of its TP, then the proposition is returned as the topic for the digression. Otherwise, the first test fails.

The first test is used to detect the digression in Demonstration 1. Figure 6 gives IDP's TP just before the listener asks the digressive question in that example. When the listener asks `Why is there heavy traffic now?`, the Analyzer tries to find a way to expand the DTP-portion of its

TP on the active path (Section 5.3). This involves searching the growth points in the DTPs for motivate, concede, and recommend. When this fails, the Analyzer tries to process the feedback as a digressive question.

Testing for a digression (Figure 4) with the first digression rule (Figure 5), the Analyzer checks the TP in Figure 6 to see if the proposition in the listener's question was said by the system. The Analyzer finds that it has been said (Figure 6 in **boldface**). Therefore, the first digression test succeeds, and the following proposition is returned as the topic of the digression:

OBJ-PROP(now, heavy-traffic)

The Analyzer finds another CTP that uses the topic as a nucleus:

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

and collects reasoning that is related to the topic

ACT-PLAN(avoid(heavy-traffic), *Maple-plan)
OBJ-PROP(now, rush-hour))
SECONDARY-GOAL-ACT(avoid(heavy-traffic))

The CTP's effect,

KNOW(listener,
    OBJ-PROP(now, rush hour))

lets the listener know a new satellite proposition: now is rush hour. Since the listener model does not assert that the listener knows this proposition, and since this proposition is related by domain reasoning to the topic, the CTP remains as a candidate for expansion. So that IDP's Planner-Actor can instantiate a plan for this CTP, its effect is made a CG. Then the CTP is returned for the Planner-Actor to expand into the TP. This leads to the restatement found in the last line of IDP's response in Demonstration 1.

6.2 Detecting Digressive Questions About Inferred Propositions

In Demonstration 2, IDP never expressed the proposition that the listener questions. Therefore, the listener must have inferred it. In this case, the system cannot determine that the question is digressive simply because the answer is something that the listener does not know. Such a rule would not distinguish digressive questions from incoherent ones.

To test for digressive questions about propositions that the listener has inferred, the Analyzer uses the second test, IS-DIGRESSION-TYPE-2, that is described in Figure 7. The Analyzer checks to see if the feedback is a why-question about a proposition (**proposition**) that has not been expressed by the system, but that can be inferred (IS-INFERABLE) from what has been said. If the proposition satisfies these requirements, then, as in the previous case, the proposition is returned as the topic for the digression. Otherwise, the second test fails.

A question is coherent only if the listener has inferred it correctly from something that the system has said. Figure 8 describes the test, IS-INFERABLE, that the Analyzer uses to determine if the questioned proposition is inferable. First, the Analyzer gathers all the propositions that IDP has said as the content of a say act (**expressed-props**). For each expressed proposition (**prop**), the Analyzer collects propositions that are within two reasoning steps of each expressed proposition. These are added to the propositions that are inferable (**inferable-props**). Some of the propositions that can be inferred may have been said already. Therefore, after collecting the inferable propositions, the Analyzer removes any proposition that is also one of the expressed propositions. Finally, the Analyzer checks to see if the proposition that the listener has questioned is among those that are unsaid, but inferable, from what IDP has said previously. If it is, the test succeeds.

```
IS-DIGRESSION-TYPE-2(fb)

  If   fb = "Why" {"not"} proposition
       and
       not(DONE(system, say(proposition)))
       and
       IS-INFERABLE(proposition)
  Then return(proposition)
  Else return('nil)
```

Figure 7: Test 2: For Detecting Digressive Questions that the Listener has Inferred

```
IS-INFERABLE(proposition)

  Let expressed-props := get-said-props
  Loop-for-each prop in expressed-props
    Let inferable-props:=
        union(get-reasoning(2, expressed-props), inferable-props)
  Endloop
  Let inferable-props :=
      remove(expressed-props, inferable-props)
  If member(proposition, inferable-props)
    Then return('true)
    Else return('nil)
```

Figure 8: Determining if a Proposition is Inferable

In Demonstration 2, the second digression test is used to help determine that the listener's question is digressive. Figure 3 gives the system's TP just before the digressive question `Why is there heavy traffic now?`. In this interaction, the questioned proposition has not been expressed by the system. Therefore, the first digression test (Figure 5) fails immediately. In the second digression test (Figure 7), the form of question is satisfactory and the system has not expressed the questioned proposition. Therefore, IS-INFERABLE is invoked to see if the proposition

OBJ-PROP(now, heavy-traffic)

can be inferred from what has been said.

At this point in the discourse, the propositions that IDP has expressed with say acts are

DO(listener, *Maple-plan)
ACT-PLAN(avoid(heavy traffic), *Maple-plan)
FEWER-BUSINESSES(*Maple-plan, *Sheridan-plan)

The Analyzer collects unsaid propositions that are related by reasoning to them.

OBJECT-PROP(now, heavy-traffic)
OBJECT-PROP(now, rush-hour)
SECONDARY-GOAL-ACT(avoid(heavy-traffic))
NEWER-ROUTE(*Maple-plan, *Sheridan-plan)

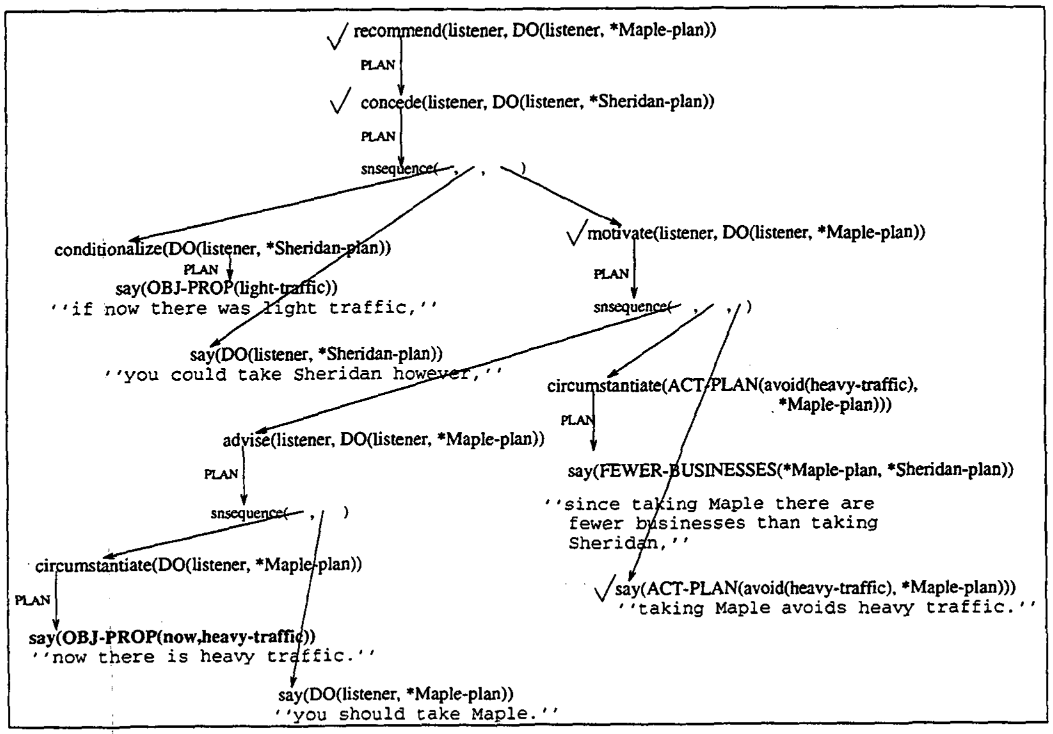
√ recommend(listener, DO(listener, *Maple-plan))
PLAN
√ concede(listener, DO(listener, *Sheridan-plan))
PLAN
snsequence(, ,)
conditionalize(DO(listener, *Sheridan-plan))
PLAN
say(OBJ-PROP(light-traffic))
''if now there was light traffic,''
say(DO(listener, *Sheridan-plan))
''you could take Sheridan however,''
advise(listener, DO(listener, *Maple-plan))
PLAN
snsequence(,)
circumstantiate(DO(listener, *Maple-plan))
PLAN
say(OBJ-PROP(now,heavy-traffic))
''now there is heavy traffic.''
say(DO(listener, *Maple-plan))
''you should take Maple.''
√ motivate(listener, DO(listener, *Maple-plan))
PLAN
snsequence(, ,)
circumstantiate(ACT-PLAN(avoid(heavy-traffic), *Maple-plan)))
PLAN
say(FEWER-BUSINESSES(*Maple-plan, *Sheridan-plan))
''since taking Maple there are fewer businesses than taking Sheridan,''
√ say(ACT-PLAN(avoid(heavy-traffic), *Maple-plan)))
''taking Maple avoids heavy traffic.''

Figure 6: The System's TP Just Before the Digressive Question – Demonstration 1

```
IS-DIGRESSION-TYPE-3(fb)
  If     fb = "Why", {"not"}
         Then Let proposition := get-last-proposition-said
```

Figure 9: Test 3: For Detecting Garden-path Digressions

These are the inferable propositions. Since the questioned proposition is among them, the test for the second type of digressive question succeeds.

Next, the Analyzer tries to find a CTP that uses the proposition as a nucleus (Figure 4). As before, the CTP that it finds is

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

Since the listener does not know its effect, the Analyzer selects the CTP to answer the digressive question by making its effect a CG, and returning the CTP to be expanded and executed by the Planner-Actor.

6.3 Detecting Garden-path Digressions

Demonstration 3 shows that when the listener repeatedly asks `Why?`, IDP eventually recognizes a *garden-path digression*. A garden-path digression will always occur at some point in an interaction, if the listener asks a series of simple why-questions. The test is a simple one that is given in Figure 9. If the feedback is `Why?` or `Why not?`, then the test succeeds and the last proposition that was expressed is returned as the topic of the digression.

In Demonstration 3, when the listener asks `Why?` a third time, the state of IDP's TP is the same as in Demonstration 2 (Figure ??). The first two digression tests fail because the form of the question is incorrect. Using the third digression test (Figure 9), the last proposition that the system expressed, and therefore, the topic of the digression becomes

FEWER-BUSINESSES(*Maple-plan, *Sheridan-plan)

IDP finds a CTP that uses this topic as a nucleus

circumstantiate(FEWER-BUSINESSES(*Maple-plan, *Sheridan-plan))

Reasoning that is related to the topic is

ACT-PLAN(avoid(heavy-traffic), *Maple-plan)
NEWER-ROUTE(*Maple-plan, *Sheridan-plan)

The effect of the CTP is to let the listener know the second proposition above. Since the listener does not know this proposition, and since it is reasoning that is related to the topic, the CTP remains as a candidate for answering the digressive question (Figure 4). After asserting that the effect of the CTP

KNOW(listener,
NEWER-ROUTE(*Maple-plan, *Sheridan-plan))

is a CG, the Analyzer returns it to be expanded to answer the digressive question. In Demonstration 3, this results in the system response:

```
since taking Maple is a newer route than
taking Sheridan.
```

6.4 Incoherent Questions

In the IDP model, an *incoherent* question is a question that cannot be related to the system's TP either by replanning/expanding the system's TP, or by identifying it as a digression. Demonstration 2a shows how IDP processes incoherent questions. When the Analyzer tries to replan or expand its DTP to answer the question (Section 5.3), it cannot find a TP that uses the propositional content of the question

OBJ-PROP(now, heavy-traffic)

as a nucleus. Therefore, analyzing the question to recognize a way to extend IDP's TP fails.

Next, the Analyzer attempts to process the question as a digression (Section 6). IDP answered this question in Demonstrations 1 and 2. In Demonstration 2a, the questioned proposition has not been expressed previously. Therefore, the first digression test fails immediately. The third digression test also fails because the question is not a simple why-question. In the second digression test, the utterance satisfies the first two conjuncts. That is, it has the correct form, and it has not been expressed by the system. However, it is not inferable from what has been said so far. At the point in the discourse where the question is asked, the only proposition that IDP has expressed is

DO(listener, *Maple-plan)

Since this proposition is not a true fact in the knowledge base, no propositions are inferable from it. Therefore, the Analyzer determines that the questioned proposition is not a digression. Having failed all the tests, IDP responds with Huh?.

Note that in Demonstration 2a, the proposition that the listener questions

OBJ-PROP(now, heavy-traffic)

and the answer to it

OBJ-PROP(now, rush-hour)

are among the localized unknowns (Section 5.1) that can be associated with the TP at the point in the discourse where the question is asked. This means that the information that the listener refers to with her question is highly relevant domain information. In spite of this, the question is incoherent. This suggests that discourse focus and purpose are more important factors for determining the coherency of a question than the domain structure.

7 The System Implementation

In the knowledge base, the several sources of information that the system needs to analyze and plan the discourse are all represented uniformly using the Semantic Network Processing System (SNePS) [17]. This includes knowledge of the text plan operators, the domain plans, entities in the domain, the user model, the discourse plan executed so far, and rules for reasoning about all of the above. The SNePS Actor models a cognitive agent operating in a single-agent world. It integrates inference and acting by representing beliefs, plans, and acts as structured intensional entities in the network formalism. Because the agent's world and planning knowledge is represented uniformly, the agent can discuss, reason about, formulate, and also execute its plans.

Based on the TOUR model [8], the various driving routes that my system can discuss are represented as preconstructed plans that are composed of two types of actions: *going* and *turning*. The domain plans are represented at various levels of detail and, as conceptual entities, can have properties. Whenever the system reasons about the domain, the reasoning that leads to deductions is recorded in the knowledge base along with the deductions themselves and is available as content for explanations.

References

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

[2] A. Cawsey. Generating explanatory discouse. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*. Academic Press, 1990.

[3] 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.

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

[5] S. M. Haller. Interactive generation of plan justifications. In *Proceedings of the Fourth European Workshop on Natural Language Generation*, 1993.

[6] S. M. Haller. A model for cooperative interactive plan explanation. In *Proceedings of the The Tenth IEEE Conference on Artificial Intelligence for Applications*, San Antonio, Texas, 1994.

[7] 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.

[8] B. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2, 1978.

[9] W. C. Mann and S. A. Thompson. Rhetorical structure theory: Towards a functional theory of text organization. *TEXT*, 8(3), 1988.

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

[11] J. Moore and W. Swartout. A reactive approach to explanation. In *International Joint Conference on Artificial Intelligence*, 1989.

[12] J. Moore and W. Swartout. A reactive approach to explanation: Taking the user's feedback into account. In C. Paris, W. Swartout, and W. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Kluwer Academic Publishers, 1991.

[13] J. D. Moore and C. L. Paris. Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics*, 19(4), 1993.

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

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

[16] 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.

[17] 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.

[18] 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.