# MULTIMODAL DISCOURSE MODELLING IN A MULTI-USER MULTI-DOMAIN ENVIRONMENT[1]

*Stephanie Seneff, Dave Goddeau, Christine Pao, and Joe Polifroni*

Spoken Language Systems Group
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139 USA
http://www.sls.lcs.mit.edu

## ABSTRACT

This paper describes the discourse component of GALAXY, a multi-domain, multimodal conversational system. In designing this module, we are attempting to develop domain-independent mechanisms, controlled via declarative tables, to promote convenient instantiation of a discourse component for each new domain. Direct anaphoric reference as well as elliptical reference are dealt with appropriately. Users can also refer verbally to items selected via mouse clicks. Cross domain references are particularly challenging, as is the ambiguity problem arising from different case roles for different subdomains. Users often utter fragments, sometimes in response to server-initiated dialogue exchanges, so an extensive fragment interpretation mechanism is supported.

## 1. INTRODUCTION

GALAXY is a multi-domain, multi-user, multi-modal conversational system that has been under development in the Spoken Language Systems Group at MIT-LCS for the last three years [1]. GALAXY focuses on information of interest to a traveller, including world wide weather and air travel information, and tourist assistance for the city of Boston. In addition to text and speech input, GALAXY understands integrated speech and mouse-click references to items in a list or on a map.

This paper mainly concerns GALAXY's discourse module. While we have applied this module thus far only in the context of travel related domains, we believe it is capable of supporting more generic discourse solutions. The main role of the discourse module is to interpret sentences in context. Users can refer back to previous information either directly through anaphoric reference (e.g., "this one,") or indirectly by not repeating prior constraints that are implied. Users may also utter queries that are unevaluable, due to missing critical information. Part of the discourse module's role is to identify such problems and initiate a subdialogue to fill in the missing elements. Users often utter fragments, particularly in response to such explicit requests for information, and these are usually interpreted by incorporating them into preceding queries. Finally, it is the discourse module's responsibility to determine the appropriate domain server for the query.

GALAXY is implemented in a client-server framework, with the client interfacing with the user and consulting several distinct knowledge servers to answer a query. Generic inheritance mechanisms are applied in the client's multi-domain discourse component, and servers can augment or supercede the discourse actions with additional inheritance requirements that are dependent on more specific domain knowledge than is available to the client. In practice, such augmentations are usually associated with situations where the server has momentarily taken control of the dialogue. For example, the AirTravel server might ask the user for a return date.

## 2. GENERAL ISSUES

In our prior experience with many components of our systems we have come to appreciate the advantages of maintaining the specifications of properties particular to a domain or language in external declarative tables. In designing the discourse component, we tried to adhere to this same philosophy as much as was feasible. Ideally, this would mean that an effective discourse mechanism could be put in place for a new domain by simply filling out a table specifying the new domain's inheritance requirements.

A challenging problem that has emerged as a consequence of multiple domains is that some words/phrases are ambiguous as to domain, or even as to case role. The system supports the possibility of underspecifying the case role and/or domain association, leaving multiple options open for possible resolution at a later time. Such ambiguity is resolved through conjunction with the domain specified by other constituents, either in the current frame or the history record. The city of Boston is probably the best example of this problem. The CityGuide domain understands TOWN to mean a delimited *geographical area* defining spatial limits for a search, as in "the bookstores in Boston." The AirTravel domain, on the other hand, understands the concept CITY to be a *point location*, as in "flights from Denver to Boston." A query such as "What about Boston?" cannot be properly interpreted until context is considered. Other cases include, for example, restaurants in Boston named "Hong Kong" and "India," which obviously have other interpretations as well.

[1] This research was supported by DARPA under contract N66001-94-C-6040, monitored though Naval Command, Control and Ocean Surveillance Center.

## 3. PROCESSING STAGES IN CLIENT

The GALAXY client is concerned with high-level control processing, with each instantiated client being devoted to a single user. A block diagram of control flow in a client is shown in Figure 1. The highest level process monitors constantly for input from one of three distinct sources – keyboard, audio, and mouse.

A user can click on any item displayed in a list. Each item is represented internally as a semantic frame, and any clicked item is recorded as the "frame in focus" (FIF). In some cases, a mouse click can invoke an immediate action. For example, a request for weather in Florida will result in a list of Florida's cities, and a click on any item in the list will automatically bring up the weather for that city. Clicked items are always available for pronominal reference in a follow-up query, such as "Show me the flights *there* from Paris." Each point displayed on a map is also clickable, and each such point is associated with a single item in the displayed list.

Audio input is sent to the recognizer server [2], which returns an $N$-best list of hypotheses. This list is in turn sent to the natural language parser [3], which selects the "best" alternative and returns a semantic frame. The selected semantic frame, along with the list of displayed items and any existing FIF, are sent to the discourse component, which interprets the sentence in context and determines the appropriate domain. A paraphrase is generated from the discourse-resolved frame and displayed to the user. The frame is then dispatched to the domain server. A similar process transpires for keyboard entries.

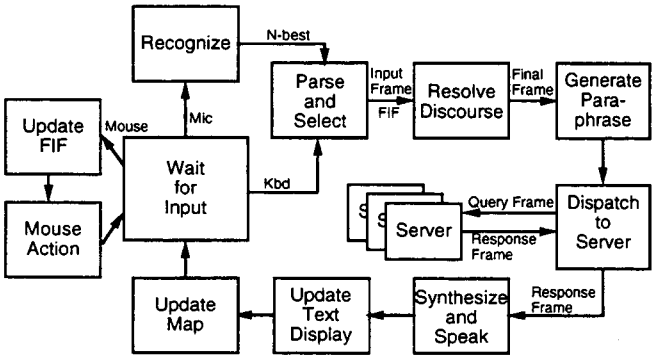


**Figure 1:** Block diagram of GALAXY client control flow.

The server evaluates the semantic frame and returns a response frame, which includes a response string and an optional list of items, also represented in semantic frame format. Servers can optionally return a discourse-update frame, which is incorporated into the discourse in the same way as a user query. The client sends the response string to the synthesizer for spoken responses, updates the display elements, and returns to the wait loop for further direction from the user.

## 4. DISCOURSE ALGORITHM

The discourse component maintains a history table which contains a record of prior reference objects that could be needed to fully interpret future queries. The entries are all represented in the form of semantic frames, and are keyed on conceptual labels. In general, only the most recent instance of a particular reference category is kept. The system identifies the main topic of the new query as the "focus," which plays a critical role in history resolution. Other categories include source, destination, date, the most recent lists for particular semantic classes, objects of certain predicates such as "in" and "about," etc.

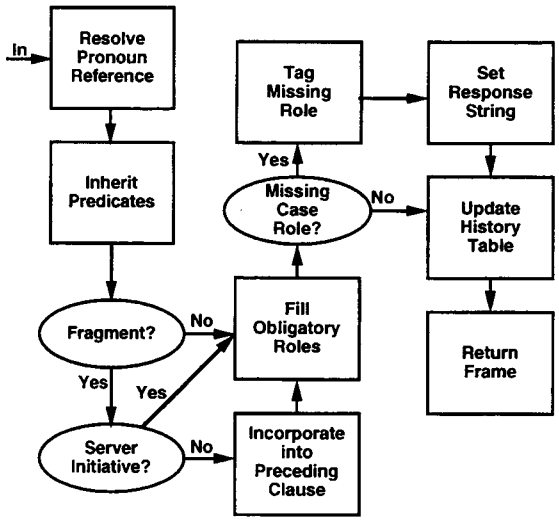


**Figure 2:** Block diagram of flow control in discourse module.

A three-level distinction (immediate, new, and old) is maintained for the source, destination, and focus, and this information is utilized during the decision process. "Immediate" includes clicked items as well as NP's culled from the *current* semantic frame, that may be needed for sentence-internal pronominal reference. "New" refers to references first appearing in the *preceding* query. All other instances are "old."

Each new query is processed through a series of intermediate steps to resolve different aspects of the discourse, such as explicit anaphora and ellipsis. The temporal order of the procedures has been determined empirically and is subject to change. The current instantiation is presented in block diagram form in Figure 2. The system first attempts to resolve direct pronominal reference, where any clicked item takes precedence over all other sources if it passes semantic restrictions. Verbal references to "the $n$-th one" are treated in the same way as clicked items. The history is then updated to include any potential referent in the current frame. For example, for the sentence, "How do I get from LCS to the closest bank?" *LCS* is entered into the immediate history, and is later used to resolve the argument of "closest."

After resolving explicit pronominal reference, the system then applies inheritance rules for any predicates that were mentioned in prior queries and should carry forward. We determined through our prior experience with the flight domain [4, 5] that a good strategy for implementing predicate inheritance is to specify two tables, one indicating which predicates should be inherited for each NP seman-

tic class, and the other specifying which predicates, if present in the current frame, mask inheritance of particular other predicates. We adopted this same strategy for GALAXY.

After dealing with explicit pronominal reference and implicit predicate inheritance, the system then makes a branch point decision based on whether or not the current query is a fragment. Fragments require special treatment, as they are typically incorporated into the preceding clause either by insertion or replacement. In our system a fragment can only contain a topic or one or more predicates. To interpret the fragment, the system must "find a home" for the fragment's topic or predicate(s), splicing it into the clause in history.

A fragment could also be a response to a specific question in a server-initiated dialogue exchange. For such cases, the discourse table contains a list of semantic categories that would be appropriate responses for each such server-initiated exchange. If the fragment matches the conditions, the discourse component bypasses inheritance, deferring to the server to deal with a subdialogue without further complications.

Following fragment analysis, the final discourse step is to fill in any obligatory case roles. For example, directions and flights require a source and destination, "nearest" requires an argument for comparison, and a "property" (such as phone number) requires a possessor or "of" predicate. In the event that no suitable filler for the role can be found, the entry in the history is marked as "missing," and an appropriate response string, such as "Where are you?" is generated. Since such an interchange is likely to provoke a fragment response by the user, the fragment analyzer gives priority to filling any missing slots.

Once the inheritance for the current frame is completed, the history table is updated. This includes replacing the source and destination (or marking them as "old" if the utterance contains no source or destination), and updating the focus and the slots for inheritable predicates, such as *in* or *date*. Any topic whose semantic class is specified in the history table as a potential coreference class is also stored in the history keyed under its class. The entire frame is entered into the history as the most recent clause, which would be recalled for any subsequent fragment analysis. If the system displayed a list of items, the topic of the current clause is entered as the frame associated with both the most recently displayed list and the most recent list *for this particular semantic class*. These two entries are needed to resolve requests such as "Go back to the list," or "Show me the restaurants again," respectively.

Figure 3 shows several examples of entries in the discourse control file for GALAXY. We have adopted a standardized format for entering knowledge under a diverse set of headings, to facilitate development of a new domain. The symbol "&" is a generic join that may in practice mean "AND," "OR," or some other relationship, depending upon the table heading. The heading "TOPIC_DOMAINS" is used to determine the appropriate domain server, and also to check for consistency within a single utterance. The entry under "DOMAIN_DEFAULTS" indicates, for example, that any references to "weather *there*" should be interpreted as "weather in Boston," in the context of any CityGuide question. The first entry under "PREDICATE_INHERITANCE" states that any NP's in the semantic classes EVENT or WEATHER should inherit a prior date.

TOPIC_DOMAINS

| | |
|---|---|
| CityGuide&AirTravel&Weather: | CITY&TOWN |
| AirTravel&Weather: | CITY MONTH_DATE |

DOMAIN_DEFAULTS

| | |
|---|---|
| CityGuide: | IN CITY *Boston* |

SEMANTIC_CLASS

| | |
|---|---|
| LOCATION&AirTravel: | CITY AIRPORT |
| EVENT: | FLIGHT FARE |
| PROPERTY: | PHONE HOURS MENU... |

PREDICATE_INHERITANCE

| | |
|---|---|
| MONTH_DATE: | EVENT WEATHER |
| IN&STATE: | CITY |

PRONOUN_REFERENCE

| | |
|---|---|
| LOCATION: | *there to* |
| FIRST_PERSON: | *I me from here* |

**Figure 3:** Representative entries from the discourse table for GALAXY

## 5. SERVER DISCOURSE ACTIVITIES

The server response can affect discourse context in a number of ways. First, the server returns information at the user's request, and the user may refer to that information in later interactions. As mentioned previously, the servers provide information in list form, accessible by clicking or numerical reference. The server can also ask the user for clarification, and the user is likely to respond with fragments that the client may not be able to interpret on its own. The server may also interpret parts of the user frame more fully, returning a replacement for discourse update. This is especially crucial for dates that are expressed relative to other dates, as in "three days later." If not replaced, the date would keep incrementing by three with each subsequent AirTravel query! Finally, the server may take initiative in helping the user toward a common goal.

The AirTravel server provides a good example of server discourse activities, since it tends to take the initiative during flight reservations dialogues. For instance, it may ask questions not directly related to the user's immediate request. A frequent server response to a booking request, "Please book this flight," is "Will this be one way or round trip?" The referent in this case is not the flight (all flights are one way!), but the entire itinerary.

The AirTravel server maintains a distinction between browsing mode, when the user takes most of the initiative, and booking mode, when the system takes some initiative. The discourse component must keep track of both sides of the user-system dialogue. During booking mode, the server may display information that the user did not specifically request, for instance by showing fares after both legs of a round trip flight have been booked. When the system is taking initiative, a semantic frame created by the server is incorporated directly into the history. The server may also set context for non-speech interaction, allowing mouse clicks to be interpreted in a domain-specific context, for example clicking to get more information on a flight or to book a fare.

| | |
|---|---|
| **Utt1:** | WHAT IS THE FORECAST FOR DALLAS TOMORROW |
| **Action1:** | <show forecast for Dallas tomorrow> |
| **Utt2:** | HOW ABOUT BOSTON |
| **Action2:** | <show forecast for Boston tomorrow> |
| **Utt3:** | ARE THERE ANY FLIGHTS THERE FROM DALLAS |
| **Action3:** | <show flights from Dallas to Boston on May 3rd> |
| **Utt4:** | WHAT IS THE CLOSEST BANK TO HERE |
| **Action4:** | <request missing source> |
| **Utt5:** | AT MIT |
| **Action5:** | <show the closest bank to MIT> |
| **Utt6:** | HOW DO I GET THERE |
| **Action6:** | <give directions from MIT to the closest bank to MIT> |
| **Utt7:** | HOW FAR IS THE ROYAL EAST FROM THIS BANK |
| **Action7:** | <give distance between the Royal East and the closest bank to MIT> |
| **Utt8:** | HOW ABOUT LAGROCERIA |
| **Action8:** | <give distance between Lagroceria and the closest bank to MIT> |

**Figure 4:** An example dialogue between a user and our GALAXY system, illustrating domain switching.

```
Clause:   [what_about
           Topic: [CITY&TOWN    name: Boston]
           Domain: AirTravel&Weather&CityGuide  ]
```

**Figure 5:** The semantic frame for the fragment, "What about Boston?"

## 6. AN EXAMPLE

Figure 4 gives an example dialogue, particularly exercising cross-domain discourse reference. Utt1 is the context setting query for Utt2, a straightforward "what about" question. Utt2 results in the semantic frame shown in Figure 5. "Boston" is ambiguous as to both category and domain, and these ambiguities are resolved based on the fact that the previous query was a weather query. The system substitutes "Boston" for "Dallas," returning the reconstructed history frame to the client.

The user switches domains in Utt3. Nonetheless, two items are inherited from the history, "Boston," through direct anaphoric reference, and "tomorrow," elliptically. The AirTravel server converts "tomorrow" into the appropriate date and sends the reconstructed date back to the client.

When the user abruptly switches to CityGuide in Utt4, the discourse process tries to find a referent for "here," but rejects the source "Dallas" because CITY is not a point location in the CityGuide domain. The system responds appropriately with the query, "Where are you?" Utt5 is then an example of a fragment in the context of a missing element, so "MIT" is entered into the history as a source.

Utt6 has a pronominal reference "there" to tag the destination, along with an elliptical source. The system knows that source and destination are obligatory predicates for "directions" clauses. It correctly picks up "the closest bank to MIT" as the destination, by retrieving it from the "focus" slot, and then finds "MIT" itself in the "source" slot, introduced during Utt5. Utt7 has a reference to "this bank," which is easily resolved via an unambiguous match on semantic class. Utt8 is analagous to Utt2 – both "Lagroceria" and the "Royal East" are restaurants, so the system infers that the former should substitute for the latter in the preceding clause.

| No Parse | Discourse Used Correctly | Discourse Used Incorrectly | Discourse Not Used |
|---|---|---|---|
| 142 (24%) | 154 (26%) | 33 (5%) | 271 (45%) |

**Table 1:** Breakdown of discourse performance on wizard-collected data.

## 7. ASSESSMENT

We have been collecting data for GALAXY in a wizard mode over the past several months. Subjects were informed that the system was able to understand some utterances in context, and we were hoping this would encourage them to use discourse capabilities. Table 1 summarizes the system's performance on a designated training set. We were encouraged to see how often the discourse module was needed, although we clearly still have some problems that need to be addressed.

As data were collected, we slowly augmented the system to accomodate newly identified discourse phenomena. We have observed that subjects tend to try out discourse, and, if it works correctly, they continue to make use of it. If they encounter a discourse problem, they tend to revert to speaking fully specified utterances, for fear that discourse will not work correctly. By dividing our wizard data into an earlier half and a later half, we observed that there was a 50% increase in the use of discourse during the later time period. We suspect this increased usage reflects the improved behavior of the discourse model over time.

Discourse processing is particularly vulnerable to logical programming defects, since errors can propagate across both utterances and domains. Therefore, it is important to be able to confirm that the system is still healthy after changes have been made. To this end, we have established a procedure to evaluate the system on a series of sentences specially designed to exercise most of the discourse capabilities. For each sentence, the output of the current system is compared to a verified reference. This has been extremely valuable for detecting inadvertently introduced errors during active system development.

## 8. REFERENCES


1. D. Goddeau, E. Brill, J. Glass, C. Pao, M. Phillips, J. Polifroni, S. Seneff, and V. Zue, "GALAXY: A Human-language Interface to On-line Travel Information," *Proceedings, ICSLP-94*, pp. 707-710, Yokohama, Japan, Sept. 1994.
2. M. Phillips and D. Goddeau, "Fast Match for Segment-based Large Vocabulary Continuous Speech Recognition," *Proceedings, ICSLP-94*, pp. 1359-1362, Yokohama, Japan, Sept. 1994.
3. S. Seneff, "TINA: A Natural Language System for Spoken Language Applications," *Computational Linguistics*, Vol. 18, No. 1, pp. 61–86. 1992.
4. Glass, J., D. Goddeau, L. Hetherington, M. McCandless, C. Pao, M. Phillips, J. Polifroni, S. Seneff, and V. Zue, "The MIT ATIS System: December 1994 Progress Report", *Proc. ARPA Spoken Language Technology Workshop*, pp. 252-256, Austin, TX, January 1995.
5. V. Zue, S. Seneff, J. Glass, D. Goddeau, D. Goodine, C. Pao, M. Phillips, and J. Polifroni, "PEGASUS: A Spoken Dialogue Interface for On-Line Air Travel Planning," *Speech Communications*, Vol. 15, pp. 331–340, 1994.