A Context Resolution Server for the Galaxy Conversational Systems

Edward Filisko, Stephanie Seneff

Spoken Language Systems Group
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139 USA
{filisko, seneff}@sls.lcs.mit.edu

Abstract

The context resolution (CR) component of a conversational dialogue system is responsible for interpreting a user's utterance in the context of previously spoken user utterances, spatial and temporal context, inference, and shared world knowledge. This paper describes a new and independent CR server for the GALAXY conversational system framework. Among the functionality provided by the CR server is the inheritance and masking of historical information, pragmatic verification, as well as reference and ellipsis resolution. The new server additionally features a process that attempts to reconstruct the intention of the user given a robust parse of an utterance. Design issues are described, followed by a description of each function in the context resolution process along with examples. The effectiveness of the CR server in various domains attests to its success as a module for context resolution.

1. Introduction

Context resolution (CR) is an intuitive process in the course of a human-human dialogue. Consider the deceivingly simple utterance, *Put it there*. The interpretations of *it* and *there* can assume many different meanings, depending on the prior conversation, the environmental context, the accompanying hand gestures, etc. In human-human dialogue, a participant draws upon the history of the dialogue, physical and temporal context, inference, shared world knowledge, and even common sense to interpret and, if necessary, disambiguate another dialogue participant's utterance.

In spite of significant effort in designing and building human-computer conversational systems to date [1, 2, 3, 4, 5, 6, 7, 8, 9], there has not yet emerged from the research community a clear view of how to organize all the activities that need to take place in the gap between recognition of the user query and synthesis of the system response. In particular, there is by no means a common convention established on how to distribute discourse-related functionality among system components. We suspect this lack of consensus is due, in part, to the fact that almost every task performed by such an entity depends on contextual information that is distributed among many components. Pressured to localize all the knowledge in one place, some systems choose to handle all discourse and dialogue modeling in a single component [1, 2], while other systems take the approach of distributing discourse functionality to several specific modules, each of which has a clearly assigned role [3]. For example, the architecture presented by LuperFoy et al. [4] adopts three components – Dialogue Management, Context Tracking, and Pragmatic Adaptation.

Many of the phenomena of interest are explored in the study of discourse. Discourse can be defined as any form of conversation or talk. From arguments to soliloquies, task-oriented dialogues to protests, we participate in many types of discourse on a daily basis. Grosz posits that the study of discourse seeks answers to two main questions [10]:

1. What information is conveyed by groups of utterances that extends beyond their individual meanings?
2. How does context affect the meaning of an individual utterance?

This point of view informs our design of dialogue systems, where we have tried to partition these two tasks into distinct servers, which we call a context resolution (CR) server and a dialogue manager. The first question deals with observing the relationships between utterances in order to recognize a goal in the mind of the user. Since a user's ultimate goal is often domain-dependent, we accomplish such goal recognition via a separate dialogue manager for each domain.

Our CR server was designed to address the second of the above questions, that is, how are the ambiguities of a single utterance resolved using history, physical space, and common knowledge as context? The CR server is designed to be completely domain-*independent*, controlled exclusively using domain-*dependent* information contained in external files. This approach greatly facilitates the instantiation of context resolution capabilities for a new domain.

The CR server takes over processing of the user query after the natural language understanding server and passes control on to the dialogue manager. Its main input is a *semantic frame* encoding the meaning of the user's query, as deduced by the recognizer and the parsing engine. Its main output is a *frame in context*, which is intended to represent an expansion of the user query into a meaning representation that fully specifies the user's intention at this point. The dialogue manager also includes context knowledge, such as the fact that the user has reserved a specific flight, but the knowledge needed to *resolve* each specific user query is held in the CR server. The CR server is engaged once again after the dialogue manager has created a reply frame. While it has no responsibility for altering the reply, it does incorporate into its history at this time appropriate *system initiative* information that will help it resolve subsequent user queries.

The specific capabilities composing the CR server were chosen in part because they can be formalized to follow common procedures for all domains, obtaining domain-dependent

This research was supported by DARPA, under contract N66001-99-1-8904 monitored through Naval Command, Control and Ocean Surveillance Center, and by NTT.

```
{c display
   :topic {q library
             :number "plural"
             :pred {p in
                      :topic {q town
                                :name "Cambridge"}}}}
```

Figure 1: Semantic frame for *Show me libraries in Cambridge*.

knowledge from external files. The information in these files represents simple relationships among various concepts, which can be consulted and utilized systematically in the context resolution process. More complex functionalities, such as identifying user goals, are handled by the dialogue manager, since the domain-dependent nature of such capabilities often makes the process unique for a specific domain.

The CR server draws on information from the past via its history record. The semantic frame of each utterance is stored in the history after context resolution is completed. A time-ordered *discourse entity* list [11] is also maintained, which contains every topic mentioned in the dialogue that the user may subsequently reference. Entities presented in a Web browser that are selected via a mouse click are also stored there. Furthermore, the CR server may at times need to make use of pragmatic knowledge that can only be found in external knowledge resources. In such a case it draws on the expertise of other servers to solve the problem, by dispatching requests to a central hub in a *module-to-module sub-dialogue* [12].

2. Galaxy Framework

Researchers in the Spoken Language Systems group at MIT have been developing human-computer dialogue systems for nearly fifteen years. These systems are developed within the GALAXY Communicator architecture, which is a multi-modal conversational system framework [12]. A GALAXY system is configured around a central programmable hub which handles the communications among various human language technology servers, including servers for speech recognition and synthesis, language understanding and generation, dialogue management, and context resolution, the topic of this paper.

In the GALAXY domains under development at MIT (JUPITER for weather information [6], MERCURY for flight reservations [5], ORION for off-line task delegation [7], PEGASUS for flight arrival and departure information [8], VOYAGER for city traffic and landmark information [9], and many others), the linguistic knowledge from an utterance is encoded in a hierarchical *semantic frame*. The top-level frame is always labeled as *clause* (c), but subcomponents in the hierarchy can be classified as any of *clause*, *topic* (q), or *predicate* (p), which carry linguistic meaning. Each such entity is also called a *frame*, and typically contains additional information represented as keys with associated values, which can also be frames. TINA [13], the parsing component of the natural language server, parses from a word graph produced by the recognizer and encodes the best hypothesis as a semantic frame, organized according to the parse tree structure. Figure 1 shows a semantic frame for the utterance, *Show me libraries in Cambridge*.

3. Context Resolution Algorithm

The natural language server passes the frame to the CR server, which then walks the frame through a sequence of stages, each of which addresses a specific aspect of the context resolution algorithm, as shown in Figure 2. Besides resolving context, the CR server may also reorganize the frame into a more meaningful arrangement, controlled by an external table of semantic relationships. The stages in Figure 2 that do this reorganization are *Repair Topic Relationships*, *Form Obligatory Topic Relationships*, and *Repair Predicate Assignments*. This type of frame reorganization is not typically viewed as a discourse task, but it seemed productive to us to give the CR server this responsibility, since it already possesses extensive knowledge about the relationships among clauses, topics, and predicates within the domain. We have found that a powerful approach to handling the problem of recognition errors leading to incoherent sentence structure is to allow the parser to extract a set of topics through a *robust parse* mechanism. The CR server can then infer missing relationships and attempt to repair the frame as well as possible, given external domain knowledge.

Pronoun, deictic, and definite noun phrase resolution (*Resolve Anaphora and Deixis* in Figure 2) are additional functionalities performed in the context resolution process. These phenomena become more interesting in multi-modal domains where the physical context of the output display becomes relevant to the interpretation of a user's input. GALAXY's multi-modal interface, WebGALAXY, allows spoken, typed, and clicked input to a Web browser [14]. The CR server can handle the integration of two mouse clicks in a single turn.

The inheritance of information from the history is another component of context resolution, handled by the *Inherit/Mask History Using Pragmatics* stage in Figure 2. The CR server consults its history record, incorporating any relevant information into the semantics of the current user utterance. On the other hand, information that is contradictory or obsolete, given the current utterance, is not propagated and is, therefore, unavailable as future context.

It is often the case that the user utters a short fragment, particularly in response to a system prompt. For instance, the utterance, *Yes*, can only be interpreted in the context of the system's previous query. If the system asks, *Did you say to Boston, Massachusetts?* and the user answers, *No, Austin, Texas*, then the fact that *Austin* plays the role of a *destination* is known only because the system prompted for the validity of a different destination. In other words, *Austin* is first incorporated into the system-initiative frame (replacing *Boston*) before being incorporated into the further context of the prior user utterances, to yield, perhaps, *Show me flights from Denver* to Austin, Texas *leaving on March 23*. These phenomena are handled by the *Handle System Initiative* and *Resolve Ellipsis and Fragments* stages in Figure 2.

3.1. The Sequence of CR Functions

Determining the exact functions to be carried out by the CR server and the optimal order of execution is a major challenge. The CR server exploits and extends the set of functionalities, which were experimentally defined by examining collected dialogues within various GALAXY domains [15]. The sequence of functions achieving the best performance, established by trial-and-error, is described below. Experimenting with alternative control sequences is simple, since the sequential order of function execution is specified in an external table provided by the system developer, and thus can be easily rearranged. All domain-dependent knowledge and constraints are developer-specified in external tables.

We will proceed through the context resolution algorithm explaining the operation of each stage. The examples in Table

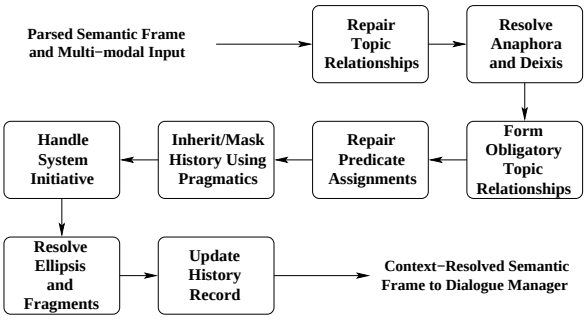

Figure 2: Context resolution algorithm.

Example 1
Spoken:
Like to get some pricing information from Dayton to LA.
Recognized:
What is the price the information from Cleveland to LA?
Example 2
1. *I want a flight to Denver.*
2. *I'll leave from Boston, and connect in Chicago.*
3. *Do you have a nonstop flight?*
Example 3
1. *What is the weather like in Atlanta, Georgia?*
2. *Is it snowing in Augusta?*

Table 1: Examples to demonstrate stages of the CR algorithm.

1 will be referenced to help clarify the various functions.

Example 1 is taken from the log of a real user telephone interaction in the MERCURY domain. This example will demonstrate the effect of *Repair Topic Relationships*, *Form Obligatory Topic Relationships*, and *Repair Predicate Assignments*. Examples 2 and 3 demonstrate the effect of *Inherit/Mask History Using Pragmatics*.

Repair Topic Relationships: In the event that a user's utterance cannot be fully parsed, the parser backs off to a robust mode and places a collection of parsed fragments into a semantic frame. These fragments are often topics with undefined relationships. The CR server consults a table and attempts to infer a relationship, thereby linking the fragmentary topics.

The recognized utterance of example 1 contains the sequence *the price the information...* which the parser analyzes as two distinct noun phrases with an undetermined relationship. The CR server attempts to infer a relationship between them by consulting an external rule file. Finding the relationship, *information for fare*, the server links the two topics appropriately in the semantic frame.

This stage occurs first in the algorithm since it functions as a type of pre-processor, making sure that the base of topic relationships is established before context resolution continues.

Resolve Anaphora and Deixis: The next step is to identify appropriate *antecedents* for any anaphors present in the frame, including pronouns (*it*, *them*) and definite and demonstrative noun phrases (*the bank*, *that flight*). It also deals with deictic (*this*, *here*) and ordinal (*the third one*) references. When any of these references appears in the semantic frame, the discourse entity list must be searched for an appropriate entity to resolve the reference. The entity list is bounded and contains every topic, explicitly or implicitly invoked, which may serve as an antecedent. The first entity in the list that matches detailed constraints is chosen as the antecedent for a given anaphor.

Entities may have been selected via a mouse-click on a displayed list of items. For example, a user may click successively in a single turn on two elements on a map while saying, *I want to go from here to there*. Mouse-clicked entities are inserted at the top of the entity list, so that they take priority. Temporal order is honored in associating multiple clicks with multiple deictics. We are in the process of incorporating time stamps to make such deictic reference resolution more precise.

Form Obligatory Topic Relationships: We have developed a mechanism to more fully represent the information in a semantic frame by allowing the developer to specify topic relationships in which some topic *must* exist to form a relationship with another topic already present in the semantic frame. If such an obligatory topic relationship is not satisfied, the required relationship and topic will be created by the CR server.

In example 1, a constraint finds that, in the MERCURY domain, a fare is always associated with a flight. Therefore, a *flight* topic is created to honor the obligatory *fare for flight* relationship. At this stage, all the topics have been related as *information for fare for flight*.

The reason this stage follows the *Resolve Anaphora and Deixis* stage is that the resolution of a pronoun may result in an obligatory topic, in which case the relationship would no longer need to be created. For example, *How much is it?* may be spoken by a user desiring the price of a flight. The parser would output the semantic relationship *fare for it*, *it* would be resolved to *flight*, and there would no longer be an obligatory topic relationship to create.

Repair Predicate Assignments: A generic mechanism has been developed whereby each clause and topic element is licensed to contain specific predicates within its frame. The server consults constraints to determine whether or not a given predicate is licensed to be in its current location. If not, a search will ensue to find another location in the semantic frame where the predicate *is* licensed.

In example 1, neither the *information* topic nor the *fare* topic supports the *source* (*from Cleveland*) or *destination* (*to LA*) attributes. However, the *flight* topic, just created as an obligatory topic in the previous stage, *does* support both *source* and *destination*. These attributes are moved into the new *flight* topic in the semantic frame, resulting in a fully licensed arrangement and a properly formed phrase structure: *information for fare for flight from Cleveland to LA*.

Inherit/Mask History Using Pragmatics: Once the input semantic frame has been rationalized, a process of inheritance ensues to incorporate elements from the history record. Just as predicates can be licensed to occur within specific frames, so too can they be licensed to be inherited from history into specific frames.

Before an element can be inherited from the history, however, there must first be a check to determine if there is any *masker* element already present in the current frame, which would block its inheritance. A pragmatic verification step can also be invoked, to ensure that an inheritance makes sense in the physical world. Examples 2 and 3 in Table 1 will be used to demonstrate this mechanism.

In example 2, the first utterance contributes a topic, *flight*, and a destination, *Denver*, to the history record. The second utterance contributes a source, *Boston*, and a connection city, *Chicago*. In the third utterance, it occurs to the user that a non-

stop flight might be available. *Source* and *destination* are propagated from the previous utterance. However, *connection_city* is masked by the presence of *flight_cycle = nonstop*. The resulting intention is *nonstop flight from Boston to Denver*. Many similar masking constraints can easily be specified by the developer.

In example 3, after the first utterance has been spoken, the history contains *Atlanta* and *Georgia*. The second utterance contributes a new city, *Augusta*, which prevents *Atlanta* from being inherited. *Georgia* has no masker and, thus, only depends upon pragmatics to confirm that *Augusta, Georgia* is a valid city. If the user had instead said, *Is it snowing in Chicago?*, the pragmatics check on the city-state pair, *Chicago, Georgia*, would fail and inheritance of *Georgia* would be blocked. The CR server relies on the generation server to convert the hypothesized city-state combination into an SQL query that will then be processed by the database server, with the result returned to the CR server for interpretation.

Handle System Initiative: When the system initiates a query, such as *Where are you traveling from?*, it anticipates that the user might simply respond with an isolated city or airport, e.g., *Dayton*. The dialogue manager provides a *system initiative* to the CR server, informing it to interpret such an utterance fragment as a *departure* city in this case.

There are situations in various GALAXY domains, where the system prompts for a date, in which the role of the date (return date for flight, check-in or check-out date for hotels, etc.) is embedded in the prompt and provided to the CR server via a system initiative. Similar to the example above, the CR server assigns the appropriate role for the date in the user's response.

Resolve Ellipsis and Fragments: An ellipsis is an utterance from which the user has omitted information assumed to be understood by the system. The CR server often handles this by inheriting the *understood* information from history during the inheritance and masking stage of context resolution. A sentence fragment, i.e., an utterance consisting of a stand-alone phrasal unit rather than a complete clause, is handled by incorporating it into the clause of the previous utterance, replacing one or more elements. Consider the utterance, *Show me a library in Boston*, followed by, *What about Cambridge?* The latter phrase contributes *Cambridge* as a concept, which is spliced into the clause of the preceding utterance. Semantic constraints are utilized to identify *Boston* and *Cambridge* as more semantically consistent than *library* and *Cambridge*. Thus, the system understands, *Show me a library in Cambridge*, instead of *Show me Cambridge*.

In a fragmentary response, whether the user had said *from Dayton* or simply *Dayton* in response to a system initiative, the utterance would be treated identically in this stage.

Update History Record: The last stage of context resolution is updating the history record. First, the semantic frame is traversed as each topic is appended to the discourse entity list, which may already contain other entries from earlier in the dialogue. This updated list can then be used for reference resolution in subsequent utterances. The semantic frame is stored as the new history, which is used to propagate information as context for future utterances. The stored frame may also serve as the context into which a fragment may be spliced.

4. Discussion

The capabilities provided by the CR server are essential in a spoken dialogue system. The server was developed in the MERCURY domain, but its generic design has facilitated the incorporation of its functionality into all of our major GALAXY domains (JUPITER, ORION, PEGASUS, VOYAGER). We believe that the effectiveness of the CR server in this wide array of domains attests to its success as a module for the context resolution aspect of discourse processing.

We are in the process of designing and implementing a generic *dialogue manager* for GALAXY [16]. It is likely that some of the newly emerging generic capabilities, such as date/time or geography processing, will be migrated into the CR server. We will continue to refine the definition of the boundary between these two components.

We are extending the multi-modal capabilities in GALAXY to include hand gesture and pen-based interaction, which we expect to integrate easily into our framework. In parallel, we are providing the capability to time-stamp individual linguistic elements of the parse tree, such that we can make more intelligent decisions about deictic reference resolution.

5. References

[1] O. Lemon et al., "The WITAS multi-modal dialogue system I," in *Proc. European Conf. on Speech Communication and Technology*, Aalborg, Denmark, Sept. 2001, pp. 1559–1562.

[2] A. Stent et al., "The CommandTalk spoken dialogue system," in *Proc. Mtg. of the Assoc. for Computational Linguistics*, College Park, MD, June 1999, pp. 183–190.

[3] J. Allen et al., "An architecture for more realistic conversational systems," in *Proc. Intl. Conf. on Intelligent User Interfaces*, Santa Fe, NM, Jan. 2001, pp. 1–8.

[4] S. LuperFoy et al., "An architecture for dialogue management, context tracking, and pragmatic adaptation in spoken dialogue systems," in *COLING-ACL '98*, Montréal, Québec, Aug. 1998.

[5] S. Seneff, "Response planning and generation in the MERCURY flight reservation system," *Computer Speech and Language*, vol. 16, pp. 283–312, 2002.

[6] V. Zue et al., "From Jupiter to Mokusei: Multilingual conversational systems in the weather domain," in *Proc. MSC2000*, Oct. 2000, pp. 1–6.

[7] S. Seneff et al., "ORION: From on-line interaction to off-line delegation," in *Proc. Intl. Conf. on Spoken Language Processing*, Beijing, China, Oct. 2000, pp. 142–145.

[8] V. Zue et al., PEGASUS*: A Spoken Language Interface for On-Line Air Travel Planning*, ARPA Human Language Technology Workshop, Princeton, NJ, March 6-11, 1994.

[9] J. Glass et al., *Multilingual Human-computer Interactions: A Case Study in the* VOYAGER *Domain*, ATR International Workshop on Speech Translation, Kyoto, Japan, Nov., 1993.

[10] B. Grosz et al., "Discourse and dialogue," in *Survey of the State of the Art in Human Language Technology*, R. Cole et al., Ed., chapter 6. Cambridge University Press, 1998.

[11] J. Allen, *Natural Language Understanding*, The Benjamin/Cummings Publishing Co., Inc., Redwood City, CA, 1995.

[12] S. Seneff et al., "GALAXY-II: A reference architecture for conversational system development," in *Proc. Intl. Conf. on Spoken Language Processing*, Sydney, Australia, Nov. 1998, pp. 931–934.

[13] S. Seneff, "TINA: A natural language system for spoken language applications," *Computational Linguistics*, vol. 18, no. 1, pp. 61–86, 1992.

[14] R. Lau et al., "WebGALAXY: beyond point and click–a conversational interface to a browser," *Computer Networks and ISDN Systems*, vol. 29, pp. 1385–1393, 1997.

[15] S. Seneff et al., "Multimodal discourse modelling in a multi-user multi-domain environment," in *Proc. Intl. Conf. on Spoken Language Processing*, Philadelphia, PA, Oct. 1996, pp. 192–195.

[16] J. Polifroni et al., "Promoting portability in dialogue management," in *Proc. Intl. Conf. on Spoken Language Processing*, Denver, CO, Sept. 2002.