To appear, *AI Magazine, 2001*

Towards Conversational Human-Computer Interaction

James F. Allen, Donna K. Byron, Myroslava Dzikovska,
George Ferguson, Lucian Galescu, Amanda Stent

Dept. of Computer Science
University of Rochester
Rochester, NY 14627

Abstract

The belief that humans will be able to interact with computers in conversational speech has long been a favorite subject in science fiction. This reflects the persistent belief that spoken dialogue would be the most natural and powerful user interface to computers. With recent improvements in computer technology and in speech and language processing, such systems are starting to appear feasible. There are significant technical problems that still need to be solved before speech-driven interfaces become truly conversational. This paper describes the results of a ten-year effort building robust spoken dialogue systems at the University of Rochester.

What is a Dialogue System?

The term "dialogue" is used in different communities in different ways. Many researchers in the speech recognition community view "dialogue methods" as a way of controlling and restricting the interaction. For instance, consider building a telephony system that answers queries about your mortgage. The ideal system would allow you to ask for what you need in any way you chose. The variety of possible expressions you might use makes this a challenge for current speech recognition technology. One approach to this problem is to have the system engage you in a *dialogue* by having you answer questions such as "What is your account number?" "Do you want your balance information?" and so on. On the positive side, by controlling the interaction, your speech is much more predictable, leading to better recognition and language processing. On the negative side the systems has limited your interaction. You may need to provide all sorts of information that isn't relevant to your current situation, making the interaction less efficient.

Another view of dialogue involves basing human-computer interaction on human conversation. In this view, dialogue *enhances* the richness of the interaction and allows more complex information to be conveyed than is possible in a single utterance. In this view, language understanding in dialogue becomes *more* complex. It is this second view of dialogue to which we subscribe. Our goal is to design and build systems that approach human performance in conversational interaction. We believe that such an approach is feasible and will lead to much more effective user interfaces to complex systems.

Some people argue that spoken language interfaces will never be as effective as graphical user interfaces (GUI) except in limited special-case situations (e.g., Schneiderman, 2000). This view underestimates the potential power of dialogue-based interfaces. First, there will continue to be more and more applications for which a GUI is not feasible because of the size of the device one is interacting with, or because the task one is doing requires using one's eyes and/or hands. In these cases, speech provides a worthwhile and natural additional modality (Cohen and Oviatt, 1995).

Even when a GUI is available, spoken dialogue can be a valuable additional modality as it adds considerable flexibility and reduces the amount of training required. For instance, GUI designers are always faced with a dilemma—either they provide a relatively basic set of operations, forcing the user to perform complex tasks using long sequences of commands, or they add higher-level commands which do the task the user desires. One problem with providing higher-level commands is that in many situations there is a wide range of possible tasks, so the interface becomes cluttered with options, and the user requires significant training to learn how to use the system.

It is important to realize that a speech interface by itself does not solve this problem. If it simply replaces the operations of menu selection with speaking a predetermined phrase that performs the equivalent operation, it may aggravate the problem, because the user would need to remember a potentially long list of arbitrary commands. Conversational interfaces, on the other hand, would provide the opportunity for the user to state what they want to do in their own terms, just as they would do to another person, and the system takes care of the complexity.

Dialogue-based interfaces allow the possibility of extended mixed-initiative interaction (Chu-Carroll and Brown, 1997; Allen, 1999). This approach models the human-machine interaction after human collaborative problem solving. Rather than viewing the interaction as a series of commands, the interaction involves defining and discussing tasks, exploring ways to perform the task, and collaborating to get it done. Most importantly, all interactions are contextually interpreted with respect to the interactions performed so far, allowing the system to anticipate the

[1] *This work was supported in part by DARPA grant F30602-98-2-0133, ONR grant N00014-95-1-1088 and NSF grant IRI-9711009.*

Technique Used	Example Task	Task Complexity	Dialogue Phenomena handled
Finite-state Script	Long-distance dialing	least complex	User answers questions
Frame-based	Getting train arrival and departure information		User asks questions, simple clarifications by system
Sets of Contexts	Travel booking agent		Shifts between predetermined topics
Plan-based Models	Kitchen design consultant		Dynamically generated topic structures, collaborative negotiation subdialogues
Agent-based Models	Disaster relief management	most complex	Different modalities (e.g., planned world and actual world)

Figure 1: Dialogue and Task Complexity

user's needs and provide responses that best further the user's goals. Such systems will create a new paradigm for human-computer interaction.

Dialogue Task Complexity

There is a tremendous range of complexity of tasks suitable for dialogue-based interfaces, and we attempt a broad classification of them in Figure 1. At the simplest end are the fintie-state systems that follow a script of prompts for the user. Such systems are in use today for simple applications such as long-distance dialing by voice, and have already proved quite successful. This technique works only for the simplest of tasks.

The frame-based approach includes most of the spoken dialogue systems constructed to date. In this approach, the system interprets the speech to acquire enough information in order to perform a specific action, be it answering a question about train arrivals, or routing your call to the appropriate person at a bank. The context is fixed in these systems for they do only one thing. Specialized processing techniques are used that take advantage of the specific domain. One can view the context as being represented as a set of parameters that need to be before the system action can be taken instantiated (e.g., Seneff and Polifroni, 2000). For example, to provide information about train arrivals and departures, the system needs to know parameters like the train id number, the event involved (e.g., arriving or departing), the day of travel, and so on (see Figure 2). The action is performed as soon as enough information has been identified. This approach has been used for systems providing information about current movies (*e.g.*, Chu-Carroll, 1999), information about train schedules (*e.g.,* Sturm et al, 1999), and for describing routes to restaurants (*e.g.,* Zue *et al.*, 2000).

Because of the simplicity of these domains, it is possible to build very robust language processing systems. One does not need to obtain full linguistic analyses of the sentences, and in fact most information can be extracted by

Parameter	Possible Values
The train ID?	BN101, ...
The event?	Departure, arrival
The location?	Avon, Bath, Corning, ...
The date/time range?	Monday, Aug 3, afternoon, ...

Figure 2: Context for a Train Information Task

simple patterns designed for the specific domain. For example, given the utterance "When does the Niagara Bullet leave Rochester?" pattern-matching techniques could identify values for the following parameters: the train? (Answer: The Niagara Bullet); the event? (Answer: leaving); the location? (Answer: Rochester). Even if speech recognition was poor and the recognized utterance was "Went up the Niagara Bullet to leave in Chester", patterns could still extract the train (*i.e*., the Niagara Bullet) and event (*i.e.*, leaving) and continue the dialogue.

The next level up in complexity involves representing the task by a series of contexts, each represented using the frame-based approach. For instance, for a simple travel booking agent, the system may need to book a series of travel segments, and each one would be represented by a context containing the information about one travel leg. It might also be able to book hotels and rental cars. With multiple contexts, such systems must be able to identify when the user switches contexts. It can be quite challenging to recognize cases where a user goes back and wants to modify a previously discussed context, say to change some detail about the first leg of a trip after discussing the second leg. Examples of such work can be found within the DARPA Communicator project (*e.g.*, Xu and Rudnicky, 2000).

At Rochester, we are primarily interested in the design of systems for the next two levels of complexity shown in Figure 1. In these, the tasks are too complicated to represent as a series of parameterized contexts. In fact, these tasks require the system to maintain an explicit model of

the tasks and/or world and reason about these models. The language and the dialogues become significantly more complicated, and one also needs to start explicitly modeling the collaborative problem solving process that the system and user engage in. In the plan-based approach, the dialogue involves interactively constructing a plan with the user (e.g., a design for a kitchen, a plan to evacuate personnel off an island). The last level of complexity involves agent-based models. These dialogues may still involve planning, but also may involve executing and monitor operations in a dynamically changing world (e.g., emergency rescue coordination).

Practical Dialogue

Note that while this classification probably covers most of the potential applications for human-computer interaction, it by no means captures the extent of full human conversation. In all these settings, the dialogue is focussed on accomplishing a concrete task. We call such dialogues *practical dialogues*. The dialogue literature identifies many specific subclasses of practical dialogue such as *task-oriented dialogues*, *information-seeking dialogues*, *advice and tutoring dialogues*, and *command and control dialogues*.

It is important to remember this distinction because it does seem clear that full natural-language understanding by machine will not occur in the foreseeable future. Our belief, however, is that sufficient understanding of practical dialogues is feasible. This is captured in a hypothesis:

> *The Practical Dialogue Hypothesis* : The conversational competence required for practical dialogues, while still complex, is significantly simpler to achieve than general human conversational competence.

But even if this hypothesis is true, it might still be too time-consuming to construct understanding systems for different domains. Even if we could build a demonstration system in one domain, it might still be infeasible to apply this technique to other domains. This leads us to our second hypothesis, which essentially says that while practical dialogues in different domains may appear quite different at first glance, they all share essentially the same underlying structures:

> *The Domain-Independence Hypothesis*: Within the genre of practical dialogue, the bulk of the complexity in the language interpretation and dialogue management is independent of the task being performed.

If this hypothesis is true, it is worthwhile to spend a considerable effort building a generic dialogue system that can then be adapted to each new task relatively easily. Our experiences, detailed later in this article, have thus far lent support to the truth of these hypotheses. The practical dialogue hypothesis is explored in more detail in (Allen *et al*, 2000).

(1) USR: We need to get the woman in Penfield to Strong.
(2) SYS: OK
(3) USR: What vehicles are available?
(4) SYS: There are ambulances in Pittsford and Webster.
(5) USR: OK. Use one from Pittsford.
(6) SYS: Do you know that Route 96 is blocked due to construction?
(7) USR: Oh
(8) USR: Let's use the interstate instead.
(9) SYS: OK. I'll dispatch the crew.

Figure 3: A Short Example Dialogue

Example: A Fragment of a Practical Dialogue

The following fragment of a dialogue illustrates some of the power of dialogue-based interaction. Dialogues of this complexity can be handled by our system (to be described later). The entire dialogue fragment is shown as Figure 3. The task is an emergency rescue scenario. Specifically, the user must collaborate with the system to manage responses to 911 calls in a simulation of Monroe County, New York. In this situation, Penfield, Pittsford and Webster are towns, Strong is the name of a hospital, and the dialogue starts after a report of an injured woman has been received. The first utterance serves to establish a joint objective to get the woman to Strong Memorial Hospital. In utterance (2) the system confirms the introduction of the new objective. This context is crucial for interpreting the subsequent interaction. The system must identify the question in (3) as initiating a problem-solving act of identifying resources to use to solve the problem. This has significant impact on how the question is interpreted: First, while the user asked about vehicles, the system needs to realize that it should only consider ambulances, because they are the only vehicles that can perform this part of the task. Second, the term "available" means different things in different contexts. In this context it means the vehicles are not currently assigned to another task, and that a crew is ready. Finally, while there may be many available ambulances, the system chooses in its response in (4) to list the ones that are closest to Penfield.

Note that in a frame-based system, such contextual interpretation can be built-in to the specialized language processing. In our system, however, the context is dynamically changing. If we next talk about repairing a power line, for instance, the "vehicles" will now need to be interpreted as electric utility trucks.

Utterance (5) is interpreted as an attempt to specify a solution to the objective by sending one of the ambulances from Pittsford. Specifically, the user has asked to use an ambulance in Pittsford to take the woman to the hospital. In this case, the system does not simply agree to the request because it has identified a problem with the most direct route. Thus it responds by giving the user this information in the form of a clarification question in (6). At this stage,

the solution has not been agreed to and is still the active focus of the discussion.

In (7) the user indicates that the problem was not known (with the "Oh"). Even with a longer pause here the system should wait for some continuation, as the user has not stopped their response. Utterance (8) provides further specification of the solution, and is interpreted as confirming the solution of using an ambulance from Pittsford. (Note that "Let's use one from Webster instead" would have been a rejection of this solution and the introduction of a new one.). Again, it is reasoning about the plan and the situation that leads the system to the correct interpretation.

In utterance (9), the system confirms the solution and takes initiative to notify the ambulance crew (thus starting the execution of the plan).

While this is a short example, it does show many of the key issues that need to be dealt with in a planning-based practical dialogue. Note especially that the interaction is collaborative, with neither the system nor the user being in control of the whole interaction. Rather, each contributes when they can best further the goals of the interaction.

Four Challenges for Dialogue Systems

Before giving a brief overview of our system, we discuss four major problems in building dialogue systems to handle tasks in complex domains and how we approached them. The first is handling the level of complexity of the language associated with the task. The second is integrating a dialogue system with a complex "back-end" reasoning system (*e.g.*, a factory scheduler, a map server, an expert system for kitchen design). The third is the need for intention recognition as a key part of the understanding process. The fourth is enabling mixed-initiative interaction, in which either the system or the user may control the dialogue at different times in order to make the interaction most effective. We will consider each of these problems in turn.

Parsing Language in Practical Dialogues

The pattern-matching techniques used to great effect in frame-based and sequential-context systems simply do not work for more complex domains. They do not capture enough of the subtlety and distinctions that people depend on in using language. We need to produce a detailed semantic (*i.e.*, "deep") representation of what was said–something that captures what the user meant by the utterance. Currently, the only way to get such a system is to build it by hand. While there are techniques for automatically learning grammars from corpora (*e.g.*, Charniak, 2000), such systems produce a "shallow" representation of the structure of written sentences, not representations of meaning.

There have been many efforts over the years to develop broad coverage grammars of natural languages such as English. These grammars have proven to be of little use in practice because of the vast ambiguity inherent in natural languages. It would not be uncommon for a twelve-word sentence to have hundreds of different parses based on syntax alone. One of the mainstay techniques for dealing with this problem has been to use semantic restrictions in the grammar to enforce semantic as well as syntactic constraints. For example, we might encode a restriction that the verb "eat" applies to objects that are edible, for instance, to disambiguate the word "chips" in "He ate the chips" to be the ones made out of corn rather than silicon. The problem with semantic restrictions, however, is that it is hard to find them if we want to allow all possible sentences in conversational English. This is one place where the practical dialogue hypotheses come in to play. While semantic restrictions are hard to find in general, there do appear to be reasonable restrictions that apply to practical dialogues in general. Furthermore, we can further refine the general grammar by specifying domain-specific restrictions for the current task. This can significantly reduce the possible interpretations allowed by the grammar.

In TRIPS, we use a feature-based augmented context-free grammar with semantic restrictions as described in (Allen 1995). We have found little need to change this grammar when moving to new applications, except to extend the grammar to cover new general forms that hadn't happened to occur yet in previous domains. We do, of course, have to define any new words that are specific to the new application, but this is done relatively easily.

Another significant aspect of parsing spoken language is that it is not sentence-based. Rather, a single utterance may realize a sequence of communicative acts called *speech acts*. For instance, the utterance "OK let's do that then Send a truck to Avon" is not a grammatical sentence in the traditional sense. It needs to be parsed as a sequence of three speech acts: an acknowledgement ("OK"), an acceptance ("let's do that") and a request ("send a truck to Avon"). Our grammar produces act descriptions rather than sentence structures. To process an utterance, it looks for all possible speech acts anywhere in the utterance and then searches for the shortest sequence of acts that covers the input (or as much of the input as can be covered).

Integrating Dialogue and Task Performance

The second problem is how to build a dialogue system that can be adapted easily to most any practical task. Given the range of applications that might be used, from information retrieval, to design, to emergency relief management, to tutoring, we cannot place very strong constraints on what the application program looks like. As a result, we chose to work within a agent-based framework, where the back-end is viewed as a set of agents providing services, and we define a broker that serves as the link between the dialogue system and the back-end, as shown in Figure 4.

Of course, the dialogue system has to know much about the task being implemented by the back-end. To accomplish this, the generic system (applicable across a practical domain) is specialized to the particular domain by integrating domain-specific information. The challenges here lie in designing a generic system for practical dialogue together with a framework in which new tasks can be defined

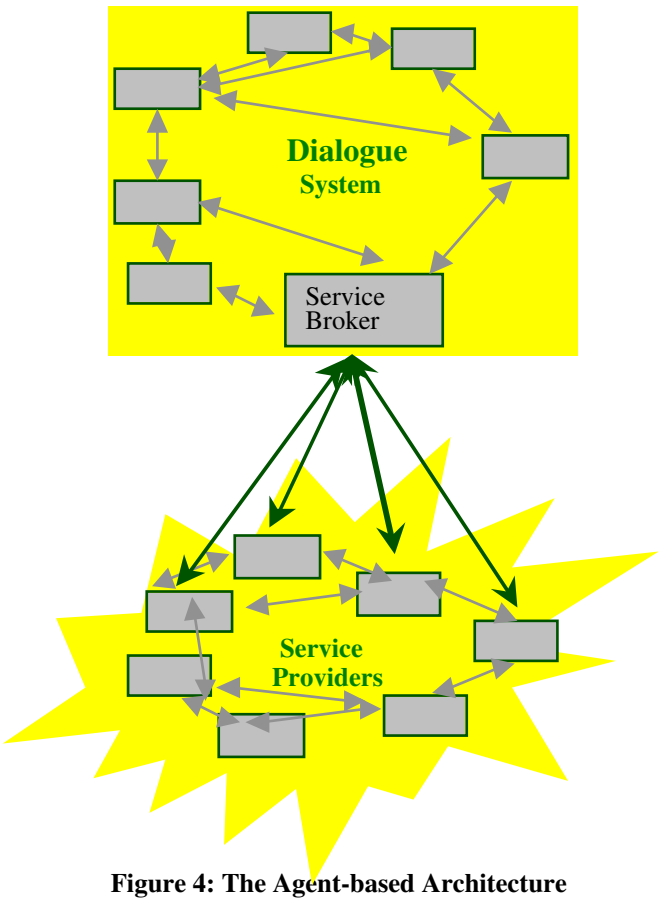

Figure 4: The Agent-based Architecture

relatively easily. Key to this enterprise is the development of an *abstract problem-solving model* that serves as the underlying structure of the interaction. This model includes key concepts such as:

- **Objectives**: The way you want the world to be (*e.g.*, goals and subgoals, constraints on solutions);
- **Solutions:** Courses of action intended to move closer to achieving the objectives;
- **Resources**: Objects and abstractions (*e.g.*, time) that are available for use in solutions; and
- **Situations**: The way the world currently is (or might be).

Utterances in a practical dialogue are interpreted as manipulations of these different aspects, *e.g.*, creating, modifying, deleting, evaluating and describing objectives, solutions, resources and situations. A domain-specific task model provides mappings from the abstract problem-solving model to the operations in a particular domain by specifying what things count as objectives, solutions, resources, and situations in this domain and how they can be manipulated. In this way, the general-purpose processing of practical dialogue is separated from the specifics of, *e.g.*, looking up information in database, verifying design constraints, or planning rescue missions.

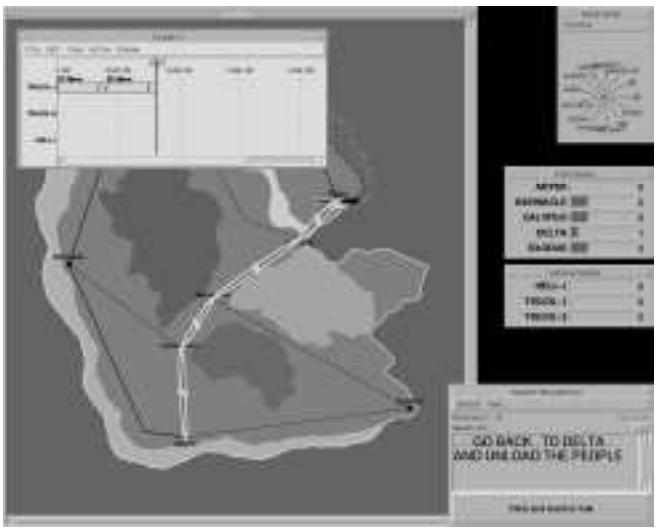

Figure 5: Interacting with TRIPS

Intention Recognition

Many areas of natural language processing have seen great progress in recent years with the introduction of statistical techniques trained on large corpora, and some people believe that dialogue systems will eventually be built in the same way. We do not think that this is the case, and one of the main reasons is the need to do intention recognition, *i.e.,* determining what the user is trying to do by saying the utterance. Let us illustrate this point with one particular example from an application in which the person and the system must collaborate to construct, monitor and modify plans to evacuate all the people off an island in the face of an oncoming hurricane. Figure 5 shows a snapshot of an actual session in progress with an implemented version of the system. On the map you can see the routes developed so far, and the plan itself is displayed in a window showing the actions of each vehicle over time.

For the following example, the context developed by the interaction so far is as follows:

Objectives: The overall objective is to evacuate the island. So far, one subgoal has been developed: evacuating the people in the city of Abyss to Delta.

Solutions: A plan has been developed to move the people from Abyss to Delta using truck one.

Within this setting consider the interpretations of the following utterances:

- *Can we use a helicopter to get the people from Abyss?* In this setting, this utterance is most naturally talking about using a helicopter rather than the truck to evacuate the people at Abyss. It is ambiguous as to whether the user wants to make this change to the plan (*i.e.*, a request to modify the plan), or is asking a question about feasibility. With the first interpretation, an appropriate system response might be “Sure” and modifying the plan. With the second, it might be “Yes we could, and that would save us 10 hours”.

- *Can we use a helicopter to get the people at Barnacle?* The only change is the city mentioned, but now the most natural interpretation is that the user wants to introduce a new subgoal (evacuating Barnacle) and is suggesting a solution (fly them out by helicopter). As before, this is ambiguous between request and question interpretations. A good response to the request might be "OK" (and adding the goal and solution to the plan), while a good response to the question would be "Yes".
- *Can we use a helicopter to get the people from Delta?* Changing the city to Delta changes the most likely interpretation yet again. In this case, the most natural interpretation is that once the people from Abyss arrive in Delta, we should pick them up by helicopter. In this case, the user is talking about extending a solution that was previously discussed. And, as in the other two cases, it could be a request or a question.

These examples show that there are at least six distinct and plausible interpretations of an utterance of this form (changing only the city name). Distinguishing between these requires reasoning about the task to identify what interpretation makes sense rationally given the current situation. There is no way to avoid this reasoning if we are to respond appropriately to the user. The techniques in statistical NLP, while useful for certain subproblems such as parsing, do not suggest any approach to deal with problems that require reasoning in context.

Note also that even if the system only had to answer questions, it would be necessary to perform intention recognition in order to know what question is truly being asked. This reveals the complexity, and the power, of natural language. Approaches that are based purely on the form of language rather than its content and context will always remain extremely limited.

There are some good theoretical models of intention recognition (e.g., Kautz and Allen 1986), but these have proven to be too computationally expensive for real-time systems. In TRIPS, we use a two-stage process suggested in Hinkelman & Allen (1989). The first stage uses a set of rules that match against the form of the incoming speech acts and generate possible underlying intentions. These candidate intentions are then evaluated with respect to the current problem solving state. Specifically, we eliminate interpretations that would not make sense for a rational agent to do in the current situation. For example, it is unlikely that one would try to introduce a goal that is already accomplished.

Mixed-Initiative Dialogue

Human practical dialogue involves mixed-initiative interaction, i.e., it involves the dynamic exchange of control of dialogue flow, increasing dialogue effectiveness and efficiency and enabling both participant's needs to be met. In contrast, in fixed initiative dialogue one participant controls the interaction throughout.

Finite-state systems are typically fixed system-initiative. At each point in the dialogue, the user must answer the specific question the system asks. This can work well for very simple tasks like long-distance dialing, and typically works well for tasks such as finding out information about your bank accounts (although it often takes many interactions to get the one required piece of information). It does not work well when you need to do something a little out of the normal path—in which case you often may need to go through many irrelevant interactions before getting the information you want, if ever. As the task becomes more complex, these strategies become less and less useful.

On the other extreme, a frame-based system can be fixed user–initiative. The system would do nothing but interpret user input until sufficient information was obtained to perform the task. This has the problem that the user may not know what information he or she still needs to supply.

Because of these problems, many current spoken dialogue systems offer limited mixed-initiative interaction. On one hand, the system may allow the user to give information that is in addition to what the system asked for. And on the other, the system may ask for clarification, or may prompt for information it needs to complete the task. These are the simplest forms of initiative that can occur. In more complex domains, initiative may occur at different levels (e.g., Chu-Carroll and Brown, 1997) and dramatically change the system behavior over extend periods of the dialogue.

There are cases where conflicts can arise between the needs of the system and the requirements of the dialogue. For example, consider the 911 domain discussed earlier. A situation can arise in which the user has asked a question (and thus is controlling the dialogue), but the system learns of a new emergency, and thus wants notify the user of the new problem. In such cases, the system must balance the cost of respecting the user's initiative by answering the question, against the cost of ignoring the question and taking the initiative so as to more quickly deal with the new emergency. For example, we might see an interaction like:

USER: What is the status of clearing the interstate from the accident?
SYSTEM: Hold on a minute. There's a new report that a tree just fell on someone in Pittsford.

While this seems incoherent at the discourse level, it may very well be the preferred interaction when viewed from the perspective of optimizing the task performance.

Current spoken dialogue systems have only supported limited discourse-level mixed initiative. As we will see below, in TRIPS we have developed an architecture in which much more complex interactions can occur. The system's behavior is determined by a component called the Behavioral Agent that, in responding to an utterance, considers its own private goals in addition to the obligations it has because of the current state of the dialogue.

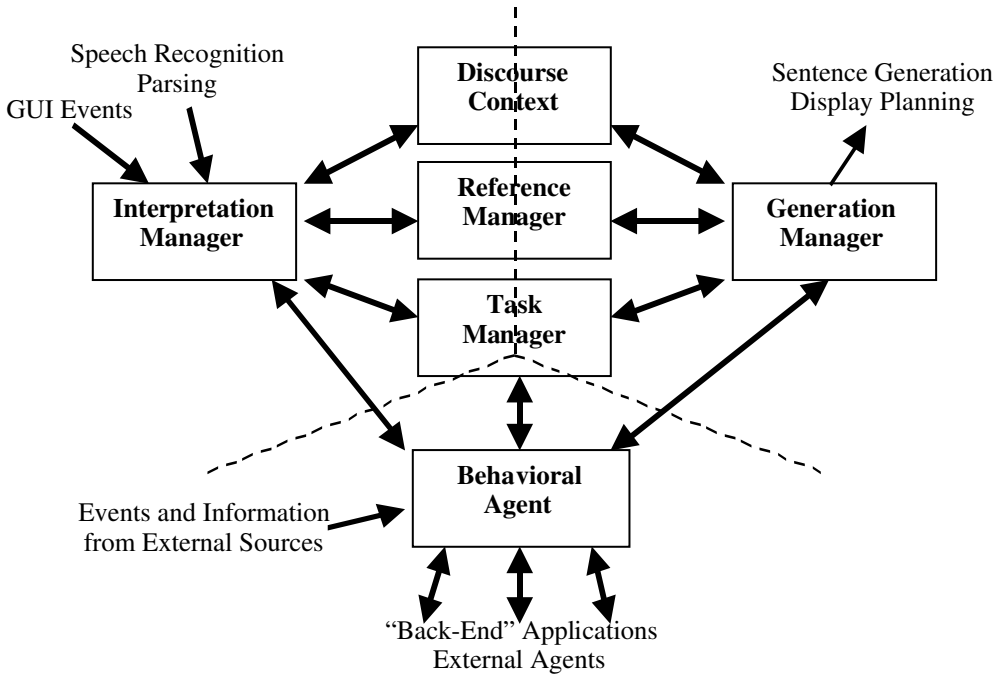

Figure 6: The TRIPS System Architecture

The TRIPS System: A Prototype Practical Dialogue System

TRIPS, The Rochester Interactive Planning System, is the latest in a series of prototype practical dialogue systems that have been developed at the University of Rochester (Allen *et al.*, 1995; Ferguson, Allen, and Miller, 1998). We started by collecting and studying human-human dialogues where people collaborate to solve sample problems (Heeman and Allen, 1995; Stent, 2000). We then used this data to specify performance goals for our systems. Our ongoing research plan is to incrementally increase the complexity of the domain while at the same time increasing the proficiency of the system. In this section, we provide a brief overview of the latest TRIPS system, concentrating on the responsibilities of the core components and the information flow between them.

As mentioned previously, TRIPS users an agent-based component architecture. The inter-agent communication language is a variant of the Knowledge Query and Manipulation Language (Labrou and Finin, 1997). This architecture has proven to be very useful for supporting the development and maintenance of the system, and facilitates experimentation with different algorithms and techniques. More details on the system architecture can be found in (Allen *et al.,* 2001).

The components of the system can be divided into three areas of functionality: interpretation, generation and behavior. Each area consists of a general control/management module that coordinates the behavior of the other modules in the cluster and shares information and messages with the other managers, as shown in Figure 6. As we discuss the components, we will consider the processing of an utterance from our 911 domain.

The speech recognizer in TRIPS uses the Sphinx-II system (Huang 1993) which outputs a set of word hypotheses to the parser. A keyboard manager allows the user to type input as well and passes this to the parser. The parser is a best-first bottom-up chart parser along the lines described in (Allen 1995) and, as discussed earlier, uses a grammar that combines structural (i.e., syntactic) and semantic information. TRIPS uses a generic grammar for practical dialogue. A general semantic model and a generic set of predicates represent the meanings of the common core of conversational English. This grammar is then specialized to an application domain by defining domain-specific lexical items, domain-specific semantic categories, and mappings from the generic predicates to the domain-specific predicates, using a set of declarative "scenario" files defined for the application.

The output of the parser is a sequence of speech acts, with the content of each specified using the generic predicates as well as the predicates defined in the scenario files. Consider the utterance "We need to get the woman in Penfield to Strong", where Penfield is a town and Strong is the name of a hospital. A simplified version of the output of the parser would be the speech act

```
(ASSERT
 :ID SA11
 :SPEAKER USR
 :HEARER SYS
 :CONTENT
  (NEED
```

```
:AGENT (PRO "we")
:THEME
 (TRANSPORT
  :OBJECT
   (THE ?w (AND
            (TYPE ?w WOMAN)
            (AT-LOC ?w
              (NAME ?n "Penfield"))))
  :TO-LOC (NAME ?s "Strong"))))
```

The output of the parser is passed to the Interpretation Manager, which is responsible for interpreting such speech acts in context and identifying the discourse obligations that the utterance produces as well as the problem solving acts that the user is attempted to accomplish. It invokes the Reference Manager to attempt to identify likely referents for referring expressions. The Reference Manager uses the accumulated discourse context from previous utterances plus knowledge of the particular situation in order to identify likely candidates. The analysis of references in the current sentence is shown in Figure 7.

The Interpretation Manager then uses the Task Manager to aid in interpreting the intended speech and problem solving acts. In this case, it uses general lexical knowledge that sentences asserting needs often serve to introduce new active goals in practical dialogues (not always, *e.g.*, they can be used to extend solutions as well, as in "Go on to Pittsford. We will need to get fuel there"). To explore this hypothesis, it checks whether the action of transporting an injured woman to a hospital is a reasonable active goal in this domain, which it is. Thus it identifies the following as what the user intended to accomplish by their utterance (the intended problem solving act):

```
(INITIATED
  :WHO USR
  :WHAT
    (CREATE
      :ID PS22
      :AGENT USR
      :WHAT
        (OBJECTIVE
          :WHAT
            (TRANSPORT
              :OBJECT
                (THE ?w (AND
                  (TYPE ?w WOMAN)
                  (AT-LOC ?w PENFIELD)
                  (REFERS-TO ?w WOM1)))
              :TO-LOC SMH1))))
```

The Interpretation Manager also identifies a discourse obligation to respond to the user's utterance:

```
(OBLIGATION
  :WHO SYS
  :WHAT
    (RESPOND-TO
      :WHAT SA11))
```

where SA11 is the ASSERT speech act shown earlier.

The Behavioral Agent must decide how to handle the proposed act of establishing a goal. Assuming that it has no difficulty doing this, and that it has nothing else more pressing, it can plan a response of confirming this suggested goal, thereby completing the problem solving act initiated by the user. Thus it could send the following request to the Generation Manager to simply confirm the user's suggestion:

```
(REQUEST
   :CONTENT
     (CONFIRM
       :WHAT SA11
       :WHY
         (COMPLETE
           :WHO SYS
           :WHAT PS22)))
```

The Behavioral Agent could also, if it chose to take more task-level initiative, search for available ambulances and suggest one to use for transporting the woman. In this example, however, the system remains more passive. The Generation Manager, receiving this request as well as knowing the pending discourse obligation, can satisfy both using a simple "OK" (and possibly updating a screen showing active goals if there are any).

Referring Expression	Likely Referent	Source Used
"We"	SS1: The set consisting of USR and SYS	The general setting of the dialogue
"The woman in Penfield"	WOM1: The injured woman in Penfield previously discussed	Discourse history
"Strong"	SMH1: Strong Memorial Hospital	General world knowledge and lexicon

Figure 7: Reference Resolution

Discussion

While there are still serious technical issues remaining to be overcome, dialogue-based user interfaces are showing promise. Once they reach a certain level of basic competence, they will rapidly start to revolutionize the way the people interact with computers, much like the direct-manipulation interfaces (using menus and icons) revolutionized computer use in the last decade.

We are close to attaining a level of robust performance that supports empirical evaluation of such systems. For instance, we performed an experiment with an earlier dialogue system that interacted with the user to define train routes (Allen *et al.*, 1996). For subjects, we used undergraduates who had never seen the system before. They were given a short videotape about the routing task, taught the mechanics of using the system, and then left to solve routing problems with no further advice or teaching, except that they should interact with the system as though it were another person. Over 90% of the sessions resulted in successful completion of the task. While the task was quite simple, and some of the dialogues fairly lengthy given the task solved, this experiment does support the viability of

dialogue-based interfaces, and validated the claim that such systems would be usable without any user training.

In the next year, we plan a similar evaluation of the TRIPS 911 system, in which untrained users will be given the task of handling emergencies in a simulated world. This experiment will provide a much more significant assessment of the approach using a task that is near to the level of complexity found in a wide range of useful applications.

More Information

This paper has given a very high-level overview of the TRIPS project. More information on the project, including downloadable videos of system runs, is available at our web site www.cs.rochester.edu/research/cisd.

References

Allen, J. F, *Natural Language Understanding*, Benjamin Cummings, 1995.

Allen, J. F., L. K. Schubert, G. Ferguson, P. Heeman, C.-H. Hwang, T. Kato, M. Light, N. G. Martin, B. W. Miller, M. Poesio, and D. R. Traum, "The TRAINS project: A case study in defining a conversational planning agent," *Journal of Experimental and Theoretical AI*, **7**:7–48, 1995.

Allen, J.F, G. Miller, B., Ringger, E. and Sikorski, T. "A Robust System for Natural Spoken Dialog", *Proc. of the Association for Computational Linguistics (ACL*, Santa Cruz, CA., 1996.

Allen, J.F. "Mixed Initiative Interaction", *Proc. IEEE Intelligent Systems, 14*, 6. 1999.

Allen, J., D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent, "An Architecture for a Generic Dialogue Shell," *Journal of Natural Language Engineering*, **6**(3), 2000.

Allen, J., G. Ferguson and A. Stent, "An Architecture for More Realistic Conversational Systems", *Proc. of Intelligent User Interfaces (IUI-01)*, Santa Fe, NM 2001.

Charniak, E. "A Maximum-Entropy-Inspired Parser", *Proc. Of 1st Mtg of the North American Chapter of the American association for Computational Linguistics*, Seattle, WA, 2000.

Chu-Carroll, J., and M. K. Brown, "Tracking Initiative in Collaborative Dialogue Interactions," *Proc. 35th Annual Meeting of the Association for Computational (ACL-97)*, Madrid, Spain, 1997.

Chu-Carroll, J. Form-based reasoning for Mixed-Initiative Dialogue Management in Information-Query Systems, *Proc. EUROSPEECH,* 1999.

Cohen, P. R., and S. L. Oviatt, "The Role of Voice Input for Human-Machine Communication," *Proc. National Academy of Sciences*, **92**(22): 9921–9927, 1995.

Ferguson, G., J. Allen, and B. Miller, "TRAINS-95: Towards a mixed-initiative planning assistant," in Brian Drabble, ed., *Proc. Third Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 70–77, Edinburgh, Scotland, 1996.

Ferguson, G., and J. F. Allen, "TRIPS: An Integrated Intelligent Problem-Solving Assistant," *Proc. of the National Conference on Artificial Intelligence (AAAI-98)*, Madison, WI, pages 567–573, 1998.

Heeman, P. A., and J. Allen, "The TRAINS-93 Dialogues", TN94-2, Dept. of Computer Science Dept., University of Rochester, March 1995.

Hinkelman, E. and J.F. Allen, "Two Constraints on Speech Act Ambiguity", *Proc. of the Association for Computational Linguistics (ACL*), Vancouver, Canada, 1989.

Huang, X. D., F. Alleva, H. W. Hon, M. Y. Hwang, K. F. Lee, and R. Rosenfeld, "The Sphinx-II Speech Recognition System: An Overview." *Computer, Speech and Language*, 1993.

Kautz, H. and J.F. Allen, "Generalized Plan Recognition",*Proc. of the National Conference on Artificial Intelligence (AAAI-86)*, Philadelphia, PA. 1986.

Labrou, Y., and T. Finin, "A Proposal for a New KQML Specification," TR CS-97-03, Dept. of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 1997.

Schneiderman, B. "The Limits of Speech Recognition", *Communications of the A.C.M., 43:9,* 2000.

Seneff, S. and J. Polifroni, "Dialogue Management in the Mercury Flight Reservation System," *Proc. ANLP/NAACL 2000 Workshop on Conversational Systems*, Seattle, WA, 2000.

Stent, A.J., "The Monroe Corpus", TR728 and TN99-2, Dept. of Computer Science, University of Rochester, March, 2000.

Sturm, J., E. den Os and L. Boves, "Dialogue Management in the Dutch ARISE Train Timetable Information System, *Proc. EUROSPEECH*, 1999.

Xu, W., and A. Rudnicky, "Task-based Dialogue Management Using an Agenda," *Proc. ANLP/NAACL 2000 Workshop on Conversational Systems*, Seattle, WA, 2000.

Zue, V., S. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen and L. Hetherington, "Jupiter: A Telephone-based Conversational Interface for Weather Information," *IEEE Trans. on Speech and Audio Processing,* **8**(1), 2000.