Toward Conversational Human-Computer Interaction

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

■ The belief that humans will be able to interact with computers in conversational speech has long been a favorite subject in science fiction, reflecting 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 article describes the results of a 10-year effort building robust spoken dialogue systems at the University of Rochester.

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 example, 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 system 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?" 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 system has limited your interaction. You might 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 graphic user interfaces (GUIs) except in limited special-case situations (for example, 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 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 because it adds considerable flexibility and reduces the amount of training required. For example, 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 that 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

Technique Used	Example Task	Task Complexity	Dialogue Phenomena Handled
Finite-state script	Long-distance calling	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 (for example, planned world and actual world)

Table 1. Dialogue and Task Complexity.

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 can aggravate the problem because the user would need to remember a potentially long list of arbitrary commands. Conversational interfaces, however, would provide the opportunity for the user to state what he/she wants to do in his/her own terms, just as he/she would do to another person, and the system takes care of the complexity.

Dialogue-based interfaces allow the possibility of extended mixed-initiative interaction (Allen 1999; Chu-Carroll and Brown 1997). 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 to this point, allowing the system to anticipate the 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 table 1. At the simplest end are the finite-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 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 because 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 instantiated before the system action can be taken (for example, Seneff and Polifroni [2000]). For example, to provide information about train arrivals and departures, the system needs to know parameters such as the train ID number, the event involved (for example, arriving or departing), and the day of travel (see table 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 (for example, Chu-Carroll [1999]), train schedules (for example, Sturm, den Os, and Boves [1999]), and routes to restaurants (for example, 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 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), and 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 (that is, the Niagara Bullet) and event (that is, 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 example, for a simple travel booking agent, the system might 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 Defense Advanced Research Projects Agency COMMUNICATOR Project (for example, Xu and Rudnicky [2000]).

At the University of Rochester, we are primarily interested in the design of systems for the next two levels of complexity, shown in table 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 or world and reason about these models. The language and the dialogues become significantly more complicated, and one 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 (for example, a design for a kitchen, a plan to evacuate personnel off an island). The last level of complexity involves agent-based models. These dialogues might still involve planning but also might involve executing and monitoring operations in a dynamically changing world (for example, emergency rescue coordination).

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

Table 2. Context for a Train Information Task.

Practical Dialogue

Note that although 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 focused 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.

The fact that we are dealing with practical dialogues is important 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, captured in the following hypothesis:

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

However, 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. Thus, we arrive at our second hypothesis, which essentially says that although practical dialogues in different domains might 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.

(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.

(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 1. A Short Example Dialogue.

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).

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 1. The task is an emergency rescue scenario. Specifically, the user must collaborate with the system to manage responses to 911 calls in a simulation for 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, which has significant impact on how the question is interpreted: First, although 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, although there might 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 example, 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 because the user has not stopped his/her 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). Although this example is short, 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 in control of the whole interaction. Rather, each contributes when he/she/it 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 (for example, 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 can control the dialogue at different times to make the interaction most effective. We 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 (that is, 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. Although there are techniques for automatically learning grammars from corpora (for example, 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 12-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 example, 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. Although 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, which 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 haven't happened to occur in previous domains. We do, of course, have to define any new words that are specific to the new application, but this defining is done relatively easily.

Another significant aspect of parsing spoken language is that it is not sentence based. Rather, a single utterance can realize a sequence of communicative acts called *speech acts*. For example, 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 acknowledgment ("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 that 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 strong constraints on what the application program looks like. As a result, we chose to work within an 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 2.

Of course, the dialogue system has to know much about the task being implemented by the back end. Thus, 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 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 (1) *objec-*

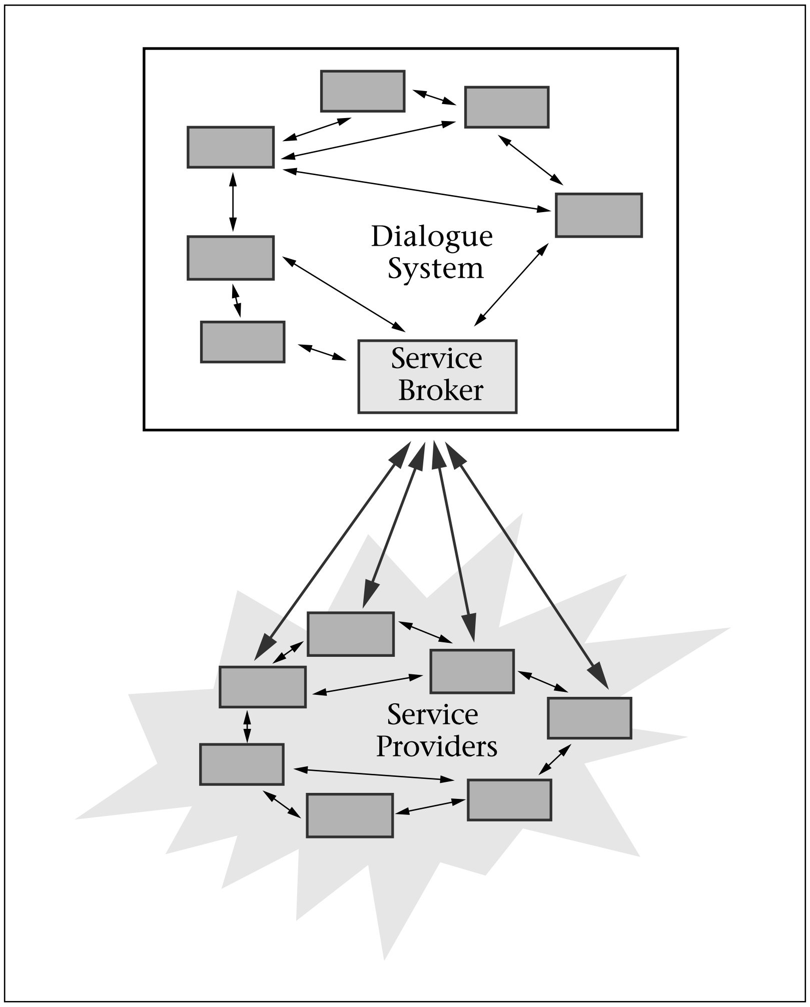

Figure 2. Service Providers.

tives, the way you want the world to be (for example, goals and subgoals, constraints on solutions); (2) *solutions,* courses of action intended to move closer to achieving the objectives; (3) *resources, objects,* and *abstractions* (for example, time) that are available for use in solutions; and (4) *situations,* the way the world currently is (or might be).

Utterances in a practical dialogue are interpreted as manipulations of these different aspects, for example, 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, for example, looking up information in a database, verifying design constraints, or planning rescue missions.

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, that is, 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 3 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 to this point, 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 to this point 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 about whether the user wants to make this change to the plan (that is, 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 question is ambiguous because of request and question interpretations. A good response to the request might be "OK" (and adding the goal and solution to the plan), but 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. 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 interpretations 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 natural language processing, although 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 to know what question is truly being asked, which 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 (for example, Kautz and Allen [1986]), but these models have proven to be too computationally expensive for real-time systems. In TRIPS, we use a two-stage process suggested in Hinkelman and 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; that is, the control of the dialogue changes between the conversants. In contrast, in a fixed-initiative dialogue, one participant controls the interaction throughout. Mixed-initiative interaction increases dialogue effectiveness and efficiency and enables both participants' needs to be met.

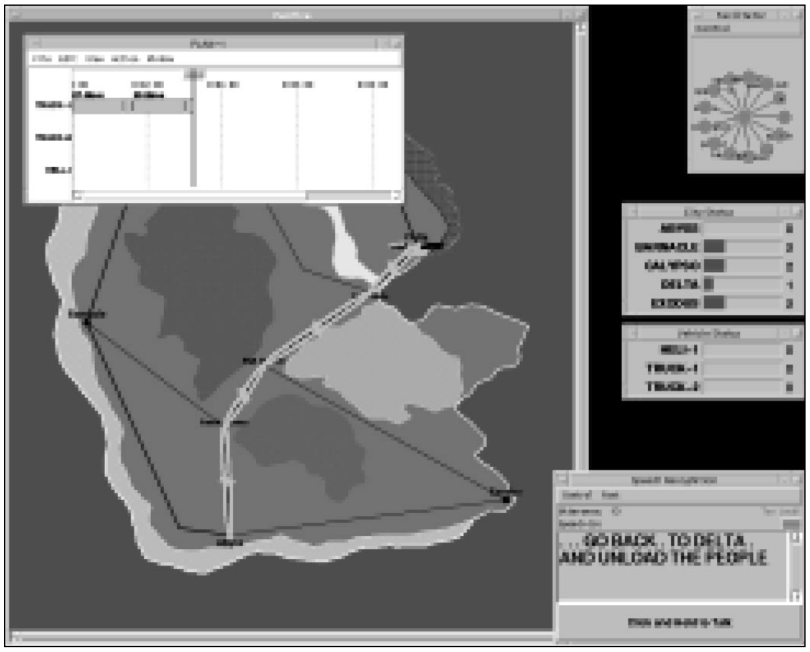
... GO BACK .. TO DELTA
AND UNLOAD THE PEOPLE

Figure 3. Interacting with TRIPS.

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 approach can work well for very simple tasks such as 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 might 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.

At 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 approach is problematic in that the user might not know what information he/she still needs to supply.

Because of these problems, many current spoken-dialogue systems offer limited mixed-initiative interaction. On the one hand, the system might allow the user to give information that is in addition to what the system asked for. On the other, the system might ask for clarification or might prompt for information it needs to complete the task. These are the simplest forms of initiative that can occur. In more complex domains, initiative can occur at

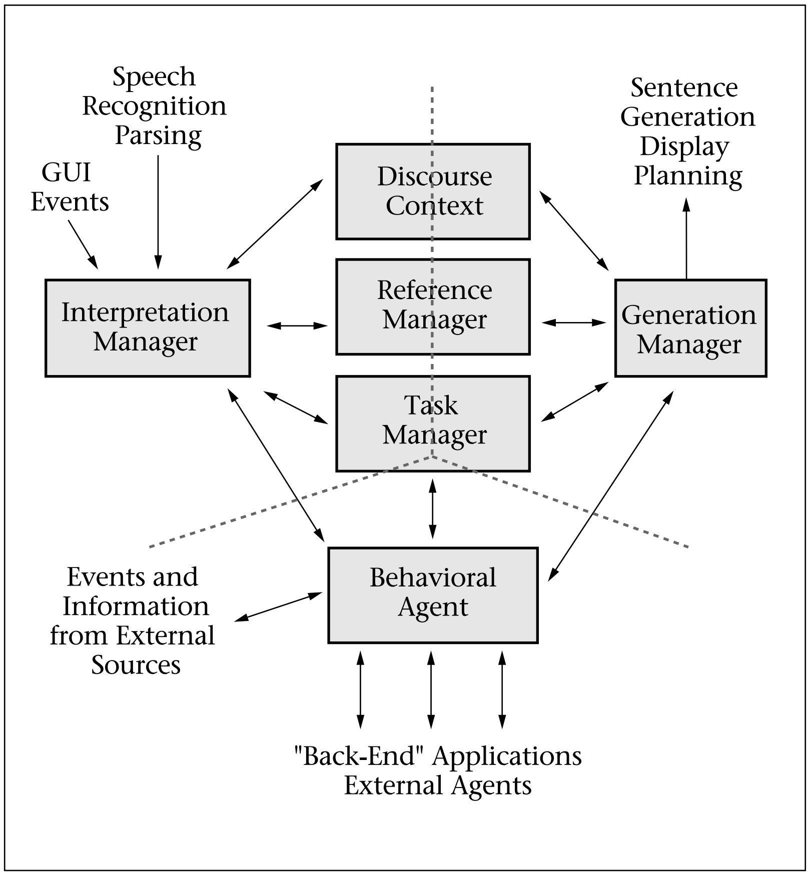
Speech Recognition Parsing
GUI Events
Sentence Generation Display Planning
Discourse Context
Reference Manager
Task Manager
Interpretation Manager
Generation Manager
Behavioral Agent
Events and Information from External Sources
"Back-End" Applications External Agents

Figure 4. The TRIPS *System Architecture.*

different levels (for example, Chu-Carroll and Brown [1997]) and dramatically change the system behavior over extended 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 to 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 to more quickly deal with the new emergency. For example, we might see an interaction such as the following:

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.

Although this interaction seems incoherent at the discourse level, it might 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 see later, 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.

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 and Allen 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 these data to specify performance goals for our systems. Our ongoing research plan is to incrementally increase the complexity of the domain and, at the same time, increase 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 uses an agent-based component architecture. The interagent 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, Ferguson, and Stent (2001).

The components of the system can be divided into three areas of function: (1) interpretation, (2) generation, and (3) 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 4. As we discuss the components, we consider the processing of an utterance from our 911 domain.

The speech recognizer in TRIPS uses the SPHINX-II system (Huang et al. 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 input 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 (that is, 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. The grammar is then specialized using a set of declarative "scenario" files defined for each application. These files define domain-specific lexical items, domain-specific semantic categories, and mappings from generic predicates to domain-specific predicates.

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 given in figure 5.

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 attempting 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, to identify likely candidates. The analysis of references in the current sentence is shown in table 3.

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, for example, 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 figure 6 as what the user intended to accomplish by his/her utterance (the intended problem-solving act):

The interpretation manager also identifies a discourse obligation to respond to the user's utterance (figure 7), where SA11 is the assert speech act shown earlier.

```
(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"))))
```

Figure 5. A Simplified Parser Output.

```
(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))))
```

Figure 6. The Intended Problem-Solving Act.

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 request in figure 8 to the generation manager to simply confirm the user's suggestion.

The behavioral agent could also, if it chose to take more task-level initiative, search for

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

Figure 7. The Discourse Obligation.

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

Figure 8. The Request to the Generation Manager.

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

Table 3. Reference Resolution.

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).

Conclusion

Although 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 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 example, 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 percent of the sessions resulted in successful completion of the task. Although 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.[1]

Note

1. This article gave 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.; Byron, D.; Dzikovska, M.; Ferguson, G.; Galescu, L.; and Stent, A. 2000. An Architecture for a Generic Dialogue Shell. *Journal of Natural Language Engineering* 6(3): 1–16.

Allen, J.; Ferguson, G.; and Stent, A. 2001. An Architecture for More Realistic Conversational Systems. Paper presented at the Intelligent User Interfaces Conference (IUI-01), 14–17 January, Santa Fe, New Mexico.

Allen, J. F. 1999. Mixed Initiative Interaction. In Proceedings of the IEEE Intelligent Systems, 6. Washington, D.C.: IEEE Computer Society.

Allen, J. F. 1995. *Natural Language Understanding*. Wokingham, U.K.: Benjamin Cummings.

Allen, J. F.; Miller, G.; Ringger, B.; and Sikorski, T. 1996. A Robust System for Natural Spoken Dialogue. In Proceedings of the Association for Computational Linguistics, 62–70. New Brunswick, N.J.: Association for Computational Linguistics.

Allen, J. F.; Schubert, L. K.; Ferguson, G.; Heeman, P.; Hwang, C.-H.; Kato, T.; Light, M.; Martin, N. G.; Miller, B. W.; Poesio, M.; and Traum, D. R. 1995. The TRAINS Project: A Case Study in Defining a Conversational Planning Agent. *Journal of Experimental and Theoretical AI* 7(2): 7–48.

Charniak, E. 2000. A Maximum-Entropy-Inspired Parser. Paper presented at the First Meeting of the North American Chapter of the American Association for Computational Linguistics, 29 April–4 May, Seattle, Washington.

Chu-Carroll, J. 1999. Form-Based Reasoning for Mixed-Initiative Dialogue Management in Information-Query Systems. Paper presentec at EUROSPEECH, 5–9 September, Budapest, Hungary.

Chu-Carroll, J., and Brown, M. K. 1997. Tracking Initiative in Collaborative Dialogue Interactions. Paper presented at the Thirty-Fifth Annual Meeting of the Association for Computational (ACL-97), 7–12 July, Madrid, Spain.

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

Ferguson, G., and Allen, J. F. 1998. TRIPS: An Integrated Intelligent Problem-Solving Assistant. In Proceedings of the National Conference on Artificial Intelligence (AAAI-98), 567–573. Menlo Park, Calif.: American Association for Artificial Intelligence.

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

Hinkelman, E., and Allen, J. F. 1989. Two Constraints on Speech Act Ambiguity. Paper presented at Conference of the Association for Computational Linguistics (ACL), 26–29 June, Vancouver, Canada.

Huang, X. D.; Alleva, F.; Hon, H. W.; Hwang, M. Y.; Lee, K. F.; and Rosenfeld, R. 1993. The SPHINX-II Speech-Recognition System: An Overview. *Computer, Speech, and Language* 7(2): 137–148.

Kautz, H., and Allen, J. F. 1986. Generalized Plan Recognition. In Proceedings of the National Conference on Artificial Intelligence (AAAI-86), 32–37. Menlo Park, Calif.: American Association for Artificial Intelligence.

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

Schneiderman, B. 2000. The Limits of Speech Recognition. *Communications of the ACM* 43(9): 63–65.

Seneff, S., and Polifroni, J. 2000. Dialogue Management in the Mercury Flight Reservation System. Paper presented at the Workshop on Conversational Systems, 30 April, Seattle, Washington.

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

Sturm, J.; den Os, E.; and Boves, L. 1999. Dialogue Management in the Dutch ARISE Train Timetable Information System. Paper presented at EUROSPEECH, 5–9 September, Budapest, Hungary.

Xu, W., and Rudnicky, A. 2000. Task-Based Dialogue Management Using an Agenda. Paper presented at the Workshop on Conversational Systems, 30 April, Seattle, Washington.

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

James Allen is the John Dessauer professor of computer science at the University of Rochester. He was editor-in-chief of *Computational Linguistics* and has authored several books, including *Natural Language Understanding* (Benjamin Cummings, 1995), 2d edition. He received a Presidential Young Investigator Award for his research and is a fellow of the American Association for Artificial Intelligence. Allen's research interests lie at the intersection of language and reasoning and span a range of issues, including natural language understanding, dialogue systems, knowledge representation, commonsense reasoning, and planning. His e-mail address is james@cs.rochester.edu.

George Ferguson is a research scientist in the Computer Science Department at the University of Rochester. He received his Ph.D. from Rochester in 1995 for work on adapting AI planning and reasoning to the needs of interactive, mixed-initiative systems. His research interests continue to be at the interface between AI technology and human-computer interaction. His e-mail address is ferguson@cs.rochester.edu.

Myroslava Dzikovska is a Ph.D. student in computer science at the University of Rochester. She graduated with a B.Sc. in applied mathematics from the Ivan Franko Lviv State University in Lviv, Ukraine. Her research interests are natural language understanding, spoken dialogue, AI, and computational linguistics. Her e-mail address is myros@cs.rochester.edu.

Lucian Galescu received an M.S. in computer science in 1998 from the University of Rochester, where he is currently a Ph.D. candidate. He also received an M.S. in mathematics–computer science in 1992, with a thesis on the field of theorem proving, from the A. I. Cuza University of Iasi, Romania, where he held a junior faculty position until 1996. His current research interests are in the area of statistical modeling of natural language. His e-mail address is galescu@cs.rochester.edu.

Amanda Stent is a Ph.D. candidate at the University of Rochester. Her research area is natural language generation for task-oriented spoken dialogue. In 2002, she is joining the faculty in the Department of Computer Science at State University of New York at Stony Brook. Her e-mail address is stent@cs.rochester.edu.