AI Magazine Volume 28 Number 2 (2007) (© AAAI)

Mixed-Initiative Systems for Collaborative Problem Solving

George Ferguson and James Allen

■ Mixed-initiative systems are a popular approach to building intelligent systems that can collaborate naturally and effectively with people. But true collaborative behavior requires an agent to possess a number of capabilities, including reasoning, communication, planning, execution, and learning. We describe an integrated approach to the design and implementation of a collaborative problem-solving assistant based on a formal theory of joint activity and a declarative representation of tasks. This approach builds on prior work by us and by others on mixed-initiative dialogue and planning systems.

Our goal is the design and implementation of collaborative assistants that help people solve problems and get things done. We've all had the bad experience of working with someone who had to be told everything he or she needed to do (or worse, we had to do it for them). A good assistant is one that not only does what it's told, but can also take initiative itself. Collaboration means working together as a group. Taking *initiative* to mean the ability to direct the group's behavior, a mixed-initiative system is one that allows the participants to separately contribute what they can to the group's overall success. In collaborative problem solving, this means coming up with solutions to problems, with initiative varying depending on who can solve which problems.

This type of mixed-initiative collaboration requires a flexible interface that allows the participants to interact naturally in order to make their contributions. For example, a system that rigidly controls the interaction, such as a telephone menu system, can hardly be considered to be working with you, much less for you. As Eric Horvitz put it:

> I shall use the phrase [mixed-initiative] to refer broadly to methods that explicitly support an efficient, natural interleaving of contributions by users and automated services aimed at converging on solutions to problems. (Horvitz 1999)

In our research, we are primarily concerned with systems that interact using spoken natural language dialogue since (1) this is a very efficient means of communication for people; (2) it requires little or no training to use; and (3) it gives us the greatest insight into the nature of human communication and collaboration. Despite the well-known complexities of natural language, this has seemed to us the most likely way to achieve the true mixed-initiative, collaborative systems envisioned by Horvitz and others. Frankly, we can't see any other interface being both flexible enough and expressive enough to support mixed-initiative interaction with a reasonably intelligent system.

In this article, we describe our approach to building mixed-initiative systems for collaborative problem solving. The emphasis is not on the details of understanding natural language dialogue (for that, see, for example, Allen et al. [2001]). Instead we focus on the design of a collaborative agent that, naturally, communicates using dialogue.

On Collaboration

During collaborative problem solving, agents are involved in a variety of activities related to the problems they are trying to solve. They rea-

son about what they (and others) are doing, what they ought to do, and whether what they are doing is going to solve their problems. They communicate with others about what they are doing, what they know or need to know, and what they need others to do with or for them. They plan how tasks will be performed and problems solved. They make commitments that allow other agents to work with them and that also focus their own attention and resources towards their goals. They perform tasks for which they are capable and which it is appropriate for them to perform at that point in time. They learn new ways of performing tasks and solving problems, either by observing other agents or by communicating with other agents (for example, by being told how to do them). Finally, they respond to and solve new problems that arise during the performance of tasks, which often involves learning on the fly.

True collaboration requires an integrated approach to all these different activities. Point solutions to any one of them might yield a useful tool but will not result in a truly collaborative assistant. For example, given the long history of work in AI planning, one might try to build a collaborative planning system (Ferguson, Allen, and Miller 1996; Ferguson and Allen 1998; Allen and Ferguson 2002). While constructing plans collaboratively is an interesting challenge, one quickly finds that these plans then need to be executed in order to actually solve problems for users. Robust execution is then typically itself a collaborative process, so-called plan repair is definitely collaborative, and throughout there are important opportunities for learning (about the domain, the plans, the planning process, the user, the way to work with the user, and so on). In fact, it turns out to be very difficult even to construct realistic, effective plans with users in the loop without an integrated view of the different aspects of reasoning about and using plans.

These considerations have led us to the approach described in the following sections. First, a system architecture that embodies the *agency* required for a collaborative assistant. Next, representations of tasks that support the activities required in order to collaborate. These representations are used to define abstract tasks that guide the system's collaborative behavior and allow it to interpret the behavior of others. And finally, the agent's behavior is driven by the shared beliefs and commitments that arise during collaboration. After briefly describing each of these, we will present an extended example of the approach in action, helping a user perform a collaborative task.

Architecture of a Collaborative Assistant

The need for an integrated approach to designing collaborative assistants has led us to develop the agent architecture shown in figure 1. On the left is shown the overall architecture of a collaborative dialogue system. As described by Allen, Ferguson, and Stent (2001) and Ferguson and Allen (2006), the main goal of this architecture is to avoid the "dialogue pipeline" approach to such systems. The system architecture is itself agent oriented. The three main components, labeled "Interpretation," "Generation," and "Collaborative Agent," operate continuously, asynchronously, and in parallel.[1] This allows continuous interpretation of user action and input, interleaved and overlapping generation of system output, and independent operation of the system in pursuit of its own desires and goals. Further details and discussion are available in the references cited previously.

On the right side of figure 1 is a detailed view of the core Collaborative Agent component. The structure of the Collaborative Agent is based on the belief desire intention (BDI) model of agency (Rao and Georgeff 1991). In the BDI framework, an agent is driven by its beliefs, its desires, and its intentions. The beliefs correspond to the agent's knowledge of itself and the world, the desires correspond to the states it wants or is willing to work to achieve, and the intentions represent its commitments to do certain things towards those desires. BDI agents are driven by their intentions. In the case of collaborative behavior, agents also make joint commitments with other agents that constrain their behavior. Our model describes how the need to make joint commitments drives the dialogue behavior of a collaborative agent.

Interaction between the Collaborative Agent and the other components of the system is in terms of collaborative problem-solving acts, which are speech acts in the sense of Austin (1962), and Searle (1969). Examples are to request that another agent perform an action, or to ask them to inform us of some fact. This abstraction separates the Collaborative Agent from any specific interface modality.

The top subcomponent in figure 1, labeled "Collaborative Action," controls the system's overall communicative behavior. On the one hand it handles collaborative acts performed by the user (as reported by the Interpretation components) and updates the BDI state. On the other hand, it is itself an agent, attempting to achieve collaborative goals (such as agreeing on the value of a parameter or jointly commit-

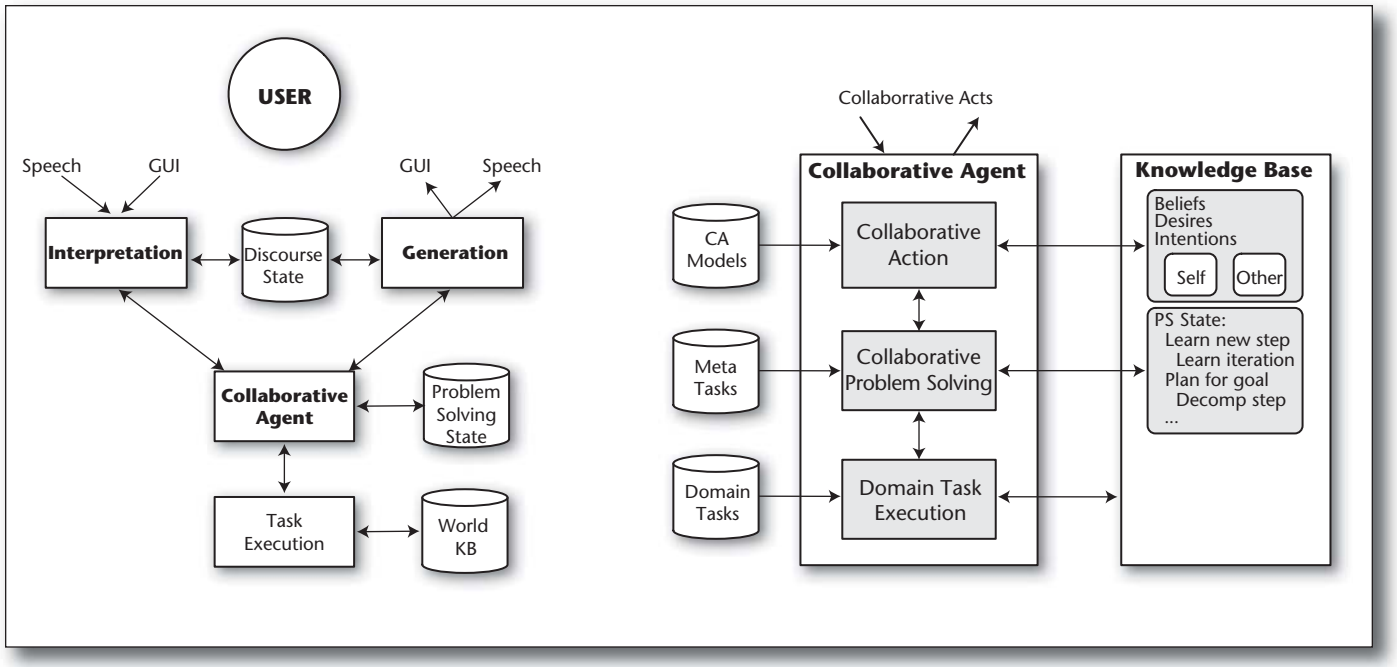

Figure 1. Architecture of a Collaborative Dialogue Agent, and Detail of the Collaborative Agent Component.

ting to performing a task). It does this by requesting performance of collaborative acts by the Generation components, which will eventually realize the acts as linguistic (or other) communication.

The middle subcomponent, labeled "Collaborative Problem Solving," is responsible for the system's overall problem-solving behavior. That is, it maintains the state involved in performing a task, or learning a new task, or planning to perform a task, and so on. Execution of these metatasks generates collaborative goals that result in communication as described above.

Finally, the subcomponent labeled "Domain Task Execution" stands for the various services that the system can perform on its own during the course of collaborative problem solving. It is crucial, for reasons described in the next section, that these tasks be explicitly represented so that the system can reason about them.

Task Representation for Collaboration

As shown in figure 1, several components of our system are driven by task models. Collaborative action is driven by models of collaborative acts such as agreeing on something or making joint commitments. Collaborative problem solving is driven by models of abstract metatasks such as performing a task, planning a task, learning a task, and so on; and domain task execution involves executable models of tasks in the domain, for example, filling out a web form, booking a meeting room, or commanding a robot.

To support the many things people do during collaboration, such as communication, planning, execution, and learning, we are developing a representation of tasks with the following features:

The ability to represent partial knowledge about tasks: Most task or plan formalisms assume that one is representing "complete" knowledge of how to do things, much like how a program completely characterizes its execution. But during collaborative problem solving, one often encounters partial knowledge about tasks. For example, while learning how to buy a computer, the system may have learned that choosing a model, finding the best price, and completing a requisition are all necessary subtasks but may not yet know the precise ordering and other constraints between the steps. Or it might be able to analyze the state of its learned task and realize that something is missing between two of the learned steps. Partial knowledge about tasks is not limited to learning. During collaborative execution, users will typically give only partial descriptions of what they want to get done. The system must be able to represent these partial tasks and then reason with them, for example, matching the partial description

against its task knowledge to determine how to help.

The ability to represent the knowledge requirements of tasks: The system needs to know what it needs to know in order to perform a task successfully. Note that these are different from the parameters of a task, although knowing the value of a parameter may be a knowledge requirement. These knowledge requirements drive the system to find information as needed during the performance of a task. In a collaborative setting, knowledge requirements often involve other agents, for example, that need to agree on the value of a task parameter.

The ability to represent tasks at different levels of abstraction: Most tasks that one encounters in practice are specializations of more abstract tasks. For example, buying a plane ticket is a specialization of purchasing something, which is itself a specialization of moving or obtaining something. Representing these abstraction/specialization relationships is crucial if task knowledge is to be applied in novel circumstances. This is particularly true for abstracting, generalizing, and transferring learned knowledge to related tasks.

The ability to represent execution of tasks by agents: In particular, collaborative tasks necessarily involve multiple agents. Representing and being able to reason about the roles agents can play in tasks is crucial for effective collaboration.

Suitable for use in interpreting and generating natural language and multimodal interaction: The representation must support communication about the task and task-related explanations, such as question-answering. It should support the modes of reference typical in language-based descriptions for language-based learning and for generating descriptions of known tasks. This means, for example, explicit representation of and ability to reason about the roles played by objects in tasks. This requirement is somewhat specific to our belief in the role of natural language in effective collaboration, but any reasonably expressive interface is going to require something similar.

The representation we are developing treats tasks as objects that gather together a set of assertions about related events and propositions. Space precludes a detailed description here, but it currently provides a vocabulary for asserting the following types of information: (1) knowledge about how tasks specialize each other; (2) knowledge about how objects get used in tasks; (3) knowledge about causing events to occur, the conditions under which events happen, and the effects of events happening; and (4) knowledge about how higher-level events are broken down into lower-level ones, under some constraints.

Tasks are treated as descriptions, not programs. However under certain conditions they can be translated into programs in a given programming language for execution. They can also, however, be directly executed themselves, or an agent can reason about its behavior on the fly.

Integrated Problem Solving

We have argued that collaborative interaction with people requires an integrated approach to planning, execution, learning, and the many other things people do while working together. To this end, the subcomponent labeled "Collaborative Problem Solving" in figure 1 is based on abstract models of these metatasks. The Collaborative Agent pursues goals such as learning a new task collaboratively or collaboratively executing a known task at this level. These tasks involve the performance of domain-level tasks, such as learning to extract a specific item of knowledge from a web page or looking up some information in a database.

These metalevel models (and indeed the task models at all levels) serve two purposes. The first role is the standard one in BDI systems, namely guiding the agent's behavior, whether collaborative or not. For example, to plan what to do there would be an explicit model of planning that built up complex task descriptions (plans) using the domain-level task models. While this metalevel is typically not explicit in stand-alone, autonomous AI systems such as planners, the more cognitively inspired AI architectures (such as Soar [Rosenbloom, Laird, and Newell 1993]) do a similar thing. Even executing a task is represented by an explicit model of execution that allows the agent to reason about its own operation.

In the collaborative setting, the metatask models at the problem-solving level have a second, equally important role. This is that they allow the Collaborative Agent to form expectations about what might (or should) happen next. These expectations provide the task-level focus that is crucial for the interpretation of user actions or utterances. For example, if we are learning a task collaboratively, we might expect that the user is likely to either perform or describe the next step of the task or indicate that there are no more steps. If we are planning collaboratively, the task model would indicate that any agent that is involved in the task could add an action to the plan to achieve an open precondition or add a constraint to resolve a conflict, and so on.

This is similar to the metareasoning possible in principle in other BDI systems such as PRS (Georgeff and Lansky 1987) or SPARK (Morley and Myers 2004). To our knowledge, though, the metalevels in those systems have never been extensively developed in practice. Since these systems are concerned primarily with task execution, they generally have not considered the need to use the expectations of task and metatask models for interpretation of the actions of other agents.

Collaboration as Joint Activity

We follow (at least in spirit) the view of collaboration and collaborative activity as following from shared or joint intentions (Cohen and Levesque 1991; Levesque, Cohen, and Nunes 1990; Grosz and Sidner 1986; Grosz and Sidner 1990; Rao and Georgeff 1991; Rao and Georgeff 1992; Sadek 1992; Sadek and de Mori 1997). In particular, we believe that (1) an agent's behavior must be driven by its desires and its prior commitments, (2) a collaborative agent must be able to make commitments with other agents to achieve goals of mutual interest, and (3) a collaborative agent must be able to communicate with other agents to agree about items of mutual interest, including commitments.

Without dwelling on the details, we use a language based on that of Cohen and Levesque (1990a), with the following main modal operators:

(Bel a P): Agent a believes that P is true.

(Des a P): Agent a desires that P be true (there can be many sources of desires).

(Goal a P): Agent a is committed to bringing about that P be true.

(JGoal (a1 a2 ...) P): The given agents have the joint goal P (called a JPG in Cohen and Levesque [1990a]).

(MB (a1 a2 ...) P): The given agents mutually believe that P is true.

This language is used to represent the content of user and system utterances. It is also used in task models, for example, to specify knowledge requirements. The system is able to reason about beliefs, desires, goals, and the rest, although our current implementation does not use a full-blown modal logic theorem prover (but we are considering the possibility).

Initiative and Collaboration

How do the pieces we have described come together to form a collaborative assistant? And how does mixed-initiative interaction arise? We will illustrate both user and system initiative, and the overall operation of the system, with an extended example. In the example, the system is designed to help a user with everyday office tasks, such as purchasing equipment and supplies. All of the functionality required for the example has been implemented, although some parts of the current prototype have not been updated to reflect the newest aspects of the model described in this article.

User Initiative

Let's start with an example of user initiative, since this is what a traditional dialogue system would support. Suppose the user says to the system: "I want to purchase an LCD projector for my class." Without dwelling on the internal details of interpretation, it is worth observing that there are three possible interpretations of this utterance (in all three cases, PURCHASE123 will be newly defined to be a task of type Purchase, whose object is an LCD projector, PROJ123, and so on).

First, it could be a direct report of a want or need:

```
(inform USR SYS
  (Des USR (Done PURCHASE123)))
```

In this case, a suitable response might be "OK," and the fact about the user's desires would be recorded in the system's knowledge base.

Second, it could be a statement of a goal that the user is pursuing independently:

```
(inform USR SYS
  (Goal USR (Done PURCHASE123)))
```

A suitable response to this might be "Good luck with that," and again the system might record this information about the user's goals in the knowledge base.

Finally, it could be a proposal that this be adopted as a joint goal:

```
(propose USR SYS
  (JGoal (SYS USR) (Done PURCHASE123)))
```

This is the interpretation that drives collaboration.

These interpretations are evaluated using the same model that drives the system's own communicative behavior. That is, the Collaborative Agent evaluates the user's utterances by considering whether it would have performed a similar act, given the current state. The Interpretation subsystem will decide among the possible interpretations, using dialogue context, user model, reasoner support, and heuristics such as preferring the collaborative interpretation whenever possible on the grounds that the system is an assistant.

The mechanism for handling user proposals is, in general, to determine how the proposed action can be integrated into the current tasks at the problem-solving level. For this example,

we have to assume the existence of a top-level task that permits new goals to be accommodated as subtasks. We whimsically call this task BE-HELPFUL. Note that the explicit representation of this task allows the system to (1) know that it is being helpful; and (2) reason about what is involved in being helpful. The definition of BE-HELPFUL provides that if the system adopts the goal, then not only is performing the purchase acceptable from its perspective, but also the system is now committed to performing the joint task.

In deciding whether to adopt the proposed goal, the system needs to be able to reason about the suitability of what the user has proposed to determine whether or not it makes sense before taking it on as a goal. Our approach for proposals of actions is to check whether the system knows a task model matching the proposed action (which, of course, is likely only partially specified). A more sophisticated model for BE-HELPFUL would allow us to handle more complicated examples, such as the following. Suppose that the user requests the LCD projector at 9:00 a.m. Then at 5:00 p.m. she requests one again. Given that LCD projectors are purchased very rarely, it may be that the system should double-check with the user as to whether she wants to buy a second projector or whether she simply forgot that she already asked the system to get it. This would be the sort of proactive help that one would expect from a good assistant that understands your intentions.

Note that while this reasoning is proceeding, the dialogue components are not necessarily idle. For example, the Generation subsystem knows from purely dialogue principles that the system has an obligation to respond to the user's proposal. It can therefore generate appropriate communicative behaviors (for example, taking and holding the turn with aural or visual gestures) even in the absence of the content of the response. And of course, crucial to supporting natural user initiative, if the user continues and offers further information or an additional proposal, the Interpretation components can start processing it asynchronously. This may result in the addition of information that would affect the Collaborative Agent's behavior, perhaps even changing its response to the original utterance.

To wrap up the example, eventually the system adopts the joint goal and the dialogue subsystem can generate an acceptance of the user's proposal (for example, "Ok, let's do that"). The system's overall behavior is now driven by the need to achieve the joint goal of (eventually) purchasing an LCD projector.

System Initiative

If the user initiative case seems simple, the system initiative case shows off our model more completely. Let's suppose that the system has the goal of purchasing an LCD projector, whether because the user proposed doing so or by some other means (perhaps our user and her system are responding to a projector order from elsewhere).

Needing to Collaborate

First, the task model for purchasing needs to represent the knowledge requirements of purchasing. One approach might have the Collaborative Agent incrementally deciding what needs to be agreed as it works on achieving the goal of purchasing the projector. Another approach would use general principles and ontological knowledge to "precompile" the decisions about collaboration. Or in some domains, it might even make sense to simply encode what are the "necessarily joint" elements of the solution. Finally, it may be that what needs to be agreed is specifically what needs to be agreed, a possibility that we do consider and are implementing.

Regardless of the details, let's assume that as part of the performance of PURCHASE123, the system needs to agree with the user about the budget for the purchase. This requirement is represented as:

```
(MB (SYS USR)
 (KnowRef (SYS USR)
  (the BUDGET of PURCHASE123)))
```

That is, the knowledge requirement is that the user and the system must have a mutual belief regarding the identity of the budget of our previously committed-to purchasing.[2] The Collaborative Agent adopts the goal of achieving this mutual belief and starts working to achieve it.

Collaborative Behavior

The Collaborative Action subcomponent reasons about collaborative goals (like MB and JGoal) and how to achieve them through dialogue. To do this, it relies on the BDI state of the system as represented in its KB and a set of task models treated as reactive procedures. Given the above goal, the Collaborative Agent would perform the following procedure:

1. If there is a mutually believed value for the budget, then the goal has been achieved.

2. Otherwise, if the system believes it has already committed to a value for the budget, it will inform the user of that.

3. Otherwise, if the system desires some value for the budget, then it will propose that to the user.

4. Otherwise, if it believes that the user desires some value, then it will check that with the user.

5. Otherwise, so far as the system believes, neither party has a preference, so it will ask the user (other strategies are possible).

This procedure is essentially a compiled version of the definitions of the speech acts such as in (Cohen and Levesque 1990a; Cohen and Levesque 1990b). Interestingly, we also use these same procedures in reverse for recognition of user intent during interpretation.

In this example, let's assume that the first three queries fail and that the system decides to ask the user about the budget. The collaborative act will be sent to the Generation subsystem and (eventually) realized as something like "What is the budget of the purchase?" The Collaborative Agent has now done all that it can do towards this goal, so it suspends work on that goal pending new circumstances (other strategies are possible, depending on the system's knowledge of the situation and of the user).

User Proposals

Suppose the user responds to the system's question with: "Fifteen hundred dollars." Skipping the details of interpretation, which would include, for example, using the fact that the system just asked a question about the budget and checking that $1500 could conceivably be the value of the budget property of PURCHASE123, it arrives at the following interpretation (glossing the representational details):

```
(propose USR SYS
 (JGoal (USR SYS)
  (choose (PURCHASE123 budget $1500)))
```

The standard semantics of the propose act are twofold: (1) the speaker desires the content of the act be performed, in this case making the budget of the purchase be $1500 and (2) the speaker will commit to this if the hearer will also. The choose action corresponds to the agent updating its mental state so as to make the indicated property true. The fact that it is a (proposed) JGoal means that both agents must make the same choice (if accepted).

As with the user-initiative case described previously, the system must decide whether to accept or reject the proposal. Regardless of whether it asked for a value or whether the user proposed something on his or her own, the system needs to reason about whether this value is acceptable. In many cases this is different from the initial test that it is a coherent proposal. For example, although $1500 is a perfectly reasonable thing to propose for the budget, the system might know that there is only $1000 available and that it should thus reject the proposal (with that explanation).

Reaching Agreement

Assume for purposes of the example that the system accepts the proposal and agrees for its part to make the budget $1500. The Collaborative Agent generates an accept act, which would be realized as something like "OK."

The crucial next step is to observe that when the Collaborative Agent next executes, it will notice that there is joint agreement as to the identity of the budget. That is:

```
(MB (SYS USER)
 (the BUDGET of PURCHASE123))
```

will be true thanks to the semantics of KnowRef and MB. This subgoal is therefore marked as achieved. If, for example, the PURCHASE123 task was blocked, it might now be able to proceed. Furthermore, note that additional knowledge may have been asserted to the knowledge base during the interaction, either because of extended interactions or during the interpretation process itself. The state in which the agent resumes PURCHASE123 may be quite different from that when it was suspended, even beyond knowing the value of the budget.

Related Work

This work is based on a long tradition of research in AI and computational linguistics. The semantics of speech acts and the relation to intentions is derived from Cohen and Perrault (1979) and Allen and Perrault (1980). The logic of intentions and commitment is loosely based on Cohen and Levesque (1990a). The challenge for us has been to apply these principles in a practical system that supports natural language dialogue.

Basing interagent collaboration on joint commitments is key to the Shared Plans formalism (Grosz and Sidner 1986; Grosz and Sidner 1990). Collagen (Rich and Sidner 1998) builds on Shared Plans and implements a collaborative assistant that performs actions with and for a user of an on-screen computer application. Rich and Sidner refer to Collagen as an application-independent "collaboration manager," which corresponds to our view of the separate Collaboration component of the mixed-initiative dialogue subsystem. They also emphasize that it is left to the underlying "black box" agent to actually make decisions, corresponding to our separation between collaborative dialogue manager and Task Manager, although it is somewhat unclear exactly what is communicated between the levels in Collagen. There are some differences between our approaches. We have concentrated on the problems of interpreting natural language in

practical dialogue and, in particular, how the same knowledge that drives collaboration can be used to interpret the user's input. The Collagen approach (based on Lochbaum [1991]) to "discourse interpretation" is something that we separate into BDI reasoning (which may involve domain- or task-specific reasoning).

Driving dialogue behavior from models of rational behavior is also proposed by Sadek (Bretier and Sadek 1996; Sadek et al. 1996; Sadek, Bretier, and Panaget 1997). The specific application that is described involves very simple question-answering dialogue on specific topics. It is hard to tell whether their deductive approach would work well in practice for more general collaboration. We imagine that in less-constrained situations there would be difficulties similar to those we face in trying to handle true mixed-initiative problem-solving dialogue. Finally, another deductive approach to collaborative dialogue is the recent STAPLE system (Subramanian, Kumar, and Cohen 2006), based directly on Joint Intention Theory. We are confident that the principles involved are very similar to our own. As with the previous comparison, any differences are likely to come down to the practicalities of natural language dialogue.

Discussion

Several issues come up repeatedly in discussion of mixed-initiative systems. In this section we describe how our approach addresses each of these in turn.

Task Allocation

On the issue of task allocation and division of responsibility between human and system, there are two crucial points to make. First, task allocation is treated as a matter of agreeing on the allocation of responsibility and then jointly committing to successful performance of the tasks. The process of agreement occurs naturally in dialogue. The joint commitment and its underlying formal basis forces the agent to behave properly whether it can do its part or not. In fact, it will drive the agent to try and do another agent's part if it feels this is necessary and to tell its collaborators if it knows it cannot do its part. The second point is that any division of responsibility must be dynamic and flexible, able to be discussed and renegotiated at any time. Of course different agents will be able to do different things. But keeping the task specifications separate from the capabilities of the agents who perform them will allow the tasks to be performed again by other combinations of agents or under different conditions.

Control

On the issue of controlling the shift of initiative and proactive behavior, our approach leaves this to emerge from the interaction between the agents. The human and the system operate asynchronously and in parallel. Communicative initiative is driven by the system's "need to know," that is, by the knowledge requirements of what it needs to do. Agreeing on something with the user (asking the user, telling the user, clarifying with the user, and so on) is another kind of task that the system performs in service of its goals. Proactive behavior ("system initiative") is a natural consequence of the system being goal-driven rather than simply reacting to user utterances and actions. When desirable, the "level" of initiative can be adjusted by designing the system to have more or less of the knowledge required by the tasks.

Awareness

The issue of maintaining shared awareness between human and system is in some sense the guiding principle of our approach. Communication and dialogue are about maintaining this shared state (and exploiting it for effective interaction). Agreement (or mutual belief) is often necessary in order for a task to succeed. Joint commitments between system and user drive the system's behavior. Communicative acts are performed to update and maintain the shared beliefs.

Communication

On the issue of communication protocols between human and system, we are primarily concerned with spoken natural language interaction. As we have said before, we find it unlikely that any other interface is going to be flexible and expressive enough to support intuitive interaction with a reasonably capable system. Dealing with language also forces us to confront a variety of representation and inference challenges, which, again, would have come up sooner or later anyway. Having said this, we do believe that our model of interaction as collaboration is more broadly applicable. For the approach to be useful, however, the interface must meet two requirements. First, to support interpretation, the context displayed or implied by the interface must be made explicit and available for use by the Interpretation and Collaboration components. For example, for a graphical interface, the interface must explicitly represent what is visually salient (rather than simply rendering it), what information is being communicated (rather than just having it in a data structure associated with a widget), and what are the ontological relationships between various elements (rather than their being simply tokens or labels). Second, the actions permitted by the interface must be expressed in terms of communicative acts with semantically meaningful content (rather than simply being tied to programmed callbacks). These two requirements taken together allow modalities other than natural language to be used for collaboration. As a bonus, if the interface is designed with these properties, natural language could be used in place of the interface if desired or required.

Evaluation

On the issue of evaluation of mixed-initiative and collaborative systems, we feel that this is an important area for further work. Evaluation of mixed-initiative systems has always been challenging, in part because users are unpredictable (note that this is a feature not a bug). For several reasons, we have concentrated on end-to-end or task-based measures of system performance. On the one hand, poor performance by any given component (for example, speech recognition) might be compensated for by another. On the other hand, stellar performance by a single component (for example, a machine-learning algorithm)

is not going to translate into user satisfaction unless the results can be integrated with the rest of the system's knowledge and effectively communicated to the user. Supporting iterative and "drill-down" behaviors has often been observed to be more important to users than pure computational prowess (which they take for granted these days). Of course end-to-end systems may not be feasible when faced with limited human or financial resources. One solution we endorse is to build connections to other groups with complementary interests and rely on clear, well-founded representations to serve as interlingua. But in general, as mixed-initiative systems become more common, the community is going to need to figure out how to evaluate such systems and communicate their utility to potential users.

Architecture

On the issue of architectural principles for mixed-initiative systems, our position should be clear by now. We feel that the right system architecture is crucial to the specification and development of collaborative agents. One important aspect of our approach is that by representing the system's core competencies as tasks at the metalevel, we can discuss, modify, and improve any aspect of the system's performance. As McCarthy put it at the dawn of AI, "all aspects of behavior except the most routine must be improvable" (McCarthy 1959).

Personalization

Finally, on the issue of personalization and adaptation to the user's preferences, we believe that this is crucial to effective collaboration. Such adaptation will not only make the interaction more enjoyable, it will also allow the system to produce solutions more quickly that are more likely to be acceptable or desirable to the user. We have not done much principled work in this area ourselves. However, we strongly believe that there is not a single magical set of "preferences" that the system ought to need to know. Naive translation of English sentences about likes and dislikes is unlikely to produce intuitive results when applied to AI algorithms. Better to approach the problem by trying to understand how people solve problems and then design the algorithms and systems to understand those techniques (even if they don't use them themselves). Then customization becomes meta-knowledge about problem solving (perhaps similar to "advice" [Myers 1996]).

Conclusions

We have described an architecture and representations for a collaborative agent. The system's behavior with respect to its commitments is driven by a formal model of collaboration based on a theory of joint intention. The model of collaboration and its role in the overall dialogue system is entirely application- and domain-independent. It does depend on reasoning that may be specific to the problems at hand, but in our opinion this is the right place for such specialization. The system takes an integrated view of the many capabilities required for collaboration, such as reasoning, communication, planning, execution, and learning. We believe that the architecture described in this article is a practical way to develop collaborative assistants based on knowledge-based systems. We are currently applying these ideas in several domains including personal health care, command and control of agent teams, office assistants, and several artificial domains used to explore further the use of mixed-initiative systems for collaborative problem solving.

Acknowledgements

This article is an extended and revised version of a paper originally presented at the AAAI 2006 Fall Symposium on Mixed-Initiative Problem-Solving Assistants. We thank the participants of that symposium for helpful comments and discussion, particularly Chuck Rich and Candy Sidner. Phil Cohen helped develop aspects of the role and representation of joint intentions. Phil Michaluk and Hyuckchul Jung contributed to the development of the representation of tasks. This material is based upon work supported by DARPA/SRI International subcontract #03-000223 and from the National Science Foundation grant #IIS-0328811. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the above-named organizations.

Notes

1. In our previous work, the component labeled "Collaborative Agent" was referred to as the "Behavioral Agent." The new name reflects the emphasis on collaboration and the component's central role in managing collaborative behavior, as described in this article.

2. For discussion of KnowRef, see Allen (1979), Moore (1985), and Morgenstern (1991). For this example we have also used a syntax similar to that of KM (Clark and Porter 1997) for this article, although the real thing is more complex.

References

Allen, J. 1979. A Plan-based Approach to Speech Act Recognition. Ph.D. dissertation, Deptartment of Computer Science, University of Toronto. Technical Report 131/79, Toronto, ON.

Allen, J., and Ferguson, G. 2002. Human-machine collaborative planning. Paper presented at the Third International NASA Workshop on Planning and Scheduling for Space, Houston, TX, October 27–29.

Allen, J.; Byron, D.; Dzikovska, M.; Ferguson, G.; Galescu, L.; and Stent, A. 2001. Towards Conversational Human-Computer Interaction. *AI Magazine* 22(4): 27–38.

Allen, J.; Ferguson, G.; and Stent, A. 2001. An Architecture for More Realistic Conversational Systems. In *Proceedings of Intelligent User Interfaces 2001 (IUI-01)*, 1–8. New York: Association for Computing Machinery.

Allen, J. F.; and Perrault, C. R. 1980. Analyzing Intention in Utterances. *Artificial Intelligence* 15(3): 143–178.

Austin, J. L. 1962. *How to Do Things with Words*. Cambridge, MA: Harvard University Press.

Bretier, P., and Sadek, D. 1996. A Rational Agent as the Kernel of a Cooperative Spoken Dialogue System: Implementing a Logical Theory of Interaction. In *Proceedings of the ECAI-96 Workshop on Agent Theories, Architectures, and Languages*, 189–204. Berlin: Springer.

Clark, P., and Porter, B. 1997. Building Concept Representations from Reusable Components. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-97)*, 369–376. Menlo Park, CA: AAAI Press.

Cohen, P. R., and Levesque, H. J. 1990a. Intention Is Choice with Commitment. *Artificial Intelligence* 42(2–3): 213–361.

Cohen, P. R., and Levesque, H. J. 1990b. Performatives in a Rationally Based Speech Act Theory. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, 79–88. New York: Association for Computing Machinery.

Cohen, P. R., and Levesque, H. J. 1991. Teamwork. *Nous* 25(4): 487–512.

Cohen, P. R., and Perrault, C. R. 1979. Elements of a Plan-Based Theory of Speech Acts. *Cognitive Science* 3: 177–212.

Ferguson, G., and Allen, J. 1998. TRIPS: An Intelligent Integrated Problem-Solving Assistant. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 567–573. Menlo Park, CA: AAAI Press.

Ferguson, G., and Allen, J. 2005. Mixed-Initiative Dialogue Systems for Collaborative Problem-Solving. In Mixed-Initiative Problem-Solving Assistants: Papers from the 2005 AAAI Fall Symposium, ed. D. Aha and G. Tecuci, 57–62. Technical Report FS-05-07, Association for the Advancement of Artificial Intelligence, Menlo Park, CA.

Ferguson, G.; Allen, J.; and Miller, B. 1996. TRAINS-95: Towards a Mixed-Initiative Planning Assistant. In *Proceedings of the Third Conference on Artificial Intelligence Planning Systems (AIPS-96)*, 70–77. Menlo Park, CA: AAAI Press.

Georgeff, M. P., and Lansky, A. L. 1987. Reactive Reasoning and Planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, 667–682. Menlo Park, CA: AAAI Press.

Grosz, B. J., and Sidner, C. L. 1986. Attention, Intention, and the Structure of Discourse. *Computational Linguistics* 12(3): 175–204.

Grosz, B. J., and Sidner, C. L. 1990. Plans for Discourse. In *Intentions in Communication*, ed. P. Cohen, J. Morgan, and M. Pollack, 417–444. Cambridge, MA: The MIT Press.

Horvitz, E. 1999. Uncertainty, Action, and Interaction: In Pursuit of Mixed-Initiative Computing. *IEEE Intelligent Systems* 14(5): 17–20.

Levesque, H. J.; Cohen, P. R.; and Nunes, J. H. T. 1990. On Acting Together. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, 94–99. Menlo Park, CA: AAAI Press.

Lochbaum, K. E. 1991. An Algorithm for Plan Recognition in Collaborative Discourse. In *Proceedings of the Twenty-Ninth Annual Meeting of the Association for Computational Linguistics (ACL-91)*, 33–38. East Stroudsburg, PA: Association for Computational Linguistics.

McCarthy, J. 1959. Programs with Common Sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, 75–91. London: Her Majesty's Stationary Office.

Moore, R. C. 1985. A Formal Theory of Knowledge and Action. In *Formal Theories of the Commonsense World,* ed. J. Hobbs and R. Moore, 319–358. Norwood, NJ: Ablex Publishing.

Morgenstern, L. 1991. Knowledge and the Frame Problem. *International Journal of Expert Systems* 3(4): 309–343.

Morley, D., and Myers, K. 2004. The SPARK Agent Framework. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-04)*, 712–719. New York: Association for Computing Machinery.

Myers, K. 1996. Strategic Advice for Hierarchical Planners. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR96)*, 112–123. San Francisco: Morgan Kaufmann Publishers.

Rao, A., and Georgeff, M. 1991. Modeling Rational Agents within a BDI-Architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR-91),* ed. J. Allen, R. Fikes, and E. Sandewall, 473–484. San Francisco: Morgan Kaufmann Publishers.

Rao, A. S., and Georgeff, M. P. 1992. An Abstract Architecture for Rational Agents. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR-92*), ed. B. Nebel, C. Rich, and W. Swartout, 439–449. San Francisco: Morgan Kaufmann Publishers.

Rich, C., and Sidner, C. L. 1998. COLLAGEN: A Collaboration Manager for Software Interface Agents. *User Modeling and User-Adapted Interaction* 8(3/4): 315–350.

Rosenbloom, P. S.; Laird, J. E.; and Newell, A., eds. 1993. *The Soar Papers: Research on Integrated Intelligence*. Cambridge, MA: The MIT Press.

Sadek, D., and de Mori, R. 1997. Dialogue Systems. In *Spoken Dialogue with Computers,* ed. R. de Mori. Boston: Academic Press.

Sadek, M.; Ferrieux, A.; Cozannet, A.; Bretier, P.; Panaget, F.; and Simonin, J. 1996. Effective Human-Computer Cooperative Spoken Dialogue: The AGS Demonstrator. In *Proceedings of the Fourth International Conference on Spoken Language Processing (ICSLP-96)*, volume 1, 546–549. Newark, DE: University of Delaware.

Sadek, M.; Bretier, B.; and Panaget, F. 1997. Artimis: Natural Dialogue Meets Rational Agency. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, 1030–1035. San Francisco: Morgan Kaufmann Publishers.

Sadek, M. D. 1992. A Study in the Logic of Intention. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR-92)*, ed. B. Nebel, C. Rich, and W. Swartout, 462–473. San Francisco: Morgan Kaufmann Publishers.

Searle, J. R. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge, UK: Cambridge University Press.

Subramanian, R. A.; Kumar, S.; and Cohen, P. R. 2006. Integrating Joint Intention Theory, Belief Reasoning, and Communicative Action for Generating Team-Oriented Dialogue. In Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06), 1501–1506. Menlo Park, CA: AAAI Press.

George Ferguson is a research scientist in the Department of Computer Science at the University of Rochester. He has more than a dozen years experience in the design and implementation of knowledge- and language-based systems in a number of domains and is a primary architect of the TRIPS conversational assistant system. He is the founding chair of the AAAI Intelligent Systems Demonstrations program and was program chair of the 2004 National Conference on Artificial Intelligence (AAAI-2004).

James Allen is the John H. Dessauer Professor of Computer Science at the University of Rochester. His research interests span a range of issues covering natural language understanding, discourse, knowledge representation, commonsense reasoning, and planning. He is the author of *Natural Language Understanding,* 2nd ed. (Benjamin Cummings, 1995), editor of *Reasoning About Plans* (Morgan Kaufmann, 1991), and coeditor of *Readings in Planning* (Morgan Kaufmann, 1990). He is a Fellow of AAAI.