TRIPS: An Integrated Intelligent Problem-Solving Assistant

George Ferguson and **James F. Allen**
Department of Computer Science
University of Rochester
Rochester, NY 14627-0226
{ferguson,james}@cs.rochester.edu

Abstract

We discuss what constitutes an integrated system in AI, and why AI researchers should be interested in building and studying them. Taking integrated systems to be ones that integrate a variety of components in order to perform some task from start to finish, we believe that such systems (a) allow us to better ground our theoretical work in actual tasks, and (b) provide an opportunity for much-needed evaluation based on task performance. We describe one particular integrated system we have developed that supports spoken-language dialogue to collaboratively solve planning problems. We discuss how the integrated system provides key advantages for helping both our work in natural language dialogue processing and in interactive planning and problem solving, and consider the opportunities such an approach affords for the future.

Content areas: AI systems, natural language understanding, planning and control, problem solving, user interfaces

Introduction

It is an interesting time to be an AI researcher. Computer speeds and capacities have increased to the point that tasks that used to take days can now be done in seconds. The pervasive role of computers in everyday life has emphasized the need for people and computers to co-exist and, one hopes, complement rather than hinder each other. Within the field of AI itself, we now have a solid understanding of many of the core issues. AI concepts like search, planning, learning, natural language, and so on have mature theoretical underpinnings and extensive practical histories. In many cases we understand just how hard some of these problems are to solve, and in these cases we often have a good understanding of how approximate solutions can be found and what the tradeoffs are in using them.

Supported by these two trends, it is now possible to build AI systems that actually perform tasks that people find difficult (and that require intelligence, however exactly you define it) in reasonable amounts of time. A good example is Deep Blue (Hamilton & Garber 1997), which, despite some controversy, certainly counts as a highly successful, working AI system that leverages both of the trends described above. But on a smaller scale, researchers are starting to report on experiences with building AI systems that solve some task from start to finish, whether it is answering questions using a database, large-scale planning and scheduling of shipments, or autonomous robots and robotic assistants.

This paper argues first for the utility of building end-to-end systems that integrate a variety of capabilities in the performance of some task. We discuss some of the dimensions along which such integration can take place, and how the different aspects of integrated systems interact in the design, implementation, and operation of the system. We then describe in detail one particular implemented, integrated system: TRIPS, the Rochester Interactive Planner System. This system integrates speech recognition, natural language understanding, discourse processing, planning and plan recognition, and much more in order to provide the human user with an interactive, intelligent problem-solving assistant in a transportation/logistics domain. TRIPS is fully functional and will be demonstrated at AAAI in the Intelligent Systems Demonstrations program.

Integrated Systems

We start with what it means to be an integrated AI system. The AI part seems straightforward: the system should perform some task or tasks that people find to require intelligence. As for integration, we feel that there are two important dimensions to consider. First, the system can integrate the functionality of multiple more specialized components in the performance of the task. In an integrated AI system, these components are things like planners, natural language parsers, learning algorithms, speech recognizers, temporal reasoners, logic engines, and so on. These are AI technologies that do not themselves solve a task, but that provide services necessary to do so. The second, less obvious, dimension to consider is the integration between the AI system and the person or people using it. Most, if not all, fielded AI systems have to interact with people at some time. It is in our interests to make this interaction as natural for the people as possible, both to increase the acceptance of the AI technology and to leverage people's skills in the performance of the task. So an integrated system, to us, is one that both integrates multiple capabilities in service of a task and integrates its functions with those of its human users.

Why then should AI researchers want to build integrated AI systems? The first benefit is that it gets us to the "criti-

cal mass" of an end-to-end system, rather than toy programs in toy domains. This has two huge advantages. First, the task grounds the research. For example, natural language understanding has to be *about* something, planning has to be *for* something, and so on. Second, end-to-end performance allows evaluation using task-based metrics. AI systems are notoriously difficult to evaluate in isolation—what does it mean for an NLU system to "understand" a sentence? Task-based metrics allow us to compare different approaches, components, or architectures. They also allow us to compare the system's performance to that of people on the same task, which is surely the ultimate test of an intelligent system.

The second benefit of building integrated AI systems is less clearcut but just as significant. The need to integrate disparate components often forces us to broaden our perspective, add functionality, increase robustness, and so on. This makes one-off, *ad hoc* solutions less likely, since it is unlikely that the same shortcuts will serve different components equally well. For example, a planner might prefer not to reason about time, but an NLU system might absolutely require it. Integration requires a clear understanding of the representations being used and clear specification of component interfaces, both of which are issues that the AI community is well-positioned to address.

Finally then, if integrated AI systems are desirable research goals, how do we build them? We cannot be definitive on this point, of course. But one methodological principle has been useful in our research, to be described below. This is the need to clearly define the task that the system is to perform. Without a crisp task definition and a definition of what it means to "do better," evaluation is impossible. Further, the task should support a range of incrementally more difficult problems, allowing us to bootstrap the system-building process by building a simple system for a simple task, then increasing the complexity of the task and improving the system. Task-based evaluation throughout provides a guide to what is working and what isn't. A corollary of this is that we have to choose hard enough tasks and big enough problems. Integrated systems using current AI technologies are already, in fact, capable of performing some very challenging tasks.

TRIPS: The Rochester Interactive Planning System

We next describe a particular integrated AI system that we have developed based on the previous observations. TRIPS, the Rochester Interactive Planning System, is the latest in a series of prototype collaborative planning assistants. Our research plan is to design and build a series of progressively more sophisticated systems working in progressively more realistic domains. TRIPS builds on our experiences with the TRAINS system (Ferguson *et al.* 1996; Allen *et al.* 1995; Ferguson, Allen, & Miller 1996), but (a) functions in a more complicated logistics domain compared to TRAINS' simple route-planning domain, (b) supports the construction of much more complex plans than TRAINS could produce or understand, and (c) embodies a more complex model of collaborative problem-solving than TRAINS. All three of these developments required significant analysis, redesign, and reimplementation (the extension of the domain *per se* was perhaps the most straightforward part). TRIPS does build directly on TRAINS, however, reusing many of the components and all of the infrastructure directly.

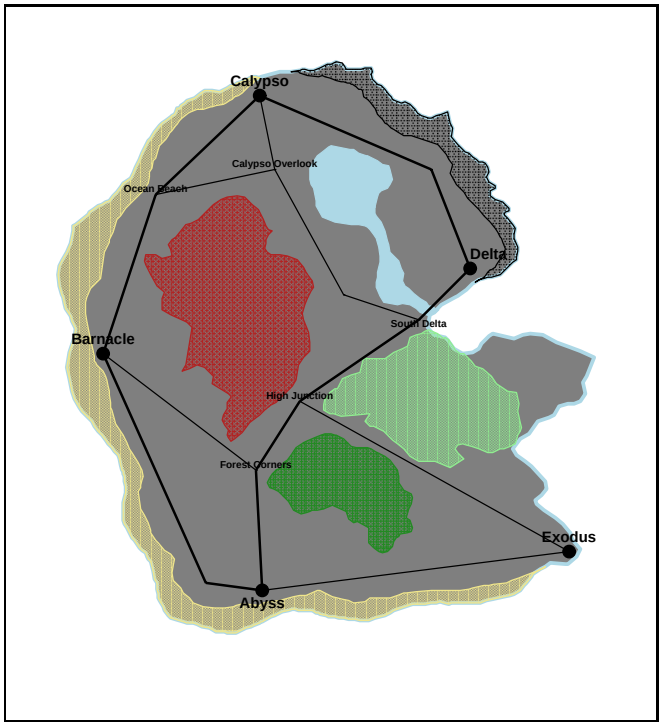

Figure 1: Map of Pacifica

TRIPS is an assistant to a human manager. The two of them collaborate to construct plans in crisis situations. Our example scenario involves the island of Pacifica (see Figure 1), which is in the path of an approaching hurricane. Because of the hurricane, the population must be evacuated to one of the cities (the "safe" city) and from there presumably to safety. The manager's task is to plan the evacuation, using a variety of vehicles (with varying capabilities) at his or her disposal. There may be a variety of constraints placed on the final plans, such as time, cost, weather-related factors, and so on. The important features of this scenario as a research environment are: (a) it is clear what the task is and when it is completed (*i.e.*, we can easily describe tasks to people, they can know when they're done, and we can verify that); (b) the relative quality of the solution is easily measured (in terms of time, violated constraints, or some measure of robustness, for example); and (c) it is easy to vary the complexity of the task from trivially solvable to problems that people would have difficulty solving and, in fact, that TRIPS could not solve on its own (*e.g.*, increase numbers of people, change numbers and types of vehicles, modify road network, and so on). The first two aspects make evaluation possible, the final aspect lets us work incrementally on progressively harder problems.

0:00 H: Hello.

0:02 S: Hi there.

0:03 H: Show me a map of Pacifica.

0:07 S: Ok. *Displays large map of the island of Pacifica*

0:08 H: Where are the people?

0:11 S: There are two groups of people at Exodus, two at Calypso, two at Barnacle, and one group of people at Abyss. *Displays a bar chart showing the numbers of people in the cities. This chart is updated as the conversation progresses to show the state of the world given the plan under consideration.*

0:17 H: And where are the transports?

0:19 S: There are two trucks and one helicopter at Delta. *Displays a chart showing vehicle status and contents (also updated through the conversation)*

0:24 H: Use a truck to get the people from Calypso to Delta.

0:31 S: Your wish is my command. *Displays a construction window with a timeline view of the plan under development, with the vehicles on the vertical axis and time horizontally. The map is also updated to show the route taken to Calypso and back.*

0:34 H: How long will that take?

0:37 S: It will take six hours and forty-two minutes.

0:43 H: What if we went along the coast instead?

0:47 S: That option would take ten hours and twenty-seven minutes. *Displays another construction window showing the option alongside the original plan.*

0:53 H: Forget it.

0:55 S: Ok. *Deletes second construction window from the display.*

1:05 H: Use the other truck to get the people from Exodus to Delta.

1:13 S: Ok. *Adds the new task to the original construction window, using a different color, and updates the map. In this case, the task is clearly seen to take more than ten hours.*

1:22 H: Let's use the helicopter instead.

1:34 S: Ok. *Changes the construction window view of this task and updates the map display.*

1:38 H: Now send truck one to Barnacle.

1:41 S: Your wish is my command. *Adds a third task to the construction window (in a third color) and updates the map.*

1:44 H: Pick up the people there.

1:48 S: Ok. *Updates displays*

1:50 H: Go on to Abyss.

1:54 S: Ok. *Updates displays*

1:57 H: And take the people there to Delta.

2:05 S: Ok. *Updates displays*

2:10 H: That looks good.

2:12 S: Glad to be of service.

We now suppose that the manager learns of a change in road status from an external source, and informs the system so that the plan can be adjusted. Delta Bridge crosses the inlet between Delta and South Delta.

2:18 H: Delta bridge is out.

2:21 S: Ok. The highlighted actions may need to be revised. *Flashes Delta Bridge in red, and highlights actions whose routes use the bridge in red in the construction window.*

2:37 H: Send truck two along the coast instead.

2:39 S: Ok. *Displays new route (to Calypso and back) on the map and updates the construction window (longer trips).*

2:42 H: Have the other truck stop at South Delta instead.

3:02 S: Ok. *Display new route (along coast to Barnacle, then to Abyss, then to South Delta) and updates the construction window.*

3:06 H: Unload the people there.

3:13 S: Ok. *Updates displays*

3:18 H: And use the helicopter to get the people from South Delta to Delta.

4:03 S: No problem. *Multiple flights with the helicopter are added to the third task in the construction window and the map is updated to show them.*

Figure 2: Example Dialogue

Example Dialogue

An example dialogue is presented in Figure 2. This is a verbatim transcript of an actual session, annotated to describe the graphical displays generated by TRIPS. The times shown are wall time—the actual time of each utterance. The exact times are not important,[1] but note that most utterances are understood within a few seconds, including speech recognition and all processing. This level of responsiveness is required to maintain an effective dialogue with the user.

TRIPS System Architecture

The organization of TRIPS is shown in Figure 3. Modules communicate by exchanging KQML messages (Finin *et al.* 1993) using our own central message-passing Input Manager (not shown in the figure). Most modules are in fact separate Unix processes, and TRIPS can run on any combination of machines that can run the individual modules. The TRIPS infrastructure allows any program that can read standard input and write standard output to exchange messages.

[1] This session was run using two Sun Ultra 1 workstations with 167 MHz SPARC processors.

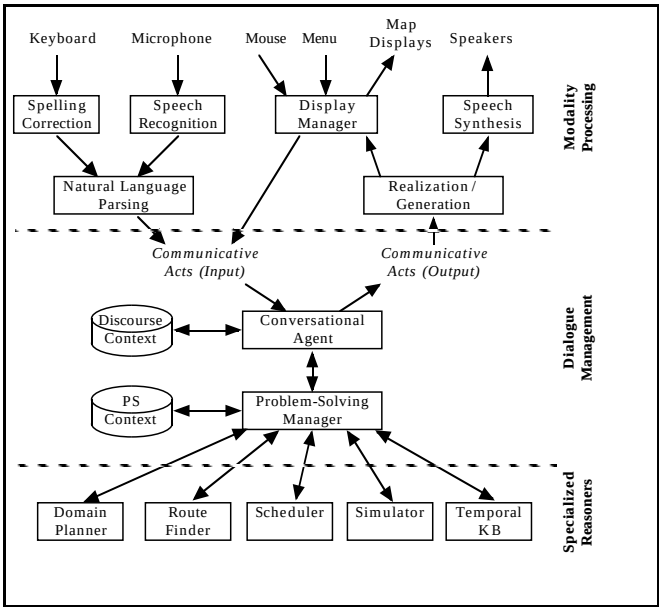

Figure 3: TRIPS System Architecture

As shown in the figure, the components of TRIPS can be divided into three groups:

1. **Modality Processing**: This includes speech recognition and generation, graphical displays and gestures, typed input, and so on. All modalities are treated uniformly. For input, words and gestures are parsed into meaning representations based on treating them as communicative acts. For output, communicative acts generated by the system are realized using speech or graphics.
2. **Dialogue Management**: These components are the core of TRIPS, and are responsible for managing the ongoing conversation, interpreting user communication in context, requesting and coordinating specialized reasoners to address the needs of the conversation, and selecting what communicative actions to perform in response.
3. **Specialized Reasoners**: These components provide the "brains" of TRIPS, in the sense of being able to solve hard problems such as planning courses of actions, scheduling sets of events, or simulating the execution of plans. The goal here is to provide a form of plug-and-play interoperability, where new or improved specialized reasoners (including, for example, network-based sources or agents) can be easily added to the suite of resources at TRIPS' disposal.

Space obviously precludes discussing all these components here. The robust speech recognition work is described in (Ringger & Allen 1996b; 1996a). Descriptions of the approach to robust language understanding can be found in (Allen *et al.* 1996). The approach to planning, which is interesting for its emphasis on plan modification and its use of an expressive, temporal world model, is described in (Ferguson & Allen 1998). The rest of this section will concentrate on the dialogue management components of TRIPS, namely the Conversational Agent and the Problem Solving Manager, since it is these that really effect the integration in TRIPS. In both cases, the contexts shown in the figure support the incremental specification and development so necessary for effective integration.

Interaction as Conversation

The Conversational Agent coordinates all system activity as it interacts with the user. The key idea for integrating various different input and output modalities is that all user interaction is viewed as *communicative acts*, a generalization of speech acts. As a consequence, all communication between the Conversational Agent and the modality processing modules is in terms of possible communicative acts that have been or should be performed. TRIPS supports a wide range of speech acts, ranging from direct requests (*e.g., show me the map*), questions (*where are the transports?*), assertions (*The bridge is out*), suggestions (*Let's use a helicopter instead*), acceptances and rejections (*ok*, *no*), as well as a range of social acts including thanks, apologies, and greetings. In addition, there are a limited range of gestural acts such as pointing and dragging screen objects using the mouse. The modality processing modules typically produce a set of possible *surface acts* based on the form of the act. The Conversational Agent then combines these interpretations with the discourse context in order to determine the *intended acts*, and then in coordination with the problem solving manager, determines the system's response, which again is expressed in terms of communicative acts.

Since the task in TRIPS is interactive problem solving, most of the acts that are performed relate to different problem solving operations. The operations most common in TRIPS include introducing a new task or subtask to the plan, modifying or deleting an existing task, defining all or part of a solution for a task, modifying an existing solution, or evaluating all or part of the plan. As well, as is common in human-human problem-solving, either the person or the system can create alternate solutions (options) for comparison purposes, remove an option from consideration, or adopt a particular option. Note that the problem solving operation involved is orthogonal to the communicative act used to communicate it. For example, one might request a modification to a plan, suggest a modification, accept or reject a modification, or promise to do a modification.

The Conversational Agent is driven by a set of rules that identify possible interpretations intended by the user and plan a system response for each. These interpretation/response pairs are then ranked and the system's response is selected. Currently, this selection process is based on a static ranking of the strength and reliability of the rules. In the future, more complex deliberation processes will be introduced that allow the system to generate a wider range of responses, including taking the initiative in the conversation if desired.

Integration as Collaborative Problem-Solving

While the Conversational Agent coordinates the interpretation of communicative acts and chooses system responses, it does not have direct knowledge of the task or current state of

the problem solving process. This information is handled by the Problem Solving Manager, which maintains an abstract representation of the task and the current solution (or solutions) under consideration, and coordinates reasoning by the specialized reasoners when necessary. The key idea supporting the PSM in this is a very general representation of plan-related information, including a hierarchical task structure, explicit representation of the possible solutions under consideration, and a temporal knowledge base that represents the world over time relative to different solutions.

The abstract plan representation is general purpose and is used for a wide range of purposes including: providing the context for recognition of the user's intentions, driving displays that summarize the state of the world and/or plan, answering queries about the plan, building the commands and context required by the specialized reasoners, and integrating and managing the results from all the specialized reasoners. Note that while it supports a wide range of purposes, this representation is not typically used directly in any planning process. Rather the PSM converts the general representation into specific commands to each specialized reasoner using a representation that that reasoner understands. It then converts the results returned back into the general representation for use by other components. A key feature of this representation is that it allows the system to represent and reason about plans that it could not have generated on its own. This allows the user to incrementally develop plans that are more complex than the TRIPS planner can generate.

To support the interpretation processes in the conversational agent, the PSM and conversational agent interact using a propose-evaluate-confirm protocol. The conversational agent suggests a possible problem solving action, say to modify a certain subtask by using a different vehicle. The PSM then evaluates this interpretation based on *coherence* (*i.e.*, did this operation make sense in the current problem solving context) and *feasibility* (*i.e.*, could the requested operation be performed). Based on these evaluations, the CA chooses a particular set of interpretations and then informs the PSM so it can update its context for the next interaction. In order to evaluate each interpretation, the PSM may invoke the specialized reasoners as necessary to perform the actual reasoning required.

Consider a brief example from the sample dialogue, when the user says *Let's use the helicopter instead*. From its surface form, this utterance is interpreted as a suggestion to use the helicopter in some unspecified part of the plan instead of some other (also unspecified) vehicle. The CA uses coherence heuristics to explore the most likely possibility: that the modification should apply to the last subtask being discussed, namely getting the people from Exodus to Delta. It asks the PSM to evaluate modifying this plan by replacing something with the helicopter. The PSM uses its abstract representation of the plan to find likely objects that the helicopter could replace in this task, in this case the truck mentioned in the previous interaction. It then calls the Planner with the task to replace this truck with the helicopter in this task. The Planner performs this operation and returns a revised plan. This plan, however, is quite different from the previous solution. Rather than making one trip with the truck, the plan now involves making two trips with the helicopter (due to different vehicle capacities). The PSM then calls the Router to instantiate routes for the helicopter for its two trips, and then invokes the Scheduler to produce a nominal plan that can be used to generate a display. Since all these operations were performed successfully, the PSM returns to the CA an evaluation saying that this interpretation ranks high on both coherence and feasibility. The CA has no other viable options, and so notifies the PSM that this is the interpretation selected. The PSM updates its problem solving state to reflect the new plan, and updates the world model used to drive the plan display. The CA executes the response corresponding to this interpretation, causing display updates and a spoken confirmation.

Related Work

Few integrated systems have been constructed that have the robustness and depth of TRIPS. In most other work, either the interface to the system is highly constrained, or the task that the system performs is highly constrained. For example, most speech-based query systems (such as the ATIS systems (DARPA 1989–1991)) may handle a wide range of questions but only do one thing—answer queries about a trip. The system does not have to explicitly reason about what needs to be done as the task is fixed in advance, and remains constant during the interaction. Nevertheless, such systems do fit the criteria we laid out for an integrated system to the extent that they truly handle a task that a person would have in obtaining travel information.

Other systems provide a richer back end task but limit the interface. The Circuit Fix-it Shop (Smith, Hipp, & Biermann 1995), for instance, handles a complex task of diagnosing and repairing circuits, but the user must use one of a fixed set of sentences. The COLLAGEN system (Rich & Sidner 1997) involves an architecture with some strong similarities to our approach, but the interface is quite constraining and cumbersome for a person to use as it does not support natural language, and only supports plan modification by chronological backtracking. In both of these systems however, as in TRIPS, treating the interaction as a *dialogue* between the system and the user provides the *context* required to integrate system functions and coordinate system behaviours.

In fact, one of the earliest integrated systems is still one of richest to date. This is the Basic Agent (Vere & Bickmore 1990), in which a person could interact in natural language with a submarine robot in a simulated world. This system lacked a compelling task to be accomplished, however, and the interactions were more of the flavor of demonstrating the capabilities of the system. Nonetheless, the Basic Agent integrated a wide range of capabilities within a system that could be used effectively by a person.

Conclusions and Future Work

Integrated systems like TRIPS take time and effort to build. What have we gained from the experience?

First, we can evaluate performance by having people solve problems. In work on TRIPS' predecessor TRAINS, we developed a methodology for testing groups of naive

users solving a set of predefined problems (Sikorski & Allen 1996; Stent & Allen 1997). The results were encouraging (90% of sessions resulting in success), but these were mostly a reflection of the simplicity of the TRAINS task. We have not yet evaluated TRIPS on its more complex task and domain, but applying the same methodology, we expect to be able to show that people can solve problems faster with TRIPS than with another person. We hope to report those results in the near future. The point for this paper, however, is that this experiment could not even be conceived without an integrated, end-to-end system with which to work.

Second, the emphasis on integrating multiple specialized reasoners at the problem-solving level has had several benefits. The drive towards integration has resulted in a very general shared representation of plans, objectives, domain objects, and the like. This representation is used by a range of components from natural language understanding to planning and simulation. And of course, the integration of multiple specialized reasoners allows us to plug new technologies into TRIPS or interact with external agents. The key to making this more generally effective is the specification of component or agent *capabilities* so that meta-reasoners like the TRIPS PSM can task them and understand their responses. Specifying and using such capabilities is an active area of our current research.

Finally, the close, intuitive integration between person and computer in TRIPS has several benefits. Viewing the interaction as a conversation is far more natural for the person than learning arcane command languages or GUIs. This will translate into more effective performance with less training (as we hope to show in our evaluation experiments). The closely-integrated, mixed-initiative interaction is also easier for the computer, since it is possible for it to indicate when it can't solve a problem and literally ask for help. The generality of the representations described above ensures that the system can understand and use suggestions that it would never have come up with by itself. The emphasis for the specialized reasoners then changes from complete but impractical reasoning, to flexible and expressive but almost certainly incomplete forms of reasoning.

In the end, the whole, integrated system is greater than the sum of its parts. TRIPS allows the human and the system to collaboratively solve harder problems than either could solve on their own.

Acknowledgements

The TRIPS development team includes Eric K. Ringger, Lucian Galescu, Donna Byron, Amanda Stent, and Myroslava Dzikovska, in addition to the authors. Neal Lesh developed the TRIPS simulator. Further thanks are due to the "Big Picture" group at Rochester, especially Len Schubert, and to the members of the original TRAINS group from which TRIPS emerged.

TRIPS is funded in part by ARPA/Rome Laboratory contract no. F30602-95-I-1088, ONR grant no. N00014-95-I-1088, and NSF grant no. IRI-9623665.

References

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* 7:7–48.

Allen, J. F.; Miller, B. W.; Ringger, E. K.; ; and Sikorski, T. 1996. A robust system for natural spoken dialogue. In *Proceedings of the 1996 Annual Meeting of the Association for Computational Linguistics (ACL-96)*, 62–70.

DARPA. 1989–1991. *Proceedings of the DARPA Speech and Natural Language Workshops*, San Mateo, CA: Morgan Kaufmann.

Ferguson, G., and Allen, J. F. 1998. From planners to plan management via a hybrid planning architecture. To appear as a University of Rochester CS Dept. Technical Report.

Ferguson, G.; Allen, J.; and Miller, B. 1996. TRAINS-95: Towards a mixed-initiative planning assistant. In Drabble, B., ed., *Proceedings of the Third Conference on Artificial Intelligence Planning Systems (AIPS-96)*, 70–77.

Ferguson, G.; Allen, J. F.; Miller, B. W.; and Ringger, E. K. 1996. The design and implementation of the TRAINS-96 system: A prototype mixed-initiative planning assistant. TRAINS Technical Note 96-5, Department of Computer Science, University of Rochester, Rochester, NY.

Finin, T.; Weber, J.; Wiederhold, G.; Genesereth, M.; Fritzson, R.; McKay, D.; McGuire, J.; Pelavin, R.; Shapiro, S.; and Beck, C. 1993. Specification of the KQML agent-communication language. Draft.

Hamilton, S., and Garber, L. 1997. Deep blue's hardware-software synergy. *IEEE Computer* 30(10).

Rich, C., and Sidner, C. L. 1997. COLLAGEN: When agents collaborate with people. In *First International Conference on Autonomous Agents*, 284–291.

Ringger, E. K., and Allen, J. F. 1996a. Error correction via a post-processor for continuous speech recognition. In *Proceedings of the 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-96)*.

Ringger, E. K., and Allen, J. F. 1996b. A fertility channel model for post-correction of continuous speech recognition. In *Proceedings of the 1996 International Conference on Speech and Language Processing (ICSLP-96)*.

Sikorski, T., and Allen, J. 1996. A task-based evaluation of the TRAINS-95 dialogue system. In *Proceedings of the ECAI Workshop on Dialogue Processing in Spoken Language Systems*.

Smith, R. W.; Hipp, D. R.; and Biermann, A. W. 1995. An architecture for voice dialog systems based on prolog-styletheorem proving. *Computational Linguistics* 21(3):281–320.

Stent, A. J., and Allen, J. F. 1997. TRAINS-96 system evaluation. TRAINS Technical Note 97-1, Department of Computer Science, University of Rochester, Rochester, NY.

Vere, S., and Bickmore, T. 1990. A basic agent. *Computational Intelligence* 6(1):41–60.