User Modeling in Conversational Interfaces

Pat Langley

Department of Computer Science
University of Auckland
New Zealand

We thank Alfredo Gabaldon, Mehmet Goker, Benjamin Meadows, Ted Selker, Cynthia Thompson, and others for contributions to the research presented here.

Benefits of Conversational Interfaces

We typically interact with computers using keyboard and mice for input and screens for output.

Spoken-language interfaces offer clear advantages outside the home or office, say when driving or walking.

However, current dialogue systems are limited in that they:

- Offer limited forms of interaction with users
- Make limited inferences about users' mental states

We hold that *user models* – both short term and long term – have important roles to play in conversational interfaces.

Types of Task-Oriented Dialogue

The most useful dialogues are task related; we can identify at least four broad classes:

- Recommendation
- Command and control
- Mentoring and teaching
- Collaborative problem solving

Each category constrains the form of conversational interaction that one must support.

In this talk, I will discuss the first and third types of dialogue, focusing on inferring models of the user.

The Recommendation Task

A major class of interactive service involves *recommendation*:

- *Given:* A type of item the user wants to select (e.g., books);
- *Given:* Knowledge about items that are available;
- *Given:* Constraints about items the user cares about;
- *Find:* One or more items that satisfy these constraints.

Recommender systems have been available on the World Wide Web since the 1990s, but they are not *conversational*.

Such interactive variants have a very different feel from more traditional recommender systems.

Recommending Destinations

One service that people might find valuable is recommendation of destinations. We can state this task as:

- *Given:* Some type of destination the user wants to select;
- *Given:* Knowledge about available nearby destinations;
- *Given:* Constraints about destinations the user desires;
- *Find:* A few destinations that satisfy the constraints.

Destination advisors available on the Web are not suitable for driving or similar contexts.

At DaimlerBenz, we designed / implemented a conversational destination advisor for use in such settings.

System Vision (Basic Version)

Inquirer: Where do you think I should eat tonight?

Advisor: What type of food would you like?

Inquirer: What types are there?

Advisor: You can say things like Chinese, Indian, and Mediterranean.

Inquirer: Oh, maybe a French place with free parking.

Advisor: What quality rating would you like?

Inquirer: I don't care, as long as it's cheap.

Advisor: I'm sorry, I don't know any restaurants like that. Would you like to search for any price?

Inquirer: No, I think I'd like Chinese instead.

Advisor: In what city would you like to eat?

Inquirer: How about Berkeley?

Advisor: How does the Long Life Vegi House on 2129 University Avenue sound?

Inquirer: Sure, that sounds fine.

This interaction is effective but tedious because the advisor does not know the inquirer's tastes.

System Vision (Personalized Version)

Inquirer: Where do you think I should eat tonight?

Advisor: What type of food would you like?

Inquirer: Cheap Chinese.

Advisor: What city do you prefer?

Inquirer: How about Palo Alto?

Advisor: How does Jing Jing Szechuan Gourmet on 443 Emerson sound?

Inquirer: Sure, that sounds fine.

Once an advisor knows about the user's preferences, dialogues should be much shorter and faster.

The Adaptive Place Advisor

We implemented this approach in the *Adaptive Place Advisor* (Thompson, Goker, & Langley, 2004), which:

- carries out spoken conversations to help user refine goals;
- incorporates a dialogue model to constrain this process;
- collects and stores traces of interaction with the user; and
- personalizes both its questions and recommended items.

We focused on recommending restaurants about where the user might want to eat.

This approach is appropriate for drivers, but it also has broader applications.

Speech Acts for Adaptive Place Advisor

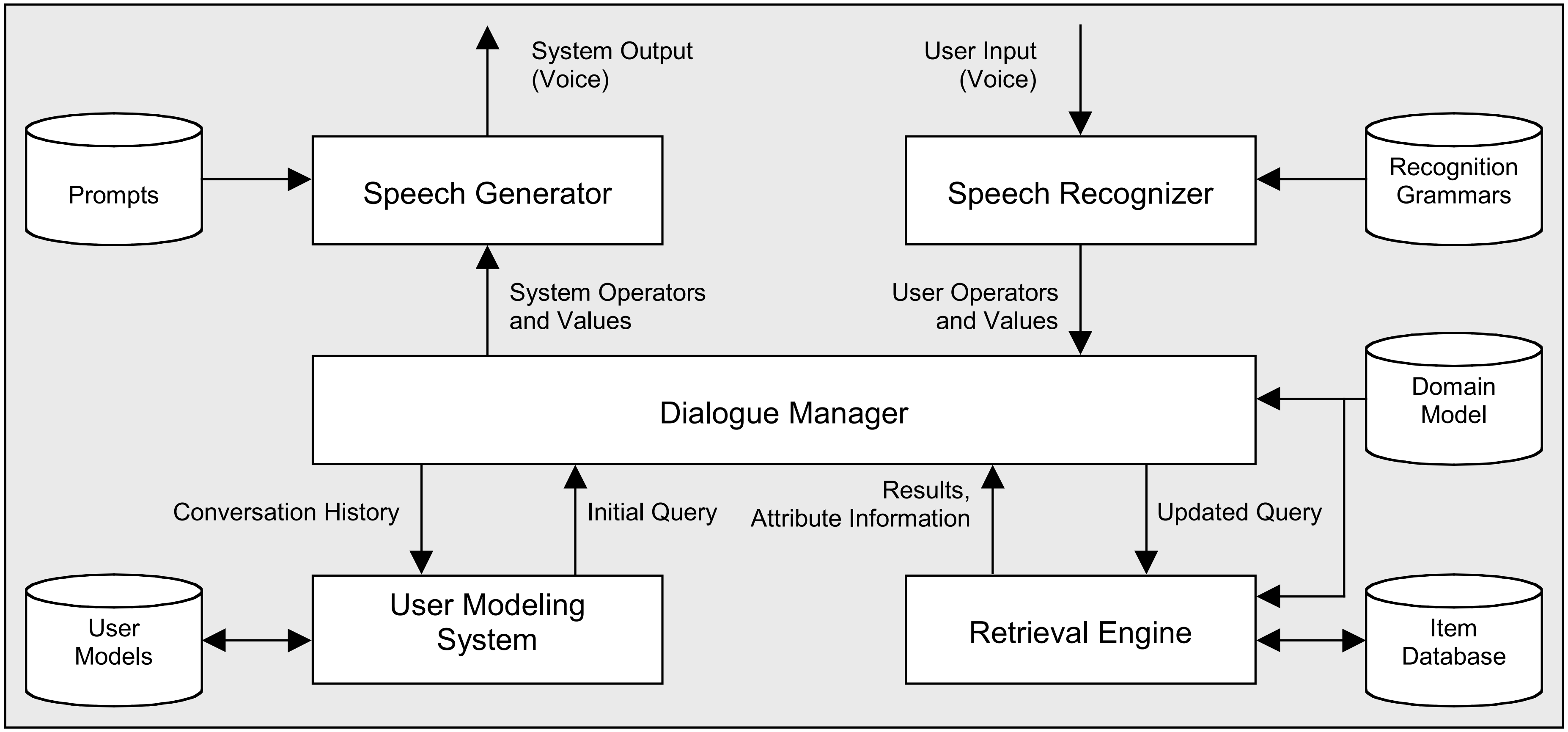
System Speech Acts

Ask-Constrain	Asks a question to obtain a value for an attribute
Ask-Relax	Asks a question to remove a value of an attribute
Suggest-Values	Suggests a small set of possible values for an attribute
Suggest-Attributes	Suggests a small set of unconstrained attributes
Recommend-Item	Recommends an item that satisfies the current constraints
Clarify	Asks a clarifying question if uncertain about latest user operator

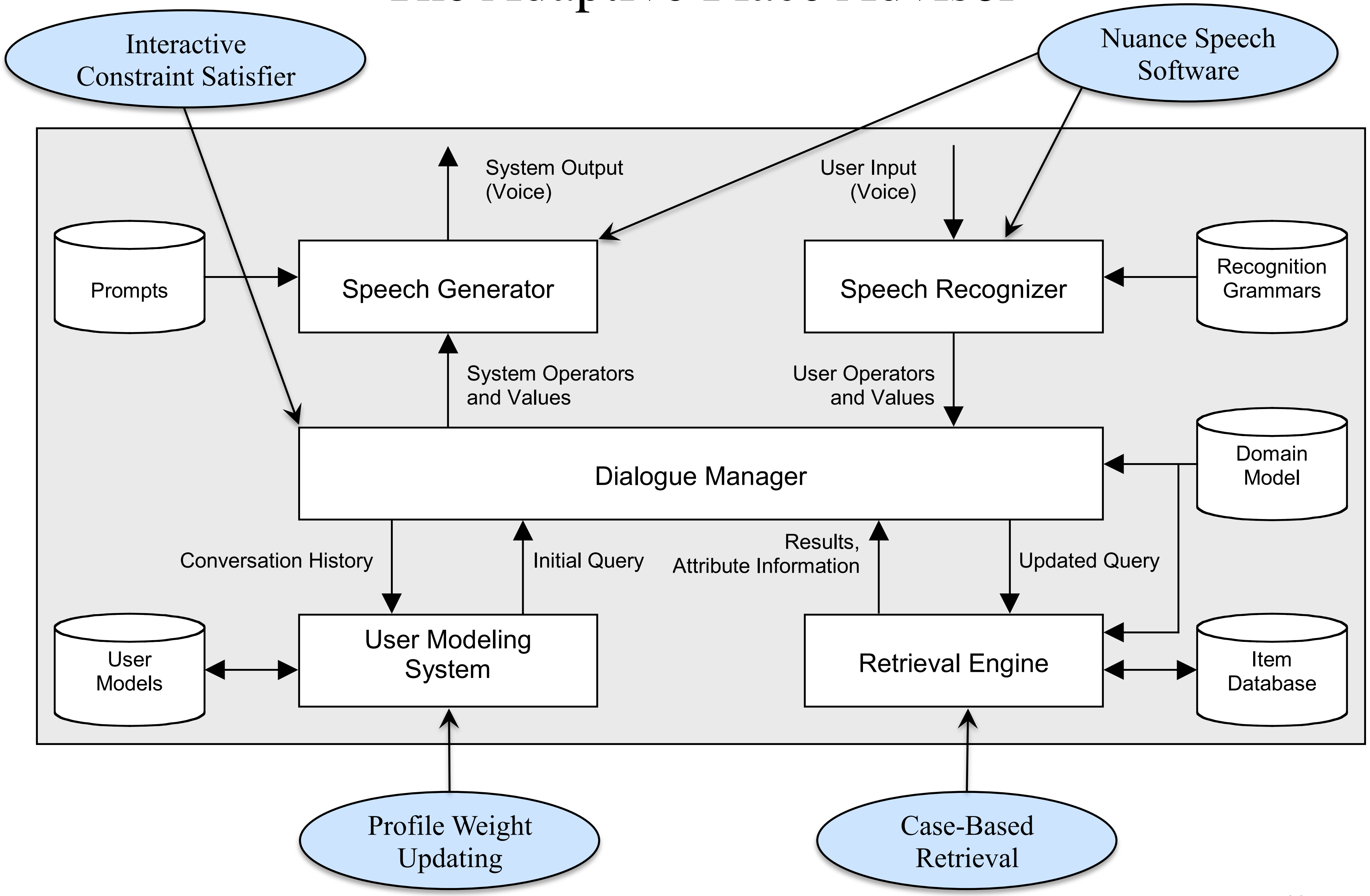
User Speech Acts

Provide-Constrain	Provides a value for an attribute
Reject-Constrain	Rejects the proposed attribute
Accept-Relax	Accepts the removal of an attribute value
Reject-Relax	Rejects the removal of an attribute value
Accept-Item	Accepts the proposed item
Reject-Item	Rejects the proposed item
Query-Attributes	Asks system for information about possible attributes
Query-Values	Asks system for information about possible attribute values
Start-Over	Asks the system to re-initialize the search
Quit	Asks the system to abort the search

The Adaptive Place Advisor



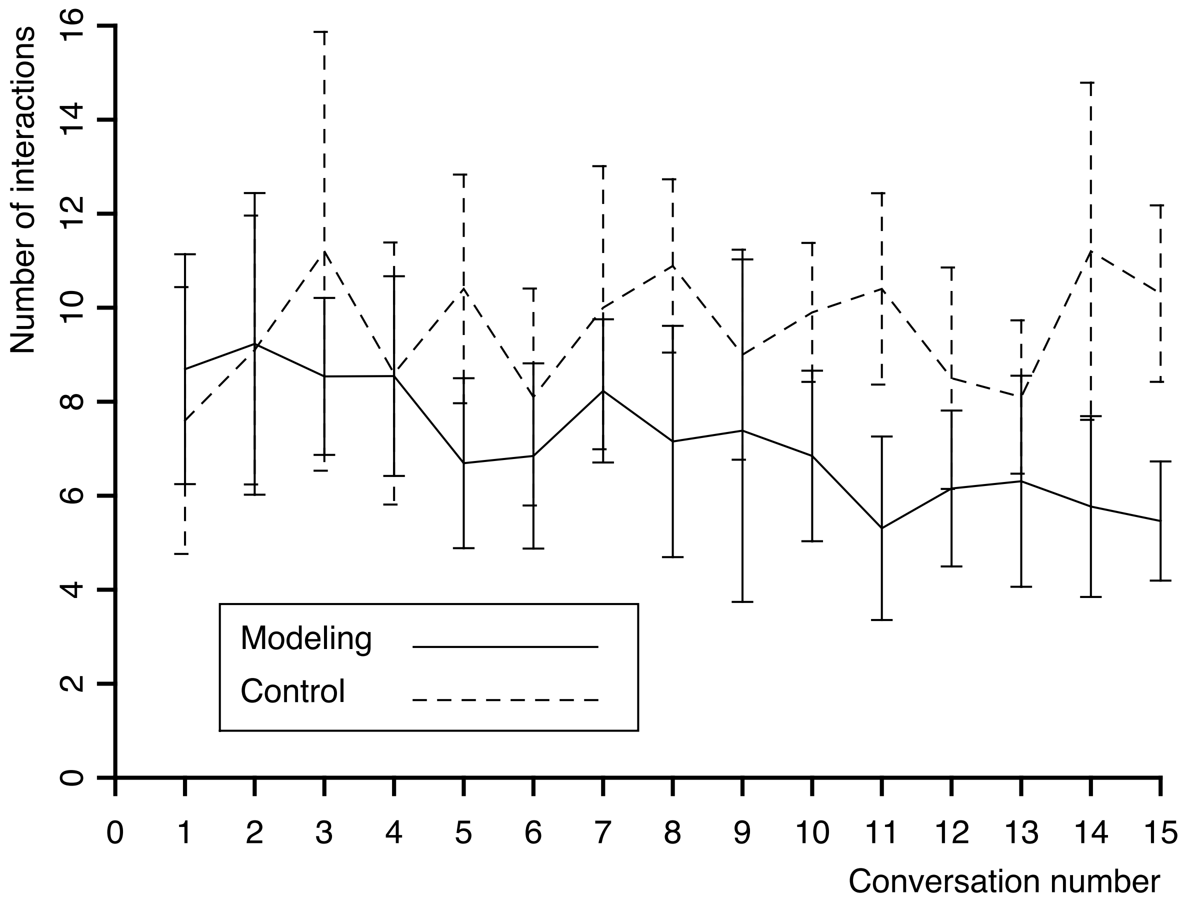
The Adaptive Place Advisor



Evaluation: User Interactions

We ran experiments to evaluate the Place Advisor with 13 subjects in the personalized condition and 11 subjects as controls.

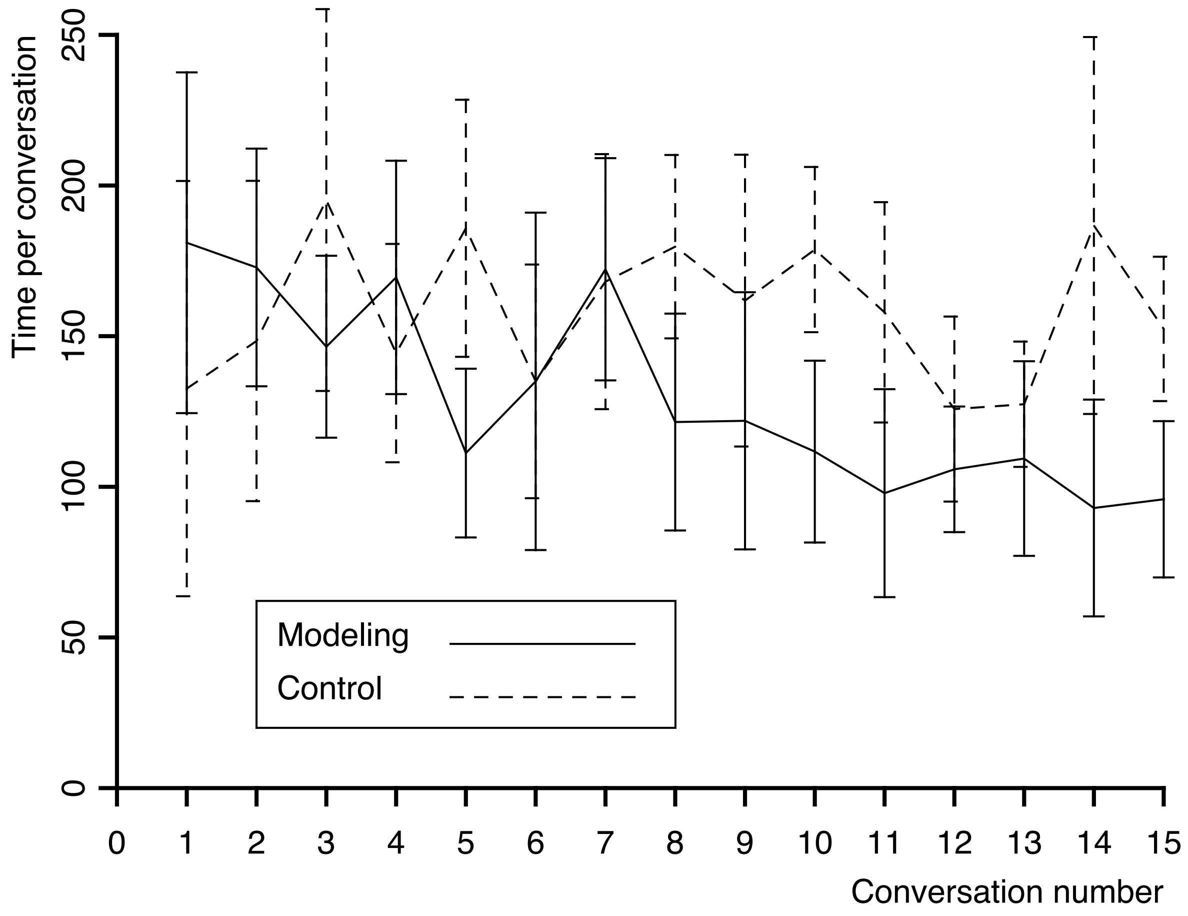
Over time, spoken interactions for the personalized subjects decreased substantially, as the system got to know the user.



Evaluation: Conversation Time

We also measured the total time for each recommendation session in both experimental conditions.

Again, this metric decreased as the system became familiar with the user, shortening conversations by about 50 percent.



Interim Summary

We have described an implemented approach to conversational recommendation that:

- Interactively constrains destination choices through dialogue
- Uses a personal profile to bias questions and recommendations
- Updates this profile based on interactions with the user

Experiments with the system have demonstrated that:

- Interaction constraint satisfaction is an effective technique
- Over time, personalization reduces interaction time / effort

Modern dialogue systems like Siri would benefit from including such forms of interaction.

Mentoring Systems

Another broad class of problems involves *mentoring*, which we can state as:

- *Given:* Some complex task the user wants to carry out;
- *Given:* Knowledge about this task's elements and structure;
- *Given:* Information about the user's beliefs and goals;
- *Generate:* Advice that helps the user complete the task.

As before, we can develop *conversational* mentoring systems; a common application is *help desks* (Aha et al., 2006).

Our CMU work has focused on medical scenarios, which have a more procedural character.

A Medic Assistant

One application focuses on scenarios in which a human medic helps injured teammates with system assistance:

- The medic has limited training but can provide situational information and affect the environment;
- The system has medical expertise, but cannot sense or alter the environment directly; it can only offer instructions;
- The medic and system collaborate to achieve their shared goal of helping the injured person.

We have developed a Web interface similar to a messaging app, but could replace it with a spoken-language interface.

Sample Dialogue* for Medic Assistant

M: We have a man injured!

A: Ok. What type of injury?

M: He' s bleeding.

A: How bad is the bleeding?

M: Pretty bad. I think it is the artery.

A: Ok. Where is the injury?

M: It' s on the left leg.

A: Apply pressure on the leg's pressure point.

M: Roger that.

A: Has the bleeding stopped?

M: No. He's still bleeding.

A: Ok. Apply a tourniquet.

M: Where do I put the tourniquet?

A: Just below the joint above the wound.

M: Ok. The bleeding stopped.

A: Good job.

M: medic A: advisor

* We provide the system with the meaning for each of M's utterances.

Some Observations

This sample dialogue, although simple, raises several issues:

- Behavior is *goal-directed* and involves communicative actions about *joint activity*;
- The participants develop *common ground* (Clark, 1996*)* about each other's *beliefs* and *goals;*
- Many beliefs and goals are not explicitly stated but rather *inferred* by conversing agents; and
- The process alternates between inference of mental states and executions goal-directed activities.

We can address these issues without tackling other challenges, such as speech recognition.

Representing Beliefs and Goals

Domain content: reified relations/events in the form of triples:

[inj1, type, injury]   [inj1, location, left_leg]

Speech acts: inform, acknowledge, question, accept, etc.

inform(medic, computer, [inj1, location, left_leg])

Dialogue-level predicates:

belief(medic, [inj1, type, injury], ts, te)

goal(medic, [inj1, status, stable], ts, te)

belief(medic, goal(computer, [inj1, status, stable], t1, t2), ts, te)

goal(medic, belief(computer, [inj1, location, torso], t1, t2), ts, te)

belief(computer, belief_if(medic, [inj1, location, torso], t1, t2), ts, te)

belief(computer, belief_wh(medic, [inj1, location], t1, t2), ts, te)

belief(computer,inform(medic,computer,[inj1,location,left_leg]),ts,te)

Representing Domain Knowledge

The framework assumes that domain knowledge includes:

- Conceptual rules that specify situational patterns with relational predicates
  - E.g., situations in which an injury threatens a patient’s life
- Skills that associate conditions, effects, and subskills with relational predicates.
  - E.g., courses of actions that may preserve the patient’s life

Both concepts and skills are organized into hierarchies, with complex structures defined in terms of simpler ones.

Representing Dialogue Knowledge

The framework assumes three kinds of dialogue knowledge:

- Speech-act rules that link belief/goal patterns with act type;
- Skills that specify domain-independent conditions, effects, subskills (e.g., a skill to communicate a command); and
- A dialogue grammar that states relations among speech acts (e.g., ‘question’ followed by ‘inform’ with suitable content).

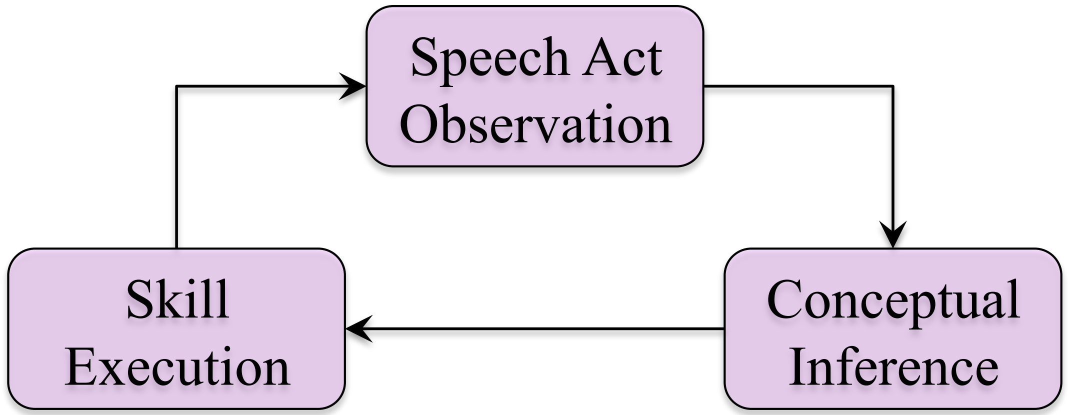
Together, these provide the background content needed to carry out high-level dialogues about joint activities.

Architectural Overview

The architecture operates in discrete cycles during which it:

- Observes new speech acts, including ones it generates itself;
- Uses inference to update beliefs/goals in working memory;
- Executes hierarchical skills to produce utterances based on this memory state.

At a high level, it operates in a manner similar to production-system architectures like Soar and ACT-R.



Dialogue Interpretation

The inference module accepts environmental input (speech acts and sensor values) and incrementally:

- Retrieves rules connected to working memory elements
- Uses factors like coherence to select some rule to apply
- Makes default assumptions about beliefs and goals as needed

This abductive process carries out heuristic search for coherent explanations of observed events.

The resulting inferences form a *situation model* that influence system behavior.

Dialogue Generation

On each architectural cycle, the hierarchical execution module:

- Selects a top-level goal that is currently unsatisfied;
- Finds a skill that should achieve the goal whose conditions match elements in the situation model;
- Selects a path downward through the skill hierarchy that ends in a primitive skill; and
- Executes this primitive skill in the external environment.

In the current setting, execution generates speech acts by filling in templates.

Evaluation of Dialogue Architecture

We have tested the dialogue architecture on three domains:

- Medic scenario: 30 domain predicates and 10 skills
- Elder assistance: six domain predicates and 16 skills
- Emergency calls: 16 domain predicates and 12 skills

Dialogue knowledge consists of about 60 rules that we held constant across domains.

Domain knowledge is held constant across test sequences of speech acts within each domain.

Results were encouraging but tests on a wider range of speech acts / dialogues would strengthen them.

Interim Summary

We have created an architecture for conversational mentors that:

- Cleanly separates domain-level from dialogue-level content;
- Integrates inference for situation understanding with execution for goal achievement;
- Utilizes these mechanisms to process both forms of content.

Experimental runs with the architecture have demonstrated that:

- Dialogue-level content works with different domain content;
- Inference and execution operate over both knowledge types.

These results suggest that the approach is worth pursuing further.

Concluding Remarks

Human-level dialogue processing involves a number of distinct cognitive abilities:

- Representing other agents' beliefs, goals, and preferences
- Understanding and generating speech acts
- Drawing domain-level and dialogue-level inferences
- Using utterances to alter others' mental states

Dialogue systems can operate without these facilities, but they are then poor imitations of human conversationalists.

Related References

- Aha, D., McSherry, D., & Yang, Q. (2006). Advances in conversational case-based reasoning. *Knowledge Engineering Review*, *20*, 247–254.
- Gabaldon, A., Langley, P., & Meadows, B. (2014). Integrating meta-level and domain-level knowledge for task-oriented dialogue. *Advances in Cognitive Systems*, *3*, 201–219.
- Gabaldon, A., Meadows, B., & Langley, P. (in press). Knowledge-guided interpretation and generation of task-oriented dialogue. In A. Raux, W. Minker, & I. Lane (Eds.), *Situated dialog in speech-based human-computer interaction*. Berlin: Springer.
- Langley, P., Meadows, B., Gabaldon, A., & Heald, R. (2014). Abductive understanding of dialogues about joint activities. *Interaction Studies*, *15*, 426–454.
- Thompson, C. A., Goker, M., & Langley, P. (2004). A personalized system for conversational recommendations. *Journal of Artificial Intelligence Research*, *21*, 393–428.