Knowledge-Guided Interpretation and Generation of Task-Oriented Dialogue

Alfredo Gabaldon, Ben Meadows, Pat Langley

Abstract In this paper, we present an architecture for task-oriented dialogue that integrates the processes of interpretation and generation. We analyze implemented systems based on this architecture – one for meeting support and another for assisting military medics – and discuss results obtained with the first. In closing, we review some related dialogue architectures and outline plans for future research.

1 Introduction

Systems that use natural language to assist a user in carrying out some task must interact with that user as execution of the task progresses. The system in turn must interpret the user's utterances and other environmental input to build a model of what both it and the user *believe* and *intend* – in regard to each other and the environment. The system also requires knowledge to use the model it constructs to participate in a dialogue with the user and support him in achieving his goals.

In this paper we report on two systems we have built for task-oriented dialogue and describe the architecture that underpins them. The architecture integrates two processes: *dialogue interpretation*, which builds an expanding model of the user's context in terms of their beliefs and goals, and *dialogue generation*, which uses this interpretation of the situation and background knowledge to help the user achieve his goals through a task-directed conversation. .

Alfredo Gabaldon
Silicon Valley Campus, Carnegie Mellon University, Moffett Field, CA 94035 USA
e-mail: alfredo.gabaldon@sv.cmu.edu

Ben Meadows
Department of Computer Science, University of Auckland, Auckland 1142, NZ
e-mail: bmea011@auckland.ac.nz

Pat Langley
Silicon Valley Campus, Carnegie Mellon University, Moffett Field, CA 94035 USA
e-mail: patrick.w.langley@gmail.com

In addition to integrating interpretation and generation, the architecture incorporates several other important features. Although we believe that domain-specific knowledge is essential in intelligent systems, we also believe that intelligent behavior relies on abstract *meta-level* knowledge that generalizes across different domains. In particular, we are interested in high-level aspects of dialogue: knowledge and strategies relevant to dialogue processing that are independent of the actual content of the conversation. The architecture separates domain-level from meta-level content, using both during interpretation and generation. The work we report is informed by cognitive systems research, a key feature of which is arguably integration and processing of knowledge at different levels of abstraction [7].

Another feature of our architecture is the incremental nature of its processes. We assume that dialogues occur within a changing environment and that the tasks to be accomplished are not predetermined but discerned as the dialogue proceeds. Our architecture incrementally expands its understanding of the situation and the user's goals, acts according to this understanding, and adapts to changes in the situation, sometimes choosing to pursue different goals. In other words, our architecture supports *situated* systems that carry out *goal-directed* dialogues to aid their users. In the next section we discuss two implemented prototypes that demonstrate this key functionality. We follow this with a detailed description in Section 3 of the underlying architecture and a discussion of results in Section 4. We conclude with comments on related work and plans for future research.

2 Application Systems for Task-Oriented Dialogue

In this section, we discuss two prototypes that incorporate our architecture as their dialogue engine. The first system facilitates cyber-physical meetings by interacting with humans and equipment; the second is an advisory system that collaborates with a military medic to address the mutual goal of treating a patient. In each case, we discuss the setting, the knowledge that drives behavior, and a sample dialogue that illustrates operation. Neither system interacts with users in spoken language, but each could be adapted to this end in a straightforward manner.

2.1 A System for Meeting Support

We have developed a system that supports 'cyber-physical' meetings in which a number of human users interact with each other and with a number of devices to pursue some joint activity. The specific scenario we examine here involves a session between a physical therapist, a patient, and a relative of the patient who have the shared aim of completing a physical therapy session. During the meeting, the users carry out a dialogue that includes the system as a participant. The users and system also interact with distributed sensors that measure leg motion, heart rate, and ambient light and temperature, as well as distributed actuators that include a 'smart' television and a text-to-speech generator.

Table 1 Sample dialogue involving a physical therapist (Doctor), a patient (John), and the meeting support system.

Doctor:	John, please do leg lifts.
Patient:	Okay. I will do leg lifts.
TV:	[starts playing tutorial video about leg lifting]
Sensor:	[sends movement signal to system]
System:	Great start John!
Sensor:	[sends movement signal to system]
	[after patient does a few more leg lifts . . .]
System:	You are doing great, John!
	[after a few more, the patient stops . . .]
Patient:	I can't do any more leg lifts.
Doctor:	It's okay, John. That was good.
System:	John, you did eight repetitions.
Patient:	How many did I do in the previous session?
System:	In the previous session you did five repetitions.

During a physical therapy session, users communicate by entering sentences through a menu-based smart-phone interface, although we intend to replace this with a spoken-language interface in future work. The current interface accepts sentences in a controlled version of English, translates them into a logic-like representation, and sends the result to the dialogue system for processing. At the same time, the sensing devices send messages to the dialogue system via `HTTP`. Each user can see the utterances of other users and those of the system via the phone interface.

The system has knowledge about activities that arise during a therapy session, including physical activities the patient can carry out, such as repeatedly lifting a leg, and therapist procedures, such as obtaining relevant patient information (from a database or the patient), measuring range of movement, and tracking heart rate and blood pressure. In a typical session, the system asks the patient standard questions and the therapist asks additional ones. Then the therapist decides on the task for the session and suggests it to the patient, relative, and system. If the patient or family member accepts the proposed task, the system updates its model of the current situation and proceeds accordingly. The system supports the patient's execution of a task by tracking his progress, instructing the television to show a tutorial video, and providing feedback. For instance, once sensor input reveals the patient has started doing an exercise, it might encourage him by saying "Great start!"

Specific components of the meeting support system include a menu-based interface on a smart phone to input English sentences, a phone application that serves as a motion detector; a television for displaying tutorials and other support videos, a heart-rate monitor, environmental sensors for temperature and lighting, an `HTTP` client/server module for component communication, and the dialogue system. Table 1 shows a sample dialogue for one of the physical therapy scenarios. In this case, the patient John participates in a session in which he partially complete a leg exercise under supervision of a therapist at a remote location. We will return to this case study in Section 4, where we examine it in more detail.

Table 2 Sample dialogue between a human medic and a more knowledgeable computer advisor.

Medic:	We have a man injured!
Advisor:	Okay. What type of injury?
Medic:	He's bleeding.
Advisor:	How bad is the bleeding?
Medic:	Pretty bad. I think it is an artery.
Advisor:	Okay. Where is the injury?
Medic:	It's on the left leg.
Advisor:	Apply pressure on the leg's pressure point.
Medic:	Roger that.
Advisor:	Has the bleeding stopped?
Medic:	No. He's still bleeding.
Advisor:	Okay. Apply a tourniquet.
Medic:	Where do I put the tourniquet?
Advisor:	Just below the joint above the wound.
Medic:	Okay. The bleeding has stopped.

2.2 A Medic Assistant

Our second prototype involves scenarios in which a military medic on the battlefield helps an injured teammate. Because the medic has limited training, he interacts with the dialogue system to get advice on treating the person; the system plays the role of a mentor with medical expertise. The medic and system collaborate towards achieving the shared goal of stabilizing the patient's medical condition. The system does not know the specific task in advance. Only after the conversation starts, and the medic provides relevant information, does the system act on this content and respond in ways that are appropriate to achieving the goal. The system does not effect change on the environment directly; the medic provides both sensors and effectors, with the system influencing him by giving instructions.

During an interaction, the system asks an initial sequence of questions that lead the medic to provide details about the nature of the injury. This sequence is not predetermined, in that later questions are influenced by the medic's responses to earlier ones. Table 2 shows a sample dialogue in which the medic-system team attempts to stabilize a person with a bleeding injury. The system possesses domain knowledge about how to treat different types of injuries, taking into account their location, severity, and other characteristics. The program can also adapt the treatment according to the medic's situation. For instance, it may try a different treatment for a wound if the medic claims that he cannot apply a particular treatment because he lacks the supplies necessary for that purpose.

This system uses a Web interface similar to a text-messaging application, although again we plan to replace this with a spoken dialogue module in the future. The medic types English sentences into a form element within the interface, which it sends to the dialogue system via an `HTTP` request. The system in turn sends the content to a natural language processor that translates into a logical form our system can interpret. We have used *Skyphrase* (`http://www.skyphrase.com`), a proprietary,

semantically-driven language interpeter, for this purpose. Skyphrase sends back the translation, again via `HTTP`, and the dialogue system uses the information to update its model of the current situation. This drives the system's continuing behavior in pursuing the joint task. Lack of space constrains our discussion of the system here, but we have provided further details elsewhere [6].

2.3 Discussion

Although the two systems we have just described are limited in many respects, developing them raised a number of new challenges that have not been fully addressed by existing dialogue systems:

- As the dialogue and the users' situation evolves, the system must update its account of the situation by interpreting new information as it arrives, including dialogue utterances and sensor inputs.
- The system's interaction with users is goal directed and involves joint activity over time; this activity includes not only actions carried out by the users, but also communicative actions and commands to device actuators.
- The system must establish a *common ground* [4] with its users, that is, it must develop a shared model of the environmental situation and others' mental states; it must also update this joint model as circumstances change and information flows among participants.
- The beliefs and goals of each participant are not stated explicitly, but the system must infer enough of them to be effective; this involves using not only domain-specific knowledge, but also more abstract knowledge that relates mental states to communication events.
- The overall process is highly dynamic, as the system continuously draws inferences from users' utterances and other input to expand its understanding of the evolving situation, and as it carries out activities to achieve goals as they arise.

Our application systems and architecture represent first steps towards addressing these challenges. In the next section we describe the integrated architecture, an implementation of which serves as the main component of the two systems above.

3 Agent Architecture

Now we can turn to our framework for task-oriented dialogue. We have focused on supporting goal-directed behavior that is physically situated in dynamic contexts. The architecture depends on a knowledge base that lets it generate inferences, introduce goals, and execute actions. Input is *multi-modal* in that it might come from speech, text, visual cues, or external sensors. We have implemented the architecture in Prolog, making use of its support for embedded structures and pattern matching, but its representation and control mechanisms diverge substantially from the default Prolog inference engine, as we will see shortly.

3.1 Representation and Content

As in research on cognitive architectures [9], we distinguish between a dynamic short-term or *working* memory, which stores external inputs and inferences based upon this information, and a more stable *long-term* memory, which serves as a store of knowledge that is used to make inferences and organize activities.

Working memory is a rapidly changing set of ground literals that contains the system's beliefs and goals as it models the evolving situation. Literals for domain-level content, which do not appear as top-level elements in working memory, are stored as relational triples, as in *[i1, type, injury]* or *[i1, severity, major]*. This reification lets the system examine and refer separately to different aspects of a single complex concept, including its predicate.

Our representation also incorporates meta-level predicates, divorced entirely from the domain level, to denote speech acts [1, 13]. The literature contains many alternative taxonomies for speech acts; wew have adopted a reduced set of six types that has been sufficient for our current purposes. These include:

$inform(S,L,C)$:	speaker S asks L to believe content C;
$acknowledge(S,L,C)$:	S tells L it has received and now believes content C;
$question(S,L,C)$:	S asks L a question C;
$propose(S,L,C)$:	S asks L to adopt goal C;
$accept(S,L,C)$:	S tells L it has adopted goal C;
$reject(S,L,C)$:	S tells L it has rejected goal C.

All domain-level and meta-level concepts in working memory are embedded within one of two predicates that denote aspects of mental states: *belief(A, C)* or *goal(A, C)* for some agent *A* and content *C*, as in *belief(medic, [i1, type, injury])*. A mental state's content may be a triple, *[i, r, x]*, a belief or goal term (*nested* mental states), an agent's belief that some attribute has a value, as in *belief_wh(A, [i, r])*, a belief about whether some propositional content is true, as in *belief_if(A, C)*, or a meta-level literal, such as the description of a speech act.

Long-term memory contains generic knowledge in the form of rules. Each rule encodes a situation or activity by associating a set of triples in its head with a pattern of concepts in its body. High-level predicates are defined by decomposition into other structures, imposing an organization similar to that in hierarchical task networks [11]. Structures in long-term memory include conceptual knowledge, skills, and goal-generating rules.

Conceptual knowledge comprises a set of rules which describe classes of situations that can arise relative to a single agent's beliefs or goals. These typically occur at the domain level and involve relations among states of the world. Conceptual rules define complex categories in terms of simpler ones and organize these relatonal predicates into taxonomies.

Skills encode the activities that agents can execute to achieve their goals. Each skill describes the effects of some action or high-level activity under specified conditions. The body of a skill include a set of preconditions, a set of effects, and a set

of invariants, along with a sequence of subtasks that are either executable actions, in the case of primitive skills, or other skills, in the case of nonprimitive skills.

Goal-generating rules specify domain-level knowledge about the circumstances under which an agent should establish new goals. For example, an agent might have a rule stating that, when a teammate is injured, it should adopt a goal for him to be stabilized. These are similar to conceptual rules, but they support the generation of goals rather than inference of beliefs.

The architecture also includes more abstract, domain-independent knowledge at th meta-level. This typically involves skills, but it can also specify conceptual relations (e.g., about transitivity). The most important structures of this type are *speech act rules* that explain dialogue actions to patterns of agents' beliefs and goals *without* making reference to domain-level concepts. However, the *content* of a speech act is instantiated as in any other concept. For example, the rule for an *inform* act is:

$$\begin{aligned} \mathit{inform}(S,L,C) \leftarrow\ & \mathit{belief}(S,C), \\ & \mathit{goal}(S,\ \mathit{belief}(L,C)), \\ & \mathit{belief}(S,\mathit{belief}(L,C)). \end{aligned}$$

Here S refers to the speaker, L to the listener, and C to the content of the speech act. Rules for other speech acts take a similar abstract form.

Finally, the architecture assumes additional meta-level knowledge in the form of a *dialogue grammar* that recursively specifies valid patterns of speech acts. For example, we can decompose a dialogue into a pattern consisting of a speaker S proposing P to a listener L, followed by L's acceptance A to S, followed by a dialogue. To ensure a coherent account of the conversation, the framework includes meta-level rules that indicate 'conceptual agreement' between the arguments of speech acts; these ensure that answers to questions are consistent with the agent's beliefs.

3.2 Architectural Processing

Our dialogue architecture uses these structures to operate in dynamic settings, both interpreting and responding to inputs in terms of its available knowledge and current model of the situation. Like a traditional cognitive architecture, it operates in cycles that access relevant knowledge and use it to guide processing. This includes incrementally extending its view of the common ground and its relation to active goals, then applying skills that are appropriate to achieving those goals. On each cycle, the architecture invokes a module for dialogue interpretation followed by another for dialogue generation. We discuss the operation of each of these in turn.

Dialogue Interpretation. The most basic task confronting a dialogue system is to understand its common ground with other agents. In natural settings many utterances are elided and others may be misheard, yet it must still construct models of participants' mental states, making reasonable assumptions about necessary elements that are missing from working memory. To this end, the architecture's interpretation stage incorporates a form of abductive inference. This abduction mecha-

nism prefers explanations that introduce few assumptions as possible while accounting for many of the 'observations' that arrive through speech acts.

The process first attempts to build support for a top-level rule, such as the existence of a dialogue in the pattern of speech acts, without making any assumptions. If it cannot derive the rule's head in this manner, then increases the tolerance to one default assumption, then two, and so forth, continuing until reaching a maximum. If the interpretation module finds a proof within this limit, then it adds the assumed elements to working memory, where they become available for use on later rounds.

The abduction mechanism incorporates new utterances and other observations into working memory at the start of each cognitive cycle. Their arrival can lead it to introduce beliefs and goals for the participating agents as default assumptions, with dialogue grammar rules building upon speech acts and other conceptual rules lower in the proof tree. The module can also introduce omitted speech acts, such as implicit acknowledgements, as default assumptions, which serve as terminal nodes in the extended explanation.

Dialogue Generation. The architecture must also produce some response to continue the dialogue, which is the responsibility of a second module. On each cycle, the first stage in this process inspects the goal-generating rules, finding which ones have conditions that match against the current contents of working memory, instantiating their arguments, and adding new top-level goals as a result.[1] Next, an execution stage selects a top-level goal to pursue and finds a skill clause with this goal in its head and with conditions that match working memory. The module repeats this step recursively, finding a path down through the skill hierarchy that, if executed, should help in achieving the top-level goal. Upon reaching a primitive skill, the architecture instantiates its variables and carries out its associated actions.

On the next cycle, the module might select the same top-level goal and repeat this process, but, typically, the conditions of some skills along the previous path will no longer be satisfied, so the architecture follows a slightly different route. This leads the agent to carry out subskills in sequence, much as in the ICARUS cognitive architecture [8]. The execution process is reactive in that it responds to changes in the situation, but the influence of top-level goals also provides continuity over time. The result is hierarchical behavior in which the agent traverses the branches of an AND-tree, in which each terminal node is an executed primitive skill, across multiple cognitive cycles.

The response of the dialogue generation mechanism also varies based on the type of goal. Abduced goals typically result in the execution of a meta-level skill, say one to communicate an instruction. On the other hand, goals inferred from goal-generating rules typically result in the execution of domain-specific skills. Interestingly, meta-level and domain-specific knowledge always interact at some point during processing. For instance, a domain-specific skill may have a meta-level skill as one of its subskills, while a generic skill, like one for communicating an instruction, is eventually instantiated with some domain-specific content.

[1] The abductive inference mechanism can also introduce new top-level goals as default assumptions during its processing.

4 Empirical Evaluation

As mentioned earlier, we have used the architecture to implement two dialogue systems, one for meeting support and another for advising medics. We only have space here to report results of test runs with the first of them. We will use the interaction in Table 1 to illustrate the structures and processes that arise during the system's operation. For instance, after the doctor's utterance "John, do leg lifts," the abductive interpretation module produces a working memory that contains:[2]

$belief(dr, propose(dr, john, [[e1, exercise\_type, leg\_lift], [e1, agent, john]]))$
$belief(john, propose(dr, john, [[e1, exercise\_type, leg\_lift], [e1, agent, john]]))$
$goal(dr, [[e1, exercise\_type, leg\_lift], [e1, agent, john]])$
$goal(dr, goal(john, [[e1, exercise\_type, leg\_lift], [e1, agent, john]]))$
$belief(john, goal(dr, [[e1, exercise\_type, leg\_lift], [e1, agent, john]]))$
$belief(john, goal(dr, goal(john, [[e1, exercise\_type, leg\_lift], [e1, agent, john]])))$

In other words, after the utterance, the system believes that both the doctor and John believe a speech act occurred in which the speaker (doctor) proposes that the listener (John) does a leg-lifting exercise, that the doctor has the goal that John do leg lifts, that the doctor has the goal that John adopt the goal of leg lifting, and that John also believes the doctor has these two goals. Upon entering the dialogue generation module, the system does not find any goal-generating rules or any skills with conditions that match. For this reason, it does not produce any new goals or generate any utterances before it completes the cognitive cycle.

The next utterance is John's response "Okay. I will do leg lifts," which indicates that he accepts the doctor's proposal. The system starts a new cycle, with the first step using abductive inference to expand its model of the common ground by adding to working memory:

$belief(john, accept(john, dr, [[e1, exercise\_type, leg\_lift], [e1, agent, john]]))$
$belief(dr, accept(john, dr, [[e1, exercise\_type, leg\_lift], [e1, agent, john]]))$
$goal(john, [[e1, exercise\_type, leg\_lift], [e1, agent, john]])$
$goal(john, belief(dr, goal(john, [[e1, exercise\_type, leg\_lift], [e1, agent, john]])))$
$belief(dr, goal(john, [[e1, exercise\_type, leg\_lift], [e1, agent, john]]))$
$goal(sys, [[e1, exercise\_type, leg\_lift], [e1, agent, john]])$

At this point, the system believes that both the doctor and John believe an accept speech act occurred, that John has adopted the goal of leg lifting and wants the doctor to believe that he now has this goal, that the doctor believes that John has adopted the goal, and, since it aims to support the joint task and both parties have adopted the goal, the system adopts the goal for itself.

In this case, the dialogue generation module matches a goal-generating rule against these elements, producing a new goal to command the television to play a physical therapy tutorial for the patient:

[2] For readability, we omit the top level predicate *belief(sys,Content)* and only show the *Content*.

$$goal(sys,\ sys\_message(tv, leg\_tutorial, nil))$$

The system acts on this goal during the execution stage, invoking a skill that sends a command to the television to play the corresponding video.

During the next cognitive cycle, the system does not receive any utterance from the human users, but a signal does arrive from the motion detector indicating that the patient has lifted his leg. The interpretation module adds this information to working memory as the fact:

$$observation(motion, [[ep2, type, leg\_lift], [ep2, agent, john]])$$

where *ep2* is a new constant that denotes an event of type *leg_lift* whose *agent* is John. In response, the abductive inference process extends the current explanation by adding the elements:

$$belief(sys, [ep2, type, leg\_lift])$$
$$belief(sys, [ep2, agent, john])$$
$$belief(sys, [e1, current\_state, active])$$
$$belief(sys, [e1, reps\_done, 1])$$
$$belief(sys, [e1, last\_rep\_time, 1382124783.0])$$

The system now believes that a leg-lifting event is ongoing and that the first lift has occurred, so it adds a time stamp for the last repetition of the activity, as the system's knows that a leg-lifting exercise involves ten repetitions. Goal generation then produces an intention for the system to utter an encouragement to the patient:

$$goal(sys, support(sys, john, activity\_start))$$

The execution process focuses on this goal and carries out a skill that produces the utterance "Great start John!", which it sends to the text interface, making it available to everyone involved in the meeting.

We lack the space to completely analyze the remaining interaction, but it is important to note how the system reacts to divergences from the above sequence of events. Consider the case in which the doctor instead proposes "John lie down" and in which John counters "No, I will do leg lifts." As there is no agreed upon goal, in this case the system does not play the tutorial and instead reminds John of the doctor's goal by uttering "John, the doctor wants you to lie down."

Alternatively, consider a variation in which the interaction starts with the original utterances by the doctor and John about leg lifts, followed by the tutorial, but in which no signal arrives from the motion detector. In this case, after some time has passed without the expected motion signal, the system generates a goal to utter "John, you should strap on the motion detector" and executes a skill that communicates this content to the patient.

The different interactions illustrate the system's ability to respond appropriately based on its beliefs about the mental state of the users (e.g., whether they adopted the same goal) and the environmental situation (e.g., that the patient forgot to wear the motion detector). The dialogue framework supports such reactive responses within the broader context of the high-level goals it has adopted.

5 Discussion

Our architecture and the two systems that utilize it take steps towards robust, task-oriented dialogue systems, but there are some issues that we have not addressed fully. We remarked earlier that we plan to replace the text-based interfaces with spoken language interfaces. That move will come with the additional complication of uncertainty in the meaning of utterances, but we believe our abductive approach to incremental inference is well situated to handle this issue. We must increase the scope of explanations to include hypotheses about the meaning of each utterance, possibly using some measure of uncertainty. We should also introduce the ability to revise faulty assumptions that arise in dialogue misunderstandings, to which abduction also lends itself [10]. At the same time, one motivation for developing the architecture was to support robust cognitive systems. This suggests additional research goals, including the ability to execute skills in parallel and to handle unfamiliar tasks through problem solving. We believe that our architecture's representations and mechanisms could be adapted to other tasks beyond task-oriented dialogue that involve social cognition. Examples include settings in which agents provide help without verbal communication and in which self-interested agents take advantage of ignorance and deception [3].

The literature reports a number of advanced dialogue managers. RavenClaw [2] separates from the domain level some domain-independent aspects of dialogue management, including turn taking, timing, and error handling. In contrast, we have focused on domain-independent principles at the abstract level of dialogue knowledge. Moreover, RavenClaw emphasizes generation, while our architecture balances interpretation and generation. Our architecture is similar to Collagen [12] in that both utilize hierarchical plan structures and construct models of agents' beliefs during interpretation and generation, but a key difference is that Collagen does not separate meta-level from domain knowledge. Also, despite sharing some high-level assumptions, our abduction mechanism makes the two frameworks operate quite differently. We should also mention TRIPS [5], an integrated system that carries out dialogues to help users generate plans, drawing on knowledge to interpret user input and generate responses. However, TRIPS was designed for the task of plan creation, while our architecture can support any collaborative task given suitable domain knowledge.

6 Concluding Remarks

In this paper, we presented an architecture for task-oriented dialogue that integrates interpretation and generation, along with two implemented systems that build on it. We discussed results obtained from runs with the meeting support system, demonstrating how it interprets the current situation and, by combining meta-level and domain-level knowledge, supports users by participating actively in the dialogue and issuing commands to actuators.

In addition to integrating processes for dialogue interpretation and generation, the framework provides a clear separation of meta-level content from domain expertise,

which we maintain is a desirable feature in a cognitive architecture. These suggest that it can serve as a solid foundation for future research on both dialogue systems and other software agents that interact with humans.

Acknowledgements

This research was supported by Grant N00014-09-1-1029 from the Office of Naval Research and a gift from Ericsson. We thank Chitta Baral, Paul Bello, Will Bridewell, Herb Clark, Tolga Könik, Nimish Radia, Ted Selker, David Stracuzzi, Chihiro Suga, and Richard Weyrauch for discussions that influenced the approach reported here.

References

[1] Austin JL (1962) How to do things with words. Harvard University Press, Cambridge, MA
[2] Bohus D, Rudnicky A (2009) The RavenClaw dialog management framework: Architecture and systems. Computer Speech & Language 23(3):332–361
[3] Bridewell W, Isaac A (2011) Recognizing deception: A model of dynamic belief attribution. In: Advances in Cognitive Systems: Papers from the 2011 AAAI Fall Symposium, pp 50–57
[4] Clark HH (1996) Using language. Cambridge University Press, Cambridge, UK
[5] Ferguson G, Allen JF (1998) TRIPS: An integrated intelligent problem-solving assistant. In: Proceedings of the 15th National Conference on Artificial Intelligence, pp 567–572
[6] Gabaldon A, Langley P, Meadows B (2013) Integrating meta-level and domain-level knowledge for interpretation and generation of task-oriented dialogue. In: Proceedings of the Second Annual Conference on Advances in Cognitive Systems
[7] Langley P (2012) The cognitive systems paradigm. Advances in Cognitive Systems 1:3–13
[8] Langley P, Choi, D, and Rogers, S (2009) Acquisition of hierarchical reactive skills in a unified cognitive architecture. Cognitive Systems Research 10: 316–332.
[9] Langley P, Laird J E, and Rogers S (2009) Cognitive architectures: Research issues and challenges. Cognitive Systems Research 10: 141–160.
[10] McRoy S, Hirst G (1995) The repair of speech act misunderstandings by abductive inference. Computational Linguistics 21(4):435–478
[11] Nau DS, Cao Y, Lotem A, Munoz-Avila A (2001) The SHOP planning system. AI Magazine 22:91–94
[12] Rich C, Sidner CL, Lesh N (2001) Collagen: Applying collaborative discourse theory to human-computer interaction. AI Magazine 22:15–25
[13] Searle J (1969) Speech acts: An essay in the philosophy of language. Cambridge University Press, New York, NY