Integrated Interpretation and Generation of Task-Oriented Dialog

Alfredo Gabaldon, Ben Meadows, and Pat Langley

Abstract In this paper, we present an architecture for high-level aspects of task-oriented dialog that integrates interpretation and generation, as well as domain-level and dialog-level knowledge. We report and analyze two implemented systems based on this architecture, including one for meeting support and another military medic assistance, and we discuss our experiences with the first system. We conclude with comments on related work and plans for future research.

1 Introduction

Any system that uses a natural language interface to assist its users in carrying out some task must let them talk with the system about that task as its execution progresses. The system in turn must interpret the user's utterances and other environmental input, then use this content to infer their *common ground*, what the system and user jointly *believe* and *intend* about each other and the environment. The system must also combine this model with its knowledge to generate utterances that carry the dialog forward and help the user achieve his or her goals. In this paper we report two systems that we have developed for such task-oriented dialog, along with the architecture that underpins them, which combines interpretation and generation in the manner just described.

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
Department of Computer Science, University of Auckland, Auckland 1142 NZ
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 knowledge is essential for intelligent behavior, we also believe that it benefits from more abstract knowledge that generalizes across domains. In particular, we are interested in knowledge and strategies relevant to high-level aspects of dialog processing that are independent of conversational content. The architecture separates such meta-level knowledge from domain expertise but uses both during interpretation and generation. Our approach is informed by cognitive systems research [7], a key feature of which is integrated processing of knowledge at different levels of abstraction.

Another feature of the architecture is its incremental processing of dialogue. We assume that conversations occur within a changing environment and that details of the task being discussed are not pre-determined, but rather as the interaction proceeds. For this reason, our architecture elaborates its model of the situation and the user's mental state incrementally, responds according to this understanding, and adapts to new information by choosing both what to say and what goals to adopt. In other words, the framework is designed for *situated* systems that carry out *goal-directed* dialogs to support their users. In the next section, we report two implemented prototypes that demonstrate this functionality. After this, we describe the underlying architecture in some detail and then discuss our experiences with the first system. In closing, we discussed related work and plans for future research.

2 Task-Oriented Dialog Application Systems

As noted earlier, we have developed two prototype systems that use our architecture as their main dialog engine. One aims to facilitate cyber-physical meetings by interacting with both humans and equipment. The other collaborates with a military medic to help him provide emergency treatment to a patient in the field. In this section we discuss each of these systems in turn.

2.1 Meeting Support System

We designed the first system to support *cyber-physical* meetings in which a number of humans with devices and each other to achieve some joint task. Our specific scenario involves a meeting between a physical therapist, a patient, and a relative, with the shared goal of completing a physical therapy session. During the meeting, the users take part in a dialog that includes our system as a participant. They also interact with sensors, such as motion detectors, heart-rate monitors, and room thermometers, as well as effectors such as controls for a video player.

Within a therapy session, users communicate by sending sentences in a form of controlled English through a menu-based smart-phone interface, although we intend to replace this with one that uses spoken language in future work. The interface accepts sentences, translates them into a logic-like representation, and sends the result to our dialog system for processing. At the same time, the sensing devices

Table 1 Sample meeting support dialog involving a physical therapist and a patient.

Therapist:	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.
Therapist:	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.

send messages to the dialog system via `HTTP`. Each user can see the utterances of other users and those of the system itself via the phone interface.

The system contains knowledge about activities that participants may carry out during physical therapy. These include forms of physical activities, measuring range of movement, obtaining relevant patient information, and making physiological measurements. In a typical therapy session, the system asks standard questions of the patient and the therapist may ask further questions. After this, the therapist, possibly with assistance from the system, decides on an appropriate therapy for the session and suggest it to the other participants.

If the patient or relative communicates acceptance of the proposed task, the system updates its explanation of the situation and proceeds accordingly. The system may support the patient's execution of a task, say by instructing the TV to show a tutorial video. Once the system becomes aware that the patient has started executing the task (e.g., using input from motion detectors), it may issue an encouragement utterance, such as "Great start!" The meeting support system comprises phone menu-based interfaces for sentence input, a phone application that works as a motion detector, a television for presenting tutorials and related videos, a heart-rate monitor, temperature and light sensors, the dialog system, and an `HTTP` client/server module for component communication. Table 1 presents a portion of a dialog from a sample physical therapy session. In this example, patient John participates in a physical therapy session in which he does leg lifts under the supervision of a therapist at a remote location. We will revisit this case study at more length in Section 4.

2.2 Medic Assistant

We designed our second prototype system for battlefield scenarios in which a military medic helps an injured teammate. Because the medic has limited training, the dialog system provides advice about treating the person's injuries; thus, it plays the

Table 2 Sample dialog between a medic and an 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.

role of an advisor with medical expertise. The medic and the system collaborate toward achieving the shared goal of stabilizing the patient. The system does not know in advance what injury it must address; only after the medic provides this information does the system act know which domain expertise is relevant to achieving the stabilization goal. The system does not sense or alter the environment directly; the human medic acts as its sensors and effectors, answering questions and following instructions from the system. provides.

A typical interaction begins with the system asking the medic questions to give it information about the nature of the injury. The sequence of questions is not predetermined, with later queries being influenced by the medic's responses. Table 2 shows a sample dialog that revolves around stabilizing a patient with a bleeding injury. The system possesses domain knowledge about how to treat different types of injuries based on their location, severity, and other characteristics. The system also adapts its recommended treatment in response to the situation: for instance, it suggests an alternative treatment if the medic informs it that he lacks the necessary supplies.

This system uses a Web interface similar to that in a text messaging application. footnoteAgain, in the future we intend to replace this Web interface with one that supports spoken dialog. The medic types sentences in constrained English on the interface form. These are sent to the dialog system via an `HTTP` request, which in turn sends them to a natural language processor for translation into the logical form our system assumes. For this prototype, we have used the proprietary *Skyphrase* system (`http://www.skyphrase.com`), which sends back the translation via `HTTP`. The dialog system uses this content to update its current explanation of the situation and generates utterances that it thinks will help achieve the joint goal. We do not have space to provide further details here, but we have described the system at more length elsewhere [6].

2.3 Discussion

Although the two systems described above are limited in many respects, they pose a number of new challenges that are not fully addressed by existing dialog systems:

- As the dialog and the situation evolve, the system must update its explanation of the latter by interpreting all the information received up to that point, including dialog utterances and other input.
- The system's interaction with users is goal directed and focuses on joint activity over time. This activity includes not only actions carried out by the users, but also communicative actions and commands to device effectors.
- The system must establish *common ground* [4] with its users, creating a shared model of the environmental situation and the participants' beliefs and goals. It must also update this joint model as circumstances continue to change and as information flows among the conversants.
- Most of the participants' beliefs and goals are never stated explicitly, so that the system must be able to infer enough of them to be effective. This involves using not only domain-specific expertise, but also more abstract knowledge about how communication events influence mental states.
- 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 agent architecture represent first steps toward addressing these challenges. In the next section we describe this integrated architecture, an implementation of which is the main component of the two prototypes.

3 Agent Architecture

We are now ready to describe our architectural framework, which focuses on dialogs that support physical, goal-directed behavior in dynamic settings. The architecture incorporates rich mental models that let it draw inferences from utterances and generate others in response. Input is *multi-modal* in that it may 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 representations and control mechanisms diverge substantially from the Prolog defaults.

3.1 Representation and Content

We distinguish short-term or working memory, which contains external inputs, inferences based upon these inputs, and agent goals, from long-term memory, which stores knowledge the dialog system uses to make inferences and execute actions.

Working memory is a dynamic set of ground literals that represent the system's beliefs and goals about the developing situation. Literals for domain-level content, which never appear at the top level, are encoded as relational triples, as in *[i1, type,*

injury] and *[i1, severity, major]*. This reification lets the system access different aspects of complex relations and events separately, including their predicates.

The architecture also incorporates meta-level predicates, at a higher level of abstraction than the domain level, for speech acts [1, 11]. Various taxonomies of speech acts have been developed, but we have found a set of six types suffices for our current purposes:

$inform(S,L,C)$:	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.

In working memory, all domain-level and meta-level concepts are embedded within elements that describe mental states: *belief(Ag, C)* or *goal(Ag, C)* for some agent *Ag* and content *C*, such as *belief(medic, [i1, type, injury])*. An element's content may be a triple, *[i, r, x]*, a belief or goal (*nested* elements), an agent's belief that some attribute has a value, *belief_wh(Ag, [i, r])*, a belief about whether some propositional content is true, *belief_if(Ag, C')*, or a meta-level predicate, such as a speech act type.

Long-term memory contains generic knowledge in the form of rules. The body of each rule describes some situation or activity, in terms of a set of triples, and associates it with the predicate in the rule head. Predicates may be defined by decomposition into more basic structures, imposing an organization similar to that of hierarchical task networks [9]. Three types of structures appear in long-term memory. *Conceptual knowledge* describes classes of situations that are relevant to an agent's beliefs or goals. This content usually focuses on the domain level and involves relations that arise in states of the world. *Skills* describe agent activities, including conditions for application and either executable actions or a sequence of lower-level skills). Finally, *goal-generating knowledge* comprises domain-level rules that state the circumstances in which an agent should establish a new goal, such as stabilizing a person when he has been injured.

The architecture complements this domain content with meta-level knowledge. Some of this takes the form of skills, but important counterexamples are conceptual rules for *speech acts*. These refer to patterns of agents' beliefs and goals, *sans* domain-level predicates: they generalize over the *content* being communicated. For example, in the rule for *inform*,

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

the variable C denotes the speech act's content, which may bind to any set of triples. Another important form of meta-level knowledge is the *dialog grammar*, which specifies valid patterns of speech acts. This includes rules that decompose a dialog into constituents, such as a speaker S proposing P to a listener L, followed by L's acceptance A to S, followed by a continuation of the dialog. The grammar also includes rules for ensuring 'conceptual agreement' between the content of successive speech acts, such as that questions are followed by relevant answers.

3.2 Architectural Processing

As we have noted, the architecture deals with dynamic situations, explaining and reacting to inputs in terms of available knowledge. On each cycle, it incrementally extends the explanation of the dialog to date and applies new skills as appropriate.

Dialog Interpretation. The system's input is necessarily incomplete: many of the mental states relevant to understanding are unobserved, utterances may be misheard, and observations may be missed. The architecture must make plausible assumptions to extend its explanation of observed events; for this reason, abductive inference plays a key role in interpretation. The abduction mechanism prefers explanations that make fewer assumptions while providing greater explanatory coverage.

The process first attempts to construct a proof tree using a top-level rule (e.g., that a dialog is present which satisfies the grammar) without assumptions. If this does not succeed, it increases the tolerance to one default assumption, then two, and so forth until it finds a proof or exceeds a tolerance limit. Upon finding a proof, the module adds the assumed elements to working memory, making them available in future cycles. The abduction mechanism attempts to incorporate new utterances and other observations into working memory at the start of each cognitive cycle. Default assumptions can involve the beliefs and goals of participating agents, with the dialog grammar building on speech acts and other conceptual rules lower in the proof tree. Omitted speech acts, such as implicit acknowledgements, may also serve as default assumptions, appearing as terminal nodes in the explanation.

Dialog Generation. Following the interpretation stage, the architecture matches the conditions of goal-generating rules against the current working memory elements, instantiating their arguments and adding new top-level goals to memory.[1] After this, it selects a top-level goal and finds a skill with this goal in its head and with conditions that match working memory. The generation module repeats this step recursively, generating a path down through the skill hierarchy that should achieve 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 system may select the same top-level goal, repeating this process, but typically the conditions of some skills along the previous path will no longer be satisfied, so the generation module follows a slightly different route. This causes the agent to carry out subskills and actions in a particular sequence. The architecture appears to be reactive, but the influence of top-level goals provides continuity. The result is hierarchical behavior in which the system traverses different branches of an AND-tree on each cycle.

The generation mechanism produces different behavior depending on whether the selected goal was produced by abductive inference or domain-level goal generation. In the former case, the system may have adopted a goal that leads it to access a meta-level skill to invoke the *inform* speech act, such as *goal(medic, belief(advisor, [inj1,injury_type,bleeding]))*. Obtaining this answer might in turn require domain-level actions for information gathering. In contrast, a domain-level goal might, in the process of executing domain-specific skills, lead to the agent requesting assistance.

[1] The abductive inference process may also introduce top-level goals as default assumptions.

4 Experience with the Architecture

We can now discuss our experience with the meeting support system introduced in Section 2.1. Consider again the interaction in Table 1. After the architecture processes utterance "John, do leg lifts" by the therapist (th), its working memory contains an explanation that includes:[2]

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

In other words, after the utterance, the system believes that both the therapist and John believe a speech act occurred in which the speaker (therapist) proposes that the listener (John) do a leg-lifting exercise, that the therapist has the goal that John do leg lifting, that the therapist has the goal for John to adopt the goal of leg lifting, and that John also believes the therapist has those two goals. The system then considers possible new goals, but does not generate any. Nor does dialog generation produce anything at this point, completing the cognitive cycle.

The next utterance is John's response, "Okay. I will do leg lifts," in which he accepting the therapist's proposal. The system starts a new cycle, first expanding its explanation by adding the elements:

$$belief(john, accept(john, th, [[e1, exercise\_type, leg\_lifting], [e1, agent, john]]))$$
$$belief(th, accept(john, th, [[e1, exercise\_type, leg\_lifting], [e1, agent, john]]))$$
$$goal(john, [[e1, exercise\_type, leg\_lifting], [e1, agent, john]])$$
$$goal(john, belief(th, goal(john, [[e1, exercise\_type, leg\_lifting], [e1, agent, john]])))$$
$$belief(th, goal(john, [[e1, exercise\_type, leg\_lifting], [e1, agent, john]]))$$
$$goal(system, [[e1, exercise\_type, leg\_lifting], [e1, agent, john]])$$

At this point, the system believes that both the therapist and John believe an accept speech act has occurred, that John has adopted the leg-lifting goal and wants the therapist to believe this, that the therapist believes John has adopted the goal, and, since the system is aiding two parties with a shared goal, it adopts the goal as well. This interpretation step is followed by goal generation, which produces a goal to command the TV to play a leg-lifting tutorial for the patient:

$$goal(system,\ system\_message(tv, tutorial, nil))$$

During the generation stage, the system acts on this goal and sends the command to the TV, which plays the corresponding video tutorial.

On the next cognitive cycle, the system receives no utterance from the human users. However, it receives a signal from the motion detector, leading it to add to working memory the literal:

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

where $ep2$ is a new constant that denotes a leg lift event whose *agent* is John. In

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

response, the system expands its explanation by adding a number of new beliefs:

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

The system now thinks the leg lifting event is ongoing and the first leg lift has occurred, so it adds a time stamp for the last repetition (leg lift) of the activity, as its knowledge states that leg lifting involves ten repetitions. Goal generation then produces a goal for the system to utter encouragement to the patient:

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

During the dialog generation stage, the system acts on this goal by producing the utterance "Great start, John!"

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. For instance, consider a case in which the therapist starts instead by proposing "John lie down", followed by John's counter proposal "No, I will do leg lifts." In this case, lacking a shared goal, the system would not play the video tutorial, but instead remind John of the therapist's goal by uttering "John, the therapist wants you to lie down." Alternatively, consider a variation in which the interaction starts with the same two utterances by the therapist and John about doing leg lifts, the system playing the tutorial, followed by no signal from the motion sensor (say, because John is not wearing it). In this case, after failing to receiving the expected signal for some time, the system would generate a goal to utter "John, you need to strap on the motion detector."

This interaction illustrates our dialog system's ability to respond appropriately by drawing on its beliefs about the mental state of the users, such as whether they adopted the same goal. We have also seen that it responds differently depending on the environmental conditions, such as when it fails to receive a signal about detected motion. Together, these clarify that the system is reactive to both the external world and to internal models of the participating agents.

5 Discussion

Dialogue has received increased attention in recent years, but much of the research has focused on the level of speech. However, the literature reports a number of higher-level dialog managers, one of the most advanced being RavenClaw [2]. This system separates the domain level from some domain-independent aspects of dialogue management, such as turn taking, timing, and error handling, but it appears to encode these strategies in procedural terms. In contrast, our aim is to extract domain-independent principles and encode them as knowledge that interacts directly with domain content. Another difference is that, although both systems integrate dialog interpretation with generation, RavenClaw focuses more heavily on the latter, while our architecture devotes more of its resources to interpretation.

Our approach has more in common with some older research on dialog. Collagen [10] also operated over hierarchical plan structures and constructed models of agents' beliefs for use during interpretation and generation. One key difference is that the earlier system did not separate meta-level from domain knowledge. Also, despite sharing high-level assumptions, our abductive inference mechanism leads to very different operational details. We should also mention TRIPS [5], an integrated system that carried out dialogs with users to help them generate plans. TRIPS used knowledge to interpret user input and generate appropriate responses, but it was designed for the task of plan creation, while our architecture should be able to support any collaborative task given suitable domain-specific knowledge.

Our architecture and prototype systems take some promising steps towards robust, task-oriented dialog systems, but considerable work remains. We have remarked on plans to replace our text-based interfaces with ones that use spoken language, introducing uncertainty about the meaning of utterances, which would arrive as a set of alternatives with associated probabilities. We believe our abductive approach to interpretation can be adapted to handle such inputs, but whether it work as envisioned is an empirical question. We should also extend the architecture to correct faulty assumptions and misunderstandings [8], which will involve adding methods for belief revision to the interpretation module. More generally, we hope to embed our dialog knowledge and mechanisms within broader cognitive systems that interact with other agents. We believe that our architecture's representations and processes are relevant to other tasks that involve social cognition, such as providing assistance in the absence of verbal communication and in dealing with self-interested agents who utilize ignorance and deception [3] to achieve their goals.

6 Concluding Remarks

We have presented an architecture for task-oriented dialog that integrates interpretation and generation, and two implemented systems based on it. We discussed results obtained from trial runs with the meeting support system and demonstrated how the system interprets the current situation, and together with background meta and domain-level knowledge, supports the users by participating in the dialog and issuing commands to actuators. In addition to integrating the dialog interpretation and generation processes, it also provides a clear separation of meta-level knowledge from domain-level knowledge, an advantageous feature in a cognitive architecture.

Acknowledgements

This research was supported in part by Grant N00014-09-1-1029 from the Office of Naval Research and by 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 we have 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: Klenk M, Laird J (eds) Second Annual Conference on Advances in Cognitive Systems
[7] Langley P (2012) The cognitive systems paradigm. Advances in Cognitive Systems 1:3–13
[8] McRoy S, Hirst G (1995) The repair of speech act misunderstandings by abductive inference. Computational linguistics 21(4):435–478
[9] Nau DS, Cao Y, Lotem A, Munoz-Avila A (2001) The SHOP planning system. AI Magazine 22:91–94
[10] Rich C, Sidner CL, Lesh N (2001) Collagen: Applying collaborative discourse theory to human-computer interaction. AI Magazine (22):15–25
[11] Searle J (1969) Speech Acts: An Essay in the Philosophy of Language. Cambridge University Press, New York, NY