Dialogue complexity with portability? Research directions for the Information State approach

Carl Burke, Christy Doran, Abigail Gertner,
Andy Gregorowicz, Lisa Harper, Joel Korb, Dan Loehr
The MITRE Corporation
202 Burlington Road, Bedford, MA 01730
{cburke,doran,gertner,andrewg,lisah,jkorb,loehr}@mitre.org

Abstract

We review existing types of dialogue managers (DMs), and propose that the Information State (IS) approach may allow both complexity of dialogue and ease of portability. We discuss implementational drawbacks of the only existing IS DM, and describe our work underway to develop a new DM resolving those drawbacks.

1 Introduction

Spoken dialogue systems have shown steady improvements in recent years. To continue advancing the state of the field, we must direct research towards reducing a tradeoff between complexity and portability. Otherwise, we will continue to have systems which can handle complex interactions, or systems which can be easily modified for new domains, but not both.

The simplest existing dialogue managers (DMs), **finite-state systems**, are suitable for simple, well-structured system-initiated dialogue tasks. They also make it easy for novice developers to create new dialogue systems. Yet this type of DM does not scale well to mixed-initiative dialogues or complicated tasks with a wide variety of possible input. The most well-known such DM is VoiceXML. Similar systems include Oregon Graduate Institute's Rapid Application Developer (CSLU 2002), Unisys' Dialog Design Assistant (Unisys 1998), Nuance's Speech Objects, the Swedish GULAN (yellow pages) system (Gustafson et al 1998), and several commercial systems by SpeechWorks.

More sophisticated, mixed-initiative, **frame-based** DMs often make use of semantic "frames" containing multiple "slots" or "keys", each of which can hold a "value". Either conversational partner can volunteer or request information about any slots in the frame, at any time, in any order. When enough slots are filled to the satisfaction of both parties, the task and conversation are complete. This type of DM supports a more flexible, arbitrary flow-of-control, often controlled by scripts of rules firing upon certain conditions. Examples of these types of DMs include Philips' SpeechMania (Aust and Schroer 1998), the Dialogue Design Language Tool in the Danish Generic Dialogue System (Bernsen et al 1998), and several of the DMs developed (by e.g. MIT and the University of Colorado) for the DARPA Communicator infrastructure.

Even more complex **plan-based** DMs reason about "plans" and communicative "goals", and try to move the conversation along towards achieving these goals. By representing the relationships between goals, subgoals, and primitive actions in a domain, these systems can support dialogues with a broader scope than the frame-based DMs can. Notably, they are intended to detect topic shifts as well as support dynamic re-planning when misunderstandings occur. These systems typically model communicative goals in terms of speech acts where speech acts affect goals, beliefs, intent and/or obligations of the participants. These DMs can also be complex to develop, and correspondingly difficult to port to new applications. Examples of this type are COLLAGEN (COLLaborative AGENts), by Mitsubishi Electric Research Lab (Rich et. al. 2001), and the University of Rochester's TRAINS and TRIPS systems (CISD 2000).

The approach we find most promising, however, is the **Information State (IS) approach**, which provides the developmental simplicity of finite-state machines while allowing the flexible, complex interactions characteristic of plan-based dialogues. An IS theory of dialogue proposed by Cooper and Larson (1998) models dialogue states (i.e. structured semantic objects) as dependent record types. Dialogue moves (roughly equivalent to speaker turns) are characterized as transitions between information states in a manner that is neutral with regard to semantic theory. This approach to dialogue modeling enables developers to model the system information state in such a way that arbitrary linguistic theories of dialogue may be formalized, implemented, and compared. ISs may be used to model relations between various kinds of information such as utterances, background knowledge, non-verbal events and visual scenes. This is crucial to multimodal dialogue processing. Another important feature of the IS approach is that developers have the flexibility to define levels of dialogue as well as model goals, intent, beliefs and obligations. Thus the IS approach may also be used to model more complex dialogues using concepts derived from plan-based theories of dialogue - perhaps, inheriting some of the same challenges. However, the same framework may be used to also model simpler finite-state dialogues. TRINDIKit (TRINDI 2002) is an IS-based open source Prolog toolkit. TRINDIKit itself provides the basic infrastructure of a dialogue manager. It provides structured data types and the means to define an Information State from those types, a language for defining the modules of a Dialogue Move Engine (DME), and a language for controlling the application of individual modules to dialogue management.

We have built two dialogue systems using TRINDIKit (Burke et al 2002). We first developed a multimodal information kiosk by adapting GoDiS (Gothenburg Dialogue System) (Larsson et al 2000), which implements the Questions Under Discussion model in TRINDIKit. Adapting this existing TRINDIKit DM to a new question-answer domain required very little effort (less than two staff-weeks from initial downloading of TRINDIKit to an operational system open to the public). We then modified the DM to support control of a search-and-rescue robot using a speech-and-sketch interface on a PDA, again with relatively little effort. Based on our experience, we feel that the IS approach to dialogue management as espoused by TRINDI is a strong candidate for supporting both complexity and portability. In the remainder of this paper, we discuss some implementational drawbacks of TRINDIKit, and our work underway to develop a new toolkit, inspired by TRINDIKit but re-engineered to eliminate its drawbacks.

2 Implementational Drawbacks

Data consistency. TRINDIKit does not exercise good controls over asynchronous modifications to the IS. At one point we had to build artificial delays into our system to work around these limitations. The DM we built was based on GoDiS, which requires very structured turn-taking. In several cases, however, the interactions with the user flowed better if these responses were automatic. Processing was sufficiently slow that our GUI's automatic acknowledgement often arrived and was processed before TRINDIKit was finished cleaning up from the previous utterance. As a result, it was possible to change the IS twice before the DME could respond to one change, and the system lost track of the dialogue state. Consistency of data needs to be assured throughout the design of the system.

Inconsistent semantics. We encountered situations where constructs of the GoDiS plan language were interpreted differently depending on the depth of the plan. With the proliferation of small languages implemented by different sets of macros, it was difficult to track down bugs in the rules and conversation scripts. This was made more difficult by the nature of Prolog. Clauses that fail do not normally generate any error messages, because failure is a normal aspect of program execution. Unfortunately, database bugs and misspelled names often caused unexpected failures, causing the system to generate either no response or a response that looked reasonable but was in fact incorrect. We feel it's necessary to provide explicit notification of certain kinds of failure, such as failure to find a named variable, failure to find a matching value in a table, and so on.

Multimodal processing. Neither TRINDIKit nor GoDiS provides any direct support for multimodal processing. The primary interface driving the development of these systems was language; there is no separation of events by source, no temporal tagging of input events, and no provision for assessing temporal relationships between different inputs.

3 Proposed Solutions

From our experience with TRINDIKit, we're convinced of the advantages of a kit-based approach. We feel that TRINDIKit was a good first cut at it, and hope that our efforts will lead to a second, somewhat better iteration. We are therefore moving ahead with a new DM kit, tentatively called MIDIKI (MITRE DIalogue KIt), with the following features.

Distributed information state. We've chosen to model all of our module interactions as if they were asynchronous. This provides the cleanest separation of modules, and the cleanest conceptual integration with the asynchronous requirements of robot control. Our approach to solving this problem is to define an explicit interface definition language, which will be used to define every module's interface with the outside world. We explicitly include the information state structure in this interface definition, perhaps as a module in itself. Since TRINDIKit does not include a separate language for specifying module interfaces, we are designing our own. This language is analogous to CORBA Interface Definition Language, but with less concern for the physical implementation.

Controlled extensibility. Our interface specifications will need to be translated into specific computer languages before they can be executed. The translation will vary depending on the underlying protocol used to communicate between modules. While we want to support the widest possible audience, we don't want to get bogged down in the construction of translators for every possible set of implementation language and protocol. Our approach is to exploit an existing standard set of translation software, namely XML and XSLT processors such as Xalan. We are specifying a dialect of XML for modules interface definitions, and a small set of templates for realizing interfaces with specific combinations of programming language and protocol. Additional templates can be written to extend the kit to other languages and protocols without requiring modification of the kit itself.

Rule engine. The DME rules in TRINDIKit have strong similarities to rules in expert systems. We plan to implement these rules in both a sequential form, equivalent to the current TRINDIKit, and in an expert system form which may be more efficient. We expect that there will be differences in operating characteristics between those two styles, and we want to identify and quantify those differences.

Control and synchronization. Our primary focus is multimodal communication, potentially multiparty as well. We are extending TRINDIKit's triggers to include support for consideration of temporal relationships between events, both within and across modes.

Integrated environment. An ideal toolkit would have an integrated set of tools for designing, testing, and debugging dialogues. We would like to support static and dynamic analysis of dialogues, recording and playback of dialogues, graphical dialogue design tools, a "validation suite" of tests to support extension of the toolkit to new programming languages and agent protocols, and above all, the ability to plug-in as-yet-undefined capabilities.

4 Future Work

Significant effort has been devoted to defining our mutable language capability. This capability provides both a transition path from TRINDIKit scripts and a means for specifying module interfaces and information state structure using a common XML representation.

Our intent is to provide support for several different transport mechanisms to explore the limitations of our approach. To date, we have completed an initial interface definition specification and have developed templates to realize those interfaces with the OAA. DARPA's Galaxy Communicator is the second transport mechanism we will be considering. Time and resources permitting, we will examine some additional transports with differing characteristics, such as CORBA, Java Remote Method Invocation (RMI), or Linda.

We have devoted considerable time to up-front consideration of scripting languages, portable code generation, and module communications, and are now beginning the task of implementing our versions of the TRINDIKit scripting languages. Our target realization for these scripts is a combination of Java code and expert systems that can be executed within a Java program.

We plan to port and formally evaluate our dialogue toolkit within three domains (question-answering, automated tutoring, and multimodal robot control). Our dialogue toolkit will be openly available, as well as sample implementations for each of these domains.

References

Aust, H. and Schroer, O. (1998) An overview of the Philips dialog system. DARPA Broadcast News Transcription and Understanding Workshop, Lansdowne, VA.

Bernsen, N. O., Dybkjær, H. and Dybkjær, L. (1998) Designing interactive speech systems. From first ideas to user testing. Springer Verlag.

Burke, C., Harper, L., and Loehr, D. (2002) A Flexible Architecture for a Multimodal Robot Control Interface. Intelligent Situation-Aware Media and Presentations Workshop, AAAI '02.

CISD (Conversational Interaction and Spoken Dialogue Research Group) (2000) 'TRIPS: The Rochester Interactive Planning System', URL (Mar 2003): http://www.cs.rochester.edu/research/trips.

Cooper, R., and Larsson S. (1998) Dialogue Moves and Information States, Third International Workshop on Computational Semantics.

CSLU (Center for Spoken Language Understanding) (2002) 'CSLU Toolkit', URL (Mar 2003): http://cslu.cse.ogi.edu/toolkit.

Gustafson, J., Elmberg, P., Carlson,R., and Jönsson, A. (1998) An educational dialogue system with a user controllable dialogue manager. ICSLP' 98.

Larsson, Staffan, Robin Cooper, Stina Ericsson (2000) System Description of GoDis. Third Workshop in Human-Computer Conversation, Bellagio, Italy.

Rich, C., Lesh, N. and Sidner, C. (2001) COLLAGEN: Applying Collaborative Discourse Theory. AI Magazine, Special Issue on Intelligent User Interfaces.

TRINDI (2002) 'TRINDIKit', URL (Mar 2003): http://www.ling.gu.se/projekt/trindi/trindikit.

Unisys (1998) 'Unisys Corporation: Natural language speech assistant (NLSA): capabilities overview'. Malvern, PA.