

A Description Language for Dialog Modeling and Management

Cosmin Munteanu, Marian Boldea

Computer Science Department, "Politehnica" University
Blvd Vasile Pârvan 2, 1900 Timișoara, Romania
{cosmin,boldea}@cs.utt.ro

Abstract

Appropriate dialog modeling and management capabilities are essential for human-computer spoken dialog systems, and our article presents a dialog description language aimed to be used for both. In the following, the weaknesses of the method we used previously (based on finite state automata) are analyzed, the rationale for a description language as an improved means for both dialog modeling and management is given, and the main features of the particular language we propose are described.

1 Introduction

The recent growth of the IT industry is partly attributable to the development of friendly and comfortable user interfaces, which lately started to include spoken language interaction (speech recognition and synthesis).

As a natural follow up to developments in automatic speech recognition and synthesis, spoken dialog systems appeared, integrating them in a single framework, and the last years witness their deployment in real life applications, especially for telephone-based services, as well as their extension to a multimedia paradigm, including other modalities of interaction, besides spoken language.

The general structure of a dialog system is presented in figure 1, which highlights the central role of dialog modeling and management.

Within this structure, the dialog manager is the component that has to determine, based on user utterances and other information sources, which is the next action to be performed. The other information sources mentioned above include a dialog or interaction model, guiding these decisions, and often represented using finite state automata or related formalisms.

To solve some of the problems of finite state automata as interaction models, and to build a foundation for future research on dialogue modeling and management, we designed a dialogue description language, intended to allow for both dialog model representation and dialog manager implementation using it.

The article starts with brief theoretical considerations on dialog modeling (section 2) and management (section 3), used to analyze the finite state methods and find desirable features of a dialog description language able to replace them. Based on this analysis, and taking into account the semantic representations we already chose to use, the features of the dialog description language we propose are presented in section 4, while the last section is left for conclusions and plans for future work.

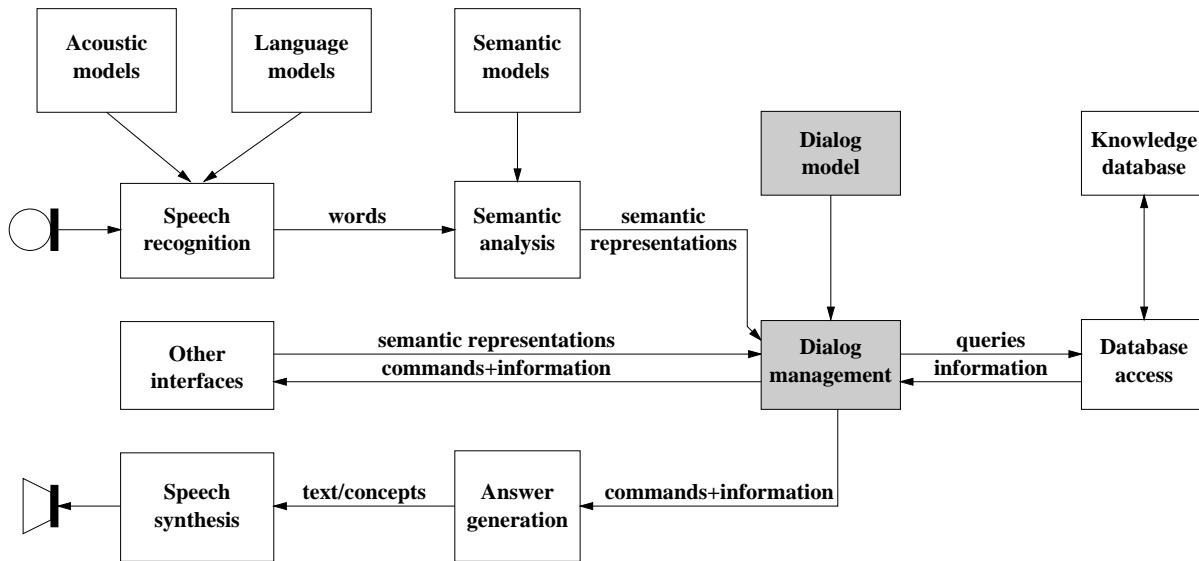


Figure 1: General structure of a dialog system

2 Dialog Modeling

In general, dialog study and modeling might be approached from multiple points of view (psychology, sociology, linguistics, computational linguistics, cognitive science, etc.) [1], but in what spoken dialog systems are concerned, the main goal of dialog modeling is to find ways to build and represent human-computer interaction models as a functional basis for the dialog manager [2], [3].

2.1 Finite state methods

During the first experiments, we used finite state methods for dialog modeling, the structure of a dialog being represented by oriented graphs in which two types of nodes can be distinguished: user and system. System nodes correspond to internal states of the system, and represent points in which decisions are made

based on user utterances, while user nodes are associated to system outputs, which could be answers to user questions, requests for reformulation or clarification, or simply questions addressed to users.

This representation has the advantage that the dialog model can be easily understood and modified using a specialized graph editor [4], [5], which enables a rapid, incremental development of the interaction model.

2.2 Dialog Description Language

The experiments using oriented graphs as described above showed that the main disadvantage of this method is the lack of support for sub-dialogs, which determined an important growth in size of the dialog model, e.g. to deal with incomplete or misrecognized input. Other weaknesses were found to be generated by limitations in representing useful information and

actions associated to graph nodes.

As an alternative to the graph representation, we chose to design a dialog description language [6] based on a context-free grammar. Although using such a language requires some programming knowledge, it offers an increased flexibility by allowing for sub-dialogs and a larger variety of information processing capabilities (e.g. system actions specified for each dialog state).

Obviously, the design of a dialog manager is strongly influenced by the representation of the interaction model, and an appropriate dialog description language could be used to specify directly its actions, so another important advantage of using DDL is the possibility of treating in an integrated manner problems related to both dialog modeling and management.

3 Dialog Management

The dialog manager is the essential part of a spoken dialog system, and its main task is to determine, at a certain moment, based on the application domain, the conversation context, and the user's last utterance, which is the next action the system must perform, and, if possible, to predict the contents of the next user utterance.

In the particular case of the application domain in which we are working (information retrieval), the manager must also be able to perform a few specific actions: to determine if there is enough information collected from user utterances to communicate efficiently with an external database engine, to actually communicate with it, and to transmit the requested information back to the user [1].

The "natural" user behavior increases the difficulty of building a dialog manager, since the next action must be chosen even when the user utterances are ambiguous or scarce in information. A powerful semantic parser could

reduce dialog manager complexity, but there remain two important tasks to be completed by the dialog manager: user actions identification and output control [7]. In the next section we will show how our DDL addresses these tasks.

4 DDL Features

To alleviate the problems mentioned earlier, DDL was designed as a procedural language offering a large flexibility for dialog modeling through the use of blocks to represent sub-dialogs (offering global solutions for local tasks), and specific data types for handling status information, so that dialog context (history) could be kept to a minimum.

DDL offers also the possibility to treat dialog modeling and management in an integrated manner, as a DDL file that describes an interaction model can be executed straight-away by an interpreter included in the dialog manager.

The main features of DDL are:

- the possibility to define and use variables of appropriate types; e.g., the dialog manager receives semantic information as frames generated from user utterances [8], and for this reason a **FRAME** data type was introduced to allow for semantic processing; its structure is shown in figure 2, from which can be noticed that it supports recursive declarations of slots, as each frame consist of one or more slots which can have the same structure as a frame;
- as a means to control the semantic analysis, the structure of a frame to be received can be specified by using an "expect" attribute for each field (slot) in the corresponding frame parameter;
- calling and processing of sub-dialogs, which can be represented in the manner of traditional imperative programming languages procedures, and instan-

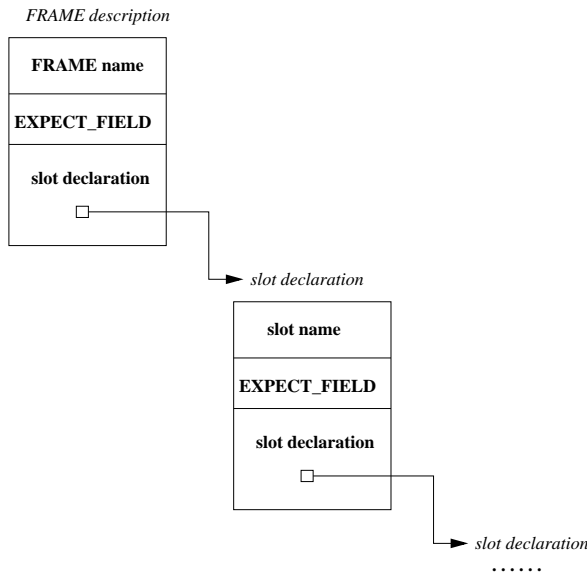


Figure 2: FRAME data type structure

tiated through specific instructions (`CALL node_name`);

- instructions for explicitly moving between different dialog states (`JUMP node_name`), which practically allow for the dialog model to be built starting from a first finite state approximation;
- dialog flow control: the dialog manager designer can control internal operations within a state using specific programming instruction (`IF - ELSE` combined with `GOTO label`);
- handling of system actions through functions that operate on frame variables:
 - getting frames from the semantic analyzer (`get_frame`);
 - database query (`print_query`);
 - output generation (`print_output`);
 - navigating through the dialog context (history), organized as a frame

list (`next_frame` and `prev_frame`); for the particular case of this list being treated as a stack, together with sub-dialogs, this allows for mixed-initiative dialogue strategies [9], [10].

To exemplify some of the possibilities that DDL offers, figures 3 and 4 present parts of a sub-dialog in DDL used for retrieving user identity (student or professor) in a dialog system for classes timetable information. More details about the language definition can be found in [11].

5 Conclusions and Future Work

Building a spoken dialog system requires a considerable effort, which could be reduce by a careful design and tuning of its main parts. In this context, the dialog manager needs special attention, and the design of the interaction model could be decisive for the final system performance [3].

One of our aims so far was to build appropriate interaction models, and to this end, after testing and evaluating interaction models based on graph representations, we designed a dialog description language (DDL) which allows not only for dialog modeling, but also for dialog management. DDL is a procedural language with specific facilities for handling sub-dialogs as structural units, and frames as knowledge representation units.

At this moment, work is in progress to implement a DDL interpreter as part of a dialog manager. Future work will evaluate the language in the framework of a dialog system, and improve it according to the feedback so collected.

```

current_frame = get_frame ((<identification> [MANDATORY]
    (<year> [OPTIONAL]) (<group> [OPTIONAL]))
    (<identification> [MANDATORY] (<prof> [MANDATORY])))

```

Figure 3: Fetching frames with an imposed slots structure. In this case, the frame is used to get user identity (student or professor).

```

if current_frame == <identification>
//an identifying frame
    no_identif = 0
    if !NULL(current_frame.identification.prof)
        //the user is a professor
        prof = current_frame.identification.prof
    else
        //the user is a student
        if !NULL(current_frame.identification.year)
            year = current_frame.identification.year
        if !NULL(current_frame.identification.group)
            group = current_frame.identification.group
    fi
else
    if current_frame == <negativ>
        //user didn't identify h*self
        no_identif = 1
    else
        //possibly correct sentence, but outside scope
        call not_understand
        goto identification_input
    fi
fi

```

Figure 4: A sequence for getting user identity. All useful information is stored in the FRAME variable *current_frame*. Based on this information, the system decides the next action to take.

References

- [1] M. McTear. Spoken Dialogue Technology: Enabling the Conversational User Interface. Submitted to ACM Computing Surveys, 2000.
- [2] E. Bilange. *Dialogue personne-machine*. Hermès, Paris, 1992.
- [3] N. O. Bernsen, H. Dybkjaer, and L. Dybkjaer. *Designing Interactive Speech Systems*. Springer-Verlag, London, 1998.
- [4] C. Munteanu. A Wizard of Oz Environment for Dialog Systems Development. Graduation Thesis, Computer Science Department, "Politehnica" University of Timișoara, June 1999.
- [5] C. Munteanu and M. Boldea. MDWOZ: A Wizard of Oz Environment for Dialog Systems Development. In *Proceedings 2nd International Conference on Language Resources and Evaluation - LREC2000*, volume 3, pages 1579–1582, Athens, Greece, May-June 2000.
- [6] H. Aust, M. Oerder, et al. The Philips train timetable information system. *Speech Communication*, 17:249–262, 1995.
- [7] A. Abella, M.K. Brown, and B. Buntschuh. Development Principles for Dialog-Based Interfaces. In *Dialogue Processing in Spoken Language Systems*, volume 1236 of *Lecture Notes in Artificial Intelligence*, pages 141–155. Springer-Verlag, London, 1996.
- [8] D. Bohuș. Stochastic Semantic Analysis in Human-Computer Dialog. Graduation Thesis, Computer Science Department, "Politehnica" University of Timișoara, June 2000.
- [9] R. López-Cózar, A.J. Rubio, et al. Evaluation of a Dialogue System Based on a Generic Model that Combines Robust Speech Understanding and Mixed-Initiative Control. In *Proceedings 2nd International Conference on Language Resources and Evaluation - LREC2000*, volume 2, pages 743–747, Athens, Greece, May-June 2000.
- [10] A.I. Rudnicky, E. Thayer, et al. Creating Natural Dialogs in the Carnegie Mellon Communicator System. In *Proceedings EUROSPEECH'99*, volume 4, pages 1531–1534, Budapest, September 1999.
- [11] C. Munteanu. Human-Computer Dialog Modeling and Management. Master Thesis, Department of Computer Science, "Politehnica" University of Timișoara, June 2000.