Database Access via Spoken Language Interfaces

Hans Dybkjær, Laila Dybkjær, Niels Ole Bernsen
{dybkjaer, laila, nob}@cog.ruc.dk
phone +45 46 75 77 11, fax +45 46 75 45 02
Centre for Cognitive Science, Roskilde University
P.O.Box 260, DK-4000 Roskilde, Denmark

Abstract

The opportunity to interact through natural language dialogue may strongly facilitate the use of databases by occasional users and non-experts. Spoken language human-computer interfaces have improved markedly in recent years and today appear in commercial contexts. The paper presents results on the use of spoken language dialogue for database access, deriving from the Danish Dialogue project in which a prototype has been developed of a speech based telephone ordering system for Danish domestic flight tickets. The dialogue functionality as regards user and database is described, and relations between dialogue and database are discussed.

Keywords: Spoken language dialogue, flexible database interfaces.

1 Introduction

Natural language is inherently a comfortable form of communication for humans. The idea of using natural language for human-computer interaction is, not surprisingly, as old as the 40 year old tradition of computational linguistics. Over the years, interfaces to computer systems have generally been much improved. [Hendrix et al. 1978] mention that business executives and other decision makers typically have a good idea of the data residing in their databases, but yet have to rely on the services of technicians to formulate and plan database queries. This has changed and standard interfaces to database management systems have become easier to use since then. But the need for even simpler forms of access increases with the rapidly growing consumption of electronic information by the public at large. The number of home computers is increasing, and via networks and telephone a long series of information services, typically drawing on databases, has become available to the public:

- Many home computers have modems which give access to the telephone network and hence to a variety of services. Recently, e.g., the Danish Pol on line service has been opened to provide access to, i.a., old and new newspaper articles and to allow people to write contributions for discussion with others. In 1995 Diatel (the Danish version of the French Minitel) will be opened and provide access to a broad variety of databases containing, e.g., news, cook book recipes and advertisements. The rapidly growing Internet allows for access to almost everything, including home shopping.
- The telephone also provides access to databases, primarily in the form of voice-response systems. Typically banks and newspapers but also many other organisations offer services to the public in this way.

Voice-response systems have the advantage that almost everybody has a telephone and is familiar with its use. No computer or specialized software is needed. However, for longer or more complex dialogues using keystrokes becomes tedious. Today a possible solution is to use speech as input instead. Speech is a natural and comfortable medium of communication for people. It is spontaneous, informal, mastered by virtually everyone [Lefebvre et al. 1993] and does not require additional technical skills of its users. Speech recognition and understanding technologies recently have made notable progress and commercial speech recognition systems are now in actual use [Dalsgaard and Bækgaard 1994, Goddeau et al. 1994, Diehl 1994, Weinstein 1994]. Speech understanding systems are still at the stage of laboratory prototypes. However, some speech understanding systems are now being tested by making them accessible via the public telephone. An example is the Philips train timetable information system in Germany [Oerder and Aust 1994].

In Denmark a prototype system for over-the-telephone booking of tickets on Danish domestic flights, P1, has been developed (cf. Figure 1) and is currently being tested with a group of naive users. The system has been developed in the Danish Dialogue project on spoken language dialogue systems [Dalsgaard and Bækgaard 1994]. P1 is a real-time, 500 word vocabulary, speaker-independent, spontaneous speech understanding system which runs on a PC and provides users with a flexible interface to a flight ticket reservation database. The Dialogue project started in 1991 and is carried out with an effort of 28 man/years by the Center for PersonKommunikation (CPK), Ålborg University, the Centre for Language Technology (CST), Copenhagen, and the Centre for Cognitive Science (CCS), Roskilde University.

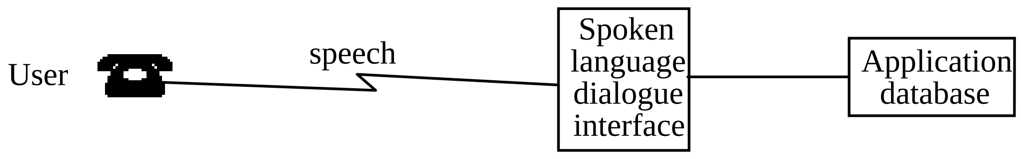

Figure 1: Abstract user view of the P1 system.

The outline of the paper is as follows. Section 2 briefly describes the P1 system architecture and the main system components including the database. Section 3 presents the dialogue functionality of the system towards the user. Section 4 describes the interaction between the dialogue model and the database (the domain model). Finally, Section 5 concludes and discusses relations between dialogue and database.

2 System Architecture and Components of P1

Figure 2 shows the architecture of P1 which is based on the SUNSTAR DDL/ICM platform [Bækgaard et al. 1992] developed in the Esprit SUNSTAR project. Communication between modules is based on messages which are passed around by a bus (the communication manager). All modules communicate with the bus through drivers. The core module is the interpretation and control manager (ICM) which interprets a dialogue description. Every time input is expected by the dialogue description the ICM looks if there is a message. Output from the dialogue description is sent as messages as well.

A user calling the system provides input to the *speech recogniser* which processes the speech signal. The speech recogniser is a further developed version of the recogniser which was developed in the Esprit SUNSTAR project. It is a speaker-independent continuous speech recogniser based on Hidden Markov Models (HMMs). In addition to user input, the speech recogniser needs predictions from the dialogue handling module on the sub-grammars and vocabulary to be used at any given point during the dialogue. The sub-grammars used in the speech recognition module are word-pair grammars represented as finite state transition networks in which the transitions represent HMMs. Viterbi search with token passing is used to find a 1-best path through the network [Brøndsted and Larsen 1994]. This path represents a string of lexical references which constitutes the output of the speech recognition module.

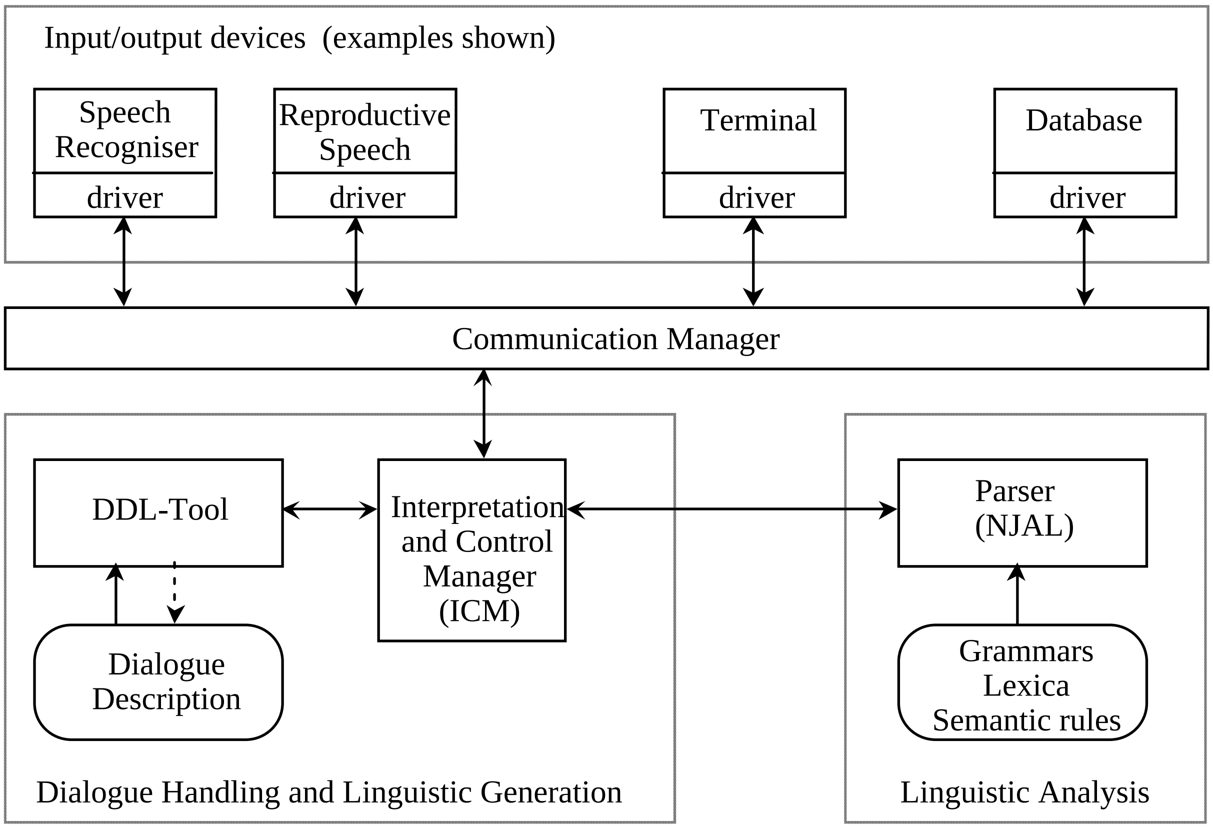

Figure 2: The main components of the P1 spoken language dialogue prototype system.

The lexical string is input to the syntactic-semantic *parser*. The dialogue handling module indicates to the parser which sub-grammars to use and which semantic objects to fill in on the basis of the input string from the recogniser. The semantic objects are frame-like structures containing a number of slots for domain relevant information. The sub-grammars used for linguistic analysis are unification-based Augmented Phrase Structure Grammars (APSGs) implemented in a formalism which is a subset of the one used in the Eurotra project [Copeland et al. 1991]. The linguistic analysis module analyses the input based on the active sub-grammars using a chart data structure and an object-oriented implementation of the Earley parsing algorithm. The parser uses semantic mapping rules for assigning semantic interpretations [Povlsen and Music 1994] which in turn are used for filling in the active semantic objects.

The *dialogue handling module* interprets the contents of the semantic objects received from the linguistic analysis module and decides on the next action to take which may be to send a query to the database, to send relevant output to the user, or to wait for the next user input [Dybkjær and Dybkjær 1994]. In the latter case, the dialogue handling module also sends predictions to the speech recogniser and the parser on the next sub-grammars and vocabulary to use, i.e. on which input now to expect from the user. The dialogue handling module, in particular the dialogue description, is discussed in detail in Section 4 below.

The output is based on *reproductive speech*. A number of words and (parts of) sentences have been recorded in advance. The dialogue description contains a simple *linguistic generation module* which selects a string of phrase names and sends it to the reproductive speech module which replays the phrases.

The *database* performs lookups, updates, integrity checks, and completions of data. The database is implemented in C++. Roughly, one part contains data on customers, travellers and flights while another part contains rules and constraints from the travel domain.

The *terminal* is used in debugging or test situations.

3 Dialogue Functionality of P1 towards the User

P1 allows users to interactively perform the ticket reservation task which is a large well-structured task well-suited for system-directed dialogue. In a well-structured task there is a prescribed amount of information which has to be exchanged. This stereotypical structure makes it possible to have a dialogue in which the system by definition has the initiative during domain communication, as is the case in P1, but which is still acceptable to users. Recordings of human-human dialogues in a travel agency showed that in reservation tasks the travel agent typically takes over and asks the questions while the customer provides the needed information after the initial customer turn [Dybkjær and Dybkjær 1993].

Originally, dialogue models were developed for ticket reservation as well as for flight information and change of reservations. However, the two latter tasks are ill-structured tasks which would require mixed-initiative dialogue to be acceptable for users [Dybkjær et al. 1994]. But the heavy technological and feasibility constraints of P1 did not allow for mixed initiative dialogue. Therefore only the reservation part was implemented.

Figure 3 next page shows an example of a dialogue with the P1 system concerning a reservation. *S* and *U* denote system and user utterances, respectively. In [Bernsen et al. 1994] a theory for task-oriented dialogues is described encompassing dialogue elements needed for different dialogue types. Below the dialogue elements needed for the system-directed P1 dialogue are described and illustrated by references to the figure.

P1 takes and preserves the *initiative* by concluding all its turns (except when closing the dialogue, *S13*) by a question to the user. The questions implicitly indicate that initiative belongs to the system. Users are only allowed to take the initiative when they want to initiate *meta-communication*. This must be done by using one of the keywords *correct* and *repeat*, *U9*. The use of keywords enables the system to simultaneously establish that the user takes the initiative and which task the user intends to perform. A *dialogue act history* is maintained to support corrections.

P1 incorporates a small amount of *user modelling* in that the system introduction can be avoided by users who already know the system, *S1, U1*.

Furthermore, P1 provides *continuous feedback* on the user commitments made, both direct *echo, S3-7, S9, S11*, and *masked echo, S2, S10, S13*. Users who accept the feedback information do not have to reconfirm their commitment as the system will carry on with the next sub-task in the same utterance.

In addition to continuous feedback, P1 offers *summarising feedback* on closing the reservation task to summarise the commitments made, *S12*.

After each system utterance the dialogue handler *predicts* the next possible user utterances in terms of sub-vocabulary and sub-grammars. Predictions are based on the *sytem focus* which is the set of sub-tasks which the user is expected to refer to in the next utterance. For instance, after *S1* the system expects a *yes* or a *no*, possibly phrased politely, and nothing else can be recognised.

All information exchanged with the user is stored in a *task record*. The information is used in domain checks at later points in the dialogue, and in the summarising feedback.

Finally, we note that the system is not a query-answering system as many traditional natural language database interfaces are. Rather, the dialogue is a series of negotiations of the values necessary to perform the task in question, and then the system handles all database access behind the dialogue with the user.

S1:	Hello, this is DanLuft reservation service for Danish domestic flights. Do you know this system?
U1:	Yes, I do
S2:	Please state your customer number.
U2:	Customer number three.
S3:	Customer number seven does not exist! Please state your customer number.
U3:	Customer number three.
S4:	Customer number three. How many persons will travel?
U4:	One.
S5:	One person. What is the id-number of this person?
U5:	Fiftyseven.
S6:	Id-number fiftyseven, Jens Hansen. Where does the journey start?
U6:	The journey is from Ålborg to Copenhagen.
S7:	From Ålborg to Copenhagen. Do you want a return travel?
U7:	No, I don't.
S8:	On which date will the travel start?
U8:	On Thursday.
S9:	Tuesday, the tenth of August. At which time of the day?
U9:	Correct.
S10:	Tuesday, the tenth of August. Is this correct?
U10:	No, on Thursday.
S11:	Thursday, the fifth of August. At which time of the day?
U11:	Forty minutes past eight.
S12:	Eight forty. You have now made a reservation for Jens Hansen from Ålborg to Copenhagen on Thursday the fifth of August, at eight forty. The ticket should be picked up in the airport at the check-in at least twenty minutes before departure. Do you want anything else?
U12:	No, thanks.
S13:	Goodbye.

Figure 3: Example session with the system. The conversation takes place on Tuesday, 3rd of August, 1993. The example is constructed, but typical for real sessions. The P1 system only speaks and understands Danish.

4 Database Interaction

In the example of Figure 3 the database is often consulted, more precisely after the utterances *U2, U3, U5, U6, U8, U10,* and *U11*. In this section we go into more detail as to how this is done.

In Figure 4 the communicational structure of the recogniser, parser, player, dialogue handler and database is shown. In particular, the database and the dialogue handler are detailed views of the database and the dialogue description of Figure 2. The files are placed outside the database since they are external, physical files. In Section 4.1 an example describes the typical processing of a user utterance, and in Section 4.2 the possible roles of the database are discussed in more detail.

4.1 The dialogue processing of utterances

The dialogue handling in P1 is task oriented. There are two classes or levels of tasks:

- *Atomic tasks* concern one item, where an *item* is a value from the application or user domains, and is tagged with current system, user, and domain status, dialogue focus, and alternative values. Moreover, all user exchanges are done within the atomic tasks, as explained below.

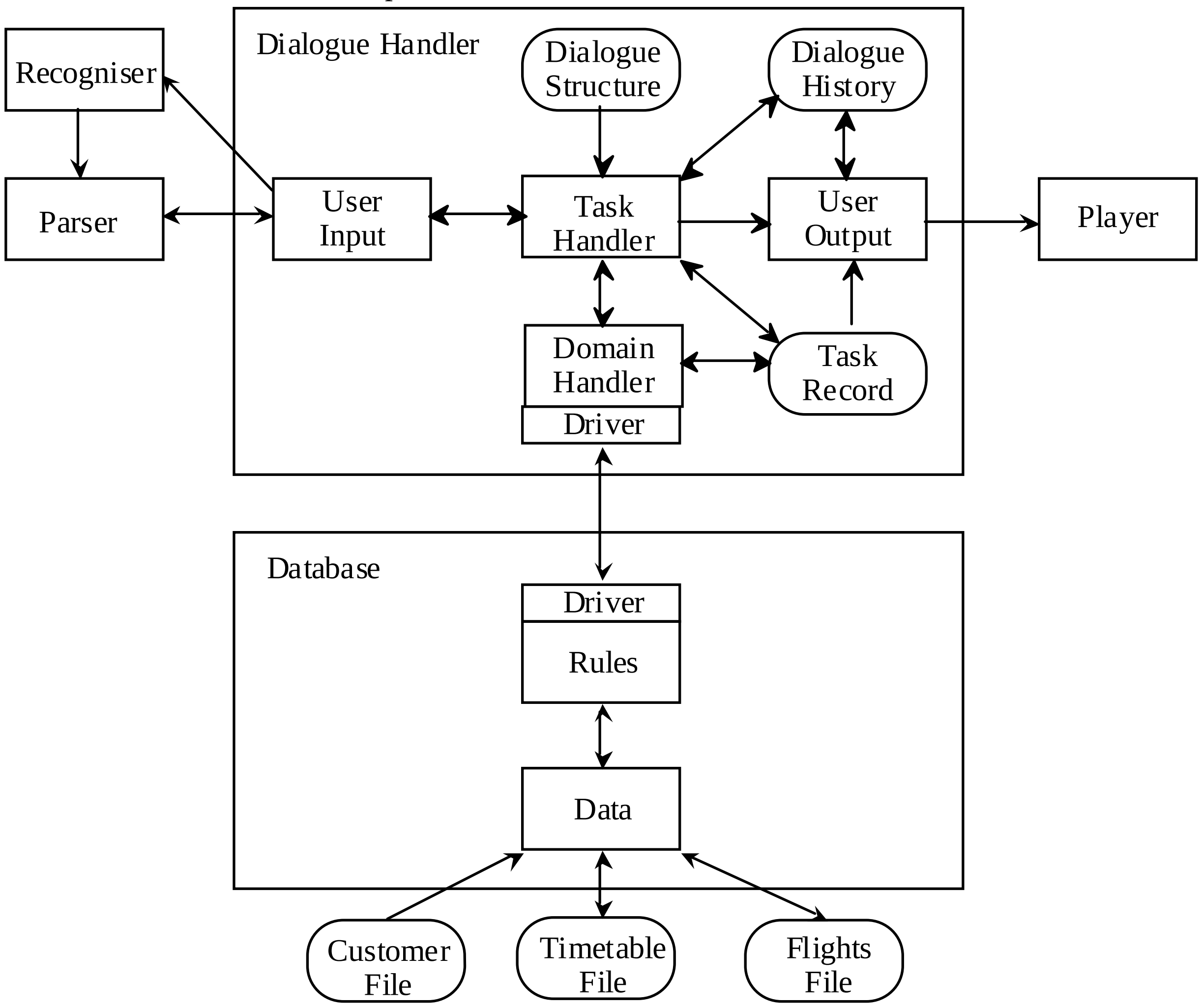

Figure 4: Architecture of dialogue handler and domain database.

- *Compound tasks* manage the temporal structure of sets of atomic tasks. Examples are the reservation task and the overall dialogue frame. Compound tasks are modelled via a *task dialogue structure*. This structure is expressed as a graph with conditional branches. The graph contains static links between tasks within a default task dialogue structure template, and the conditioned choice between these is dynamically computed on the basis of the available information, including task items and history. In addition, there are dynamic links to the previous task and to global tasks like *quit*. Dynamic links are activated via user input or via explicit system generated exceptions.

The *atomic tasks* follow a fixed scheme. The scheme has actions parametrised over the possible items, and the actions to perform are determined by the current system, user, and domain status. The actions are:

- ask the user for a value (and wait for answer),
- ask the user to select a value from a list (and wait for answer),
- ask the user if a given value is desired (and wait for answer),
- feedback of value to user,
- give the user an error message, or
- check the domain integrity of the value.

As an illustration, assume that the system has achieved the value "Copenhagen" for the departure airport, and now enters the atomic task for the arrival airport. The first thing that happens is a check of item preconditions: the dialogue handler has a table of which items the current item depends on, in the present case the departure airport which has already been established.

Next, the dialogue task handler enters a loop of user-system exchanges to obtain the required information from the user. Assume that in this loop the system has just asked the user for a value and now waits for an answer. Then Figure 5 shows the flow of information caused in the system by the user utterance. The arrow sequence refer to the modules of Figure 4, and the arrows are labeled with, to the left the type of information transferred, and to the right an example of that information.

After confirmation from the domain handler and database and after the subsequent user feedback item postconditions are checked before closing the atomic task. For instance, if a date is changed and a departure hour has previously been established, this hour value should be reestablished.

4.2 Domain handling

The domain handler is closely tied to the the task handling. In P1, for each task the domain interaction is encapsulated in a single consistency check procedure taking as argument the item record and the dialogue history. In return it may change the domain status of the current item, and deliver an item value corresponding to the new status.

In order to define the domain status of the current task, the consistency procedure may consult the database (via events and event handlers) where the main part of the domain representation is located. The database contains data on flights, customers, reservations, etc., and some auxiliary rules describing valid item values, whereas the rules of the domain related to the tasks.

The requests to the database encompass i) check of validity and integrity of new information, ii) update of the database with new information, iii) retrieval of information from the database. Of course, (i) is part of (ii,iii), too. In return, the database gives a status and relevant values. Database requests have the contents:

(request type, list of values)

and database answers have the contents:

(answer type, database status, list of values)

where the answer type matches the request type, database status is a tag, and the list of values depend on the database status. Below the database responses of P1 are explained according to the possible types of database status.

First, the request may be correct and fully specified. Then an ok-tag is returned as status, and either no values at all (case i) or a single value (case ii,iii).

Second, request may be consistent with the database, but not fully specified. An ok-tag is still returned as status, but the such underdetermined values may be treated in two different ways by the database:

- A single, appointed value is chosen (value completion). For instance, the date "the 23rd" will be interpreted as the closest, future day occurring on the 23rd, and the date item will get corresponding values for day of week, month, and year. What is "closest" must be consistent with other tasks, e.g. while `outday` is completed relative to `today`, `homeday` is completed relative to `outday`.
- A list of values is returned. For example, "morning" will become a list of feasible hour values but also errors like the non-existing departure time "7" will result in a

list—e.g. `6:45` or `7:30`, being the two closest feasible departures from Copenhagen to Ålborg.

Third, the request values may be inconsistent with the database. Then an appropriate error-tag is returned as status, and possibly some alternative values are returned. The possible error types of P1 are these:

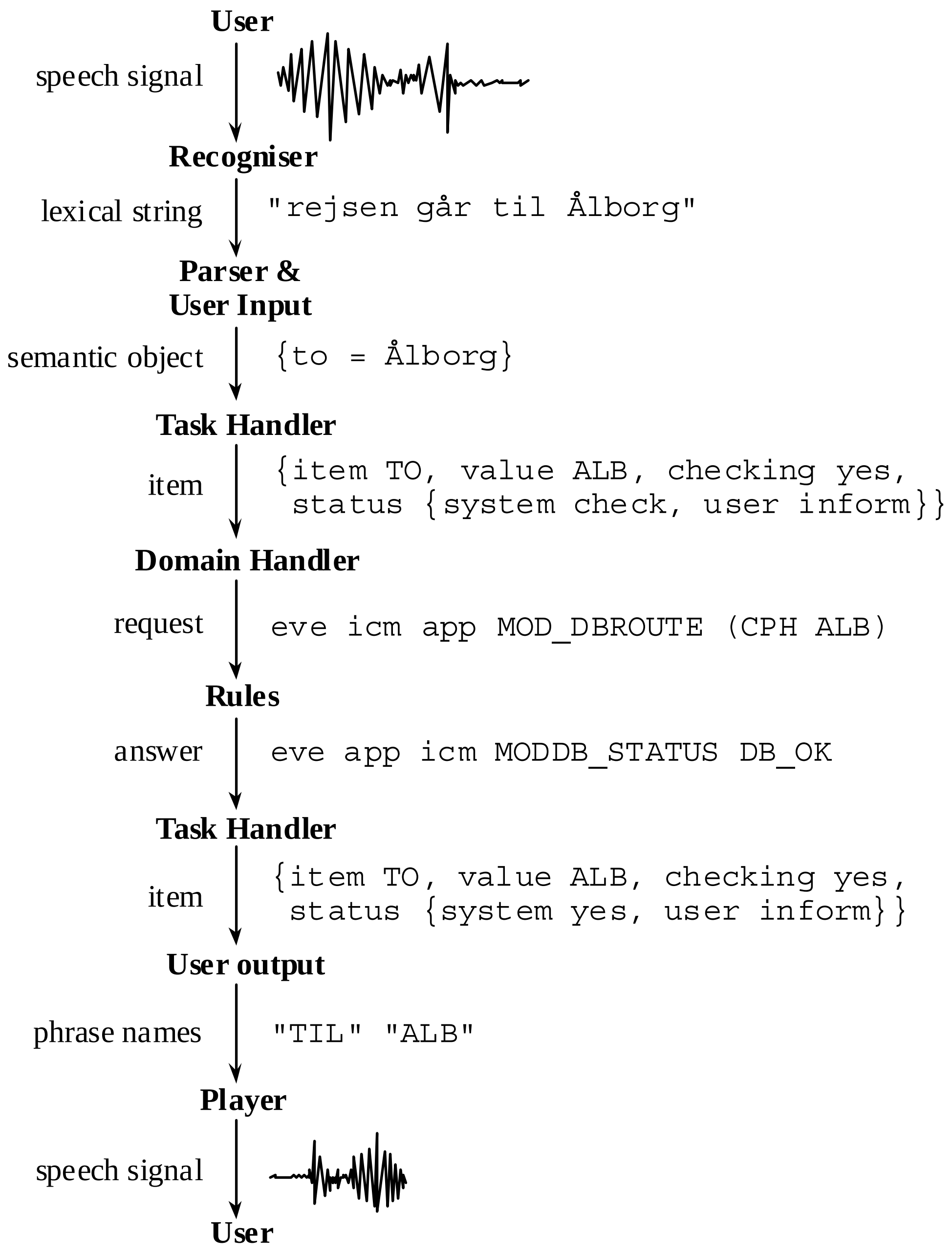

Figure 5: Typical flow from user input to user output.

Precontext: The FROM task has been completed, establishing that the travel starts in (is from) Copenhagen. Now the new task TO is initiated, the item preconditions have been checked, and the item record is `{item TO, value _, checking yes, status {system bottom, user bottom}}`.

Postcontext: The item record is completed with value `{item TO, value ALB, checking yes, status {system yes, user yes}}`. The item postconditions are checked, i.e., if the change affects any other items that should be reestablished, then the corresponding tasks are invoked. Finally, checking is set to false, and control transferred to the next task and item, typically “do you want a return travel?”.

- Format errors in the item, such as “60 minutes past”, or “February 30”. This is just marked by an error-tag, and no value is returned.
- The item is inconsistent with other items, such as the outtravel hour falling after the hometravel hour. This is just marked by an error-tag, and no value is returned.
- The item is temporarily inconsistent, such as an sold out departure hour. This is marked by an error tag, and a list of alternative values is returned.
- The item is non-existent, such as a departure hour not present in the time table. This is marked by an error tag, and a list of alternative values is returned.

Finally, there are two kinds of domain problems not handled in the database but by the domain handler which treats both kinds as preconditions of atomic tasks, cf. Section 4.1:

- Some values are missing for the computation of the current item value, such as the day of month missing in a date.
- Some values are missing for the constraint check of the current item value, for example, route and date are necessary to check an hour of departure.

All these response types are utilised by the dialogue handler. The reactions to errors are context dependent. While obtaining new information the reactions to errors are usually straightforward error messages to the user followed by a renewed questions to the user. However, if the current task turns out to be a change of a value, the situation is more complicated. For instance, if the home travel hour is changed as to fall before the out travel hour, it might as well be the latter value which is faulty.

5 Conclusion

Today, access to e.g. flight databases is usually mediated by a human travel agent who forms the flexible user interface. A system replacing the travel agent should be tolerably inferior to the human it replaces and should be able to do the tasks done by this person. This requires flexible systems, which again implies certain demands on the kind of dialogue conducted by the system.

As demonstrated through P1 and other projects the technology is about to be ready for handling the large, but still limited group of well-structured tasks acceptably. The next step, however, will be to develop spoken language dialogue systems which offer mixed-initiative dialogue. A large, additional group of tasks can be handled by spoken language dialogue systems if we can design suitable and feasible mixed-initiative dialogue systems and the tasks which can be handled acceptably by system-directed dialogue can be handled better through mixed-initiative dialogue.

The speech technology may be expected eventually to advance to a level feasible for mixed-initiative dialogues. In addition to work on mixed-initiative dialogue and improved speech recognition there is a need for better natural language analysis and generation, standard development methods and tools for reducing costs of applications and experiments, and more principled ways of separating dialogue, domain, and database.

The interface medium to P1 is a telephone but spoken language dialogue systems may also be accessed via computers and may be extended to multimodal systems. The possibility of combining different modalities will yield more flexible interfaces. Spoken input and output is fully sufficient for accessing the contents of some databases. Other databases, however, contain information which requires a different modality or a combination of modalities to provide users with the most optimal and flexible interface.

Relations between dialogue, domain and database

The development of P1 has been driven by the wish to design a dialogue that optimises the trade-off between on the one hand the desire for usable and natural conversation, and on the other hand the severe restrictions imposed mainly by the speech recogniser and the hardware. Thus the database as such plays a rather passive role in the dialogue handling. Instead, a domain handler has been designed to take care of the database communication and of more task specific constraints, in particular those affecting the default order of tasks in the converstation. This design seems rather robust and just requires the database to differentiate finely between response types—in P1, 21 error tags currently exist, and more could conveniently be used.

Other approaches exist. An interesting, almost diametrically opposed view is taken in [Smith 1991] who views task performance as theorem proving within the domain and claims that "the role of language is to supply missing axioms to complete these proofs." He further defines a sub-dialogue as "all the language interaction pertaining to one task step." This approach provides a simple and elegant framework for the system structure, and a good distinction between domain and dialogue descriptions. However, the feasible or possible proof sequences do not necessarily match a dialogue structure natural to humans. Moreover, all of the domain model is placed in the knowledge bases of the theorem prover, and as though an external database may also by viewed as a supplier of missing axioms, its addition may complicate the problem of keeping a natural dialogue structure.

The open question is when to use a careful shaping of the dialogue structure and when to use a close integration of the domain structure, and whether the two approaches can be unified in one, better framework.

Acknowledgements

The work described in this paper was carried out under a grant from the Danish Government's Informatics Research Programme whose support is gratefully acknowledged. The cooperation of colleagues at CPK and CST within the Dialogue project is appreciated.

References

[Bernsen et al. 1994] Niels Ole Bernsen, Laila Dybkjær, and Hans Dybkjær: A dedicated task-oriented dialogue theory in support of spoken language dialogue systems design, proceedings of ICSLP'94, Yokohama, pp 875-878.

[Brøndsted and Larsen 1994] Tom Brøndsted and Lars Bo Larsen: *Representation of Acoustic and Linguistic Knowledge in Continuous Speech Recognition*, report 5, Spoken Language Dialogue Systems, CPK Ålborg University, CCS Roskilde University, CST Copenhagen, 1993.

[Bækgaard et al. 1992] Anders Bækgaard, Ana Roman, and Peter Wetzel: Advanced Dialogue Design - DDL Tool and ICM, Esprit project 2094 SUNSTAR, Deliverable IV.6-2, August, 1992.

[Copeland et al. 1991] C. Copeland, J. Durand, S. Krauwer, and Bente Maegaard (Eds.): The Eurotra Formal Specifications. Studies in Machine Translation and Natural Language Processing, vol. 2, 1991.

[Dalsgaard and Bækgaard 1994] Paul Dalsgaard and Anders Bækgaard: Spoken Language Dialogue Systems, CRIM/Forwiss Workshop, München 1994.

[Diehl 1994] S. Diehl: Desktop Dictation, Byte, Vol.19, No.5, May 1994, pp145-146.

[Dybkjær and Dybkjær 1994] Hans Dybkjær and Laila Dybkjær: Representation and Implementation of Spoken Dialogue, report 6b, Spoken Language Dialogue Systems, CPK Ålborg University, CCS Roskilde University, CST Copenhagen, 1994.

[Dybkjær et al. 1994] Laila Dybkjær, Niels Ole Bernsen, and Hans Dybkjær: Different Spoken Language Dialogues for Different Tasks. A Task-Oriented Dialogue Theory, Human Comfort and Security, Springer Research Report 1994 (in print).

[Dybkjær and Dybkjær 1993] Laila Dybkjær and Hans Dybkjær: *Wizard of Oz Experiments in the Development of a Dialogue Model for P1*, report 3, Spoken Language Dialogue Systems, CPK Ålborg University, CCS Roskilde University, CST Copenhagen, 1993.

[Goddeau et al. 1994] David Goddeau, Eric Brill, James Glass, Christine Pao, Michael Philips, Joseph Polifroni, Stephanie Seneff, and Victor Zue: *GALAXY: A Human-Language Interface to On-Line Travel Information*, proceedings of ICSLP'94, Yokohama 1994, pp 707-710.

[Hendrix et al. 1978] Gary G. Hendrix, Earl D. Sacerdoti, Daniel Sagalowicz, and Jonathan Slocum: *Developing a Natural Language Interface to Complex Data*, ACM Transactions on Database Systems, 3(2), 1978, pp 105-147.

[Lefebvre et al. 1994] P. Lefebvre, G. Duncan, and F. Poirier: *Speaking with computters: A multimodal approach*, proceedings of Eurospeech'93, Berlin, 21-23 September 1993, pp 1665-1668.

[Oerder and Aust 1994] Martin Oerder and Harald Aust: *A Realtime Prototype of an Automatic Inquiry System*, proceedings of ICSLP'94, Yokohama 1994, pp 703-706.

[Povlsen and Music 1994] Claus Povlsen and Bradley Music: *Definition and Specification of the Sub-language for P1*, report 4, Spoken Language Dialogue Systems, CPK Ålborg University, CCS Roskilde University, CST Copenhagen, 1994.

[Smith 1991] Ronnie W. Smith: *A computational model of expectation-driven mixed-initiative dialog processing*, PhD thesis, Department of Computer Science, Duke University, Durham, USA, October 1991.

[Weinstein 1994] C.J. Weinstein: *Demonstrations and applications of Spoken Language Technology: Highlights and perspectives from the 1993 ARPA Spoken Language Technology and Applications Day*, proceedings of ICASSP'94, pp 337-340.