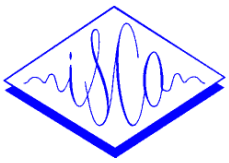

2nd International Conference on
Spoken Language Processing (ICSLP 92)
Banff, Alberta, Canada
October 12-16, 1992

AN INTEGRATED DIALOGUE DESIGN AND CONTINUOUS SPEECH RECOGNITION SYSTEM ENVIRONMENT

Boerge Lindberg, Bjarne Andersen, Anders Baekgaard, Tom Broendsted, Paul Dalsgaard and Jan Kristiansen

Speech Technology Centre, University of Aalborg
Fredrik Bajers Vej 7, DK-9220 Aalborg, Denmark

ABSTRACT

Successful and flexible establishment of a spoken dialogue system requires the integration of dialogue design, dialogue management and spoken language processing into a common system environment.

The research and development reported in this paper give details from the establishment of such an environment and its necessary development tools and devices as well as preliminary results from testing the integral continuous speech recogniser component on an artificial CAD-application.

The application is characterised by a vocabulary containing names of geometrical objects, words by which these objects may be moved, changed in size, coloured etc, such that speech activated manipulation of the objects may be observed on a display. The interactive functionality is guided by reproductive speech output telling the user if, for instance, he/she is asking the system to perform illegal actions as seen from the pragmatic/semantic component of the dialogue system.

The continuous speech recogniser part of the spoken language processing module is based on Continuous Hidden Markov Models (CHMM's) of user-configurable speech units, and the search algorithm is a token-passing Viterbi search algorithm constrained by syntactic rules stored in a finite state table (transition network). In addition to this, stub functions are associated with each state in the network, implementing simple conditions and actions and thereby enabling the implementation of more advanced grammars than otherwise available with traditional finite state tables. These stub functions may also be applied in handling of non-static conditions in the recognition process, such as pragmatic constraints.

The artificial CAD application has been used to test the system, and the performance is reported for a traditional grammar both with and without stub augmentation.

Results from these experiments show that improvements of the recognition rate may be obtained by implementing pragmatic and semantic constraints as directly associated stub functions within the states of the network.

1. INTRODUCTION

In at least two ESPRIT projects (SUNSTAR and SUNDIAL) much efforts have been spent during the last three years on designing common and flexible architectures for dialogue controlled, voice activated systems to be used as the basis for future real-world applications. To a certain degree this research and development parallels already ongoing similar activities as reported in [1,2,3,4,5].

In the research reported in this paper, the goals have been to develop the basic system architecture for a flexible spoken dialogue system in which the dialogue design, the dialogue management and the spoken language processing modules are integrated via standardised interfaces - the latter having been defined, to a large extent, within the ESPRIT Project 2094, SUNSTAR. Taking this modular approach to system architecture, the introduction of e.g. new continuous speech recognition and/or text-to-speech synthesis devices, will be a rather straightforward task, provided the requirements to standardised interfaces are met.

This paper concentrates on the three main issues of the above given strategy, namely the dialogue design, the dialogue management and the spoken language processing.

2. DIALOGUE DESIGN

The design and specification of the expected user dialogue is performed in an off-line task in which the Dialogue Developer specifies the dialogue which must govern the user-machine interactions via voice input and synthetic speech output. This specification is performed by applying a graphically oriented tool, in which interactions are described by means of a Dialogue Description Language (DDL), [6,7].

This tool, the DDL-Tool, has an extensive set of graphical functions for manipulating the dialogue specifications, as well as facilities for printing diagrams and performing syntax checks.

The DDL is a compound language encompassing three levels:

- *The graphical level (DDL/SDL)*, which is the most abstract level, describing the overall control structure of the dialogue via the use of (recursive) flowcharts. The layout of the graphical level is based on the 'Specification and Description Language (SDL)', which is standardised by CCITT [8]. In addition to this, there are a number of symbols concerned with the natural language processing part of the system - most notably user input/output symbols and user input/output grammar symbols. In traditional computer linguistics these symbols correspond to terminals and non-terminals, respectively.

- *The frame level (DDL/FL)*, which is the medium abstraction level, declaring details of the dialogue. On this level, lexical entries, semantic representation structures, etc. are defined.

- *The textual level (DDL/TL)*, which is the most concrete level, implementing e.g. procedures which may handle computations required during the dialogue.

An example of a typical sequence of interactions defined by means of the DDL-Tool is shown below in Fig. 1. However, the example only shows spoken language elements of a dialogue - in general, the DDL-Tool supports dialogue design in which multi modal events from e.g. keyboard, DTMF or mouse may occur.

Further details of the DDL-Tool and its practical use in a real-world application may be found in [6,7].

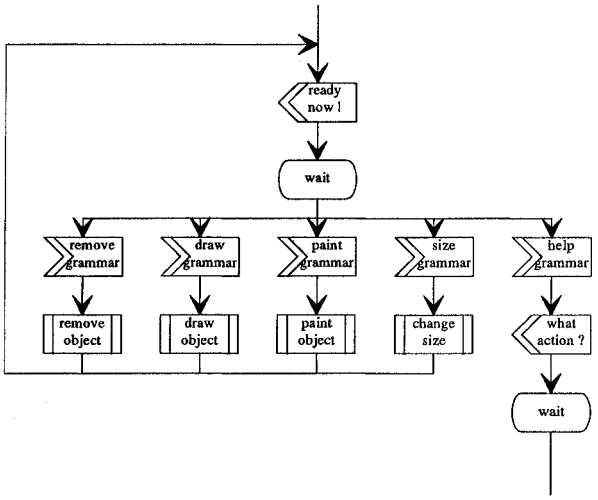

Figure 1 Example of a Dialogue Description Flow Graph, containing symbols for user input and user output grammars, procedure calls and wait states.

3. DIALOGUE MANAGEMENT

In order to execute and control the dialogue according to the defined dialogue description, an Interpretation and Control Module (ICM) is available by which all commands and messages within an application are interpreted.

The operation of the ICM is based on a dialogue description in a pure textual ICM-language, which is generated by the DDL-Tool as a compiled version of the defined dialogue descriptions.

The ICM communicates with the various input/output devices through a communication manager which enables standardised device association to the dialogue system. For a detailed system architecture, see [6].

The overall execution of the dialogue is based on the concept that a dialogue can be viewed as a state machine, which is always in a clearly defined state, waiting for input from the user.

In a typical dialogue within an application where the user communicates with the system via spoken input, a wait state is normally preceded by a speech output in the form of a prompt from the system (on the DDL/SDL level a user output symbol or a user output grammar symbol) and succeeded by an isolated word or a sentence input (user input symbol or user input grammar symbol) from the user, see Fig. 1.

Input grammars (context free grammars (CFG's)) defined at the graphical level of the DDL are used for parsing user input by a built-in parser within the ICM. The semantic result of the parsing is then filled into the slots of the semantic representation structures, defined at the frame level of the DDL.

Since each wait state always defines a certain acoustic focus, the implementation of the ICM dialogue manager as a state machine has the effect of reducing the task for the recogniser within the spoken language processing module, since only a subset of the grammars and vocabularies characterising the full dialogue are to be processed.

4. SPOKEN LANGUAGE PROCESSING

Spoken language processing involves both speech input processing and speech output processing. The speech output processing part contains, at the moment, a reproductive speech coder for replaying sequences of prerecorded words or phrases. This part will not be further described in this section. However, as indicated in sections 2 and 3 there are no limitations within the DDL-Tool and the ICM for supporting a more advanced speech output device such as a speech synthesis component, generating speech output based on textual specifications.

4.1 Speech Input Processing

The speech input processing part contains a speaker-independent continuous speech recogniser which integrates acoustic processing and natural language processing.

The natural language processing takes place fully in accordance with the specified user-machine dialogue by downloading the corresponding grammars and vocabularies from the ICM. As the process, of designing the dialogue is highly interactive and the downloading into the recogniser is done automatically, it is a rather straightforward task to make changes in the dialogue descriptions, and subsequently observe the effects on the recognition process.

The architecture of the speaker-independent continuous speech recogniser is an augmented version of the SUNCAR recogniser [9], which is based on Continuous Hidden Markov Models (CHMM) of non-tied mixtures.

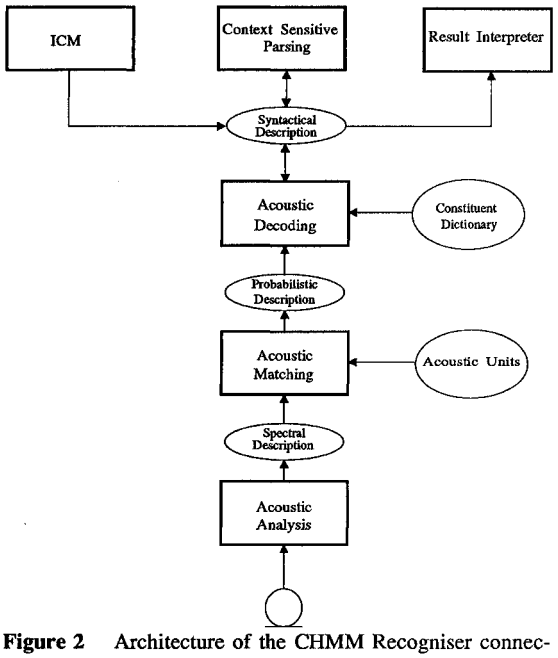

Figure 2 Architecture of the CHMM Recogniser connected to the ICM dialogue management module.

The architecture of the CHMM recogniser and its connection to the ICM is shown in Fig. 2.

In the *acoustic analysis* module, the speech signal is sampled at 8 kHz, preemphasised and limited to telephone bandwidth and then transformed into the cepstral domain in which time-derivatives are added to form a composite cepstral vector (8 cep. + 8 delta cep.) every 10 ms based on 20 ms of Hamming filtered speech.

The *acoustic matching* module provides a probabilistic description of the acoustic unit inventory based on the output from the acoustic analysis. This inventory may be phonemes, triphones, whole words, etc. and may include sink/filler models for handling extraneous speech input events.

It is worthwhile noticing that the acoustic analysis module and more important the computational expensive acoustic matching module are not dependent on information from the higher levels of processing and thus are suitable for distributed (parallel) processing implementations.

Input to the *acoustic decoding* module is the probabilistic description of the acoustic units, a dictionary of how the lexicon items are composed from the acoustic units and finally the grammar imposed syntactical constraints on the possible sequences of lexicon items, as specified in the dialogue description.

The *syntactical description* is forming the natural language constraints on the recogniser. It is implemented as a state transition network which, prior to the recognition phase, automatically is derived from the syntax description given during the dialogue specification. In the case of recursions within the syntax description, the network is expanded into a finite depth state transition network.

The state transition network remains fixed until the recognition is terminated at sentence level, and a new wait state is reached within the dialogue.

The acoustic decoding algorithm is based on the token-passing principle, as introduced in [10]. Tokens containing scores and path identifiers are continuously propagated until the whole utterance has been matched, in which case the best last emitted token in the terminal state of the network is used for tracing back to decode the recognised sequence of lexicon items.

This simple concept has, however, been extended by introducing the so-called stub functions [11]. Each state in the network has one stub associated to it, and this stub may e.g. implement simple conditions and actions. This enables the incorporation of more advanced grammars than otherwise available with the traditional transition network and furthermore it enables *context sensitive parsing* to take place at various positions in the network during the recognition process.

Thus, the main purpose of the stub functions are to integrate non-static types of conditions such as pragmatic constraints - in contrast to traditional N-best implementations, in which these constraints are implemented in a postprocessing stage.

The task of the *result interpreter* is to terminate the recognition in progress, either by observing a stable situation in the terminal state of the network or by observing extraneous speech input surrounding a syntactically correct sentence.

At present, the recogniser runs on a single AT&T DSP32C plug-in board which also contains a speech recording module and a module for reproductive speech output. If the speaker independent recogniser models whole words by 10-state CHMM's and triphones by three-state CHMM's, it can handle approximately 100 whole word models or 300 triphone single mixture models in real time. The ICM and the DDL-Tool run on SUN Sparc Workstations.

5. EXPERIMENTS AND RESULTS

An artificial CAD application has been defined for testing the system. The CAD application allows the user via spoken input to draw, paint, move, delete, change size and locate different types of graphical objects on a display.

The application consists of a number of wait states, of which the state, from where the user manipulates the graphical objects, is the most important. Each wait state of the dialogue is associated with a particular grammar (or set of grammars).

Effort was put into the design of a low perplexity "semantic" finite state network, which enabled both syntactic and semantic criterions to be met already during a 1-best recognition algorithm.

In order to access also simple pragmatic criterion in the recogniser, the FSG was augmented with conditions within the stubs. This concept attempted to integrate acoustic, syntactic, semantic and pragmatic knowledge as opposed to modular (time consuming) N-best systems, where syntax and semantics are used as selection criterions in a postprocessing parsing module. The pragmatic validity of a sentence candidate was evaluated on the basis of predefined pragmatically appropriate situations (scenarios) of what so far has been drawn, painted etc (as it makes no sense to delete graphical objects which have not been drawn, put a color to an object which it already has etc). For each state of the grammar network where a pragmatic condition is attached to the stub function, a transition cost is generated for every outgoing arc. Thus, the effect of the conditions are that transition costs are dynamically computed, as opposed to traditional static statistical grammars. In other words, the linguistic probability is in this case based on fundamental assumptions about making sense in a context, as opposed to statistics computed on the basis of language corpora.

5.1 Test results

The recogniser was tested on a database with 690 sentences and a vocabulary of 83 words. In order also to test the stub functions, a scenario was set up for each of the 690 sentences. Pragmatic conditions were added as augmentations to the basic FSG described above and implemented into the recogniser as follows: Before the main loop of the recognition algorithm the scenario was defined and compiled into tables and structures accessible from within the stub functions.

For each frame within the recognition algorithm, pragmatic conditions were evaluated for the best state only. The conditions manipulated the log-likelihood for each token associated with the outgoing arcs of the best state, as these were either set to a value, which practically "blocked" the transition, or passed on with no changes. This implementation of the stub functions merely slowed down the system imperceptibly. A general conclusion regarding increase in computational complexity is hard to point out as the complexity of pragmatic conditions is highly dependent of the state and the application in general.

The addition of conditions results in a decrease of the grammar perplexity from 6.64 to a lower value depending on the

pragmatic situation. Results from the experiments with the stubs showed an improvement of sentence recognition rate and word accuracy by 3.3 and 1.5 percent point respectably.

Table I
Performance on CAD test-sentences using a FSG with and without stub augmentation.

Grammar	Perplexity	Sent. Acc.	Word Acc.
FSG	6.64	91.0	95.8
Stub augm. FSG	< 6.64	94.3	97.3

As opposed to the standard grammar, the version augmented with conditions mainly produced confusion pairs which semantically may be classified as unimportant.

6. CONCLUSION

This paper has reported on a spoken dialogue system environment, in which dialogue design, dialogue management and spoken language processing are integrated modules communicating via standardised interfaces.

The DDL-tool enables interactive graphically oriented dialogue design with facilities which are very important for a flexible design of todays realistic spoken dialogue systems.

The ICM dialogue manager provides the necessary control of a multi-media system, in which devices are associated via a communication manager.

The spoken language processing modules perform speech input processing and speech output processing. Focus in this paper has been directed to the speech input module which consists of a continuous speech recogniser which integrates acoustic processing with natural language processing. This integration has been extended from including only traditional syntactical constraints to also including pragmatic constraints by introducing stub functions. These stub functions are associated with each state in the transition network and may be used to implement actions and non-static conditions.

On the basis of a simple (low perplexity) artificial CAD-application it has been shown that pragmatic constraints may successfully be integrated into, rather than separated from, the processing, and that this integration results in an improvement of the recognition performance.

Future work will include comparisons with often reported N-best algorithms in order to further clarify the achievements of introducing stub functions in respect of performance and computational demands.

Acknowledgments

The research reported has partly been funded by the Danish Technical Research Council by the frame programme "Natural Spoken Language Processing in Application Oriented Dialogue Systems" and partly been done within the ESPRIT project SUNSTAR.

References

[1] P. Price, V. Abrash, D. Appelt, J. Bear, J. Bernstein, B. Bly, J. Butzberger, M. Cohen, E. Jackson, R. Moore, D. Moran, H. Murveit and M. Weintraub (1990): "Spoken Language System Integration and Development", Proceedings ICSLP 90, pp 729-732.

[2] H. Höge (1990): "SPICOS II - A Speech Understanding Dialogue System", Proceedings ICSLP 90, pp 1313-1316.

[3] V.W. Zue, J.R. Glass, D. Goddeau, D. Goddine, H.C. Leung, M.K. McCandless, M.S. Phillips, J. Polifroni, S. Seneff and D. Whitney (1990): "Recent Progress on the MIT VOYAGER Spoken Language System", Proceedings ICSLP 90, pp 1317-1320.

[4] W. Boogers (1991), "Dialogue Construction By Compilation", Proceedings EUROSPEECH 91, pp 853-856.

[5] I. Nogaito, M. Takahashi, S. Kuroiwa, F. Yato (1991): "Dialogue Management In An Extension Number Guidance System", Proceedings EUROSPEECH 91, pp 857-860.

[6] A. Baekgaard, P. Dalsgaard (1990): "Tools for designing dialogues in speech understanding interfaces.", Proceedings ICSLP 90, pp 1249-1252.

[7] A. Baekgaard (1992): "Dialogue Description Language", Doc STC-DDL-2.1, EPRIT Project 2094, SUNSTAR, March.

[8] CCITT Recommandation Z.100 (1988): "Specification and Description Language SDL", APIX-35.

[9] B. Lindberg, B. Andersen, J. Kristiansen (1992): "SUNCAR Functional Description", Doc. WPIV.STC.008, ESPRIT Project 2094, SUNSTAR, March.

[10] S.J. Young, N.H. Russell, J.H.S. Thornton (1991): "The Use of Syntax and Multiple Alternatives in the VODIS Voice Operated Database Inquiry System", Computer, Speech & Language, 5.

[11] T. Broendsted (1992): "Interface between SUNCAR and a NLS Component", Internal Draft, Feb.