Spoken Language Dialogue Systems
Report 1.1, May 1992

State-of-the-art Survey
of
Spoken Language Systems

(preliminary version - do not cite or copy)

STC - Speech Technology Centre, Aalborg University.
CCS - Centre for Cognitive Science, Roskilde University.

CLT - Centre for Language Technology, Copenhagen University.

This report has been written under the responsibility of the Speech Technology Centre, Aalborg University.

The following persons have contributed to the report:

Lars Bo Larsen
Tom Brøndsted
Speech Technology Centre

Hans Dybkjær
Laila Dybkjær
Centre for Cognitive Science

Claus Povlsen
Brad Music
Centre for Language Technology

Preface

The present report is the first in a series from the strategic research programme : *Natural Language Processing in Application Oriented Dialogue Systems*. The programme is sponsered by the Danish Technical Research Council, and runs for a four year (primo 1991 - primo 1995) period. The participiants are the *Speech Technology Centre (STC)* (coordinating partner) at Aalborg University, *Centre for Cognitive Science (CCS)* at Roskilde University, and *Centre for Language Technology (CLT)* at Copenhagen University. The objective of the programme is to develop a human-machine spoken dialogue system prototype.
As a first step towards this goal, this report is concerned with a state-of-the-art survey of existing spoken language systems, mainly in Europe and the US.

Keywords :

Spoken language systems, human-machine dialogue systems, speech technology, speech recognition, natural language parsing, sublanguages, discourse, Knowledge acquisition, knowledge representation.

Danish Summary :

Denne rapport er den første i en serie fra Rammeprogrammet "Behandling af Naturligt Sprog i Applikationsorienterede Dialogsystemer", og er støttet af Statens Teknisk Videnskabelige Forskningsråd. Projektet er fireårigt med start i 1991, og projektpartnerne er : *Center for Taleteknologi*, AUC (projektansvarlig), *Center for Sprogteknologi*, KU og *Center for Kognitiv Informatik*, RUC.

Projektets formål er at udvikle prototyper af talestyrede menneske-maskine dialogsystemer. Herved skal forstås, at systemerne indenfor et begrænset emne er i stand til at føre en talt dialog med en person. Som eksempel er valgt informationer om indenrigs flytider, -priser mv.

Rapporten gennemgår en række europæiske og amerikanske systemer, som ifølge vores vurdering repræsenterer det nuværende "state-of-the-art" på området. Desuden gives et overblik over de videnskabelige discipliner og metoder, der benyttes ved udviklingen af dialogsystemer med talt input. Taleteknologi, Parsing af Naturligt Sprog, og Kunstig Intilligens.

Contents

0. Introduction

In modern society people communicate with computers in numerous situations in everyday life. The communication is often not recognised as such, as when computers are embedded in machines such as washing machines, tv-sets, cars, telephone exchanges, etc. The communicative aspect of dealing with computers is more apparent when computers are used, e.g., as word processors in office environments, as home PCs, in banks, etc., where generel input-output devices are used such as keyboard, mouse and screen display.

Such devices are not always the optimal ones to use in human-computer communication, however. It is, for instance, much faster to dictate a text than to type it. Since there is an ever increasing demand for communication, retrieval and presentation of information for which the computer is the natural instrument, the need for human-computer communication devices appropriate to the tasks at hand will rise in the future. One example of a natural application of computers is long distance database access systems such as home banking and telephone information services. Currently, because ordinary users rarely have access to keyboard and mouse for operating such systems, tasks are performed either via a human operator or via tedious series of telephone keystrokes.

An obvious alternative possibility is to let users communicate directly with the computer by using their voice.

Extensive research towards this goal has been undertaken in the last two decades in the areas of speech technology, natural language processing and artificial intelligence. This combination of efforts has now led to the possibility of constructing computer systems which are capable of not only recognising and reacting to single-word commands, but also of recognising, understanding and synthesizing human speech and hence to participate in spoken dialogue. At the present stage, however, the latter is true only for tasks within a limited and well-defined domain. Research centres in Europe, the US and Japan presently focus on integrating the above-mentioned disciplines in an effort to develop more advanced spoken dialogue system prototypes.

Spoken dialogue systems

As the term "spoken" implies, the mode of interaction in such systems is speech. Some important aspects of a spoken dialogue system are that the system is capable of understanding spoken language and of reasoning about concepts within a given (limited) domain. For instance, the system should be able to pose questions of clarification in order to obtain additional information needed for performing a given task. The main modules of a spoken dialogue system are:

_An acoustic decoding module (a speech recogniser) which performs the decoding of the acoustic signal into phonetic/linguistic elements.

_A natural language parsing module which analyses the output from the speech recogniser

and builds a semantic representation of the spoken utterance.

_A dialogue handling module which interprets the semantic representation and decides what action to take. This module also keeps track of, e.g., the possible plans and goals of the user.

_An answer generation module which generates answers or queries to be synthesized.

_A speech synthesis module which generates synthetic speech (either true text-to-speech synthesis or synthesis of pre-recorded human speech).

The present project

The present project is aimed at developing a speech understanding system which will allow users to obtain information and make reservations on Danish domestic flights. The goal is to develop a small prototype within the coming year and then use experiences with this prototype as a basis for developing a more advanced prototype. The final objective is to produce a prototype that can lead to commercial applications.

The first prototype will be very limited as regards the domain knowledge involved as well as the scope of possible communication, primarily because only a small number of words can be recognised. These limitations are mainly due to difficulties faced in the speech recognition task, where a number of problems still have to be overcome before large vocabularies can be handled with little or no linguistic constraints imposed on them. If the first prototype is to produce real-time answers it will probably be realistic to permit about 400 word forms in the recogniser.

Desirable functionalities of the system to be developed are:

1. Information on departures and arrivals of flights.
2. Information on prices.
3. Ticket reservation.
4. Information on connections to other means of transport.

All these functionalities are within the capabilities of current database technology but the last one may be infeasible (at least in the first prototype) due to limitations in the vocabulary size of the speech recogniser.

Objectives of this report

The present state-of-the-art report has the following objectives:

_to present a short list of spoken dialogue systems representing the current state-of-the-art (Chapter 1);

_to present an survey of techniques currently applied in the three areas of research

involved, each of which is addressed by one project partner, namely:

_speech technology, focusing on speech recognition (Chapter 2);

_natural language processing, focusing on parsing techniques and linguistic aspects of sublanguages, (Chapters 3 and 4, respectively);

_AI and cognitive science, focusing on discourse handling, knowledge acquisition, and formalisms for knowledge representation (Chapters 5, 6 and 7, respectively).

To aid readers unfamiliar with the technical terms, an extensive list of terms has been placed at the end of the report. Also, it might be advantaguous for readers without detailed knowledge of the involved technical concepts to read the sections 2 to 7 about the specific topics before returning to section 1. **1. Examination of existing spoken dialogue system architectures**

The first implementations of spoken dialogue systems were made about 15 years ago. Since then a number of systems have been developed—a detailed list of existing spoken dialogue systems and some of their main features is given in Appendix A—but almost all of them are research systems. Speech recognition and understanding techniques have still not matured to the point where they can be used in computer systems allowing users to conduct a relatively natural conversation involving a large vocabulary. If the vocabulary is large, which in this case means more than a few hundred words, and a relatively complicated dialogue and discourse structure is permitted, the system response will take too long and it will require a large effort to make the system speaker-independent. If real-time and speaker-independence are required features the vocabulary will be small and the dialogue structure very simple. Usually it will only allow for one-word- or at least very short replies.

Some of the more ambitious and recent research systems from both the US and Europe are described below in order to provide an overview of the state of the art. Information on various aspects of these systems such as financing, duration, goal, domain, vocabulary, speaker-dependence and especially system architecture is given to the extent that it is available.

In order to compare the performance of spoken dialogue systems it is necessary to have an evaluation methodology. Existing evaluation methodologies are described below in section 1.3. Finally a conclusion sums up what can be said in general about recent spoken dialogue systems.

1.1 US DARPA systems

MIT: SUMMIT+TINA [Phillips et al. 1991, Goodine et al. 1991, Seneff 1989, Zue et al. 1991]

MITs (Massachusetts Institute of Technology's) speech understanding system is composed of the phone based speaker independent continuous speech recogniser SUMMIT and the NLS-component TINA. The system has been coupled both to the Air Travel Information System ATIS and the VOYAGER applications which assists the user in locating hotels, restaurants and other sites in Cambridge, Massachusetts.

SUMMIT transforms waveforms into segment lattices and determines acoustic scores for phone candidates. In the most recent version left-right phone context experimentally is taken into account using mixtures of diagonal Gaussians. Lexical access is accomplished by a Viterbi search connected with syntactic constraints. TINA uses a context-free network grammar coupled with constraints on features and augmented with statistical probabilities of transitions. The probabilities are generated on the basis of external text corpora relevant to the ATIS- and VOYAGER-tasks.

Three different ways of interfacing SUMMIT with TINA have been explored: 1) A traditional N-

best interface, where syntactic constraints in SUMMIT are provided by a word pair grammar and TINA is used as a post-processor, choosing the first sentence candidate that can be parsed; 2) An interface where TINA resorts the N-best sentence candidate using parse probabilities. In this case, the chosen sentence candidate must not be the same as the first parsable hypothesis provided by the N-best algorithm; 3) A "tightly coupled" interface, where SUMMIT uses a backward N-best Viterbi search with word-pair constraints and TINA predicts a set of next-word candidates for each partial theory during a succeeding A* forward search. The terminal nodes are transformed to lists of licensed words and passed on to the recogniser which initiates the A* forward algorithm scoring the paths to each word. This interface does not involve parse probabilities.

The parse tree delivered by TINA is walked through for filling semantic frames, which in turn are translated to SQL commands for database inquiries. The walk-through process steps through each syntactic node, each of which has a syntactically motivated name (e.g. "dir-object"). Many of these names are associated with certain semantic (role) names, for example "dir-object" is associated with the semantic name "theme".
Each semantic name is in turn associated with a number of functions which control how and what frames can be filled by the given information. Thus when walking through the tree, upon encountering a "dir-object" the functions associated with "theme" are called upon to instantiate some frame. The resulting mostly empty frame is passed on for further processing, with the expectation that it will be filled out with information from other nodes later in the walk-through.

The resulting semantic frames are used as a basis for verbal responses to the user, as input to a discourse history (used for providing context to subsequent queries, and for resolution of anaphoric references), and as a basis for generation of SQL commands to the backend ATIS database. This latter step in the processing of an utterance actually has some interpretive capability. For example, the following frame representing
a flight might be instantiated from a user's utterance:

[flight qset from: Boston to: SFO]

This frame is missing the flight number. The table-driven mechanism used for generating SQL commands discovers this missing information, and can respond by generating an SQL query for retrieving the flight numbers satisfying the information that is available, i.e. from Boston to SFO (San Francisco). In this way, the mechanism can be said to interpret this frame as a query for further flight information. (See Zue et al. 1991, pp.946-7.)

The system has been tested with a test set containing both "possible" sentences (within the domain) and "impossible" sentences which cannot produce an answer from the system. In terms of overall scoring using the DARPA metric which favours false rejections ("No Answer" results) for misrecognitions ("False Answer" results), the best results were achieved with the "tightly coupled" interface. In optimizing the system with different kinds of rejection, problems arose, as many possible sentences were rejected.

CMU : CMU-ATIS SYSTEM [Lee 89 Young & Matissa 91, Young 91]

CMU (Carnegie Mellon University, Pittsburgh) has been one of the leading sites both with regard to speech recognition systems (the SPHINX system [Lee 89), and speech under- standing systems (the MINDS system e.g. [Young et. al. 89). This description is based on the latest version (ultimo 1991), as referenced above.

The CMU-ATIS system comprises four separate modules : The SPHINX continuous speech recognition system, the PHOENIX natural language interpretation module, the MINDS II dialogue control module, and the SOUL knowledge based reasoning system.

The SPHINX speech recognition system is based on a mixture of subword models (triphones) and function word HMM models. The language model used is a bigram. The recognition algorithm is a 1-best, i.e. the recognition result is one sentence hypothesis. No information about the speaker-dependence is given. The sentence hypothesis is passed to the PHOENIX module, which is a robust case frame parser designed to process spontane- ous speech. PHOENIX produces a matrix containing all meaningful phrases in the hypothesis, and a beam of possible interpretations. Using a set of slot filling heuristics, as many case frame slots as possible are filled out by recognized phrases.

A characteristic feature of spoken input is its uncertainty and unpredictability. It is therefore very difficult to express, in advance, the expected word and constituent order in whatever formal grammar is chosen. The CMU approach to dealing with this difficulty has been to make the syntactic and semantic analysis lax and less restrictive compared to other NLP interface systems (based on well-formed written input). The few constraints on the input, however, will often lead to incomplete linguistic descriptions and thus several interpretations of a given utterance (overgeneration).

In order to extend/enlarge the linguistic description an extra system component SOUL has been added to the CMU system, which is called every time the semantic interpretations generated by the case frame parser PHOENIX are either ambiguous or inadequate (incomplete). The input to SOUL is the output string processed by the recognition module, (with the probability scores for the recognized words) and the phrase matrix and the beam of the best interpretations (case slot fillings) from the PHOENIX module.

The improvement/augmentation of the semantic interpretation is done by using domain specific and extra linguistic knowledge, (organized into a multilayered frame base systems of hierarchies), and inferencing procedures: abductive reasoning and constraint satisfaction techniques, respectively.

The abductive reasoning component uses the extensive knowledge base to refine the semantic and syntactic interpretation. On the basis of this augmented representation, candidate phrase matches are evaluated in order to find alternative and more meaningful combinations of phrases.

The constraint satisfaction mechanism spots those combinations which violate domain specific constraints. If for instance the user puts as a query:

What is the shortest flight from Dallas to Fort Worth?

the user will be told that the request is unanswerable. As Both Dallas and Ford Worth are served by the same airport and as the implemented relational constraints demand that the fillings of the origin (Dallas) and the destination (Fort Worth) must be served by different airports, SOUL will regard the query as meaningless.

The SOUL component interacts with the other modules of the CMU system. If SOUL finds one

single and acceptable interpretation, it passes the result to both the dialogue module and data base. However, if SOUL detects a sequence of mis-matched words, the dubious region of the SPHINX output is identified and delimited and hypotheses regarding the semantic content of the mis-match words are made. These hypotheses are then used to find appropriate nets of lexical items, which are merged into a finite state grammar (instead of the generic bi-gram grammar) for reprocessing the original input.

Furthermore it should be mentioned that when hooked into MINDS II, SOUL uses additional knowledge of inferred speaker intentions (plans,goals) to further augment semantic interpretation of the spoken input.

1.2 European systems

In this section three European spoken language systems whose development began in the late 'eighties are described: SPICOS II, VODIS II and SUNDIAL. Especially SUNDIAL is of interest to the current project. It is the most recent among the three systems, its domain is flight information and reservation, and it is intended to operate in real-time. It is very likely that the current project could benefit from closer contact with the SUNDIAL project.

SPICOS II

SPICOS II (Siemens-Philips-IPO Continuous Speech understanding and dialogue system) is a German-Dutch system which is a further development of SPICOS I. The project is funded by the German Federal Ministry for Research and Technology. The goal of the project is a man-machine dialogue system that is able to understand fluently spoken German sentences and in this way to provide voice access to a database.

A prototype has been implemented which allows the retrieval of the contents of a database containing information about the SPICOS project itself, e.g., project members and communications such as letters and scientific papers. The mode of communication is cooperative, i.e., it is relatively free for the user. The user may—whenever he likes to—switch to another topic and he may refer back to previous utterances. In cases of ambiguity or error the system will question the user to clarify the problems in the dialogue.

The system allows continuous speech, is speaker dependent and has a vocabulary of about 1200 full words. It expects grammatically correct input. The input is either one-word replies or questions formulated as a whole sentence.

The system consists of the following components:

A *speech recognizer* takes as input a speech signal from the user. The acoustic analysis draws especially on three sources: an inventory of subword units which are phonemes based on Hidden Markov Models; a pronunciation lexicon based on 44 phoneme symbols including symbols for glottal stop and silence; and a language model for rejecting reject ill-formed sentences. Three

different language models have been tested [Ney and Paeseler 1988]: with no language constraints, a finite state network, and a stochastic trigram model based on word categories. The finite state network seems to produce the best results. However, according to [Niedermair et al. 1990] a bigram language model based on categories is used in their project. The three sources interact via the search procedure which is a left-to-right time-synchronous Viterbi-like beam search. The output from the recognition module is the N-best scored sentence hypotheses.

The *linguistic component* consists of a lexicon of full words, a syntax represented as an augmented phrase structure grammar (APSG), a modified left-to-right top-down chart parser, a semantic network for the reduction of syntactically well-formed but semantically inadmissable structures, anaphora resolution, generation of semantical representation in a typed logical language using lambda formalism suitable for database queries, and a separate representation of discourse referents and presuppositions. The parser (left-to-right char parser) checks the top-N hypothesis until it meets a well-formed and semantic admissable candidate. The parsing module is characterized by the attempt to apply syntactic and semantic constraints simultaneously while maintaining the conceptual independence of the two levels (as apposed to prevalent speech processing apporaches based on "semantic" grammars). The early application of semantic restrictions (including anaphora resolution) is achieved by checking the semantic compatibility of content words within the phrase structure rules. By applying all available knowledge as soon as possible, the search of the parser is reduced. To simplify the linguistic structures, certain restrictions have been introduced such as leaving out relative clauses and passive voice, and breaks, ahs and the like are not taken into account.

A *dialogue handler* coordinates the different parts of the system. It takes the form of a finite state network. The dialogue handler controls the dialogue for speaker adaptation, handles one-word replies and decides on the system's next move, and communicates with the linguistic component to process user questions formulated as a whole sentences.

An *answer generator* produces answers and questions on the basis of information from the dialogue handler. This is done through an appropriate transformation of the structure of the input question. The answers and questions are then synthesized and output to the user.

References: [Ney and Paeseler 1988] and [Niedermair et al. 1990].

VODIS II

The VODIS project (Voice Operated Database Inquiry System) was an Alvey-sponsered 3-year collaborative venture between British Telecom, Logica and Cambridge University Engineering Department. A key goal was to investigate ways in which higher level knowledge such as syntax and dialogue context can be used to improve the performance of existing speech recognition technoligies. Two systems have been developed: VODIS I and VODIS II. VODIS II is a further development of VODIS I and draws on experiences from that system. The domain is train table enquiries and the intended use is as a telephone-based conversational question/answer system for the general public.
The system uses a connected word recognizer and is speaker dependent.
The system consists of the following components:

The *speech recognizer* is a speaker dependent, continuous speech recogniser using dynamic time warping (DTW), whole word templates, and a Viterbi-like N-best search with token passing. The token passing approach has been adapted in the SUNCAR recogniser (cf. 2.2). Dynamically exchangeable grammar constraints are applied directly to the recogniser. The constraints are characterized as "context-free" grammar rules. However, as all non-terminals dominate a definite substring of terminals (a finite state subnetwork), the constraint networks can only model regular languages. Due to the Viterbi-like search, which works upon a finite set of link-nodes, real CFGs cannot be accessed by the recogniser. The forward, time-synchronous search of the recogniser is succeeded by a backward algorithm, which traces back from the terminal state of the syntactic constraint network and initializes a chart used by the linguistic module. Vertices are processed from right to left by the backward algorithm, whereas the chart parser builds edges bottom-up using a traditional left-to-right strategy. In order to reduce the number of vertices of the initialized chart (in VODIS called a "word lattice"), the system seems to use time alignment. Hence, the N-best algorithm of VODIS must be characterized as very inexact.

The *linguistic module* mainly consists of context-free grammars stored in a rule database and a bottom-up chart parser which processes the word lattice from left to right and reconstructs the phrase structure trees of the most likely interpretations of the input speech. The phrase structures are then converted into a frame and passed on to the dialogue controller.

A frame-based *dialogue controller* has the central control of the system. It keeps track of the dialogue and uses dialogue and domain knowledge to evaluate the alternatives received from the linguistic module in order to select a suitable response. Hence the semantic interpretation is very primitive.

A *speech output subsystem* outputs the response found by the dialogue controller.

The system has been tested in two versions: a version with "strong" and a version with "weak" syntactic constraints in the recognition component. In the version with strong constraints, the grammars used for parsing seem to be identical to the grammars used for recognition (although this is not clear). In the version based on weak constraints, wildcards are placed between the syntactic subnetworks in the recogniser. A wildcard is in VODIS a simple thresholding device in parallel with word templates. Probably a similar effect can be achieved with garbage models in a HMM-based recogniser like the SUNCAR (cf. section 2.2). Performance is evaluated only in terms of slot recognition rate (as opposed to traditional speech processing parameters like word or sentence recognition rate). Performance tests show a slightly better slot recognition rate of the weak constraint approach, when the test material involves non-syntactic input. Otherwise the strong constraint approach is superior.

References: [Young et al. 1988], [Cookson 1988], and [Young et al. 1991].

SUNDIAL

The SUNDIAL (Speech UNderstanding in DIALogue) project is an Esprit project planned to run for five years from 1989 to 1993 with an effort of about 170 man-years. The goal of the project is

to build real-time computer systems capable of maintaining cooperative dialogues with users over the telephone. By *cooperative* is meant that the conversation should be as natural as possible, i.e., the mode of communication should not be strongly directed by the system.

Systems have been developed and prototypes implemented for four languages: English, French, German and Italian. The domain is, for English and French, flight reservations and enquiries and for German and Italian, train timetable enquiries.
The systems are aimed to be speaker independent, to run in real-time and to have a vocabulary of 1000-2000 words. For the moment (spring 1992) the average response time for a whole system is about 10 seconds and the lexicon contains about 300 entries for each language.
Each system consists of five components:

A *speech recognizer* receives speech signals from the telephone and produces a word lattice. Recognition is based on Hidden Markov Models of phoneme-sized speech units using both context independent and dependent phone models. Special cases such as function words and digits are modelled separately to improve performance. Also special models for coughs, ums, etc. have been created. A word pair grammar acts as a filter on the word lattices produced. Experimentation with context dependent bigrams have been carried out and they seem to yield a better performance.

The *linguistic processing* includes a grammar—for English and French a Unification Categorial Grammar (UCG), for German an Augmented Phrase Structure Grammar (APSG), and for Italian a Dependency Grammar (DG)—and a parser which extracts a plausible string from the lattice and assigns syntactic and semantic representations to it. A left to right bottom-up chart parser works with the UCG representation and an island driven parser determines starting points for parsing. In the case of UCG the semantics are compiled into the lexicon. In APSG and DG the semantics are represented as a semantic net and as case frames, respectively.

The *dialogue manager* is a central part of the SUNDIAL project. The dialogue manager has five modules: a linguistic interface module which maintains a linguistic model of system and user utterances; a dialogue module which keeps track of dialogue history, interprets user utterances and decides how the dialogue may continue; a belief module with which the dialogue module interacts to find the correct interpretation; a message planning module with which the belief module cooperates and which plans a system utterance when it is the system's turn to speak; and a task module which checks if sufficient task information has been provided by the user, and informs the dialogue module of the current state of the task. Typically, a cycle starts by the task module requesting task information from the dialogue module.

A *text generator* constructs a detailed linguistic representation including a text which is prosodically annotated. The text generator uses the same grammar as the linguistic processing module and handles ellipses and anaphora.

A *text-to-speech system* synthesizes the output from the text generator. The synthesizer for each language is based on existing systems and is either diphone or formant based.

References: [Peckham 1991], [McGlashan et al. 1992], and [Andry and Thornton 1991].

1.3 Evaluation methodologies

Research in the area of speech technology in recent years has led to a shift of focus from speech *recognition* systems to speech *understanding* systems. In parallel, evaluation strategies have changed, focusing less on word and sentence recognition rates and more on the semantic representations and the answers given by the system based on these representations.

Speech recognition systems within the US DARPA community are normally tested with the standard scoring software developed by NIST (National Institute of Standards and Technology) [cf. Pallet et al. 90]. However, in speech understanding systems based on keyword-spotting, wildcard-techniques, error recovery etc., a correct semantic representation does not presuppose a fully and correctly recognized sentence. In this case performance scores evaluated by the NIST-software tell little about the overall performance of the system.

In 1991 a new simple metric for computing overall performance of speech understanding systems was introduced within the DARPA community [Goodine et al. 91, p. 846]. This metric presupposes a test database with a number of sampled sentences and an equal number of answers from the system. The answers are classified into three groups: Correct answers, incorrect answers, and No-answers. The latter group might in a specific dialogue system be expressed as output to the user like "Please repeat!", "This was not understood, please answer the question ...!" etc. Overall performance is computed by the algorithm:

(correct answers - incorrect answers) / all answers

The algorithm penalizes for incorrect answers, which for users of spoken language systems are more confusing than No-answers. The example given below demonstrates how incorrect answers and No-answers affect the overall performance score, while the number of right answers are constant:

I		II	
Cor. answers :	10	Cor. answers :	10
Incor. answers:	10	Incor. answers:	0
No answers :	0	No answers :	10
Perf. (10-10)/20 =	0	Perf. (10-0)/20 =	0.5

Test databases for speech understanding systems normally contain both "possible" sentences (within the domain) and "impossible" sentences which even in transcribed form cannot produce any answer. Hence, there is a trade-off between optimizing for correct-answer-rate (at the expense of many impossible sentences producing incorrect answers) and optimizing rejection (at the expense of many possible sentences producing No-Answers). However, the capability of rejecting "impossible" input is essential in speech understanding systems, as the user, contrary to the designers of the system, does not know the exact domain constraints (call to mind the anecdote about the user asking the MIT Voyager system: "Where is my dog?" !).

Normally, time is no parameter in metrics for speech understanding systems due to the fact that they so far only has been developed for lab tests and simualtions. It should, however, be pointed out, that the speech understanding systems discussed in sections 1.1-1.2 are not real-time systems. Obviously, In systems where the "realistic" aspect is stressed, also time must be a very important parameter.

1.4 Conclusion

As judged from current practice, a standard architecture for spoken dialogue systems consits ts of

_a speech recognizer often based on Hidden Markov Models.

_a linguistic module containing among other things a parser which is often a chart parser.
_a dialogue manager which is the central part having the system control.

_an answer generator.

A common feature is that none of the described systems are real-time systems whereas their vocabularies are relatively large. Preferred domains are time table enquiries for flights and trains.
For the DARPA systems a common evaluation methodology has been worked out in order to make it easier to compare the performance of different systems. The systems all draw on (parts of) the same vocabulary and are all tested with sentences from the same test database. Although the specific approach to the testing of systems may have its weaknesses, as described in section 1.3, the results can be used as a basis for further investigation of or comparison of different architectures and methods. Also new projects outside the DARPA context can easily have access to and build on the results.
In Europe there has been no initiative to work out a common evaluation methodology. Moreover on-going projects are not coordinated with each other. In fact it would be desirable to have a better dissemination of information on on-going projects, experiences and results so that new projects can exploit results and experiences which have been made already. When starting a new project within the area of spoken dialogue systems it is very important to be aware of and carefully investigate what is going on in other countries in order not just to repeat what others already have done.

2. State-of-the-art speech recognition methods

This section gives an overview of the current techniques for speech recognition. It starts by introducing the two most common modelling techniques for the process of acoustic decoding, namely Hidden Markov Models (HMMs), and Dynamic Time Warping (DTW). Artificial Neural Networks, a technique which has spread rapidly in the last few years, and often used in conjunction with the two methods mentioned, is also described broadly.

As an example of a HMM based recognition system the SUNCAR recogniser, to be used in the first prototype in this project, is described in section 2.2.

2.1 Current techniques for Automatic Speech Recognition (ASR) systems

ASR systems are categorised into a number of different classes of which the most important are : Size of vocabulary, speaker dependedness, and isolated/connected/continuous speech mode. For the present project, which is concerned with speech recognition in the context of human-machine dialogue systems, only medium sized (up to 1000 words) vocabulary continuous ASR systems will be considered.

2.1.1 Basic signal processing

The speech signal is captured by microphone and digitized with sampling rates typically ranging from 8,000 Hz up to 20,000 Hz. To extract the spectral information, either a Fourier Transformation or a speech production model is applied to the signal. The speech production model aims at identifying the reflection coefficients of the human articulatory organs, and it's parameters are termed Linear Prediction Coefficients (LPC). A further transformation of the Fourier and LPC parameters into the Cepstrum domain is widely used. The feature vectors are typically calculated every 5 - 20 ms using a windowing function. To capture the time dynamic aspects of speech the incremental (delta) values are often calculated and included in the feature vector. Also features such as energy and delta energy are widely used. Altogether, the dimension of the feature vector typically ranges from 15 up to 40, with extremes of about 80 [Mariani..].

Alternative approaches such as artificial neural networks for phonetic classification and speech perception (auditory) instead of speech production models are investigated at many laboratories e.g. at STC.

2.1.2 Dynamic Time Warping (DTW)

The DTW technique is based on the principle of matching the speech signal against a set of reference templates. The templates are simply examples of each word in the vocabulary, spoken in advance by the user(s) of the system, and typically converted into Cepstrum vectors. The term "dynamic" refers to the fact that the recognition algorithm is implemented using a dynamic programming paradigm. The "time warping" is necessary because it is not possible to do a linear alignment of the reference template to spoken utterance. This is due to the fact that when a word is pronounced more rapidly or slowly, the duration of some phonemes are fairly unchanged,

where others (typically vowels) may vary considerably. Therefore, a nonlinear alignment is necessary, which allows for "stretching" and "compr
ession" of different regions of the speech signal. This is illustrated in figure 2.1.

Figure 2.1 DTW template matching.

DTW ASR systems are more flexible than HMM systems, as new words can be included without training. However, recognition results are generally poorer (because the recognition is not based on statistical models, but just reference paterns), and a critical weakness is that the DTW technique isn't well suited for speaker independent, as well as subword based recognition.

2.1.3 Hidden Markov Models

The most successful acoustic decoding technique for ASR so far has been HMMs. They differ from DTW by using stochastic models instead of reference templates. HMM is used both for isolated word recognition, as well as for recognition of connected and continuous speech. HMMs are well suited for speaker independent recognition tasks, because they are able to effectively model the pronunciation variations across different speakers. HMM is a stochastic technique based on the assumption that human speech can be modelled as a series of distinct acoustic events. A HMM therefore models speech fragments as a number of states with possible transitions between the states. See figure 2.2. The states are not directly observable (hence the word "hidden"). As can be seen from figure 2.2 it is possible in principle to remain in one state for a in principle infinite period of time. Recognition is thus the problem of identifying the model which most likely produced a given speech fragment.

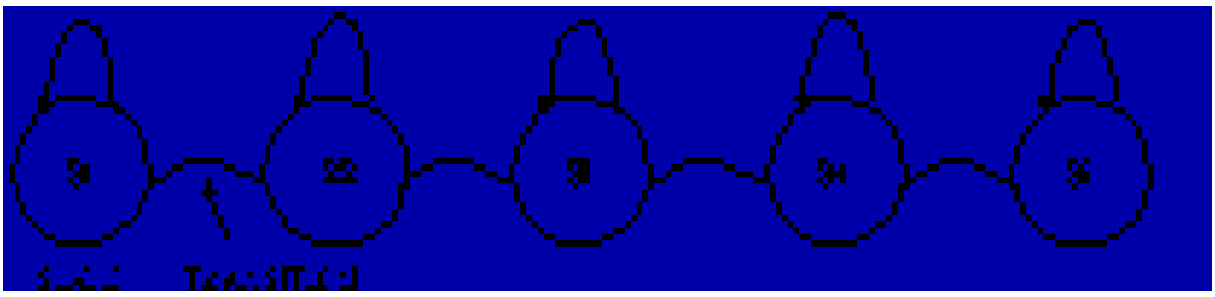


Figure 2.2. Five State Hidden Markov Model without skips. The nodes depicts the states, and the arcs the transitios between the states.

The basic recognition units are typically whole words or subword units such as context dependent phonemes (triphones) or syllables or a mixture of all three. The major drawback of the HMM technique is that it requires very large amounts of training material, and corespondingly powerful computers to do the training of the models. This problem can be overcome, though, as the training is done offline. As models are explicitly trained, it is somewhat cumbersome to add new words to existing vocabularies. That is especially true as regards speaker independent systems, as

it is necessary to use from 50 to 100 different persons to train such a system.

2.1.4 Artificial Neural Networks

(Bliver indsat mandag - lbl)

2.1.5 Recognition algorithms.

The algorithms used for continuous speech recognition are based on Bellman's principle of optimality, which can be phrased "*the optimum path from a given point must be independent of the path taken to that point"* [Husøy 1991]. This leads to the application of the DTW algorithm described above, and the corresponding Viterbi algorithm in the HMM based systems. The algorithms guarantees that the found path will be the optimal within the given syntactical constraints (and acoustic models) , given in, e.g., a finite state (FS) table.

The forward, time-synchronous search of the recogniser consists of four nested loops :

```
1. for every frame of the speech signal left to right
{
        compute feature vector

        2. for every state of the finite state table
        {
                3. for every outgoing HMM transition
                {
                        initialize start state of HMM with
                        best previous token

                        4. for every state of the HMM
                        {
                                propagate token
                        }
                }
        }
}
trace back from terminal state of the finite state
table and return best sentence candidate
```

Figure 2.3. Generic recognition algorithm shown for a HMM based system

It should be noted, that stronger network types like recursive transition networks (RTNs) or augmented transitions networks (ATNs) cannot replace the FS table in this algorithm. [Brøndsted 92]. Allmost all modern continuous speech recognition (CSR) systems are based on the algorithm shown in 2.3, though there might be differences as regards the modelled units (e.g. triphones or phonemes instead of word forms), the depth of search (*N-best* instead of *1-best*), language models (probabilistic models like *bigrams*, *trigrams* instead of the structural models mentioned above),

pruning, rejection etc.

However, in many cases it will not suffice just to identify the most likely sentence hypothesis. Further analysis in the dialogue system may lead to a rejection because of, e.g., semantic or pragmatic incompatibility. Therefore, the functionality of the recognition algorithm can be enhanced with the ability of finding the N-best hypotheses. The *A** algorithm [Rich 83] is often used for that purpose. This is very costly in terms of computations, so often approximations are made to the exact N-best search.

2.1.6 Language Models (grammars).

All continuous ASR systems include language models to constrain the acoustic search space. The language model defines the legal sequences of occurrence of the words in the vocabulary. Two classes of grammars are currently in use: structural and probabilistic.

A structural grammar is typically represented as a finite state network. This may be derived from regular grammars or stronger types like phrase structure grammars (PSG) or ATNs. A very simple word pair language model is often derived from one of the grammars mentioned, or from (sub)language corpora.

Examples of probabilistic grammars are bi- and trigrams, which are trained, and thus reflects (sub)language statistics rather than a predefined syntax constructed by the dialogue designer. The language models used within the acoustic recognition process are often very simple due to the very computation intensive search procedure. A further linguistic analysis is normally done within a natural language processing component of the dialogue system.

2.2 STC 3R/SUNCAR recogniser evaluation/description

As it has been decided that the P1 prototype will be based on the SUNCAR recogniser developed at STC, a brief account of this module follows, also as an example of a HMM based CSR system. So far only a basic version of the recogniser has been developed.

The SUNCAR is designed to be directly integrated in the SUNSTAR ICM/DDL system. This implies, that the recogniser is automatically controlled by the ICM, without concern of the dialogue designer. This setup includes automatic download of the active vocabulary and grammar, which in this way can be dynamically set throughout a dialogue session. However, the recogniser can also be integrated in other environments (e.g. DOS applications) outside the ICM/DDL system.

SUNCAR is a speaker independent recogniser based on HMMs modelling whole word forms, subword units (e.g. triphones) or a mixture of both, and a 1-best Viterbi search algorithm. A finite state table stores syntactic (in case of triphones also phonotactic) constraint rules. At the present, the table can only store simple word pair or bigram grammars, finite state grammars and context-free grammars with finite depth of recursion. The arcs of the finite state table are HMMs (in case of triphones chains of HMMs) and the states serves as link nodes in the recognition algorithm. Basically the Viterbi algorithm treats the finite state table as a Markov model as the two data

structures are largely equivalent. Therefore the computation performed over grammar nodes of the finite state table in broad terms corresponds to the computation within the HMMs associated with the arcs of the grammar network.

A provisional test with a grammar perplexity of 8,14 and a lexicon of 83 word forms (HMMs) showed a recognition rate (sentence level) of approximally 90%. The average sentence length in the test database was 2,8. It must be pointed out that the recognition rate was achieved in controlled environment as regards recording conditions, well-formedness of test sentences, no extra-vocabulary words etc. Furthermore, the tests were not truly speaker independant.

[Additonal test results using weaker grammar constraints (null-grammars and word pair grammars) will be available soon.]

SUNCAR does not recognise in real-time yet. On a system configuration with a single DSP32C-board, a near real-time version using a 1-best Viterbi search and a vocabulary of max. 100 word can be implemented. It should be pointed out that the large-voabulary speech understanding systems discussed in section 1.1 and 1.2. are *not* real-time systems and that especially the American systems seem to have unrestricted access to DSP boards.

The unit "word" (or "word form", "whole word" etc.) can in this context be defined as a speech segment that can be pronounced in isolation and modelled in a HMM. Of course, spelling forms do not always reflect these units, e.g. "femogtyve" (twentyfive) might correspond to three models <fem>, <og>, <tyve>, "til venstre" (to the left) might correspond to only one model <til_venstre> etc. Further there are instances of free variation forms like "til" - <te>,<til>, "skal" - <ska>, <skal>, or phonetic polysemi like "hvid", "vid" - <(h)vid>.

2.3 Conclusion

The current trends within speech recognition technology can be summarized in :

_Nearly all systems are based on some variation of HMMs.

_Neural networks are being investigated for classification purposes, and are used in conjunction with other modelling techniques, such as HMM.

_For CSR systems subword models are used as the basic acoustic units. Context dependent phoneme models (generalised triphones) are popular.

_There is a general agreement that large vocabulary CSR systems have to include additional knowledge sources in order to perform the recognition task. This knowledge is often represented as a language model, which is integrated closely in the recognition process.

_The language model imposes syntactical constraints upon the recogniton process. Two kinds of models are currently in use. Simple, structural models, such as finite state grammars, and simple probabilistic models, such as bi- and trigrams.

_In research laboratories speech understanding systems are currently being developed with vocabularies of 1000 - 2000 words. These systems are *not* real time systems, or in any way close to be developed into commercial products. Commercial speech recognition systems are typically isolated word recognisers. **3. Syntactic and semantic parsing in interactive language understanding systems.**

(Corrections still missing form claus/brad)

This section focusses on an important design aspect of natural language understanding systems, namely the relative roles of syntactic (structural) and semantic (meaning) parsing. First, how parsing systems can differ is discussed generally, followed by an outline how syntactic and semantic parsing are used in existing systems.

3.1 How parsing systems differ

A parser can be simply defined as a program which applies or fits a static description (typically grammatical rules) to some input (typically language). In language understanding systems, the result is an analysis or representation of the meaning of the input according to the description(s); it is another version of the input, either partially or completely translated into another form, i.e. the surface (input) form may not be recognizable from the meaning (semantic) representation. This is in contrast to purely structural (syntactic) parsing, where the output normally resembles the input which the addition of structural and lexical information.

A parsing system may be defined as a parser (or parsers, and possibly with accessory functions) with its static description(s).

The distinction between the linguistic description and the parser which applies it is one not always made in the literature, which can be a source of confusion: often the language description is the only difference between parsers that seem to *parse* quite differently, in that they produce different results. For example, the same application method can be used with either a syntactic grammar or a semantic grammar, but although the program itself is the same in each case, we would still call the former system a syntactic parser and the latter a semantic parser.

Parsing systems can thus be compared in a number of ways at a number of levels, for instance by technical details of rule application methods, or by precise comparisons of description types (e.g. dependency vs. phrase structure grammars). Here, the focus is at a higher level of generality, namely on the interrelation between syntactic and semantic descriptions in the system's attempt to construct an understanding of the input.

3.2 Parsing in language understanding

Language descriptions are either of a syntactic or a semantic nature: a simple BNF syntactic grammar exemplifies the former, (some types of) caseframes the latter. In language understanding systems, the latter, semantic descriptions are prioritized, in that all the systems must build some

kind of semantic representation, whether it be in the form of a frame, a logical language, or a representation over a semantic network.

Although syntactic and semantic descriptions are traditionally kept conceptually distinct, they are by no means mutually exclusive, and systems vary markedly in the degree to which they integrate their semantic and syntactic analyses, or whether they utilize syntactic analysis at all in the process of building a meaning representation. This leads to a natural clustering of the systems into 3 groups: non-integrated systems, which do a structural analysis as an initial step before semantic interpretation; fully integrated systems, constructing a semantic representation directly from the input elements with help from some type of structural description; and finally systems which map directly from the input to the semantic representation utilizing no structural analysis at all.

3.2.1 Non-integrated analysis

Systems which do not integrate syntactic and semantic analyses tend to implement a more thorough syntactic analysis than integrated systems. This is no doubt in part due to the fact that one motivation for implementing an integrated analysis is to reduce reliance on syntactic information, compensated by more semantic information (see discussion of integrated analyses below). In any case, given a non-integrated design, it is easy to see that the initial structural analysis must be capable of finding some constituent and dependency structure, otherwise its result can be of no assistance to further processing. Results of the structural analysis are examined by procedures for filling some semantic interpretation, this very often being a type of caseframe.

The PUNDIT system [Norton et al. 1991] exemplifies a typical modular approach to natural language understanding, and has in fact been tested with a variety of speech recognizers (see NLD91). Communication between modules is accomplished by a Dialog Manager, which passes speech recognition results on to the PUNDIT parser. Syntactic argument structure aids the subsequent filling of frames (which they call "decompositions"), which are passed on to an application module for generation of queries for the ATIS database. This modular design for production of database queries can be found elsewhere, e.g. versions of the TINA system [Seneff et al. 1991].

PUNDIT's decompositions are associated with key verbs, nouns and some adjectives from the domain; their arguments (thematic roles) are filled by rules associating them with certain semantic classes and grammatical roles. For example the regimen (i.e. object) of a "from-PP" may be used to fill the point-of-departure argument of a "flight" decomposition if the PP occurs within the phrase headed by "flight". However argument structure isn't always "transparent" for a number of reasons, three of which are mentioned: (1) the modifying constituents do not always occur immediately relative to their heads; (2) speakers use metonomy when specifying objects; and (3) the input may be parsed incorrectly, such that PPs are improperly attached.

Instead of modifying the syntactic parser to attack these problems, the developers of this system experimented with flexibility in the semantic interpretation stage. The requirement of close syntactic association to head elements was relaxed, so that a syntactic constituent which does not fill a role in the caseframe corresonding to its direct syntactic head can be "propagated down" the syntactic analysis in an attempt to match it to empty thematic roles corresponding to other parts

of the structure. Semantic constraints on potential role fillers are still in effect.

There was also a problem of synonymous input being parsed into non-equivalent decompositions. For example, "a flight from Boston" and "a flight leaving Boston" were interpreted in two separate ways. This was ameliorated with the addition of an inference component applied to the result of the semantic interpreter. Through the application of inference rules written by the developer, semantically identical elements can be transformed into a regularized format. In this example, "leaving Boston" was transformed into an argument for a "flight" decomposition, instead of instantiating its own "leave" decomposition [Norton et al. 1991, p. 126].

One problem with PUNDIT as described above is that it relies on the presence of head constituents for instantiating particular decompositions, and prepositions for marking case fillers. These shortcomings can be especially serious in a speech recognition environment, where input is often elliptical, and the input signal and recognition of it are imperfect. Prepositions are particularly difficult to recognize because they tend to be unstressed and of short duration.

The EVAR dialog system [Brietzmann and Ehrlich 1986] treats the problem of ellipsis while staying within the non-integrated framework. It is similar to PUNDIT in its overall design: syntactic analysis (valence-based, using parsing with unification) is followed by caseframe filling by specialized procedures. Partial input is matched to caseframes using a special function that attempts to fill competing frames, then evaluates them according to how well they are filled (how many obligatory slots are filled, consistency of constituents, length of time interval not covered, etc.) The filling of caseframes can be done without the head element being present, thereby tolerating its elision.

3.2.2 Integrated analysis

In the non-integrated systems above, the result of the initial structural analysis can sometimes be ambiguous: the semantic interpretation process can then select the right analysis by finding the one best satisfying semantic expectations. Other systems attempt to make this process more efficient by integrating some semantic information with the structural analysis itself, thereby eliminating useless parse results from the outset.

There is a trade-off often made by this type of approach: the efficiency sought by the close linking of structural and semantic analyses reduces general applicability of the former. In other words, such a link to the semantics of a specific domain makes the structural analysis inapplicable to other domains. In addition, without a separate general representation of syntactic information, the systems that code syntax within semantic representations (e.g. semantic grammars, or syntax within semantic caseframes) must repeat the same syntactic information in each applicable semantic representation.

Some systems have found ways of integrating syntax and semantics without loss of syntactic generality. In the system described in Hayes et al. 1986, caseframes contain information specifying which lexical elements ("heads") and syntactic roles ("syntactic cases") can fill which slots in the frame. Parsing is based on this information within the caseframes; a procedure with hard-coded syntactic knowledge seeks through the input for the heads, doing only as much syntactic analysis as necessary. This system avoided modular syntactic and semantic analyses, but

at the same time kep the syntax distinct from the semantic representations. However, it had the clear disadvantage of a relatively inflexible syntactic analysis locked into the parse procedures themselves.

Parsing in the SUNDIAL system [Baggia et al. 1991] does this one better by allowing specification of syntactic information outside both the program code and the caseframes, and then combining them by compiling the syntactic (dependency) grammar and caseframes together into what they call "Knowledge Sources (KSs)". The result is a kind of semantic grammar, with each KS owning "the syntactic and semantic competence necessary to perform a well-formed interpretation of a fragment of the input" [Giachin and Rullent 1988, p. 197].

Based on these KSs, semantic parses of input sentences are generated directly with no intervening structural analysis; rather the structure is a part of the semantic representation itself. The result, a "Deduction Instance" or DI, is then handled by actions that decide what to do next: if the DI is missing a case filler, this is either filled in by other existing DIs or broken down into subconstituents which themselves can be filled; if the DI is missing a head element, one is hypothesized, thereby handling the problem of elided head elements. Additional robustness is implemented by a kind of flag on function words (what they call "jollies"), which are treated specially under the parse process, e.g. they are not always required to be present [Giachin and Rullent 1988, p.198].

This system seems to preserve the advantages of modularity while making the analysis itself more efficient, guided by semantics. This is however at the cost of implementation complexity: a separate compiler producing the KSs must be developed. It is also unclear whether the quantity of KSs produced is inordinate, for instance if the number of KSs were the product of the number of syntactic rules and the number of caseframes.

3.2.3 Semantics without syntax

As language understanding systems typically are used interactively, they need to be both quick to react (real time) and robust in the fact of ungrammaticalities: written input to an interactive system is very often structurally ungrammatical, while spoken input is notoriously so (see case studies cited in Weischedel and Ramshaw 1987, p.157; see also Carbonell and Hayes 1983 for a discussion of error types and recovery methods). The EVAR and SUNDIAL systems above implement robustness while maintaining structural analysis. Another approach to this problem area is to implement a freer interpretative semantic analysis with no (or very little) reliance on structural information.

The straightforward case of this is what is called "word-spotting", where key words in the input are used to instantiate the semantic frames, but where minimal structural analysis is done. For instance, in entity-oriented parsing [Hayes 1984], the input is scanned for keywords that instantiate high-level entities (Hayes' version of caseframes). Once one is triggered in this way, its corresponding arguments are sought in the remainder of the input by further word-spotting. If no high-level keywords are present, there is the possibility for triggering an entity bottomup by combining those lower-level entities that are present.

This has some advantages in terms of ease of implementation and quick runtime, but has obvious

disadvantages when one considers modality, negation, passive constructions, topicalizations and other syntactic phenomena which affect the meaning of the sentence, but which cannot be identified without some kind of structural analysis. Other systems improve on the word-spotting concept by attempting to spot a larger portion of the input. This can be regarded as a type of flat sentence analysis, i.e. larger chunks of the input are identified and used to form an analysis of the sentence, but no further structuring of the analysis is present. Pattern-matching is such an analysis, wherein patterns are used to specify acceptable contexts for the keywords to be sought. The CASPAR system [Carbonell and Hayes 1983] and FlexP [Carbonell and Hayes 1987] implement such a pattern-matching activation.

Phrase-spotting, a variation of pattern-matching, also seeks specific word combinations in the input, however phrase-spotting has each chunk of text stored as meaning-bearing units. For example, "by ten o'clock" could be regarded as a meaning-bearing phrase which can fulfill a time specification (argument) in some caseframe. Thus there is a direct mapping from input to semantic representation.

The PHOENIX system [Young and Matessa 1991] parses in this fashion, evaluating all possibilities in parallel and producing a representation over a semantic network. Nodes of the network are linked to slots in caseframes, and a scoring mechanism judges which caseframe is best filled by the given input. At this point a routine puts the caseframe representation into a canonical form, which is used as a basis for generating database queries [Ward 1990].

3.3 Comparison of approaches

There are some trade-offs systems with no syntax make compared to those with syntactic analysis, in that they run the risk of missing important structural cues crucial for interpretation of input. For very restricted applications, where one can be reasonably sure of input of a limited complexity, this approach can be adequate.

In larger systems, where a greater degree of ungrammaticality can be expected, this approach is not as good. The pattern-matching and phrase-spotting types require input well-formed in other ways than structurally. For example, unless implemented flexibly, the latter types are particular about the form of the phrases and patterns, allowing little or no variation. However even if implemented flexibly (for example, by allowing 'close' matches to phrases or patterns), the system still has no syntactic information on which to base error recovery in the case of ill-formed input: in other words, all words in a pattern or phrase have the same status (except for the keyword in patterns), so that the system can have no idea whether a close match missing some key element, such as a verb, is a better or worse solution than a close match missing only a preposition or determiner.

Non-integrated systems allow error recovery of various sorts, both syntactic and semantic. They also have the advantages provided by modularity: development of various components can occur in parallel, and components from one system can more easily be applied to other domains. Efficiency is their greatest weakness: a purely structural analysis can produce several analyses for a single input sentence. An attempt at integration of syntax and semantics which preserves advantages of modularity is described, however the effectiveness of the implementation cannot be

judged based on the available literature.

3.4 Conclusion

By definition, systems implementing natural language understanding must construct meaning representations of their input. How various systems go about doing this can differ in a number of ways, some technical and some of a more general nature.

This overview focusses on the general question of the integration of semantic and syntactic knowledge in language processing. In this respect, systems can be classified into one of three groups: non-integrated structural analysis, integrated structural analysis, and systems with no structural analysis.

Some strengths and weaknesses of the three approaches are discussed, however a closer and more complete analysis of parsing in language understanding would have to cover details of the following points:

_how systems handle metonymous and synonymous input, including lexical ("plane *leaving|from* Aalborg") and structural synonymy ("plane leaving" vs. "flight that departs"; "give me" vs. "I want" vs. "could I have").

_in systems with structural analysis, how they utilize syntactic constituents (including those not forming complete sentences), and their roles and dependencies

_how robust systems are, i.e. how they maintain lexibility in both syntactic and semantic analysis

_how results of parsing can be used for discourse functions, e.g. anaphora and ellipsis resolution

_how efficiently the systems run

_how complete the parsing is, i.e. whether alternative parses can be computed

_how easy the systems' grammars are to train and modify: for example, whether the grammars can be developed independently or are hard-coded into the system (as with the CMU-system [Hayes et al. 1986])

_how powerful the grammars are, i.e. how much of a domain can they cover, both syntactically and semantically.

4. Sublanguage and Language Analysis

4.1 Introduction

With the realization that commercial product development of machine translation systems with broad lexical and grammatical coverage is at present unattainable, interest in defining and delimiting subsets of general languages has increased greatly.
Such "areas" of a general language, often labelled sublanguages, are primarily characterized by limited vocabularies and types of linguistic phenomena. It follows that it is much simpler to generate a formal linguistic description of such a language.

4.2 What is a sublanguage ?

Since the 1960s, there has been basic research to define the concept of sublanguage (SL) formally and to identify its boundaries within general languages [Sager et al. 1980]. Many different definitions have been suggested, however, most of
them agree on the following points [McNaught 1992]:

_SL is part of a natural language, although in a specific form
_SL behaves like a complete language, i.e. it is different from an artificial language
_SL is often used in special communication situations (expert-to expert)
_SL is limited to a particular subject domain

The application chosen in the current project (a ticket reservation system) and the related communication situation between the user and the system in a natural language fall squarely within this definition. Obviously the dialogue between a user and the system will be a subset of natural language although having a number of unusual characteristics. For instance, it could be expected that interrogative sentences will occur with a disproportionately high frequency in a dialogue where a costumer seeks information about domestic flights.
Moreover, as the user interface is not restricted to some kind of command language or menu-driven interaction, the sublanguage actually behaves perfectly as a natural language. The communication situation between a person seeking information on ticket reservations and the appropriate expert system surely differs from any common conversation if only as regards the restricted subject domain and the task-oriented dialogue. Finally the ticket reservation system can be characterized as being domain-specific, as the universe of ticket reservations is constituted lexically of proper names of domestic flight origins and destinations and of expressions specific to ordering flight tickets.

One of the main issues in the specialised language debate has been how to differentiate between sublanguages and general language. From a theoretical point of view, it has been claimed that the most striking feature of sublanguages is that they are closed under transformations [Harris 1968]. This notion of sublanguage is like that of *subsystem* in mathematics. For example given an algebra <A, f1,...,fn> where A is a set closed under the operations f1,...,fn, then a subset of A closed

under the same operations forms a subalgebra of <A,f1,...,fn>.
The decisive question is, however, if it is by any means possible in practice to identify the boundaries between a sublanguage and a standard language in a *concrete* text example. Research into this matter has yet to come up with an answer.
Therefore because of the fuzziness of the definition of a sublanguage, and since this type of general research into sublanguages does not fall within the goals of the present project, a more pragmatic approach has been chosen. Thus there will not be a focus on identification of different languages, general language or sublanguages, in the corpus in question: when the expected tape recorded dialogues from the real world of ticket reservations eventually become available [cf. 5.1.?], the collected corpus in itself will be regarded as constituting the sublanguage within this delimited subject domain. In other words the sublanguage will be extensionally defined without trying to differentiate between sublanguage and general language.
Even this pragmatic approach will probably not immediately define the corpus. As the collected corpus will be minor, lack of representativity may lead to an overrepresentation of random lexemes and syntactic constructions rather than *otherwise* highly frequent words and constituent order.

4.3 How to analyze the corpus

4.3.1 Syntactic analysis.

On a general level a *formal representation* of a sublanguage can be generated in 2 ways: One can start top-down, making a broad and general description of the language in question in the belief that it would be easier to constrain such a description than to enhance a narrow description based on some specific text-type; or, after having collected a corpus, one can describe a sublanguage by making a grammar and then perhaps refining and it after automatically analyzing the corpus.
The experiences from the EUROTRA machine translation project [Copeland et al. 1991] suggest that the former general approach is extremely time-consuming and costly and must be considered as being far beyond the available means of this project. Hence, an analysis from the "bottom" is likely to be preferred as a collected corpus is made available. This method also corresponds to the planned course of the project, wherein a very limited experimental system will be implemented initially to be later improved upon and augmented.
Various relevant automatic or semi-automatic methods will be used for analysis of the corpus. For instance it will be necessary to make a frequency list of all the words in the corpus; a program can then be used to identify the lemmata in the different occurrences of a word form. Moreover, in order to facilitate identification of syntactic patterns, a KWIC (keyword in context) concordance of the collected corpus will be made.

Over the past few years a great deal of research has been invested into development of different statistical and probabilistic-based programs with special attention given to processing large-scale corpora [Tomita 1991]. In this connection it would be straightforward to first apply an automatic word tagging program to systematically group the words in the corpus which statistically have the same distribution, and then, on the basis of this, make a clustering of constituents. However, since the collected corpus is expected to be of manageable size, the statistical processing approach can be regarded as overkill.
In light of this, syntactic analysis will probably have to be made by handcrafting a grammar within

the chosen grammar formalism in order to detect the inherent linguistic characteristics of the sublanguage text.
It will be necessary to begin by making a description which covers all the linguistic phenomena in the collected corpus and then, perhaps statistically, identifying the most frequently applied grammar rules in order to get an idea of the most common word orders in the domain specific dialogue. Next (as mentioned above) it may be necessary based on general linguistic knowledge to adjust the coverage of the grammar and lexicon.
In parallel with the linguistic analysis, the descriptive power of the formalism chosen (in advance) will be evaluated in order to determine whether it is adequate or not for describing the corpus.

Spoken input differs in many ways from written input [Ward 1989]. In this context it is particularly interesting to focus on ungrammatical constructions such as "wrong" constituent order and repetitions of phrases and terminals.
As an example:

Jeg vil gerne .. er der flere pladser i flyet til Aalborg?

Kan øh .. kan jeg få en billet til Billund?

These deviant constructions can be dealt with in two ways:
The "ungrammatical" constructions can be represented directly in the grammar, or a failsoft mechanism can be implemented in the syntactic parser. Although the former approach gives the user (i.e. the grammar writer) more control over how specific constructions are handled, it can lead to overcomplication and lack of perspicuity in the grammar. A reasonable compromise is to implement only the most common types of ill-formedness in the grammar and let a failsoft mechanism handle the rest.

4.3.2 Semantic analysis.

As mentioned above, there are several linguistic advantages to handling sublanguage instead of general language. Besides a reduction in quantity, semantic restrictions in special languages make it possible to describe sublanguages in a more specific and precise way. The semantic constraints, thus, make it possible to:

_reduce polysemy
_establish semantic word classes
_define patterns of word classes: frames
_determine dependencies among frames

When an lexeme is polysemous, i.e. has two semantic readings, the one not belonging to the subject matter can be excluded. For instance the reading of *plane* in a mathematical sense would be irrelevant to include in a lexicon of a ticket reservation system. Furthermore when an entry within the subject matter has both an abstract and a concrete reading, the latter will often be the only one in a sublanguage.

The three latter advantages are particularly useful in the present context, since semantical analysis

in the system will probably be based on some kind of frames or semantic grammars. A practical method of defining the frames or semantic non-terminals is to first determine the various semantic word classes of nouns, verbs, adjectives and pronouns specific to the domain.

PASSENGER = {passenger, traveller, we, I}
TICKET = {ticket, price, flight}

Classes discovered in the corpus which are deemed essential for fulfilment of known goals of the system (i.e. ticket and flight information, ordering, etc.) will require some representation as a frame or frame slot. These constitute the object frames. The ticket word class will likely be represented as an entire frame, since it is a complex set of information while passenger can be adequately represented as a single slot.
The next step would be to identify semantic templates of frames in the corpus. For instance a frame concerning ordering tickets could be as follows:

[TICKET PLANE PASSENGER TIME]

In doing this it would probably be useful to identify, synchronously, keywords in the corpus (main verbs or other operators) and the semantic word classes their arguments belong to in order to locate head concepts (actions) and attached frames revealing the dependencies among and relations between domain-specific semantic frames [Hayes 1986]. An example of an **action** in a domain of ticket reservation could be *ordering* and the lexeme filling the frame; *to order*.

4.4 Sublanguages in dialogue systems with spoken input.

In a dialogue system with spoken input, one must keep in mind that the number of words which can be handled by speech recognisers in real time performance is limited. Thus after having analysed the taped dialogues and on the basis of general linguistic knowledge extended the corpus, it will, for pragmatic reasons, be necessary to reduce the lexicon size. As an example, the current MIT Air Travel Information System, (ATIS) domain has a vocabulary of about 500 words (speaker independent) [Zue et al. 1991] which to all appearances does not lexically cover a sublanguage of air travelling.
Despite the necessity of reducing the number of words present in the sublanguage it may not necessarily restrict drastically the functionality of the system. Several investigations point to the fact [Thompson 1992] that the growth in number of different word types in sublanguages flattens as the size of the corpus exceeds 500 entries, which indicate that sublanguages compared to general language contain many more highly frequent word forms. In this context it is relevant to notice that the above mentioned investigations are based on large-scale corpora, which further emphasises the importance of making the collected corpus from the taped dialogues representative by generalising it lexically and syntactically.
The construction of the final vocabulary will be a trade-off between the following interdependent parameters:

- _recognition capacity
- _the functionality of the system
- _frequently applied sentence constructions

_highly frequent and semantically important words.

5. Domain, dialogue and discourse in spoken language systems

In their 1986 review of natural language interface systems, Perrault and Grosz claim that no natural language interface (NLI) systems exist that contain a model of the query dialogue which is sufficiently sophisticated and general to handle more than a limited range of discourse-related expressions. Most systems handle a few types of such expressions, in particular various kinds of referring expressions and elliptical expressions. This is still a fair picture of the situation: no general theory has emerged that allows practical computer implementation of a general dialogue and discourse handler (cf. also [Carberry 1990, p.242]). There are ambitious and inspiring attempts at a description of such a general model (see, e.g., [Grosz and Sidner 1986] and [Grosz et al. 1989]), but because "common sense inference" systems are still beyond the state of the art, they do not appear to be close to practical application. However, many NLI systems, including speech recognition systems, have attempted to handle aspects of discourse and dialogue.

Below is provided, first, a general description of central terms related to domain, dialogue and discourse in (spoken) natural language systems. Second, some specific methods and systems are discussed.

5.1 Some central terms

Domain, dialogue and discourse are keywords in the understanding of what is communicated between a computer and a human being. The domain determines which topics can occur in a conversation, the dialogue is the form and text of the conversation, and the discourse concerns the structure of and phenomena in the conversation. In the following sections these terms are explained in more detail along with other keywords closely connected to them. A general reference on these topics is [Smith 1991].

5.1.1 Domain

By the *domain* is here meant the application area. The computer system must contain represented knowledge or data about the area in which it is supposed to be able to conduct a conversation. A reservation and information system for domestic flight travels, for instance, must contain data on departures, arrivals, fares, etc. Such data are usually represented in a database or knowledge base. It is important for the functionality of the system that the database contains a domain description that is sufficient to the tasks it supports and that users have a clear image of its capacities and limitations. An example could be a reservation system with no knowledge of departures of flights on holidays.
The domain of a system determines the possible conversation topics of dialogue and discourse. If the user's task is t book tickets, a system with, e.g., domain knowledge on electronic circuits cannot be used. It would not be able to handle the user's goals and plans successfully.

5.1.2 Dialogue

A *dialogue* is an interactive communication between two (or more) participants. The dialogue has a certain structure to some extent determined by the mode of communication (see below). Often the whole dialogue may be seen as being composed of a number of subdialogues. A *subdialogue* concerns a certain topic, e.g., that of obtaining a customer number so that one may be allowed to make reservations.

The breakdown into subdialogues often also means a division of the vocabulary. Certain words may primarily belong to a certain subdialogue. This is important in spoken language systems where the vocabulary must be divided in a number of, e.g., 100-word lexicons. If the current subdialogue can be restricted to a certain 100-word lexicon, search time is saved.

Mode of communication

The *mode of communication* (see, e.g., [Smith et al. 1992]) determines in what order participants may speak and who decides what to speak about. The computer may have complete dialogue control (the computer is in *directive* mode), i.e., only the computer decides what to talk about next. This mode is of course comparatively easy to manage because it restricts possible user answers so that, in the extreme, only "yes" or "no" could be possible answers. The opposite of this is that the user has complete dialogue control (the computer is in *passive* mode). This mode is very difficult to handle because, among other things, predictions will be very uncertain and the user may ask questions which cannot be answered because he has not provided the relevant knowledge yet. However, if the vocabulary is small and the domain narrow it will, of course, be somewhat easier to make predictions than if many topics of conversation are possible. Finally, there is the possibility of having a system with a dialogue control somewhere in between these two opposites.

The system's form of communication may be polite, non-polite, talkative, etc., which probably influences the user's way of speaking [Zoltan-Ford 1991, Fraser and Gilbert 1991]. It is therefore very important to carefully consider the form of the phrases used by the system in addressing the user.

5.1.3 Discourse

According to [Grosz et al. 1989] *discourse* is the text or dialogue in which a sentence or utterance occurs, and which is necessary for its interpretation. It is the relation between a piece of text and its meaning. Discourse involves 1) task, plans and goals (the *discourse structure)*, and 2) anaphora, ellipsis and focus (the *discourse phenomena)*.

Tasks, plans, and goals

The *task* is a series of actions or discourse moves performed by the user in order to achieve a certain goal. A specific task may be to reserve tickets for Thursday morning 23 April on a flight

from Copenhagen to Aalborg for three persons who must attend a meeting at 9.30. A task usually includes plans and goals. The *goal* is what the user intends to achieve (which may include one or more subgoals). The goal in the example just mentioned is to obtain tickets that will get the three persons to Aalborg in due time. The user probably has one or more *plans* for reaching the goal. In the reservation example above the user's plans could be to ask for departures from Copenhagen and arrivals in Aalborg on Thursday mornings and to ask how much time it will take to get from the airport in Aalborg to the place of the meeting. Goals and plans strongly influence discourse. For goals which occur often it may be possible to construct in advance a number of plans which may be used for predicting user answers and for identifying the topic of current discourse.

Anaphora, ellipsis, and focus

Anaphora can be defined, in a very broad sense (including ellipsis), as the use of expressions whose meaning (sense and/or reference) cannot be fully determined in isolation, but only by taking into account other items with which it has a cohesive relationship, the *antecedents.* If the dialogue is not to be strictly directed, resolution of some types of anaphora and ellipses (primarily at the level of nominals) is probably necessary, since they can be expected to occur frequently. The resolution of anaphora and ellipses will impose demands not only on the speech recognition module, but also on computation and representation in other parts of the system. Few spoken dialogue systems attempt to deal with anaphora in a comprehensive and principled manner.

An anaphor is resolved by matching it against the candidate antecedents represented in the discourse history. Matching is done by grammatical features (person, gender, number), and by syntactic and semantic constraints. The discourse history is maintained by a focusing system (see below). In the case of referential anaphora several different relationships between the anaphor and its antecedent can be found. With pronouns identity of reference is the norm; with lexical items many other inferential relationships are possible as well. The nature of the relationships is crucial to the matching of anaphora with antecedents, i.e., matching is very much dependent not only on the semantic representation but also on the domain model, since it is accomplished by inference with respect to the "real world".

Resolution of anaphoric pronouns will probably turn out to be crucial in dialogues that are not strictly directed since anaphora are very frequent. The number of pronouns that the speech recognition module must handle appears to be at least 35 or 40. Also, pronouns are short words, very much reduced phonologically, and most often unstressed which may complicate their recognition. Full-NP anaphora with identical reference need resolution to get proper responses from the database. For full-NP anaphora (and especially those that specify an element or subset of an earlier set), resolution may not always be necessary in a database application: if the extra information they contain is more restrictive than the information that established their antecedents (which is often the case), their reference can be found by searching the entire database, rather than a previously referenced subset of it. In a small database search may be faster than knowledge based resolution of anaphora, but for large applications brains may beat muscle. Other types of full-NP anaphora, however, do seem to need resolution if proper references are to be found. Possibly the relatively narrow domain may reduce the problem of dealing with full-NP anaphora.

Ellipsis is a much more diverse phenomenon than anaphora. English and Danish appear to be

very different at all three levels at which ellipsis occurs, at least in the expressions applied. Generally, nominal ellipsis is less distinct from pronominal anaphora in Danish than in English, probably due to the fact that the Danish determiners carry information about gender and number. Nominal ellipsis is used so widely in Danish that it is probably indispensable. As far as practical implementation is concerned, it is suggested that nominal ellipsis might be considered a kind of pronominal reference with added or changed modifiers. Other forms of ellipsis may turn out to be necessary as well.

In Danish, verbs can be substituted by "gøre" and by the modal and auxiliary verbs (that can also substitute for each other), whereas the arguments and modifiers of the elliptical VP can be substituted by "det" or "dette". Clausal substitution is most often accomplished by "det" or "dette" and is also in other ways like verbal substitution. This very non-specific use of "det/dette" may cause problems in restricted implementations if it is confused with reference to objects.

Focusing is the process that keeps track of what the discourse is about by maintaining a *focus state* which is searched for possible antecedents of anaphora. Sidner [1979] and Carter [1987] represent the current focus state by six different focus registers. The expected focus may be syntactically marked or may be found by ordering the items mentioned in the current sentence by semantic roles and by order of occurrence. Items that are referenced anaphorically are preferred. The registers are changed according to the development of the discourse. Dahl [1986] uses a similar, but simpler algorithm and discourse representation: syntactic rather than semantic roles are used as criteria for focusing and the representation is reduced. Others have applied even simpler schemes, sometimes based on recency alone. Grosz and Sidner [1986] propose a mechanism like Sidner and Carter's for local focusing (or centering) embedded in a more complex structure for global focusing. Apparently it has not been implemented.

The choice of focusing algorithm and discourse representation is of course closely tied up with the choice of grammar formalism and parsing algorithm, as well as with considerations of system and domain restrictions.

5.2 Specific systems and methods for processing discourse

As remarked above, there is a gap between the complicated discourse of human-human conversations and what is possible in computer systems. However, some attempts at mechanizing discourse already exist. Here, we shall briefly review a few of them, in particular the mixed initiative dialogue handling in the spoken language system of [Smith et al. 1992].

5.2.1 Scripts

Script-theory [Schank and Abelson 1977] provided an early and very influential model for constructive interpretation of narrative discourse [Smith 1991, p. 384]. *Scripts* represent stereotypical sequences of the smaller events that compose routine activities in a particular context. As a formalism, they are a variation of frames directed specifically towards natural language processing, and they include the following groups of slots:

1. Roles and properties (of persons and items).

2. Preconditions (for invoking the script).

3. Effects (of going through the script).

4. Actions (that might plausibly be expressed in a single sentence in a narrative).

There is empirical support for the idea of scripts: when presented with a narrative that might mentally be represented as a script, subjects reproduce the narrative in an order and with new elements that fit with a plausible script rather than the narrative itself [Smith 1991]. But scripts lack at least three desirable properties (cf. [Sowa 1984]):

1. Shared substructures. Scripts are indivisible units, and parts of one script cannot be transferred to another.

2. Generality. Relationships in a script cannot be abstracted to a more general rule that could be applied to other situations.

3. Intentionality. Scripts specify what happens, but do not state the intent or purpose.

Scripts have among other things been used in the story-understanding system SAM [Culingford 1981]. SAM builds a representation of the entire text by using the relationships indicated by the relevant scripts.

5.2.2 Scenes

In order to overcome the deficiencies of scripts, *scenes* have been introduced as building blocks of episodic structures corresponding to scripts. A scene contains the same slot groups as scripts, but are characterized by [Smith 1991]:

1. The actions are collocational and frequently co-occur in different contexts.

2. The effect is an instrumental goal that may help the higher order goals of episodes.

3. The scenes are context free and may be reused as needed.

Successors to script theory, notably Wilensky's theory of plans [Wilensky 1983] and Schank's theory of memory organization packets (*MOPs*) [Schank 1981], use scenes. Plans and MOPs correspond to episodes, and both theories have counterparts of scenes and actions. Somewhat depending on the point of view, scenes seem to correspond to the subdialogues of a spoken language dialogue system.

5.2.3 DDL

The dialogue definition language (*DDL*) [Bækgaard 1991] together with the DDL tool [Giannantoni et al. 1991] is aimed at the description, development and maintenance of the

various interactions between a user and an application. DDL is a compound language encompassing three levels:

_*The graphical level:* this is the most abstract level, describing the overall control structure of the dialogue via the use of recursive flowcharts.

_*The frame level:* this is the medium abstract level, declaring the details of the dialogue by filling slots in frame structures.

_*The textual level:* this is the most concrete level, implementing the computational parts of the dialogue.

The DDL tool provides a graphical, direct manipulation of the dialogue specifications. Until now, DDL has been applied to a number of limited applications, e.g, public telephone services (database access, voice dialling, PABC control) and voice controlled CAD systems, none of which make actual use of natural language.

5.2.4 SPAR

Carter [1987] describes a "Shallow Processing Anaphor Resolution" system (SPAR) which is apparently the most comprehensive implementation to date. It is based mainly on the work of [Grosz 1978], [Sidner 1983] and [Webber 1980, 1983]. SPAR sticks to linguistic knowledge as long as possible in dealing with discourse problems and resorts to world knowledge only when absolutely necessary. SPAR, however, is a story understanding system, i.e., it has to build its model of the world from very general background assumptions alone. NLI systems for database applications can relatively safely assume a more specific and closed world and may therefore exploit world knowledge to a greater degree. On the other hand, SPAR does not have to deal with the multitude of dialogue problems in database querying.

5.2.5 The Circuit Fix-it Shop

The Circuit Fix-it Shop is a 125 words, connected word spoken dialogue system designed to aid a user in the repair of electronic circuits [Smith et al. 1992]. The system has a mixed-initiative, real-time, voice-interactive dialogue which makes use of a user model. The dialogue is goal-driven and divided into subdialogues each covered by a set of rules. The (sub-)goal is resolved by a theorem prover with one of the results:

a general rule; then process the antecedents of the rule.
trivially satisfied; then return success.
'vocalize(X)'; then output X verbally (*mode),*
record expected responses, and
receive a response *(mode)*:
The response may be:
_ successful; then return success.

_ negative; then no action.
_ confused; then clarify.
_ interruptive; then match and jump to another subdialogue *(mode)*.

Some notes on this central algorithm are:

_The initiative is controlled at the points marked *(mode)*. Thus, in directive mode verbal output is positively stated, responses are expected to be obedient, and interrupts are restricted. In passive mode only answers to questions are output, and interrupts to any subdialogues are acceptable.

_Interrupted subdialogues may be resumed later.

_Expected responses are compiled from the domain and dialogue controller knowledge bases and thesesupport recognition and input interpretation.

The limited vocabulary and small domain clearly facilitates the task of the system. However, the system combines mixed initiative dialogues, choice between modes (though fixed during each session), (primitive) plan recognition and focusing, resolution of anaphora, predictions utilized for recognition and input interpretation (which reduces the overall error rate), and a user model.

5.3 Conclusion

The use of domain, dialogue, and discourse in a spoken language system is more than necessary for the proper processing of linguistic data: it may also improve processing efficiency and system habitability. Some important parameters to discuss when implementing domain, dialogue and discourse are:

_The modularity of the dialogue: can it be broken down into small, manageable subdialogues?

_The mode of communication: can the system afford to be passive in the dialogue and thus risk very free and abrupt changes of focus from the user, or does the system have to exert more or less tight control of the dialogue?

_How might constraints imposed by the discourse be utilized in the speech recogniser? And in the parsing?

_The discourse phenomena: which of the anaphora and ellipses must be handled, and which can be avoided? And does the focus yield useful constraints?

_The goals: how are the user's goals detected?

_The plans: which plans can the system suppose on behalf of the user when trying to achieve a specific goal?

_The domain knowledge: should it be restricted to a database that is queried, or might it

play a more active role because it is strongly related to semantics?

6. Knowledge acquisition

Building a computer system roughly involves three phases: knowledge acquisition, knowledge representation and realization, cf. figure 6.1. In this chapter the focus is on knowledge acquisition, whereas other parts of the report concentrate on representation and realization.

Figure 6.1. The construction of a computer system. Phases, processes and intermediate results.

Knowledge acquisition is an iterative process of achieving data from different sources (e.g., books and interviews) and producing data on different storage media (e.g., paper and tape). In Section 6.1 a survey of the data sources and corresponding acquisition methods is given. The data have to be systematised (i.e., analysed and structured) on a continuing basis in order to support decisions on what to focus on in the next iteration of acquisition. Through a number of iterations, the systematisation results in a model of the system to be built and represented in, e.g., natural language. Section 6.2 briefly presents some issues of systematisation.

The data material acquired will influence different parts of the system. The parts of a spoken language dialogue system most sensitive to the data material are those directly affecting the user interface, i.e., the represented natural language (vocabulary, grammar, semantics), the dialogue (in particular the mode of communication), the discourse, and to some extent the domain. Simulation experiments (especially the Wizard of Oz method) dominate the literature on acquisition for natural language dialogue systems and are described in Section 6.3 relatively to the current project. Finally, in Section 6.4 a short conclusion is presented.

6.1 Data sources and acquisition methods

There are several sources and methods for the acquisition of the knowledge needed. Examples are the research literature, material from a company (e.g., a travel agency), interviews with and studies of experts in the domain (e.g., travel agents), and experiments. General introductions to knowledge acquisition for the construction of computer systems are [Hart 1986] and [McGraw and Harbison-Briggs 1989]. For spoken dialogue systems [Fraser and Gilbert 1991] is a useful survey. It should be noted that sources and methods are closely related. An example of a source is a book and the method is reading the book. The following is a list of sources and methods:

Research literature concerning systems with (spoken) natural language interfaces (e.g., [Darpa 1989, Darpa 1990, Darpa 1991]). Through reading of the relevant literature general ideas and specific hints of the problems that might occur can be obtained. A state-of-the-art survey like the current report is also obtainable.

Printed material concerning the application domain is obtainable from various sources. Such

material gives more precise information for developing the domain model. Concrete examples are fare and time catalogues from airline companies [Danair 1991] and [Danair 1992].

Interviews with experts provide data for the domain and task models, and to some extent the dialogue structure. Such an investigation is also part of the system analysis and may give an insight into the work processes, etc. of which the system is meant to form a part. There are many varieties of interviews, depending on factors such as the number of interviewers and the techniques used in conducting the interview itself. The data may be collected on tape or as paper-and-pencil notes. Typical, slightly structured interview notes are [Dybkjær and Povlsen 1991] and [Dybkjær and Povlsen 1992a], from interviews made with travel agents.

Field study recordings of similar manual tasks (human-human phone calls) increase one's knowledge of the task model, the dialogue structure, and the spoken language vocabulary and structure. A problem is that it may be delicate to obtain permission to record customer calls since these may involve confidential material.

Field experiment recordings of human-human phone calls also provide information on task model, dialogue structure and spoken language vocabulary and structure. In this case, and in contrast to the field study, some parameters are simulated, e.g., the customers of a travel agency may be subjects acting as customers. A field experiment may be used as a substitute if a field study is impossible.

Experiments with *simulations* of the expected end-system provide data on among other things the dialogue structure and the spoken language vocabulary and structure. In connection with spoken dialogue systems, simulation experiments are widely used and constitute an important methodology for systems development [Fraser and Gilbert 1991]. The principles of experimentation used in the current project are described in more detail in Section 6.3.

Prototyping makes possible experiments with running prototypes that are increasingly close to the final system. This provides data on the dialogue structure and the spoken language vocabulary and structure, but at a more fine-grained level than the simulation experiments. The simulation experiments may be viewed as a sort of early prototyping. How much prototyping will be necessary during the implementation of a system partly depends on how useful the results from the simulation experiments turn out to be.

The first three sources mainly provide background for the domain and task models. Therefore, the early experiments (set up on the basis of these models and the tape recordings) should focus on language and discourse.

The number of tools for knowledge acquisition is small. Paper and pencil, a tape recorder and our senses are the primary tools. There also exist programs for knowledge acquisition (cf. [Gruber 1991]), but they are still at the experimental level and seem to be out of question for the purpose at hand. However, in simulation experiments and prototyping some or all of the tasks may be effectuated by the computer.

6.2 Systematisation

Concurrently with the knowledge acquisition process the collected data must be systematised to serve as approximations to the required model.

It is important to emphasise that the construction of a spoken dialogue system just like many other system development tasks is an ill-defined design problem, i.e., there are no ready-made procedures for reaching a correct solution, and if a design has been proposed that possibly is a solution there is no method to decide if the solution is optimal or even if it is a solution; only time may show [Dybkjær 1992]. This has a number of consequences for the nature of a "rational" design process:

> _There is no single, detailed "design methodology"; each designer or designer group must develop important aspects of the design method concurrently with the design.
>
> _Potential "solutions" *(presolutions)* should be generated; each presolution is based on the designer's previous experience in the field and on evaluations of former presolutions.
>
> _Each presolution should be evaluated for the purpose of recycling.

However, even if we cannot definitely decide on the quality of some specific design, there are different ways to acquire and structure the knowledge needed and some of these may be better than others, given a specific design problem. Norman Fraser [1992] distinguishes between *design by intuition* (based on one's own knowledge from experience, literature, printed material and interviews), *design by observation* (field studies and field experiments), and *design by simulation* (based on simulation experiments and prototyping). These are all necessary to the iterative design process.

Each of the sources and methods for knowledge acquisition is coupled with "natural" ways of data registration. E.g., the simulation experiments for spoken dialogue systems are naturally recorded on tape and then transcribed into text (note the dependence on the application). However, the actual data analysis and structuring must be made with respect to the solutions which are underspecified at acquisition time.

The models implied by the presolutions should be described in flexible, changeable and adequate notations such as natural language, drawings and conventions and symbols invented for the problem at hand. Only later on more formal, fixed notations (e.g., first order predicate logic, see Chapter 7) should be applied.

6.3 Simulation experiments

As mentioned above simulation experiments serve to advance the design process beyond the observation-based stage and prior to the introduction of early prototyping. In some cases, simulation experiments might help avoiding early prototyping altogether. A simulation experiment requires an outline of the system being designed as input and yields as output data which can be

used (1) in evaluating the system design used in the experiments and (2) as input to a more detailed design of the system. The new design may then be tested in further experiments, and so on.

A series of experiments thus yields a series of increasingly refined design specifications D1, D2, D3, etc. D1 itself, however, being the first one in the series, cannot be based on simulation experiments but has to be developed on the basis of intuition and observations of human-human dialogues in the domain. Since there are many important differences between human-human dialogue and human-system dialogue, the gap between results from human-human dialogue observations and D1 have to be filled by intuition. On the other hand, once the series D1, D2, etc. is developing, its analysis often leads to specific questions concerning the properties of human-human dialogue which can be illuminated by analysing results from observation or by iterating observational studies. In this section experiments are described related to the system currently under development.

A person (the experimenter)—usually a project participant—plays the role of the spoken dialogue system, and others act as users (subjects). An important assumption is that the subjects act during the simulation in the same way as they would have done if they were real users interacting with a real computer system. Therefore a number of measures are often taken to trick subjects into believing that they really interact with a computer system as there seems to be evidence that people do not speak in the same way to a computer as they do to a human being (cf. [Amalberti 1992], [Zoltan-Ford 1991], and [Fraser and Gilbert 1991]). An example of the experimental setup is shown in figure 6.2.

*Figure 6.2. Components of the experiment. The subject S and the experimenter **E**communicate via the telephone; they cannot see each other, and **S** may or may not believe that **E** is a computer system.*

The subject **S** and the experimenter **E** communicate via telephone. They cannot see each other, and perhaps **S** believes that **E** is a computer or perhaps **S** believes or knows that **E** is a person. At the very least, **S** is required to imagine that **E** is a computer. It is possible to promote the illusion by using voice-distorting hardware (e.g., a vocoder) and by having a well-trained experimenter speaking with a constant speed in a uniform voice. **E** is supposed to have practised the relevant information and a set of standard phrases to act as much like a computer as possible. Moreover, **E** will not necessarily understand everything that **S** says but will declare his lack of understanding in cases where, e.g., **S** during the dialogue moves outside the domain of application. Physically the telephones may be arbitrarily placed. The tape recorder is coupled to **E**'s telephone because he is the constant element placed at his desk whereas the subjects may be placed anywhere.

Scenarios

Each subject is given one or more *scenarios* to perform. A scenario is a short text describing the situation **S** has to imagine him/herself to be in. On this background it should be clear to **S** what to ask for but not *how* to do it. The purpose of the experiment precisely is to record how **S** actually solves the task. An example of a scenario concerning reservation of flight tickets and a dialogue between **S** and **E** is given below.

> *_Scenario:* You live in Aalborg and you are to attend a meeting in Copenhagen at 10 a.m. on Friday, April 4.
>
> *_Dialogue:* The phrases used by the experimenter **E** are taken from [Dybkjær and Dybkjær 1992].

E: Hello, this is DanLuft's travel service for flights between Copenhagen and Aalborg. I can inform and reserve. Please state day and direction.

S: When are there flights from Aalborg arriving in Copenhagen on Friday morning April 4?

E: Friday morning, April 4, flights from Aalborg arrive in Copenhagen at 8 a.m. and 9.30 a.m.

S: Thanks, goodbye.

E: Goodbye.

Usually each subject is given more than one scenario. A fairly large amount of recordings are necessary to ensure a certain reliability of the material, in particular if statistical techniques are to be used. It may be considered to have the subjects repeat the scenarios after some time and compare the results. Similarly, scenario rehearsal may be useful before making the recordings. It may be argued that the subject, having practised the scenario at least once before, would act realistically because real life-users tend to be thoroughly acquainted with their own situation and goals. In an artificial situation it may take some time and experience to reach approximately the same state.

It may also be considered to use a number of scenarios the contents of which are more or less the same but where, e.g., the information is not given in the same order, or some extra information has been added. An example is given below:

> *_1st formulation:* You are a secretary at the University of Copenhagen. A member of staff has to participate in a meeting in Aalborg from Monday, November 25 at 9 a.m., to Tuesday, November 26 at 4 p.m.
>
> *_2nd formulation:* A researcher at the University of Copenhagen must attend a meeting in Copenhagen at 8 p.m. on November 26. He also has a meeting in Aalborg starting at 9 a.m. on November 25, and ending the next day at 4 p.m. You are the secretary who must reserve the tickets.

The purpose is to see how this influences the user's choice of words and the dialogue structure. According to [Fraser 1992], however, such rephrasings seem to have little influence.

Care must be taken that subjects are faced with scenarios relevant to their own backgrounds, and that the scenarios are representative of the intended use of the system. It may be difficult for someone to engage in acting as a person who really wants to reserve a ticket or ask for certain information. But is should be possible to avoid putting subjects in completely alien roles.

Phrases

Standard phrases are meant to ensure a certain degree of uniformity in the experimenter's language (its structure, the words used, its degree of politeness, etc.) and thus to increase the level of illusion (machines are expected to be rather one-tracked) and reduce the number of parameters of the experiment. For the latter reason the experimenter should only improvise in case of unpredicted situations arising *within* the domain. If the subjects go outside the domain the experimenter should ask the subject to call a human travel agent.

The phrases used by the system are important to customers' perception and use of the system. There is evidence that users model the vocabulary and phrase structure of a program's output [Zoltan-Ford 1991]. Moreover, the phrases yield a bottom-up approximation to dialogue structure and functionality, and to the mode of communication. The phrases have no over-all structure or order. However, if the phrases are to be used meaningfully in a conversation they cannot be put together in an arbetrary order, functionality not covered by he phrases cannot be handled, and, e.g., politeness is determined by their wordings.

The phrases should be adjusted according to changing functionality and dialogue structure, and according to the reactions of users (their understanding of and answers to the phrases). The phrases should be kept in a friendly language and be specific in the questions they pose (cf. [Fraser and Gilbert 1991]). The phrases should on the one hand be long and explicitly repeat information in order to obtain brief answers from the users, on the other hand be short in order to make the system tolerable to skilled users. This is worth some consideration: perhaps a "user adaptive" dialogue can be constructed? Another possibility is to omit parts of the phrases that are determinable from the context. Care should be taken, though, that the user understands and remembers the information given by the system and what information it asks for.

Note that even though more words may be used in the phrases produced by the system than are recognisable by the system, the phrases should guide the customers to use words within the system vocabulary.

Of the topics treated in a spoken dialogue system concerning, e.g., flight reservations and information at least one poses a problem. Consider the conversation:

_*Customer:* Can you send the tickets to me?

_*System:* Yes, your address, please?

_*Customer:* It is ...

Here the recogniser has no chance of recognising the words in the address. The system will have to resort to other means in such cases (tedious spelling, restriction of reservation to those having a customer number, reference to a human operator, not to speak of entirely omitting this function in the system) .

Expected results of the simulations
A series of iterated simulation experiments and refined system specifications should produce the following results:

_dialogue structure which allows the user and the system to interactively solve their tasks within the application domain in a way which is optimally habitable given the constraints on system architecture which do not allow, e.g., unlimited vocabulary and completely free discourse.

_A number of basic characteristics of the sublanguage needed for the application, such as vocabulary size, complexity of grammar, and number and complexity of the discourse phenomena which the system has to be able to handle.

The results of a specific experimental cycle interacts with the system specification effort in such a way that, e.g., if too large a vocabulary has been used by subjects during the experiments then giving the system increased control over the dialogue might be a means of reducing the vocabulary size; or if some specific kind of discourse phenomenon seems to be unavoidable in user-system dialogue, then some way of making the system able to interpret this phenomenon has to be identified

6.4 Conclusion

The knowledge acquisition process should be iterative and take place concurrently with systematisation (data analysis and structuring) and system, domain and sublanguage modelling. The simulation experiments are particularly important to the development of spoken language dialogue systems because they offer an experimental methodology enabling quite detailed system specification. In a spoken dialogue system, important topics for data acquisition are:

_Vocabulary size and composition.

_Complexity of grammar.

_Domain (times, fares, etc.).

_Dialogue structure, in particular realisable and habitable modes of commun-ication.

_Discourse phenomena and structure.

7. Formalisms for knowledge representation and their realisation

During and after the knowledge and language acquisition the data acquired are structured, and gradually a model (of whatever the data concern) is constructed. The model may be described using natural language or some other informal notation, or a more formal language (a formalism) may be used. This chapter will concentrate on formalisms chosen on the basis of the fact that our models are to be implemented in a computer system. The focus is on the (static) representation rather than on the (dynamic) construction of models which are discussed in the other chapters.

Also, the focus will be on formalisms that are used especially for representing the high-level knowledge of language and discourse.

The description is divided into groups according to the prime features addressed by the formalisms. The groups treated here are:

Group	Subgroup
Logic formalisms	Classical logic Non-classical logic
Associative formalisms	Semantic nets Conceptual graphs
Structured object formalisms	Frames Scripts MOPs
Production formalisms	Production systems ATNs Shape grammars
Uncertainty formalisms	Hidden Markov models Fuzzy logic
Analogical formalisms	Diagrams Dual graphs
Programming formalisms	Programming languages

Some (groups of) formalisms not included are (notations for) speech acts, higher order non-linear differential equations, and legal language—neither of which points towards implementational issues; or floating point numbers and binary trees—which concern low level implementational issues. Please also note that often notation will be ad hoc, developed along with the

module/theory construction.

In the following sections an introduction to each group of formalisms is given. For each group a definition is given, and important subgroups are briefly mentioned and a representative is chosen for a short discussion. The most salient features
of the group are described and advantages and drawbacks discussed. Also typical areas of application are mentioned. As literature on knowledge representation formalisms [Frost 1986] can be recommended. Finally the realization of formalisms is discussed.

Logic formalisms

A logic formalism contains (cf. [Frost 1986]):

1. A well-defined language for representing knowledge.

2. A well-defined model theory (or semantics) for the meaning of a statement expressed in the language (an interpretation).

3. A proof theory which is concerned with the syntactic manipulation and derivation of statements from other statements.

Thus logic captures the essence of what is necessary for the representation of knowledge, and most other formalisms can be seen as variations of logic, addressing notational (language and model theory) and computational (proof theory) efficiency.

First order logic (FOL) is a basic example of logic and is an extension of propositional logic. FOL allows the use of predicate symbols, function symbols, variables, constants, and the logical connectives _ _ _ _ _ plus the quantifiers _ and _. E.g., the following formula is expressed in FOL:

(*) _ x: *smoker*(x) _ *empty*(x)

The model theory of FOL assigns meanings to the
symbols, e.g., taking a universe U of flights, people, seats, and tickets one might assign

smoker(x)	_	x is either a smoker or allowed for smokers.
empty(x)_		x is an empty seat (on some flight).
A _ B	_	A and B are both true.
_ x: P(x)_		There is an x in U such that the predicate P(x) is true.

Then, for example, (*) will be interpreted as “there is an empty seat for a smoker” (which incidentally is false on domestic flights).

The proof theory of FOL includes rules like modus ponens which, e.g., enables the inference from *empty*(x) and (*empty*(x) _ *bookingpossible*(x)) to *bookingpossible*(x).

Other logics are often based on FOL because it has some nice and attractive properties:

1. It is *sound*: A formal deduction system A is sound if all formulas which can be derived from a set of formulas S using A, are also logical consequences of S, i.e., are satisfied by all truth assignments which also satisfy S.

2. It is *complete*: If for some formal deduction system A every logical consequence of any set of formulas S can be derived from S using A, then A is complete.

The proof theories in second order logic and higher order logic are not complete. However, FOL also has its limitations:

1. It cannot handle default knowledge since FOL is a monotonic logic where new axioms must be consistent with old ones.

2. It is not possible to refer to statements or predicate names in other statements.

3. In the real world not everything is either true or false.

4. Incomplete, uncertain, imprecise, vague, and/or inconsistent knowledge cannot be handled.

Because of the limitations in FOL a number of other logics have been invented, e.g., temporal and tense logics, many-valued logic, higher order logic, intensional logic, modal logic, and non-monotonic logic. They lack some of the nice properties of FOL but gain in expressiveness.
Logics have been developed and used within almost any area, and so also within natural language processing. Two examples are Horn clause logic and Montague logic. Horn clause logic has a proof theory with nice computational properties. Work on natural language processing by, e.g., Colmerauer in the early 'seventies led to the programming language Prolog which is based directly on Horn clauses [Kowalski 1979]. Montague logic is the result of Montague's attempt to describe meanings—in particular quantification—of ordinary English [Dowty et al. 1981].

Associative formalisms

An associative formalism is defined by:

1. A net representing intentional semantics, i.e., a directed graph in which the vertices represent items (e.g., facts, events, classes, predicates, relations) and the edges represent binary relations between the items.

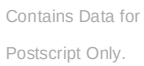
2. A set of interpreting processes operating on the net.

3. Parsimony, i.e., the same piece of knowledge should only be represented once.

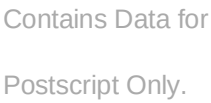
Thus, realization is an important aim of associative formalisms. In particular, the nets are intentional and embody no general principles concerning the relation between the concepts and the real world objects (extensional relations).

A number of associative formalisms exist: semantic nets, conceptual graphs, conceptual dependency graphs, partioned nets, and structure inheritance nets. The most dominant of these is semantic nets which often are used synonymously with associative formalisms.

Semantic nets have been developed within natural language processing and have—when compared to associative formalisms in general—an extra syntax analyzer (parser) that builds the net from natural language (cf. [Ringland and Duce 1988, Brachman and Levesque 1985]). Edges represent binary relations, e.g.:



N-ary relations as “The travel agent books a hotel” may be represented in the following way:



A semantic net is a flat structure. A node cannot be opened like a frame (see section 3) and has no contents. It just represents an item. There is no natural hierarchy. This causes problems if one wishes to represent " and $. Some of the problems may be solved by introducing partitions which are a kind of scopes.

It is a drawback that semantic nets are not well-defined but may be interpreted just as one likes. Moreover, semantic nets quickly grow confused with relations in all directions. An advantage is that information on a given item is usually concentrated in one place and can be found by investigating the relations of the node representing the given item.

Apart from semantic nets one of the most well worked-out associative formalisms is conceptual graphs. In [Sowa 1976] they are intended as a means of creating a database interface where the user should be able to communicate with the database using natural language. According to [Sowa 1984] conceptual graphs emphasize semantics and are very often used for representing natural language.

Associative formalisms may seem intuitively appealing. The representations reflect (conceptual) relations very directly, and certain types of inference seems to become nearly trivial. To establish a precise interpretation is, however, no trivial task, see, e.g., [Woods 1975]. Among other things default knowledge is difficult to handle, and the distinction between types and tokens may cause problems.

Associative formalisms—especially semantic nets—have been used in various areas, e.g., natural language processing and computer vision.

Structured object formalisms

The common characteristics of the formalisms in this group are:

1. Represented knowledge is organized in frames (large structures) with slots (attributes defining the frame) and fillers (the actual values).

2. A structure is treated as a single object rather than as a number of individual parts.

Typical structure formalisms are case frames [Fillmore 1968], frames [Minsky 1975], scripts [Schank 1977], and MOPs [Schank 1981]. They are inspired by psychological evidences and try to explain or simulate these. However, though only scripts and MOPs were immediately directed towards realization, they have all become mechanized in various disguises.

The most important subgroup is frames. A frame consists of a frame-name and some slots and fillers. Not only single properties but also procedures and other frames may be attached to slots. If no value is filled into a slot a default value may be assumed. A simple example is:

Name of frame: driver	
Slot:	Filler:
Age	condition: x: 18 _ x _ 70
Mood	
Number of legs	default: 2
Qualification	generic: has driver's licence

A frame system is a network of nodes (frames) and relations organized in a hierarchy. A frame at some given level in the hierarchy may inherit properties from higher-level frames since higher-level frames are more general. Frames may be related in different ways, e.g., as kind of, part of, siblings, disjoint, or similar.

Frames are intuitively appealing and easy to understand. They are fairly expressive, e.g., the representation of default knowledge causes no problems here.

There are many good ideas in the structured object formalisms such as general patterns and special instances, inheritance, and shared substructures. But there are problems in the realization of the ideas, perhaps because of the absence of a well-defined model theory on which to base the representations, e.g., Minsky was unclear in his writings on the subject.

Structured object formalisms have been used within a number of different areas. Especially frames have several areas of application. E.g., they have been used within computer vision, natural language processing, and space planning. Case frames, scripts, and MOPs, on the other hand, were developed for use in natural language processing and their usage seems to be restricted to this area. Scripts and MOPs are elaborated in Chapter 5 on discourse.

Production formalisms

A production formalism is characterized by:

1. A database with specific information on the given problem.

2. A rule base containing rules representing general knowledge of the problem domain. Every rule consists of a condition part C and an action part A:

   if (C satisfied) then (do A)

3. An interpreter (inference engine) that chooses and applies rules.

Production formalisms are all aimed at realization. Archetypical production formalisms are production systems (e.g., EMYCIN and HEARSAY-III [Waterman 1986]) and Prolog [Kowalski 1979].

ATNs (augmented transition networks) and shape grammars are other examples of production formalisms. An ATN [Woods 1970] is a recursive transition network to which are added some extra features. Shape grammars [Stiny 1980] are a special kind of production rules where the condition part as well as the action part consist of shapes.

Production formalisms may be used for recognizing as well as generating patterns. They have a number of advantages:

1. Low-level modularity. Rules may easily be added, changed, or deleted without affecting other rules.

2. Representing knowledge as a set of production rules is a natural and appropriate method for many problem domains.

3. The easily readable form (if—then) may be used for explanations.

But certainly they also have disadvantages:

1. A set of rules has no internal structure and cannot immediately be grouped.

2. Matching is inefficient because of the lack of structure.

3. It may be difficult to predict the behaviour of a system by a static analysis (and hence should not be used for areas where safety is critical).

4. The system must contain represented knowledge on the use of rules, e.g., it may be implicit in the order of the rules, but hence it is unclear where control knowledge is represented.

Production formalisms are applied within many areas. Production systems are used in expert systems to the extent that the paradigm of knowledge base systems is often used synonymously with production systems. ATNs are used within natural language processing, and shape grammars are a powerful tool within space planning.

Uncertainty formalisms

An uncertainty formalism is characterized by:

> _The use of a continuous set of numbers expressing uncertainty or belief (in contrast to a discrete, finite set of truth values).

ncertainty formalisms range from statistics (e.g., Bayes' theorem and Hidden Markov Models [Rabiner 1988]) over neural nets to fuzzy logic, and are often used in connection with other formalisms, such as production systems.

Fuzzy logic [Giles 1982] is an uncertainty formalism. It has emerged from an attempt to develop a logic that models the fuzziness of natural language, e.g., predicates like young and big, and quantifiers like almost and some. Fuzzy logic is based on fuzzy set theory which is again based on the idea that an object x belongs to a fuzzy set A not just completely or not at all but with a certain grade of membership.

Uncertainty formalisms have performed reasonably, e.g., hidden Markov models dominate speech recognition, and expert systems like MYCIN and PROSPECTOR [Waterman 1986] have become well-known. However, systems based on uncertainty require tedious training (Hidden Markov Models) or ad hoc assignments (MYCIN). If observations are sparse and the problem domain open the meaning of the numbers become problematic to define. How do we convert from human terms to numeric uncertainty factors? E.g., what does "It is very likely that" mean? And how do we normalize such numbers across people's scales (cf. [Rich 1983])?

Analogical formalisms

It is not really correct to speak of analogical (or direct) formalisms since it is rather the way in which something is represented than the formalism used for it which makes it an analogical representation. An analogical representation directly mirrors the world. There must be a direct correspondence between parts, properties, and relations in the representation and in what is represented. An analogical representation may very well be made by using a non-analogical (propositional) formalism, e.g., by involving order. Analogical representations are characterized by the following points:

1. Analogy. The structure of the representation reflect the relational structure of the represented situation.

2. Simulation. Analogical models may be manipulated by arbitrarily complex procedures which often carry out a kind of (physical or geometrical) simulation.

In contrast to this, propositional representations have the following characteristic features:

1. Abstraction. Statements are true or false without geometrical or analogical similarity.

2. Inference. Propositional models are often used for the derivation of new statements by the

use of inference rules.

Often also the following two points will be characteristic for analogical representations:

1. Connection. Every item in a represented situation is found once with all its relations to other items.

2. Continuity. The representation is analogical with continuity in movement and time in the physical world.

whereas propositional representations may be characterized by:

1. Dispersion. An item may be found in several statements.

2. Discreteness. Usually statements are not used for representing a continuous change.

However, the two kinds of representation will not always differ in these latter points. Moreover, often it is possible to make a transformation from one representation to another without loss of information.

Diagrams are used for an analogical representation in [Funt 1980]. Here diagrams represent blocks in a blocks world.

Dual graphs are a general concept in mathematics, but e.g., in [Grason 1968] an interpretation is imposed which makes the representation analogical. Dual graphs are here used for representing floor plans.

An advantage of analogical representations is that they are easy to understand because they mirror the world directly, and structure is the same as contents. But analogical representations also have important limitations. It is not easy to make abstractions, focus on certain items, and leave out information and add it later. Analogical representations are not correct if something is missing (e.g., a map with a missing road). They are not just more or less complete.

Many applications of analogical representations exist, e.g., within computer vision and theorem proving. However, analogical representations do not seem useful for languages since these are abstract.

Programming formalisms

A programming formalism or language is a notation for the precise description of computer programs or algorithms [Illingworth 1983]. A programming language is also a formalism for representing knowledge but it is different from all the above mentioned formalisms. A programming language incorporates general purpose concepts (e.g., integer and array) which traditionally are closer to the executing machine than to the problem domain. All programming languages perform on a computer.

Examples of programming languages are machine code, assembler, C, C++, Pascal, Lisp, Prolog,

Standard ML and Miranda. Each language has a number of properties each of which it may share with one or more other languages. Such properties are low-level or high-level, efficiency, portability, imperative or declarative, procedural or non-procedural, solution oriented or problem oriented, etc. E.g., low-level languages like machine code are efficient but not portable since they are tailored to specific computers. This is in contrast to high-level languages like Miranda. These languages are portable but not so efficient because they must be compiled or interpreted which takes time.

Therefore, given a formalism, there are many different possibilities of implementation. At one extreme an interpreter for the formalism may be written, at the other extreme a specialized program hardwiring the formalism may be implemented. This is an instance of the general choice of the division between input and program (cf. [Naur 1974]).

Formalisms which are not programming formalisms are at a higher level of abstraction. It is usually easier to start at this level to formalize the main concepts and terms with which it would be natural and most easy to work. Therefore one should start there to look for a suitable formalism with good metaphors. If such a one cannot be found it must be tailored. The important thing is that it is easy to handle the natural concepts of the problem. This will at the end be less time-consuming than blindly choosing some inadequate formalism and perhaps skipping a level if the formalism is a programming language. Then indeed some formalisms are more well-suited for a certain problem than others.

It should be well-known that programming languages in general have been used for solving problems within many different domains. Some programming languages are more well-suited for the solution of a given problem than others, just as it is the case with other formalisms. Cobol, e.g., is primarily used for the implementation of administrative systems. Prolog has often been used for natural language processing, and Lisp within artificial intelligence. C was originally developed for implementation of operating systems but has today widely different areas of application.

Realization

Realization means to bring ones models to execute on a machine, in our case on a computer. The realization is the last phase ending in the final computer system. However, realization is considered also in the early phases. Already during the knowledge acquisition phase realization is to some extent taken into account, just as intended users, cost-benefit, areas of application, etc., are considered in this period and affects what data are acquired. It would be fairly stupid not to think of realization early in the system development since we might then end with models which cannot be mechanized and which must be rejected.

During the knowledge representation phase the acquired data are used as the basis for models represented, e.g., in natural language. These models then goes through a formalization process and are turned into formal models. A formal model is a representation of the intended system parts in some formal language, here called a knowledge representation formalism. Finally, this representation serves as a basis for the implementation process in the realization phase, and the result is (hopefully) a computer system.

Knowledge representation formalisms were described in the above sections. They have all been

used in computer systems and hence they are closely related to the realization. The programming languages of course already execute on a computer.

There are three ways in which to mechanize the mentioned formalisms. We may build an interpreter, or we may build a compiler, or we may handcode each object represented in the formalism using an existing programming language. The last possibility will probably only be used if we just want to use the formalism in question once for a single task.

Some of the formalisms can hardly be discussed without mentioning their realization (e.g., production formalisms). Other formalisms such as logic also have an existence in themselves. Some representations are well-defined (logic). Others have no formal theory, and this leads to a number of different interpretations and uses in implementation (semantic nets, frames). There are also elements which can be formally represented but which cannot be exploited in a computer, e.g., some of the features of higher-order logic. There are formalisms which have primarily been used by a narrow group of persons and within a restricted domain. Examples are Schank's scripts and Sowa's conceptual graphs which have been used for natural language processing. Contrary to this formalisms such as logic, production rules, and frames have been used by lots of persons and within a wide variety of fields.

8. Conclusion

The goal of the current project is the integration of speech and natural language into a spoken dialogue system capable of giving reasonable responses to user answers and questions within a limited domain chosen to be Danish domestic flight reservation and information.

This report has focused on the architecture of existing spoken dialogue systems and on the techniques and methods used in constructing the different parts of such systems. A summary of important aspects of system architecture and design process is given below and it is discussed what consequences the current state-of-the-art may have for the ongoing project.

8.1 Architecture of spoken dialogue systems

A spoken dialogue system typically consists of the following five components: a speech recognition module, a natural language analysis module, a dialogue manager, an answer generation module and an answer synthesis module. The emphasis is usually on the first three mudules. The answer generation often consists in converting database retrievals into natural language, and for the synthesis already existing system components are used.

The techniques for speech recognition seem to be relatively standardized. Recognition is almost always based on statistical methods and in particular on Hidden Markov Models. Moreover, a number of comparable test parameters such as perplexity, error rates and speed are widely used. This, however, does not mean that speech recognizers are very advanced. Real-time recognizers are rare and have small vocabularies (or large error rates). Small, user adaptive, spoken command recognizers have had the greatest success so far. Although faster and bigger computers may improve the recognizers as regards speed, it is desirable to experiment with new methods in order to, e.g., decrease the search space or increase the likelihood of finding the best sentence hypothesis. Systems like MINDS and Circuit Fix-it Shop exploit linguistic knowledge and knowledge of dialogue in the recognizer and this seems to lead to smaller error rates. It is not quite clear, however, whether such improvements also lead to improved efficiency and speed.

Current natural language analysis techniques are based on work on machine translation, story understanding, grammatical analysis by machines, etc. Only limited commercial success has been achieved so far unless one also counts simple spell checkers. Natural language analysis includes a syntactic as well as a semantic part. The methods and techniques for syntactic analysis are the most well-developed. Much work still remains to be done in semantic analysis before really successful systems can be created. Usually, the semantic problems closest to syntax are dealt with most extensively, probably because they are the easiest ones to handle. Linguistic knowledge is sometimes used in the speech recognizer. This is usually done statically and is restricted to n-grams and finite automata.
The dialogue manager normally controls the system. This module takes care of dialogue and discourse but the management is often restricted to state transitions in a finite automaton. These very elementary techniques are probably used because the problem area is so large and open that

it is difficult to know what more specialized techniques would be adequate. Some systems !e.g, MINDS, SPICOS and Circuit Fix-it Shop! use techniques from computer science (theorem proving) and discourse theory (resolution of ellipsis and anaphora, focusing and plan recognition). However, these techniques are easier to use in limited applications not involving a spoken interface.

8.2 Design of a spoken dialogue system

The design of a spoken dialogue system may be viewed as a traditional system development task. Apart from the phases concerning knowledge acquisition, formalization and realization depicted in figure 6.1, the design process also includes more general considerations of application:

_It must be clear at what group of users the application aims.

_The application should only be developed if the expected user group is large enough and there is a potential need for exactly that application (unless the aim is pure research).

_It must be carefully considered what funtionalities the users will need.

_The chosen solution should be compared with other potential solutions. E.g., when choosing a spoken dialogue system alternatives such as providing the users with mouse, keyboard, and monitor instead should be considered.

_The user interface is one of the most important things of a computer system. A poor user interface may be disastrous to the usability of a system. Much work has been done on the design of user interfaces for computer systems but mostly on systems including monitor, keyboard and mouse. The design of spoken interfaces requires a special emphasis on language, dialogue structure and discourse. Current design efforts rely heavily on Wizard of Oz simulation experiments. However, most experiments are constructed with regard to ideal systems with approximately full natural language understanding capabilities within some specified domain.

8.3 The current project

The general conclusions from this state-of-the-art survey of spoken dialogue systems with respect to the current project are the following:

_It will be necessary to develop a special version of the Wizard of Oz method aiming8 at a realistic system in technological terms, i.e., a system with limited capabilities.

_It must be carefully considered where standard methods and techniques should be used and where experiments with new possibilities would be appropriate.

_The implementation should be open to great variation within each of the modules. We should aim at a feasible system but also at a system which yields numerous possibilities for

experimentation.

_The first prototype should realize a fairly small system with just a few hundred words, a simple syntax, and a relatively system-directed dialogue. With today's technology it should be feasible to develop such a system which will run in approximately real-time.

_The habitability of a spoken dialogue system should be judged in its own right, against other similar spoken dialogue systems, and against systems with no speech interaction but with similar functionality.

Appendix A. Survey of existing systems

A survey of most of the spoken language based computer systems reported in the literature is given below. The list is ordered in alphabetical order.

For each system is provided (as far as the information is available): the *name*, the *domain*, and the main *references*; for the recogniser, number of *words* in the vocabulary, *speaker* dependence, type of *speech* recognised, and *time* for recognition; explicit *knowledge*, its *representation*, and its *integration* with the recogniser; and additional *notes*. Systems described in chapter 1 refer by [Chapter 1].

The list is only complete to the extent that we have found the information in the literature. In case more than one reference is stated the first one is our main source. Unless otherwise mentioned the systems do not run in real time.

APHODEX (Acoustic-PHOnetic Decoding EXpert system) [Haton 1988, Carbonell 1988]: administrative information system. *Recogniser***:** several thousands words, "many" speakers, and continuous speech.

ARGOT:

Circuit fix-it shop [Smith et al 1992]**:** Real-time support for the repair of electronic circuits. *Recogniser***:** 125 words, speaker independent, connected words. The recogniser is 50% correct at the utterance level whereas the overall system interprets 81.5% of the utterances correctly. *Language and dialogue***:** Domain, task and user models are used in the discourse handling and the dialogue state is used to delimit the lexicon in the speech recogniser. *Representation***:** Domain, dialogue control and discourse are programmed in Prolog, the parser in C. *Note***:** See Section 5.2.

DIALOGIC:

FDmDialog [Kitano 1991]**:** Speech-to-speech real-time translation system for the domain of conference registration. *Language and dialogue***:** Uses acoustic-phonetic, linguistic and discourse knowledge. The architecture has massive parallel processing, a connectivistic network, multilevel parse trees, and marker-passing.

Dragon: *Note***:** predecessor to the HARPY system.

Ernest [Sagerer et al. 1988]**:** Train travel system. *Representation***:** Uses semantic nets for the representation of linguistic knowledge.

EVAR [Niemann et al. 1987]**:** German intercity train inquiries. *Recogniser***:** 3918 words, speaker (simulated) independent continuous speech. 36 of the 64 German phones are

used. *Language and dialogue***:** Some acoustic-phonetic and linguistic knowledge, perhaps anaphora, but little dialogue control. Only built-in information used in the recogniser. *Representation***:** HMM (EMM) in the recogniser, semantic nets, case frames, and ATN for the language handling. Implemented in Lisp, C and Fortran. *Note***:** not blackboard. Lexicon unit is spelling.

GUS [Bobrow et al. 1977]**:** Travel information system. *Language and dialogue***:** domain and dialogue reasoning attempted.

HAM-ANS (Hamburg Application-Oriented Natural Language System) [Mctear 1987, Hoeppner 1984]**:** Applied to hotel reservations, databases and image analysis. *Recogniser***:** Simulated. *Language and dialoogue***:** A user model is maintained. Ellipsis resolution and quantifier handling is included. The dialogue structure is rather fixed. *Representation***:** Semantic nets are used. Note**:** Typed dialogue.

HARPY [Erman et al. 1981]**:** General 1011 word speech recognition system.

Hearsay-II [Erman et al. 1981]**:** Document retrieval system. *Recogniser***:** 1011 word models, speaker dependent, connected speech. *Note***:** Blackboard architecture. Comparable to HARPY.

HWIM (Hear What I Mean) [Erman et al. 1981, Woods 1983]**:** Travel planning system. *Recogniser***:** 1097 word models, speaker dependent. *Representation***:** Restricted ATN grammar. Implemented in Algol-60.

KEAL-NEVEZH (KEAL, new) [Mercier et al. 1988]**:** Recognition of numbers, pseudo-LOGO commands, and interfacing for yellow pages database.*Recogniser***:** Speaker dependent continuous speech.

MAX [MAX 1991]**:** Real-time EEC information system in practical use. *Recogniser***:** 16 words, speaker independent, isolated words. *Representation***:** Domain information in database. *Note***:** Synthetic speech.

MINDS (Multi-Modal Interactive Dialog System) [Young et al. 1990]**:** Air traffic information system (ATIS). *Recogniser***:** speaker independent, connected speech. *Language and dialogue***:** Uses syntax, discourse, dialogue, user domain knowledge, and goal and task expectations. Pragmatic results are used dynamically for constraining the recogniser. *Representation***:** Semantic nets are used. *Note***:** The recogniser is based on SPHINX.

PARRY:

SDI:

SOUL/MINDS-II [Chapter 1]**:**

SPHINX:

SPICOS II (Siemens-Philips-IPO-Continuous Speech Understanding and Dialogue System) [Niedermair et al. 1990] [Chapter 1]**:** Office database system. *Recogniser***:** 1200 words, speaker adaptable, continuous speech. *Language and dialogue***:** Anaphora resolution and discourse representation is used. Semantic restrictions are used for pruning the search tree. *Representation***:** HMM, bigram, chart parser, augmented phrase-structure grammar, logic and semantic nets. Implemented in Prolog. *Note***:** Expects grammatically correct input. (SPICOS I ran in 20 times real time).

SRI [Erman et al. 1981]**:**

SUMMIT+TINA [Philips et al. 1991, Goodine et al. 1991, Seneff 1989] [Zue et al. 1991] [Chapter 1]**:** Used for VOYAGER (urban navigation) and ATIS (air travel information system) running 3-5 times real time. *Recogniser***:** 350 words in VOYAGER, 500 words in ATIS. Speaker independent, continuous speech. *Language and dialogue***:** Apart from linguistic knowledge, context and discourse history is used. *Representation***:** Semantic frames, SQL and word pair grammars. *Note***:** Pauses within utterances are allowed.

SUNSTAR:

SUNDIAL [McGlashan et al. 1992] [Chapter 1]**:** Flight reservations and enquiries in English, train enquiries in German and Italian. Curently 10 seconds response, promise is real time. *Recogniser***:** 300 words in German, Italian, French and English. *Note***:** Started 1989.

VESPRA:

VODIS II [Young et al. 1991, Cookson 1988] [Chapter 1]**:** Traffic inquiry system. *Recogniser***:** speaker dependent, continuous speech, based on dynamic time warping.

Voice Master Key [Christ 1992]**:** Real-time menu control system. *Recogniser***:** 256 words in all, 16 words at a time. Speaker dependent and adaptable, isolated words. *Note***:** Commercially available for PC at DKK 1600 + VAT.

References

Baggia, P., Ciaramella, A., Clementino, D., Fissore, L., Gerbino, E., Giachin, E., Micca, G., Nebbia, L., Pacifici, R., Pirani, G. and Rullent, C., A man-machine dialogue system for speech access to e-mail information using the telephone: implementation and first results, *Proceedings, Eurospeech 91 3*(1991), 865-8.

Brietzmann, A. and Ehrlich, U., The role of semantic processing in an automatic speech understanding system, *COLING-86*, 1986, 596-598.

Carbonell, J. G. and Hayes, P. J., Recovery strategies for parsing extragrammatical langua ge, *Computational Linguistics 9*, 3-4 (1983), 123-146.

Carbonell, J. G. and Hayes, P. J., Robust Parsing Using Multiple Construction-Specific Strategies, in *Natural Language Parsing Systems*, Bolc, L. (ed.), Springer-Verlag, Berlin, 1987, 1-32.

Giachin, E. P. and Rullent, C., Robust parsing of severely corrupted spoken utterances, *COLING-88*, Budapest, 1988, 196-201.

Hayes, P. J., Entity-oriented parsing, *COLING-84*, 1984, 212-217.

Hayes, P. J., Hauptmann, A. G., Carbonell, J. G. and Tomita, M., Parsing spoken language: a semantic caseframe approach, *COLING-86*, 1986, 587-592.

Norton, L., Linebarger, M., Dahl, D. and Nguyen, N., Augmented Role Filling Capabilities for Semantic Interpretation of Spoken Language, *DARPA (June)*, 1991, 125-33.

Seneff, S., Glass, J., Goddeau, D., Goodine, D., Hirschman, L., Leung, H., Phillips, M., Polifroni, J. and Zue, V., Development and Preliminary Evaluation of the MIT ATIS System, *DARPA (Feb)*, 1991, 88-93.

Ward, W., The CMU Air Travel Information Service: Understanding Spontaneous Speech, *DARPA (June)*, 1990, 127-9.

Weischedel, R. M. and Ramshaw, L. A., Reflections on the knowledge needed to parse ill-formed language, in *Machine Translation: Theoretical and Methodological Issues*, Nirenberg, S. (ed.), Cambridge University Press, Cambridge, 1987, 155-167.

Young, S. and Matessa, M., Using pragmatic and semantic knowledge to correct parsing of spoken language utterances, *Proceedings, Eurospeech 91 1*(1991), 223-7.

==========

(tom)

Brøndsted, T.: "Interface between SUNCAR and a NLS Component", internal draft STC, feb. 1992

$$ (flere ?)
Cookson, S.: Final Evaluation of VODIS. Voice Operated Database Inquiry System. Pro ceedings of Speech (1988).

Goodine, D., Seneff, S., Hirschman, L., Philipps, M.: Full Integration of Speech and Lan guage Understanding in the MIT Spoken Language System. Eurospeech 2 (1991), 845-848.

Ney, H. et al.: Phoneme-Based Continous Speech Recognition Results for Different Lan guage Models in the 1000-Word SPICOS System. Speech-Communication 7 (1988), 367-374

$$ (et al)
Niedermair, G.T. et al.: Linguistic Processing Related to Speech Understanding in SPICOS II. Speech Comunication 9 (1990), 565-585

Pallet, S.D., Fisher, W.M., Fiscus, J.G.: Tool for Analysis of Benchmark Speech Recogni tion Tests. ICASSP 90 (1990), 97-100

Phillips, M., Glass, J., Zue, V.: Automatic Learning of Lexical Representations for Sub- word Unit Based Speech recognition Systems. Eurospeech 2 (1991), 577-580.

$$ (Seneff + flere?)
Seneff, S.: TINA: A Probabilistic Syntactic Parser for Speech Understanding Systems. DARPA Speech and Natural Language Workshop (1989)

$$ (et al.)
Young, S.J. et al.: The use of Syntax and multiple alternatives in the VODIS voice oper ated database inquiry system. Computer Speech and Language (1991) 5, 65-80

Zue, V., Glass, J., Goodine, D., Leung, H., McCandless, M., Philipps, M., Polifroni, J., Seneff, S.: Recent Progress on the Voyager System. Proc. Speech and Natural Language Workshop (1990), 206-211

$$ (et al)
Zue, Victor et al.: Spoken Language Systems for Human/Machine Interfaces. Speech Technology and Data Compression 14 (1991), 936-955

Peckham J. : "Speech Understanding and Dialogue over the telephone: An overview of progress in the SUNDIAL project" Proceedings of EUROSPEECH 91, 24 - 26 September 1991, Genoa Italy. pp. 1469-72.
Andry, F & Thornton, S : "A parser for lattices using a UCG grammar", Proceedings of EUROSPEECH 91, 24 - 26 September 1991, Genoa Italy. pp. 219-22.

Lee, k. F : "The SPHINX system", Ph.D. Thesis CMU ...etc $$ 1989

Young, S.R. : Personal Communication 1991

Young S.R. Matissa : fra EUROSPEECH

Young S.R. et al. $$ $$ $$ 1989 fra Speech comm. or speech & na.t lang.

Husøy, P.O. : "Forward Connected Artificial Neural Networks Applied to Automatic Speech Recognition", Ph.D thesis, NTH, Trondheim 1991, Norway.

Mariani

Rich, E. : Artificial Intelligence ...

Bech A., *Hvem var ansvarlig for bombeangrebet? Om opbygning og anvendelse af en vidensbase i et tekstforståelsessystem*, in [Engberg-Pedersen et al.]

Copeland, C., J. Durand, J., S. Krauwer, B. Maegaard (eds), *Studies in Machine Translation and Natural Language Processing, vol 1: The EUROTRA Linguistic specifications*. Luxembourg, 1991.

Engberg-Pedersen et al, *Anvendt Sprogvidenskab*, København, 1990.

Grishman R., R. Kittredge, *Analyzing language in Restricted domains*, Lawrence Erlbaum, 1986.

Hayes, P.J, *The Second Naive Physics Manifesto*, in [Hobbs and Moore]

Hayes, P. J., Hauptmann, A. G., Carbonell, J. G. and Tomita, M., *Parsing spoken lan guage: a semantic caseframe approach*, in *COLING-86*, 1986, 587-592.

Hobbs J., R.C. Moore (eds), *Formal Theories of the Commonsense World*, New Jersey, 1985.

Kittredge R., J. Lehrberger, *Sublanguage*, De Gruyter, 1982.

Klein, Veltman, *Natural Language and Speech*. Springer Verlag, 1991.

Liberman M, *Title unknown*, in [Klein and Veltman]

McNaught J.,*Introduction to Sublanguage*, in [*Record of SALT/ELSNET Sublanguage Workshop, 7-8 January 1992*], 5-18.

Niedermair, G.T. et al, *Linguistic Processing Related to Speech Understanding in SPICOS II*, in *Speech Comunication 9* (1990), 565-585.

Seneff, S., *TINA: A Probabilistic Syntactic Parser for Speech Understanding Systems*, DARPA Speech and Natural Language Workshop (1989)

Thompson H., *Sublanguage and Statistics: A Bluffer´s Guide*, in [*Record of SALT/ELSNET Sublanguage Workshop, 7-8 January 1992*], 19-26.

Wayne W., *Understanding Spontaneous Speech*,in *DARPA Speech and Natural Language Workshop*, 1989, 137-141.

Zue W.V. et al., *Spoken Language Systems for Human/Machine Interfaces*, in *RIAO, Intelligent Text and Image Handling, Conference Proceedings*, 935-955.

Tomita M., (ed), *Current Issues in Parsing Technology*, Boston, 1991.

Harris Z., Mathematical Structures of Language, New York, 1968.

Sager C., *English Special Languages. Principles and Practice in Science and Technology*,
Wiesbaden, 1980.B_kgaard, Anders and Paul Dalsgaard: "Tools for Designing Dialogues in Speech Understanding Interfaces", in: "Proceedings of International Conference on Spoken Language Processing", Kobe, Japan, November 1990, pages 945-948.

Bates, Madeleine and Oliviero Stock (eds.): "Third Conference on Applied Natural Language Processing", 31 March -- 3 April 1992.

Bobrow, Daniel G., Ronald M. Kaplan, Martin Kay, Donald A. Norman, Henry Thompson, and Terry Winograd: "GUS, A Frame-Driven Dialog System", in: Artificial Intelligence, vol 8, 1977, pages 155-173, also in [Grosz et al. 1986, p595--604].

Brachman, Ronald J. and Hector J. Levesque (eds.): "Readings in Knowledge Representation", Morgan Kaufmann, 1985.

Brady, Michael and Robert C. Berwick (eds.): "Computational Models of Discourse", The MIT Press, Cambridge, Mass., 1983.

Carberry, Sandra: "Plan Recognition in Natural Language Dialogue", ACL-MIT Press Series in Natural Language Processing, The MIT Press, 1990.

Carbonell, N. and J.M. Pierrel: "Task-Oriented Dialogue Processing in Human-Computer Voice Communication", in: Niemann, H., M. Lang, and G. Sagerer (eds.): "Recent Advances in Speech Understanding and Dialog Systems", NATO ASI Series F: Computer and Systems Sciences, vol 46, Springer-Verlag, 1988, pages 491-495.

Carter, David: "Interpreting Anaphors in Natural Language Texts", Ellis Horwood, 1987.

Christ, Bj_rn: "PÕ talefod med PD'en", in: Alt om Data, vol 3, March 1992, pages 114-118.

Cullingford, R.: "{SAM}", in: Schank, R. C. and C. K. Riesbeck (eds.): "Inside Computer Understanding: Five Programs Plus Miniatures", Lawrence Erlbaum, Hillsdale, N.J., 1981.

Dahl, Deborah A.: "Focusing and Reference Resolution in PUNDIT", in: "Proceedings AAAI-86, Fifth National Conference on Artificial Intelligence, august 11-15, 1986, Philadelphia, Pa.", 1986.

Danair: "Tidtabel Vinter 91/92", 1991.

Danair: "Tidtabel Sommer 92", 1992.

DARPA: "Speech and Natural Language, Proceedings of a Workshop", Morgan Kaufmann Publishers, Inc., USA, October 1989.

DARPA: Stern, Richard (ed.): "Speech and Natural Language, Proceedings of a Workshop held at Hidden Valley, Pennsylvania", Morgan Kaufmann Publishers, Inc., USA, June 24--27 1990.

DARPA: "Speech and Natural Language, Proceedings of a Workshop", Morgan Kaufmann Publishers, Inc., USA, February 1991.

Dowty, David R., Robert E. Wall, and Stanley Peters: "Introduction to Montague Semantics", D. Reidel Publishing Company, 1981, reprint with corrections 1988.

Dybkj_r, Laila: "Knowledge Representation. The Concept and its Use", Tech. Report 88/17, DIKU, Department of Computer Science, University of Copenhagen, 1988.

Dybkj_r, Laila: "Computer-Aided Floor Plan Sketching", Tech. Report 92/1, DIKU, Department of Computer Science, University of Copenhagen, 1992, PhD thesis.

Dybkj_r, Hans and Laila Dybkj_r: "Phrases for a Travel Information System", Technical report, Centre for Cognitive Informatics, Roskilde University, Denmark, January 1992, draft, 5pp.

Dybkj_r, Laila and Claus Povlsen: "Interview med Jesper i Dan Transport".

Dybkj_r, Laila and Claus Povlsen: "Interview med Jesper og Kim i Dan Transport".

Eriksen, Bj_rn, Hans J_rgen Helms, and Mogens D. R_mer: "EDB-ordbog", Gjellerup, Denmark, 1975.

Erman, Lee D., Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy: "The {Hearsay-II} Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty", in: Webber, Bonnie Lynn and Nils J. Nilsson (eds.): "Readings in Speech Recognition", Morgan Kaufmann, 1981, pages 349-389.

Fillmore, Charles: "The case for case", in: Bach, E. and R. Harms (eds.): "Universals in Linguistic Theory", Holt, 1968, pages 1-88.

Fraser, Norman M. and G. Nigel Gilbert: "Simulating speech systems", in: Computer Speech and Language, vol 5, 1991, pages 81-99.

Fraser, Norman: "Wizard of Oz Methods", Abstract for talk given at workshop in Trento, March 30 1992.

Frost, Richard A.: "Introduction to Knowledge Base Systems", Collins, London, 1986.

Funt, Brian V.: "Problem Solving with Diagrammatic Representations", in: Brachman, Ronald J. and Hector J. Levesque (eds.): "Readings in Knowledge Representation", Morgan Kaufmann, 1985, chapter VIpages 441-456.

Giles, Robin: "Semantics for Fuzzy Reasoning", in: Int. J. Man-Machine Studies, vol 17, 1982, pages 401-415.

Grason, John: "A Dual Linear Graph Representation for Space-Filling Location Problems of the Floor Plan Type", in: Moore (ed.): "Emerging Methods in Environment Design and Planning", June 1968.

Grosz, Barbara J.: "Discourse Knowledge", in: Walker, D. (ed.): "Understanding Spoken Language", Elsevier North-Holland, New York, 1978, chapter VIII--XIV (Section 4)pages 229-346.

Grosz, Barbara J. and Candace L. Sidner: "Attention, Intentions, and the Structure of Discourse", in: Computational Linguistics, vol 12, nr 3, 1986.

Grosz, Barbara J., Karen Sparck Jones, and Bonnie Lynn Webber (eds.): "Readings in Natural Language Processing", Morgan Kaufmann, 1986.

Grosz, Barbara J., Martha E. Pollack, and Candace L. Sidner: "Discourse", in: Posner, Michael I. (ed.): "Foundations of Cognitive Science", The MIT Press, 1989, chapter 11pages 437-468.

Gruber, Thomas: "Learning Why by Being Told What. INteractive Acquisition of Justifications", in: IEEE Expert, august 1991, pages 65-75.

Halliday, M. A. K. and Ruqaiya Hasan: "Cohesion in English", Longman, Bath, 1976.

Hart, Anna: "Knowledge Acquisition for Expert Systems", Kogan Page Ltd., London, 1986, Newer version from 1988 or 1989 exists.

Haton, Jean-Paul: "Knowledge-Based Approaches in Acoustic-Phonetic Decoding of Speech", in: Niemann, H., M. Lang, and G. Sagerer (eds.): "Recent Advances in Speech Understanding and Dialog Systems", NATO ASI Series F: Computer and Systems Sciences, vol 46, Springer-Verlag, 1988, pages 51-70.

Hoeppner, W., K. Morik, and H. Marburger: "Talking it over: the natural language dialogue system {HAM-ANS}", Tech. Report ANS-26, University of Hamburg research unit for information science and artificial intelligence, 1984.

Illingworth, Valerie (ed.): "Oxford Dictionary of Computing", 3rd edition 1990, paperback 1991 edition, Oxford University Press, 1983.

Jelinek, Frederick: "Self-Organized Language Modeling for Speech Recognition", in: "Stochastic Approaches", IEEE, 1989, pages 450-506.

Joshi, A., Bonnie Webber, and Ivan Sag (eds.): "Elements of discourse understanding", Cambridge University Press, Cambridge, Mass..

Kitano, Hiroaki: "_DmDialog: A Speech-to-Speech Dialogue Translation System", in: Machine Translation, vol 5, nr 4, 1991, pages 301-338.

Kowalski, Robert: "Logic for Problem Solving", North-Holland, 1979, fourth printing 1984.

MAX: "Reference Card for MAX", ECHO, European Commision Host Organisation, B.P. 2373, L-1023 Luxembourg G.D., 1991, {MAX} is a speaking robot recognizing 16 words; call free, and ``he" will offer various information such as ecu-prices and {EEC}-unemployment rates.

McGlashan, Scott, Norman Fraser, Nigel Gilbert, Eric Bilange, Paul Heisterkamp, and Nick Youd: "Dialogue Management for Telephone Information Systems", in: Bates, Madeleine and Oliviero Stock (eds.): "Third Conference on Applied Natural Language Processing", 31 March -- 3 April 1992, pages 245-246, poster.

McGraw, Karen L. and Karan Harbison-Briggs: "Knowledge Acquisition: Principles and Guidelines", Prentice-Hall International, 1989.

McTear, Michael: "The ARticulate Computer", Basil Blackwell, Oxford, 1987.

Mercier, G., A. Cozannet, and J. Vaissiere: "Recognition of Speaker-Dependent Continuous Speech with {KEAL-NEVEZH}", in: Niemann, H., M. Lang, and G.

Sagerer (eds.): "Recent Advances in Speech Understanding and Dialog Systems", NATO ASI Series F: Computer and Systems Sciences, vol 46, Springer-Verlag, 1988, pages 459-464.

Minsky, Marvin: "A Framework for Representing Knowledge", in: "The Psychology of Computer Vision", McGraw-Hill, 1975.

Naur, Peter: "Concise Survey of Computer Methods", Studentlitteratur, 1974.

Niedermair, G. Th., M. Streit, and H. Tropf: "Linguistic Processing Related to Speech Understanding in SPICOS II", in: Speech Communication, vol 9, 1990, pages 565-585.

Niemann, H., A. Brietzmann, U. Ehrlich, S. Posch, P. Regel, G. Sagerer, R. Salzbrunn, and G. Schukat-Talamazzini: "A Knowledge Based Speech Understanding System", in: International Journal of Pattern Recognition and Artificial Intelligence, vol 2, nr 2, 1988, pages 321-350.

Niemann, H., M. Lang, and G. Sagerer (eds.): "Recent Advances in Speech Understanding and Dialog Systems", NATO ASI Series F: Computer and Systems Sciences, vol 46, Springer-Verlag, 1988, Proceedings of the NATO Advanced Study Institute on Recent Advances in Speech Understanding and Dialog Systems, Bad Windsheim, FRG, July 5--18, 1987.

Perrault, C. Raymond and Barbara J. Grosz: "Natural-Language Interfaces", in: Annual Reviews of Computer Science, vol 1, 1986, pages 47-82.

Pollack, Martha: "A Model of Plan Inference that Distinguishes between the Beliefs of Actors and Observers", in: "Proceedings of the Twenty-fourth Annual Meeting of the Association for Computational Linguistics", New York, 1986, pages 207-214.

Rabiner, Lawrence R.: "Mathematical Foundations of Hidden Markov Models", in: Niemann, H., M. Lang, and G. Sagerer (eds.): "Recent Advances in Speech Understanding and Dialog Systems", NATO ASI Series F: Computer and Systems Sciences, vol 46, Springer-Verlag, 1988, pages 183-206.

Rich, Elaine: "Artificial Intelligence", McGraw-Hill Book Company, 1983.

Ringland, Gordon A. and David A. Duce (eds.): "Approaches to Knowledge Representation, an Introduction", paperback 1989 edition, Reasearch Studies Press Ltd. (Wiley), Somerset, England, 1988.

Sagerer, G. and F. Kummert: "Knowledge Based Systems for Speech Understanding", in: Niemann, H., M. Lang, and G. Sagerer (eds.): "Recent Advances in Speech Understanding and Dialog Systems", NATO ASI Series F: Computer and Systems Sciences, vol 46, Springer-Verlag, 1988, pages 421-458.

Schank, Roger C. and Rober P. Abelson: "Scripts, Plans, Goals and Understanding. An Inquiry into Human Knowledge Structures", Lawrence Erlbaum Associates, Publishers, 1977.

Schank, Roger C.: "Language and Memory", in: "Perspectives of Cognitive Science", Ablex Publishing Corporation, New Jersey, 1981.

Schwartz, Richard M., Y. Chow, M. Dunham, O. Kimball, M. Krasner, F. Kubala, J. Makhoul, P. Price, and S. Roucos: "Acoustic-Phonetic Decoding of Speech (Statistical Modeling for Phonetic Recognition)", in: Niemann, H., M. Lang, and G. Sagerer (eds.): "Recent Advances in Speech Understanding and Dialog Systems", NATO ASI Series F: Computer and Systems Sciences, vol 46, Springer-Verlag, 1988, pages 25-50.

Sidner, Candace L.: "Focusing in the comprehension of definite anaphora", in: Brady, Michael and Robert C. Berwick (eds.): "Computational Models of Discourse", The MIT Press, Cambridge, Mass., 1983, page ??.

Smith, George W.: "Computers and Human Language", Oxford University Press, 1991.

Smith, Ronnie W., D. Richard Hipp, and Alan W. Biermann: "A Dialog Control Algorithm and Its Performance", in: Bates, Madeleine and Oliviero Stock (eds.): "Third Conference on Applied Natural Language Processing", 31 March -- 3 April 1992, pages 9-16.

Sowa, John F.: "Conceptual Graphs for a Data Base Interface", in: IBM Journal of Research and Development, vol 20, nr 4, July 1976.

Sowa, John F.: "Conceptual Structures: Information Processing in Mind and Machine", Addison-Wesley, 1984, New York.

Stiny, G.: "Introduction to Shape and Shape Grammars", in: Environment and Planning B, vol 7, 1980, pages 343-351.

Thurmair, G.: "Semantic Processing in Speech Understanding", in: Niemann, H., M. Lang, and G. Sagerer (eds.): "Recent Advances in Speech Understanding and Dialog Systems", NATO ASI Series F: Computer and Systems Sciences, vol 46, Springer-Verlag, 1988, pages 397-420.

Waibel, Alex and Kai Fu Lee (eds.): "Readings in Speech Recognition", Morgan Kaufmann, 1990.

Waterman, Donald Arthur: "A Guide to Expert Systems", The Teknowledge Series in Knowledge Engineering, Addison-Wesley, 1986.

Webber, Bonnie L.: "A Computational Approach to Discourse Anaphora", ???, New

York, 1980.

Webber, Bonnie L.: "So what can we talk about now?", in: Brady, Michael and Robert C. Berwick (eds.): "Computational Models of Discourse", The MIT Press, Cambridge, Mass., 1983, page ???.

Wilensky, Robert: "Planning and Understanding", Addison-Wesley, Reading, MA, 1983.

Woods, William A.: "Transition Network Grammars for Natural Language Analysis", in: Communications of the ACM, vol 13, nr 10, October 1970.

Woods, William A.: "What's in a Link: Foundations for Semantic Networks", in: "Representation and Understanding. Studies In: Cognitive Science", Academic Press, New York, 1975.

Woods, W.A.: "Language Processing for Speech Understanding", in: Waibel, Alex and Kai Fu Lee (eds.): "Readings in Speech Recognition", Morgan Kaufmann, 1990, chapter 8.4pages 519-533, originally from 1983.

Young, Sheryl R., Alexander G. Hauptmann, Wayne H. Ward, Edward T. Smith, and Philip Werner: "High Level Knowledge Sources in Usable Speech Recognition Systems", in: Waibel, Alex and Kai Fu Lee (eds.): "Readings in Speech Recognition", Morgan Kaufmann, 1990, chapter 8.6pages 538-548.

Yourdon, Edward: "Managing the System Life Cycle", Yourdon Press, New York, 1982.

Zoltan-Ford, Elizabeth: "How to get people to say and type what computers can understand", in: International Journal on Man-Machine Studies, vol 34, 1991, pages 527-547.

Zue, Victor W., James R. Glass, Dave Goddeau, David Goodine, Lynette Hirschman, Hong C. Leung, Michael S. Philips, Joseph Polifroni, and Stephanie Seneff: "Spoken Language Systems for Human/Machine Interfaces", in: Mariani, M. (ed.): "RIAO '91 Conference Proceedings on Intelligent Text and Image Handling, Barcelona, Spain", vol 2, 1991, pages 936-955.

List of Terms

(ikke endeligt ajourført).

Brief definitions of common terms related to human-computer interfaces based on speech and natural language are given. Danish equivalents and page references are provided. Cross references are *slanted*.

allophone (allofon): A variant of a *phoneme* or basic sound.

anaphora (anafor):

application (applikation): ``Application'' is ambiguosly used for the whole system (user, task, ...) as well as for the (usually computer based) back-end.

apprentice (l_rling) [Fraser and Gilbert 1991]: Person who assists a *wizard*; also used of software tools used by the wizard.

architecture (arkitektur): Primarily the system architecture, i.e., the specification of the system components and their interaction.

bigram (bigram): Statistical description of the word pairs that can possibly appear in (speech based) input to a computer system.

connected word recognizer (genkender af sammenk_dede ord): A speech recognizer designed to recognize connected words (it may be applied to the task of recognizing *continuous speech* with the result of degrading performance). The recognition units within the CWR are normally *whole word models*.

connected words (sammek_dede ord): Words spoken carefully, but with no explicit pauses between them.

continuous speech (kontinuert tale, flydende tale): Words spoken fluently and rapidly as in conversational speech.

continuous speech recognizer (genkender af kontinuert tale): A speech recognizer designed to recognize *continuous speech*. The recognition units within the CSR are normally *subword models*, but very often hybrid systems are found which contain mixtures of subword and *whole word models*.

data (data) [Eriksen et al. 1975]: A formalized representation of facts or ideas in such a form that it may be communicated or transformed by a

process.

database (database): A data file which in contrast to a *knowledge base* consists of explicitly stated facts (whereas rules are implicitly stored), is relatively large, and may only be used for retrieving the facts it contains.

dialogue (dialog): Communication between two (or more) persons or machines; covers language, dialogue structure, topics, ...

discourse (diskurs) [Grosz et al. 1989]: Conversation or argumentation; the larger context; the text or dialogue in which a sentence or utterance occurs, and which is necessary to its explanation.

domain (domæne): A delimited part of a world.

ellipsis (ellipse):

expert system (ekspertsystem) [Waterman 1986]: A *knowledge base system* based on expert knowledge to attain high levels of performance in a narrow problem area. Expert systems often have explanation facilities.

filler model: A statistical *subword model* of speech or words outside a given vocabulary.

formant (formant): A resonant frequency in spoken sound.

garbage model (støjmodel): A statistical *whole word model* of speech or words outside a given vocabulary. Often more than one garbage model will be used in a given *speech system*.

grammar (grammatik): Description of the (syntactical) structure of a language. In computer science usually a formal description of programming language syntax.

hidden Markov models (skjulte markovmodeller) [Rabiner 1988]: Common statistical modelling technique for acoustic decoding of speech sounds.

human-computer interaction (menneske-maskine interaktion): The communication between a human user and a computer system.

inference engine (inferensmaskine) [Dybkjaer 1988]: That part of a *knowledge base system* which contains the general problem-solving knowledge. Ideally there is a strict division between the represented general problem-solving knowledge (in the inference engine) and the represented domain knowledge (in the *knowledge base*).

information (information): Knowledge that may be communicated as data, conciously.

isolated word recognizer (isoleret-ordgenkender): A speech recognizer designed to recognize isolated words. Teh recognition units within an isolated word recognizer are normally whole word models.

isolated words (isolerede ord) \body Words spoken with pauses (typically with duration in excess of 200 ms) before and after each word.

knowledge (viden) [Dybkjaer 1988]: ``Understanding" or ``information about"; Seneca contrasts `knowledge' to `remembering': ``to remember is to preserve something committed to memory; to know, by contrast, is to make each item your own, not to depend on a model and to be constantly looking back at the teacher".

knowledge acquisition (videnindhentning, -indsamling, -hjemtagning) [Hart 1986]: The acquisition of information from different sources such as general books, articles, material from the work place, recordings, and interviews with and studies of experts. The purpose is to enable modelling of expert knowledge to be represented in a computer.

knowledge base (videnbase) [Dybkjaer 1988]: That part of a *knowledge base system* which contains the represented domain knowledge. In contrast to a *database* a knowledge base consists of explicitly stated general rules and facts, is relatively small, and may be used for different purposes while its contents remain unchanged.

knowledge base system (videnbasesystem) [Dybkjaer 1988, Frost 1986]: Traditionally a computer system with a separation into a *knowledge base* and an *inference engine*. Most *expert systems* have this structure.

knowledge-based system (videnbaseret system): The same as *knowledge base system*. Hopefully, every computer system or programme is based on knowledge.

knowledge elicitation: See *knowledge acquisition*.

knowledge engineer (videningeniør) [Waterman 1986]: The person who designs and builds the *knowledge base system* or *expert system*.

knowledge engineering (videningeniørarbejde) [Waterman 1986]: The process of building *knowledge base systems* or *expert systems*.

knowledge representation (videnrepr_sentation) [Dybkjaer 1988]: Modelling. Know\-ledge representation involves:

- A domain of discourse, i.e., some physical/abstract (part of a) world.
- A language in which to represent knowledge about the world.
- Connections (between the world and the represented knowledge): (a) The encoding (representation) of knowledge of some world in some language. (b) The decoding (interpretation) of represented knowledge.

man-machine interaction (menneske-maskine interaktion): See *human-computer interaction* (though, in general the machine need not be a computer).

model (model) [Dybkjaer 1988]: Conceptual construction; hypothesis for explaining and predicting (physical) world phenomena; representation of (part of) the world.

natural language (naturligt sprog): Communication instrument used by some human beings.

parser (parser, syntaksanalysator):

perplexity (perpleksitet) [Jelinek 1989, Young et al. 1990]: The average size of the set of words expected next by a speech recognizer.

phoneme (fonem): The basic sound unit in a language.

plan recognition (plangenkendelse) [Carberry 1990]: Under the assumption that the user has a plan of how to obtain his/her goals, reveal that plan.

prerecorded speech (forud-optaget tale): See *synthetic speech* (b).

project charter (kravspecifikation) [Yourdon 1982]: Project abstract + statement of goals and objectives + customized project life cycle + schedule constraints + technical and procedural constraints + preliminary project scenarios.

recognition unit: A basic unit of speech on which recognition operations are performed.

scalability (skalerbarhed): The possibility of adjusting the size of a system, e.g., by adding or removing functionality, or by improving performance.

speech (tale): Acoustic form of natural language produced by humans. See also *synthetic speech*.

speech corpus (taledatabase):

subject (subjekt, forsøgsperson): User (of a system) within simulation experiments (e.g., wizard of Oz).

subword models (delordsmodeller): Recognition units are models of smaller parts than words (e.g. phonemes, diphones, triphones, demisyllables, syllables).

syntax (syntaks): See *grammar*.

synthetic speech (syntetisk tale): (a) Speech generated on the basis of a phonetic interpretation of written text. (b) Computer speech composed of units of recordings of human speech.

task (opgave): That which is to be performed or solved; usually implies a goal.

user (bruger): Person who is using a (computer-based) system.

viterbi search (viterbisøgning): Dynamic programming algorithm for finding the optimal path in a search space.

whole word models (helordsmodeller): Recognition units are models of whole words.

wizard (troldmand) [Fraser and Gilber 1991]: Experimenter. Person who---unknown to the user---simulates (part of) a computer system.

wizard of Oz (Troldmanden fra Oz) [Fraser and Gilber 1991]: This is a technique to check the dialogue before implementation: the computer and the menus are simulated by a person (the simulator) while a user is trying to pretend that s/he uses the real system (or is faked into that belief). Meanwhile the simulator (or a helper) takes notes about the dialogue (structure, errors, ...). word-pair (ordpar): Simple grammatical representation of the fact that the twoo words may appear in sequence.

word rejection : The abilitiy of a speech recognizer to reject out-of-vocabulary words.

word spotting : The ability of a speech recognizer to detect one or more vocabulary words from an utterance which contains vocabulary words and unconstrained speech.