ANALYSING COHERENCE OF INTENTION IN NATURAL LANGUAGE DIALOGUE

BY

PAUL MC KEVITT, B.Sc. (Hons.) (Dublin), M.S. (New Mexico)

Submitted by Paul Mc Kevitt
to the University of Exeter
as a thesis for the degree of
Doctor of Philosophy
in the Faculty of Science

University of Exeter
Exeter, United Kingdom, E.C.
September, 1991.

I certify that all the material in this thesis which is not my own work has been identified and that no material is included for which a degree has previously been conferred upon me.

Paul Mc Kevitt

Supervisor:

Professor Derek Partridge
Department of Computer Science
University of Exeter

Examiners in charge:

Internal Examiner:

Dr. Antony Galton
Department of Computer Science
University of Exeter

External Examiner:

Dr. David Benyon
Department of Computing
The Open University

Lé
Peter, Rose, Peádar, Tara, agus Míchael

Contents

List of Figures

List of Tables

0. Preface

The work described here does not concern itself so much with the structure, or syntax, the meaning, or semantics, of language, but more the use of language. The work, then, lies firmly within the field of *pragmatics*[1]. The study of pragmatics with reference to the computer processing of language, or *natural-language processing*, is of particular interest. In recent years the study of pragmatics has become a new and emerging subarea of study within natural-language processing[2]. Specifically, the processing of written English shall be of interest here. While there has been much work done on solving the problems of syntax and semantics in natural-language processing, there are comparatively few studies on pragmatics.

Our work is about the analysis of the intentions, or goals, plans and beliefs, of people in natural-language dialogue, by computers. The ability of computers to analyse the different ways people specify what they wish to achieve in a dialogue is of particular interest. Also of interest is what makes those dialogues coherent. Hence, the work is about how a computer can distinguish different types of people's intention in dialogue, and how those distinctions might be used to determine what the person intends in the dialogue. The concern here is more with dialogue, or the language interactions between agents, whether they be people or computers, than with other forms of discourse, such as natural-language text. There is no concentration on speech processing: the focus is the analysis of written dialogues.

A theory of intention analysis is proposed to solve, in part, the problem of natural-language dialogue processing. A central principle of the theory is that coherence of a natural-language dialogue is based, in part, on the sequences of intention in the dialogue. It is proposed that the theory of intention analysis can be used to solve problems in natural-language dialogue between people and a computer, such as modelling the level of expertise of a computer user, in order to generate user-sensitive natural-language responses. Specifically, two hypotheses are tested. First, what we call the Intention-Computer hypothesis, that the analysis of intention in natural-language dialogue facilitates effective natural-language dialogue between different types of people and a computer, is tested. A computational model of the theory of intention analysis is developed and it is demonstrated that it can be used to model levels of user expertise. In turn, the model of expertise can be used to generate user-sensitive natural-language responses. The computational

[1]Gazdar (1979) provides a useful bibliography on pragmatics. Green (1989) gives an introduction to pragmatics and natural-language understanding. Levinson (1983) provides a discussion of pragmatics from a linguistic point of view. Wilks and Mc Kevitt (1990) contains the proceedings of a conference on pragmatics in artificial intelligence.

[2]See Barnden et al. (1991), Fass et al. (1991), Hovy (1988), and Wilks and Mc Kevitt (1990).

model, called Operating System CONsultant (OSCON), acts as a consultant on the UNIX[3] and MS-DOS[4] computer operating systems, through natural-language dialogues with users. Second, what we call the Intention-Person hypothesis, that the analysis of intention in natural-language dialogue indicates the degree of expertise of different people, is tested. Experiments are conducted to test the theory and both hypotheses. The results give positive evidence and have implications for a variety of research fields such as natural-language processing, human-computer interaction, artificial intelligence (AI), and philosophy.

As we can be considered to be doing artificial intelligence, we can be asked to justify the use of terms like *intention*, which are used at a representational level in the computational model, and are normally seen as being applied to people. If we are claiming that the program recognises intentions then we could be construed as saying that the program *has* intentions. Hence, we may be asked to take a stand on whether intentions can be ascribed to the program, or not. If intentions are ascribed to the program, that ascription can be underpinned by taking a stand on a theory of consciousness, and whether a form of strong, or weak AI as defined by Searle[5], and redefined by Narayanan[6], is accepted. Accepting strong AI means that we accept that humans, and certain computers, have the same type of mental predicates, like *intention*, ascribable to them, and all predicates ascribable to humans are ascribable to such computers. On the other hand, accepting weak AI means that we see that there is a certain type of mental predicate, which can only be ascribed to one type of entity, whether computer or human. However, we are currently agnostic about whether our theory and computational model reflect strong, or weak, AI.

With respect to the methodology and foundations of AI we take the stance described in the following sections. There has been much discussion in the literature as to exactly what AI is, and is not, and what it means for something to be a theory in AI. Much of the discussion has centred on whether AI is a science or not, and the rôle of programs in AI. The reason for such concerns is made clear by Narayanan (1986),

> The aim of this paper, apart from trying to steer well clear of terminological issues, such as the distinction between 'science' and 'study', is to demonstrate that unless AI is provided with a proper theoretical basis and an appropriate methodology, one can say just about anything one wants to about intelligence and not be contradicted; unless AI is provided with some reasonable goals and objectives little of current AI research can be said to be progressing.
> — Narayanan (1986, p. 44).

Hence, it is observed that it is important to provide a proper foundation for AI if it is to progress in a clear direction. We are concerned too, because the theory and Intention-Computer hypothesis centred on here concern natural-language processing which is considered a subproblem

[3]UNIX is a trademark of AT&T Bell Laboratories.
[4]MS-DOS is a trademark of Microsoft Corporation.
[5]See Searle (1980, 1987).
[6]See Narayanan (1990b, p. 238).

of AI. If we are to argue for a theory here, then questions about theories, programs, and scientific methodology concern us. First, methodology shall be discussed, and then foundational issues.

0.1 Methodology

Methodologies of AI concern the ways in which computational models of intelligent behaviour are designed and developed. There are a number of architectures available to the AI researcher for developing AI programs. Also, methodologies concern the processes by which researchers develop an idea and incorporate that idea within a computational model. First, we will discuss two major competing paradigms within AI.

0.1.1 Symbolic versus connectionist approaches

There are a number of approaches to constructing computer models within the field of AI. One of the grand distinctions of such models is that of *symbolic* versus *connectionist* architectures.

The approach taken in this work is the symbolic one, which has been the traditional approach in AI. The other is called the connectionist, distributed, or neural-network approach (see Barnden and Pollack 1991, Sharkey et al. 1986, Sharkey and Sharkey 1987, and Waltz and Pollack 1985). Connectionist models have been applied in the areas of pattern recognition and associative memory. However, for the most part, connectionist attempts to model high-level processes such as planning, common-sense reasoning, and the semantic and pragmatic aspects of natural-language understanding have come up against technical difficulties (see Barnden and Pollack 1991, p. 3). The connectionist approach has come up against difficulties in achieving the complexity of processing obtained in Cognitive Science and AI which use processes of manipulating symbolic representational structures. These problems have been discussed in Barnden (1984), Fodor and Pylyshyn (1988), and Smolensky (1988). Of course, the connectionist approach seems to provide qualities difficult to achieve in the symbolic framework (e.g. associative memory, slow adaptation, and distributed representations.). Connectionists claim that their models can effectively cater for fault-tolerance, robustness, flexibility, pattern completion, ability to match representations approximately, and much more (see Barnden and Pollack 1991, p. 3).

It is proposed here to take a symbolic approach to representing information in any algorithms or computer programs that are developed. There are no claims made that the symbolic approach is better than the connectionist one. The symbolic approach is chosen as it seems the most appropriate for modelling the high-level processes that we shall be working with.

Of course, on one hand, we could take the luxury of arguing here, that the traditional symbolic approach to developing systems is utilised, and that any algorithms developed can be translated into connectionist models. This approach has been termed *implementational connectionism* by Barnden and Pollack (see Barnden and Pollack, 1990, p. 5). On the other hand, there is a path from high-level cognitive processing to connectionism that bypasses traditional symbol processing. Smolensky (1988) takes this route, and says that traditional symbol processing is merely an approximate superficial description of underlying connectionist reality. Our algorithms could be such approximations, too. Next, another methodological issue, that of determining how a theory relates to the symbolic computational framework is discussed.

0.1.2 A grounding within Marr's framework

Marr (1990) describes two types of theory. The first type of theory, *type I*, is one where a technique is used to describe the problem under analysis before solving it. Marr refers to Chomsky's notion of a "competence[7]" theory for English syntax which follows this approach. The point is that one should describe a problem before devising algorithms to solve the problem. The second type of theory, *type II*, is one where a problem is described through the interaction of a large number of processes. Marr points out that the problem for AI is that it is hard to find a description in terms of a type I theory. He says, most AI programs have been type II theories.

Marr (1982) defines a three-level framework within which any machine carrying out an information processing task is to be understood:

Computational theory (Level 1): The goal of the computation and the logic of how it can be carried out.

Representation and algorithm (Level 2): The implementation of the computational theory and the representation for the input and output. Level 2 also involves the algorithm for the transformation of input into output.

Hardware implementation (Level 3): The physical realization (sic) of the representation and algorithm.

Our theory and Intention-Computer hypothesis can be characterised in Marr's terms. First, the goal of the computation is to model the sequences of intention from sequences of utterances in natural-language dialogue. That involves the recognition and representation of intention sequences. The logic of the computation is spelled out in Horn Clause formalism using Quintus Prolog[8]. Second, the computation is *represented* in Quintus Prolog. The representation for input/output is one of natural-language input/output. The algorithm for transformation is one of Prolog transformations which recognise intention in natural-language utterances, determine and represent the types, frequencies, and sequences of intention, build a user-model from the representation, and use the user-model to generate effective natural-language responses. The gap between levels 1 and 2 is less obvious with the use of Prolog because Prolog is a logic programming language and such languages were not prevalent in Marr's time. Third, the algorithm is realised physically as a program which runs on a Sun 4[9] computer.

[7]See Chomsky (1965).
[8]Quintus Prolog is a trademark of Quintus Corporation.
[9]Sun is a trademark of Sun Microsystems Incorporated.

It is all very well having a methodology of AI, but the following question must also be asked: What sort of study is AI? Is AI science, engineering, or technology?

0.2 Foundations

Foundational issues with respect to AI concentrate on what the subject of AI is. There seems to be a large resistance to calling AI a science.

0.2.1 AI and science

There have been many arguments as to whether AI is, or is not, a science (see Narayanan 1990b, and Partridge and Wilks 1990a, 1990b). Schank (1990) brings up the question as to whether AI is a technology of applications, or a science. He points out that researchers have taken two directions, the scientists interested in working on problems like the brain, or more neat logic problems, and the applications people working on building real practical systems.

Bundy (1990) calls AI an "engineering science." He says that it consists of the development of computational techniques, and the discovery of their properties. He argues that AI is exploratory programming, where one chooses a task that has not been modelled before, and writes a program to implement it. On the other hand, Dietrich (1990) argues that AI is a science and says,

> Then, I will suggest a new theoretical foundation, and argue that adopting it would provide a clear, unequivocal role for programs: they would be controlled experiments, and AI would become a science.
> — Dietrich (1990, p. 224).

He points out that such experiments can be operated over natural systems like ecosystems, and populations, such as ant colonies. He says,

> In the science of intelligent systems, therefore, computer programs would have a definite role: they would allow scientists to experiment with hypotheses about the nature of intelligence.
> — Dietrich (1990, p. 231).

Sparck Jones (1990) says that AI is engineering and points out that, "... AI experiments are engineering experiments, serving the designs of task systems, i.e. of artifacts," (p. 274). However, although we would agree with Sparck Jones in the sense that AI programs can be tested, and redesigned by such experiments, we would argue that AI hypotheses can also be tested with experiments.

Let's assume that AI is a science; then, a further problem arises: how is an AI theory, or hypothesis tested? Narayanan (1990b) brings up the latter problem nicely,

> The relationship between AI and cognitive psychology is strong. Does that mean that AI theories must conform to the same methodological rigour as psychological theories?

> If not, then a clear methodology must be provided for constructing and testing AI theories, otherwise AI might end up being a completely speculative subject, more akin to science fiction than science.
> — Narayanan (1990b, p. 164).

Hence, it is important to consider the problem of how to test an AI theory, if AI is a science.

0.2.2 The implementability problem

Currently, the form of the test of AI theories is one where the programs, embodying theories of intelligence[10], are implemented, and demonstrated to work, over a few selected examples.

Narayanan (1986, 1990b) points to the problem of implementability in AI. Narayanan (1986) says, "It can be argued that the criterion of implementability is vacuous at the level of the Church-Turing thesis," (p. 46-47). Hofstadter (1979) makes the following point:

> Mental processes of any sort can be simulated by a computer program whose underlying language is of a power equal to (a language) in which all partial recursive functions can be programmed.
> — Hofstadter (1979, p. 578).

Hofstadter is basically claiming that any mental process can be described by an algorithm and hence can be executed on a computer. Any AI researcher who constructs theories which can be described by an algorithm, can implement those theories. Thus, any AI theory which can be described by an algorithm can be implemented on a computer, and hence *all* AI theories are valid, as all AI theories can be described by algorithms. Sharkey and Brown (1986) also point out this problem: "To say that a theory is implementable is simply to say that it can be expressed in the form of a computer program which will run successfully," (p. 278), and suggest that a solution needs to be found, "Another question we would like to raise here is this: At what point in implementation do we decide that there are too many patches to accept that the running program is actually a test of a theory," (p. 280). Sutcliffe (1991) argues for more empiricism and says, "I see the use of norming studies and other techniques from psychology as being relevant to AI." Narayanan (1986), points out, "In any case, even if a criterion of complexity for AI programs (theories) can be found, there still remains the suspicion that no criterion exists for determining whether an AI theory is true or accurate," (p. 48).

Even taking into account the fact that many AI researchers do not argue that they have implemented AI theories, but that they just have predictions about the results of the computer program and the behaviour expected of it, this, at best, gives us either (1) an indication of how closely matched human and computer performance is, or (2) a tool for testing and sharpening our own theories. The latter is a form of weak AI (see Searle 1980, 1987). If a program produces results

[10]There are no strong claims made here with respect to the relationship between programs and theories, although we will claim to develop a theory and computational model reflecting aspects of that theory. However, this issue is discussed in Bundy and Ohlsson (1990), Simon (1990), and Wilks (1990).

unexpected of it, the question remains as to how much the programmer is allowed to modify it, and the theory it represents.

Much of the work here has appeared in, or emerged from, publications in various forms, mostly as technical reports, refereed conference proceedings, and book chapters. Descriptions of the computational model, OSCON, are given in Guthrie et al. (1989), Mc Kevitt (1991a), and Mc Kevitt (1991b). In Mc Kevitt (1986a), Mc Kevitt and Wilks (1987), and Mc Kevitt and Pan (1990a) a knowledge representation for the computational model is given. Inference rules are presented in Mc Kevitt (1988a, 1988d). Mc Kevitt (1986c) describes natural language aspects of the computational model. Details on the the use of Wizard-of-Oz experiments to enable data analysis for user modelling are given in Mc Kevitt (1990a), and for natural-language interfaces in general, are given in Mc Kevitt (1990d). In Mc Kevitt (1990b) the use of communicative acts, or intentions, for dialogue segmentation is presented. Details and data on the Wizard-of-Oz experiments, to test the hypothesis that the analysis of intentions in natural language can be used for user modelling, are given in Mc Kevitt and Ogden (1989a, 1989b), A description of an X-windows interface for the OSCON computational model is given in Mc Kevitt and Pan (1990b).

0.3 Acknowledgements

I would like to acknowledge people who have helped in many ways since I began the work reported here.

First, I would like to thank the supervisors who have contributed most to the development of my ideas in natural-language processing. My undergraduate supervisor, Arthur Cater introduced me to natural-language processing in 1984. Since 1986, Yorick Wilks has introduced, and guided me through, the enormous literature, and research topics, within natural-language processing, and is responsible, in a large part, for establishing the basic knowledge I have in the area. He has also commented on, and guided much of the work reported here. Since 1986, my current supervisor, Derek Partridge has given superb guidance and comments on the work here, and has provided many insights into the nature of artificial intelligence and software engineering.

Second, I would like to thank the examiners of this thesis. My internal examiner, Antony Galton has provided excellent comments on tightening up the wording of the thesis and its arguments. Also, my external examiner, David Benyon has provided excellent suggestions on how to improve this work. I am indebted to both examiners.

Ajit Narayanan who has not only given comments and suggestions on improving this work, but has shown me the importance of a strong grip on the philosophical aspects of artificial intelligence and natural-language processing is to be thanked. Noel and Amanda Sharkey have also given a number of useful suggestions on how to improve this research.

Useful interactions have been had with a number of other researchers in the Computing Research Laboratory (CRL) at New Mexico State University, Las Cruces, New Mexico (USA), the Computer Science Department at the University of Exeter, Exeter, United Kingdom (EC), and

the Advanced Computer Interface Group at U S West Advanced Technologies, Denver, Colorado (USA). Experience on natural-language processing has been gained from interchanges with Adrian Baldwin, Afzal Ballim, Jerry Ball, Subhanker Banerjee, John Barnden, Sylvia Candelaria De Ram, Paul Day, Ted Dunning, David Farwell, Dan Fass, Chris Fields, Maria Gonzales, John Gooday, Niall Griffith, Cheng-Ming Guo, Louise Guthrie, Steve Helmreich, Xiu-Ming Huang, Eric Iverson, Stuart Jackson, Wang Jin, Wanying Jin, Chen Li, Min Liu, Kuaresan Mylvaganam, Nicole Modiano, Bill Ogden, Tony Plate, Jordan Pollack, Lisa Rau, Martin Rajman, Jon Rowe, Richard Sutcliffe, Brian Slator, Yiming Yang, Sasumu Yasuda and Masoud Yazdani. Valuable experience on empirical work has been gained through interchanges with Hans Brunner, Mike Coombs, Paul Day, Jim McDonald, Simon Morgan, Bill Ogden, Ken Paap, Roger Schvaneveldt, Laura Thompson, and Scott Woolf. Valuable knowledge on the foundations and methodology of artificial intelligence have been gained from Eric Dietrich and Chris Fields.

The systems people at New Mexico and Exeter provided invaluable help while doing the work here. Specifically, Rick Brode and Ted Dunning at New Mexico, and Sue Charles, Khalid Sattar and Gordon Watson at Exeter are to be thanked. The secretaries at the CRL, Helen Flechsenhaar, Bea Guzman, and Jeannine Sandefur, and at Exeter, June Stevens and Marlene Teague are to be thanked for all their help.

Thanks are due to a number of colleagues for helping with some of the technical elements of the work. Louise Guthrie provided much help and guidance in programming parts of the OSCON system. Zhaoxin Pan programmed the Wizard-of-Oz system for collecting data on natural-language dialogue. Also, Bill Ogden provided valuable expertise on the development of the experimental environment for two of the experiments described here.

David Batty of the Science Faculty Office is to be thanked for being most helpful in communicating information about the Ph.D. process at Exeter, while I have been in both New Mexico and the UK.

There are a whole host of people who have been friends, and who have contributed indirectly to this work. Some great times were had with Karim Alhussiny, Linda Alhussiny, Steve Crisman, Chris Fields, Georgious Georgiou, John Gooday, Art Kellner, Iain McCorkindale, Philip Morgan, Simon Morgan, Juan Oliver-Rodriguez, Derek Partridge, Noel Sharkey, Richard Sutcliffe, Andreas Wehlau, Yorick Wilks, and Andreá Wismer. Finally, I would like to thank my family, Peter, Peádar, Rose, Tara, and Michael Mc Kevitt for all the support they have given me.

The concern here is not only to build interactive computer systems, but to enable such systems to better simulate tasks which we normally attribute only to people. The hope is that computers will be able to better understand people, and likewise, people will be better able to understand computers.

Finally, an apology is offered, for the work described here has not emerged in the clean chronological order in which it is described. The work arose from Yorick Wilks asking me if I could design and develop a small natural-language interface for a UNIX help system. In his words: "Could you, over a few stiff drinks some night, think up a quick design for a natural-language interface to

UNIX?" The answer is this.......

Exeter, U.K., E.C.

September, 1991 Paul Mc Kevitt

The critic as scientist

“Don’t let us discuss anything solemnly. I am but too conscious of the fact that we are born in an age when only the dull are treated seriously, and I live in terror of not being misunderstood. Don’t degrade me into the position of giving you useful information. Education is a useful thing, but it is well to remember from time to time that nothing that is worth knowing can be taught.”

– Oscar Wilde (1854-1900)[11]

[11] See Wilde (1990), “The critic as artist, Part I”, In “Oscar Wilde: plays, prose writings and poems,” London: J. M. Dent and Sons.

Time out of mind

The following extract appeared in the *Independent on Sunday* newspaper on Sunday, June 2nd, 1991, from an interview with Stephen Hawking:

ON THE door of Stephen Hawking's office in the Department of Applied Maths and Theoretical Physics in Cambridge is a jokey plate reading "The boss is asleep". The office opens off a stuffy common room painted some institutional shade unknown to any spectrum. There is an air of scruffiness, torpor, lack of funds; four or five people chat listlessly beneath photographs of visiting researchers and foreign students.

The door opens. The boss is not asleep. He is sitting behind a desk slouched in his wheelchair. When you close the door you see that he must have been staring at the large poster of Marilyn Monroe pinned to the back of it. Behind him, stuck to the wall, is a postcard of one of his predecessors as Lucasian Professor of Mathematics, Issac Newton. Between Monroe and Newton is Stephen Hawking, whose reputation as a scientist was established during the Seventies but whose wider fame rests on *A Brief History of Time*, published by Bantam Press in 1988, which has been on the best-seller list for 144 weeks and has sold in millions.

Not many people seem to have finished it, however. How does he account for the fact that they go on buying it even though it is well known that it is hard for the non-scientist to read; why do people want to own it?

"I know critics are always saying that very few of the people that buy my book actually read it, and that even fewer of those that read it understand it. But I don't think that's true. People keep stopping me in the street and telling me how much they've enjoyed it. I think that indicates that they have read at least a part of it. I'm sure that most people don't fully understand all the ideas in it. If they did, they would be doing research in theoretical physics. But it gives them a feeling of being in touch with the really big questions. I think that critics feel that they are very clever men, and if they can't fully understand my book, what hope have ordinary mortals?"

The answer raises supplementary questions: the streets of Cambridge may not be typical streets; the people who couldn't understand are hardly likely to stop him, and so on. But you don't want to waste Professor's Hawking's time. His very first word was in fact "time". "Time is limited. I have to leave at 6:30."

The words appear on the lower half of a large IBM screen on the desk. The upper part of the screen is taken up by a menu which basically offers an alphabet. When the cursor is on the right letter, a selected list of the words beginning with that letter that he is most likely to need replaces the menu. When the cursor then finds the word he wants, that word appears in the lower half of the screen. Thus the answer given above, which is 130 words, required approximately 500 movements and some 12 or 15 minutes' labour. Professor Hawking moves the cursor by pressure to a switch held in his left hand where the only muscle movement is his body – confined to two fingers – causes a series a (sic) small clicks to break the silence of the airless room.

Professor Hawking contracted motor neuron disease in his early twenties and was given two years to live. He is now 49........

He drives his electric wheelchair in a hectic way that alarms his friends and in the spring was knocked over by a car, which caused injuries to his head and arms. How is he now? "I'm now fully recovered from the accident I had in March. I think I'm as well as I have been in the last six years, since I had the tracheostomy operation that removed my ability to speak. My condition is fairly stable. I get weaker, but only very slowly.".......

There is an eerie kind of exhilaration in watching the words appear on the lower half of the screen. "The universe... was..." Long wait. Go to menu. The cursor flickers over rows of words. "Tikka, Tokyo, Tuesday, trinity." Strange little poems and infinite strings of options reveal themselves. A code is on the point of being cracked, but the key is held within a wasting body. "Sheepskin, sherry, shift," goes the screen as the cursor ripples through. "...expanding." At the end of the interview, the computer prints out all his answers........

– Sebastian Faulks, *Time out of mind*, Independent on Sunday, June 2nd, 1991.

It is already apparent to many of us that the processing of natural languages by computer, or natural-language processing, will have many uses and implications in the years to come.

Analysing coherence of intention in natural language dialogue

by

Paul Mc Kevitt, B.Sc. (Hons.) (Dublin), M.S. (New Mexico)

Submitted by Paul Mc Kevitt
to the University of Exeter
as a thesis for the degree of
Doctor of Philosophy
in the Faculty of Science
September, 1991.

Abstract

Much of the work on the computer processing of natural languages, or *natural-language processing*, has concentrated on studying the structure, meaning, and usage of individual utterances. One of the problems in natural-language processing is to build theories and models of how individual utterances cling together into a coherent discourse. The problem is important because, to properly understand natural language, a computer should have some sense of what it means for a discourse to be coherent and rational. Current theories and models of natural-language processing argue for a measure of coherence based on three themes: meaning, structure, and intention. Most approaches stress one theme over all the others. Here, a theory of intention analysis is developed for solving, in part, the problem of natural-language dialogue processing. A central principle of the theory is that coherence of natural-language dialogue can be modelled by analysing sequences of intention. In addition, it is shown that the theory of intention analysis can be incorporated within a computational model, and that the computational model can be used for applications such as user-modelling. In turn, user modelling can be utilised for the generation of user-sensitive natural-language responses. The computational model, called Operating System CONsultant (OSCON), implemented in Quintus Prolog, understands, and answers in English, English questions about computer operating systems. In particular, what we call the Intention-Computer hypothesis, that the analysis of intention in natural-language dialogue facilitates effective natural-language dialogue between different types of people and a computer, and what we call the Intention-Person hypothesis, that the analysis of intention indicates the degree of expertise of different types of people, are tested. Experiments provide positive evidence for the theory and hypotheses. The results have implications, both theoretical and practical, for a variety of research fields including natural-language processing, human-computer interaction, artificial intelligence and philosophy.

Supervisor: Prof. Derek Partridge, University of Exeter.
Internal Examiner: Dr. Antony Galton, University of Exeter.
External Examiner: Dr. David Benyon, The Open University.

Part I

Prologue

Chapter 1

Introduction

Much of the work on the computer processing of natural languages, or *natural-language processing*, has concentrated on modelling the structure, meaning, and usage of individual utterances[1]. However, it is not often that natural-language utterances occur on their own, independent of some context or other. Hence, one of the problems in natural-language processing is to build theories and models of how individual utterances cling together into a coherent discourse. It can be argued that to understand a discourse properly, a computer should have some sense of what it means for a discourse to be coherent. Current theories and models of natural-language processing argue that the coherence of a discourse can be measured in terms of three main themes: meaning, structure, and intention. Most of the approaches stress one theme over all the others.

The aim here is to provide a theory of intention analysis for solving, in part, the problem of natural-language discourse processing, and to incorporate that theory within a computational model which can be tested. It is also proposed that the theory can be used for applications such as user[2]-modelling. In particular, what we call the *Intention-Computer* hypothesis, that the analysis of intention in natural-language dialogue facilitates effective natural-language dialogue between different types of people and a computer, and what we call the *Intention-Person* hypothesis, that the analysis of intention in natural-language dialogue indicates the degree of expertise of different types of people, are tested. Hence, concentration shall lie on natural-language dialogue, or the form of natural language that exists between communicating agents, whether they be people, or machines. There will be concentration on written dialogues rather than spoken ones, and there will be little interest in other forms of natural-language discourse such as text. A theory of intention analysis is proposed and incorporated within a computational model, called Operating System CONsultant (OSCON). The central principle of the theory is that coherence of natural-language dialogue can be modelled by analysing sequences of intention. The *analysis of intention* has at least two properties: (1) that it is possible to recognise intention, and (2) that it is possible

[1]The term *utterance* will be used to refer to any unit of natural language, whether it be a word, phrase, sentence, or exclamation. An utterance may be well-formed or ill-formed. Reference will usually be made to written, rather than spoken utterances, unless indicated otherwise.

[2]The term *user* is used to refer to any person interacting with a natural-language processor. A *user-model* is a model of the level of any user's expertise.

to represent intention. It is shown that syntax, semantics and pragmatics of natural-language utterances can be used for intention recognition. It is shown that intention sequences in natural-language dialogue can be described by what are called *intention graphs.* Intention graphs can be used to represent phenomena existing in natural-language dialogue. It is argued that a number of orderings of intentions can be defined. More specifically, it is shown that an ordering of intention *satisfaction* exists, and when used in conjunction with intention sequences, indicates the *local* and *global* degrees of expertise of a speaker in a dialogue. Three experiments provide data to support an empirical evaluation of the theory and two hypotheses. The results have a number of implications, both theoretical and practical, for a variety of research fields such as natural-language processing, human-computer interaction, artificial intelligence and philosophy.

Let us begin with a discussion of the most important factor involved in natural-language discourse: the factor of *context*[3], and how it affects the interpretation of natural-language utterances.

1.1 The influence of context

One of the most interesting aspects of our ability to use natural languages like English is the capability of understanding natural-language utterances in context. So, when we are at the bank, and are discussing drafts, we know from the context that it is money at issue, and not calling up men, and women, for war. Likewise, if you have just arrived in the USA from the E.C., then you may find the following sign, shown in Figure 1.1, difficult to interpret[4]. The problem arises from interpreting "X" as the third last letter of the alphabet, rather than as "Cross." However, US citizens have no problem interpreting the sign as they are used to it from context.

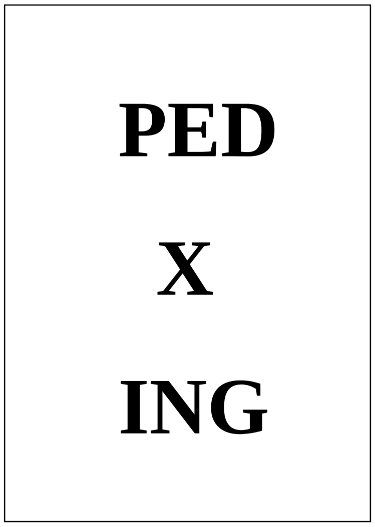


Figure 1.1: A pedestrian sign in the USA

Likewise, if an American walks into a bakery in the UK, and does not know about "Hot Cross

[3]By *context* we mean spatial, or visual, and linguistic context.
[4]This actually happened to the author, and a number of other foreign students, while studying in the USA.

Buns," then the following notice in Figure 1.2 might force him/her to ask the counter clerk for "Hot X Buns" and she/he would be terribly confused. The counter clerk, or someone who eats "Hot Cross Buns" quite frequently, has no problem understanding the sign as they are used to it from context. Our ability to understand language in context enables us to be standing in line at

HOT

X

BUNS

Figure 1.2: A bakery notice in the UK

a restaurant and utter "same please" to the server, just after someone has asked for lasagne and salad. The phenomenon of using, and understanding, utterances in context is central to the work here.

Although some researchers have tried to give definitions of utterances independent of their context, it is difficult to conceive of any utterance as being so. Searle (1978) characterises the notion of context-independence, commonly called, *literal meaning*, as follows:

> Sentences have literal meanings. The literal meaning of a sentence is entirely determined by the meanings of its component words (or morphemes) and the syntactical rules according to which these elements are combined... For sentences in the indicative, the meaning of a sentence determines a set of truth conditions; that is, it determines a set of conditions such that the literal utterance of the sentence to make a statement will be the making of a true statement if and only if those conditions are satisfied... The literal meaning of the sentence is the meaning it has independently of any context whatever; and, diachronic changes [changes in the language over time] apart, it keeps that meaning in any context in which it is uttered.
> — Searle (1978, p. 117).

However, Winograd and Flores (1986) take another view,

> It is an edifying exercise to look at the statements made both in writing and in everyday conversation, to see how few of them can even apparently be judged true or false without an appeal to an unstated background.

— Winograd and Flores (1986, p. 55).

It is argued here that context is extremely important in determining the meaning of words or utterances. Much research in the processing of language by computer has concentrated on processing the meaning of individual sentences, without taking other utterances, or utterance context, into account. The disadvantage of taking the latter approach is that utterances can be understood incorrectly, or not understood at all. To see how the computer would be limited, while not taking context into account, just think how difficult it would be for you to understand any sentence in this text, if you did not have any of its preceding sentences, or any of the sentences following it. Hence, a central theme of the work reported here is to argue that theories and models of natural-language processing can not be completely effective while processing entities, whether they be words or utterances, without analysing their context.

Those researchers who do take context into account, while proposing theories for the computer processing of natural language, do so in interesting and intricate ways, and we shall contribute to the development of those theories. As the concern here is with natural-language dialogue, it would seem appropriate to investigate the sorts of problems which occur during the processing of natural-language dialogue.

1.2 Some problems in natural-language dialogue processing

Before we go on to look at some of the problems in dialogue in more detail, we shall discuss the phenomenon of indexicality[5] which prevails during any discussion of such problems. Bar-Hillel (1954) introduces indexicality in language where he speculates that 90% of declarative sentences include implicit references of the speaker, addressee, time and/or place of utterance in expressions such as first and second pronouns (*I*, *you*), demonstratives (e.g. *this*), tenses, and adverbs like *here, now, yesterday.* He argued that indexicality is an inherent and unavoidable property of natural language. The argument is basically that utterances like those in (1) below cannot be understood in terms of their truth unless we take their context into account. Thus, with respect to the Gulf crisis in 1990, (1a) would make sense if it was uttered by say, the leader of Japan, but would be less sensible if uttered by say, the president of the USA. Hence, the speaker of an utterance, or *who* says the utterance, is of importance. It is possible, in general, to map utterances like (1a) into some representation, but it is not possible to say whether that representation is true, or false. A full account of the context of the utterance would provide such information. (1b) is interesting by the very fact that *when* it occurred is important. If it was uttered before late 1990 then it would seem reasonable, yet soon after that period it would seem unbelievable. Finally, the last utterance is of interest because of where it is uttered. For example, if uttered aloud in front of Mrs. Thatcher, while she is leader of the Conservative party, it may sound quite strange, unless it was meant in humour.

[5]Indexicals are utterances whose meanings are functions of their context.

(1) a We will not send soldiers into the Gulf.
b The USA sold arms to Iraq.
c I am secretly interested in contesting the Leadership of the Conservative Party.

Hence, what we see here, is that the truth of utterances depends on who, when, and where they are being uttered. In particular, Bar-Hillel (1954) noted the problem with the word *this*. He says,

> 'This' is used to call attention to something in the centre of the field of vision of its producer, but, of course, also to something in his spatial neighbourhood, even if not in his center of vision, or not in his field of vision at all, or to some thing or some event or some situation, etc., mentioned by himself or by somebody else in utterances preceding his utterance, and in many more ways.
> — Bar-Hillel (1954, p. 373).

Morgan (1978) gave an example that illustrates the problem of identifying the referent of an indexical:

> imagine a jar of sugar with a glass lid, on which the word *sugar* is painted in blue; and imagine that someone puts her fingertip just under the letter *u* of the word *sugar* and says, "What's that?" Our answer might be, among other things, *the letter* **u**, *the word* **sugar**, *paint, blue, English, a lid, glass, a glass lid, a jar, sugar, a jar of sugar, and so on, depending on our interpretation of the person's interests* — is she learning English, the use of seasoning, physics, or what?
> — Morgan (1978, p. 264).

It is noteworthy that although Morgan hints at the person's interests, or *intention*, in determining the intended referent of utterances, Bar-Hillel does not do this at all. We will see later that the issue of whether intention is important in dialogue, or whether dialogue can be tackled by other means, crops up a number of times in answering questions about natural-language dialogue.

There are a number of problems which occur in natural-language dialogue due to the ability of people to use language in context. To give the reader a feel for such problems, we shall discuss some of them.

1.2.1 Tense

Tense enables sentence tokens to be used, where complete meaning cannot be derived without taking context into account. Time is usually important in these cases. Take for example the utterance in (2a) below.

(2) a The European Community (EC) is going to start a war in the Gulf.
b The European Community (EC) is starting a war in the Gulf.

The complete meaning of (2a) is not determinable, due to an underspecification of time, whereas in (2b) the time is more uniquely determinable. The reason for this is that in (2a) it is not clear if the war will be started in days, weeks, or months, whereas in (2b) the time of the start of the war is more precise. A study on the temporal relationships of utterances through verbs, from a logical point of view, is given in Galton (1984). Also, the beliefs[6] and intentions of the speaker come into play, because if utterance (2a) was made by the president of the EC, and it was just at a point where it looked likely that the EC would attack Iraq, then the time reference would be on the order of days rather than say, years.

1.2.2 Anaphoric reference

In natural language discourse it is possible to use words such as *me*, *here*, and *then* to refer back to previous entities mentioned in, or related to, the discourse. The spectrum of entities referred to can be numerous. This phenomenon is called *anaphoric reference.* Also, people can use pronouns like *he*, *her*, and *their* to refer to other entities. Such pronouns are called *anaphoric references.*

Some researchers believe that reference is indexical in that it is determined by the linguistic "context of an utterance," but we believe that reference is indexical only in that it is determined by speaker intention. This is shown in (3) below. We argue that the only way the referent of *he* can be determined is by monitoring the intentions of the participants in the dialogue, as *he* could refer to either Mr. Howe, or Heseltine.

(3) Mr. Howe told Heseltine he should resign because of her attitude to Europe.

The use of *pronouns*, where words are used to refer to people, is a common type of reference. The determination of the referent of pronouns is important, i.e. the person being referred to. For example in (4) the utterance depends on who "I" is. If the person saying this utterance is the president of the USA then that is fine, otherwise it might be a case for committing someone to an institution. In (5a)[7] the object "file" is introduced and is referred to in (5b) with "it". It is only apparent that "it" refers to "file" from (5a). In dialogue (6a) the suer asks about seeing files and then about deleting them. The pronoun reference, "them," is used, and probably refers to "files". In (7b) "it" refers to "file," and "there" refers to the "printer".

(4) I am president of the USA.

[6]The term *belief* is used to refer to the points of view, or attitudes, an agent has towards some objects, or other agents. There has been much work on the processing of beliefs in natural-language processing. See, for example, the work of Ballim and Wilks (1990, 1991), Barnden et al. (1991), and Wilks and Ballim (1987).

[7]This dialogue, and a number of others immediately following it, constitutes a dialogue between a user and a system. The user is obviously asking questions about computer operating systems.

(5) a U[8]: How do I see a file?
S[9]: 'more <filename>' will display filename on the screen.
For example, 'more paul.1' will display paul.1 on the screen.
b U: How do I remove it?
S: 'rm <filename>' will remove filename from your current directory.
For example, 'rm paul.1' will remove paul.1 from
your current directory.

(6) a U: How do I see my file?
S: 'ls <directoryname>' will display directoryname on the screen.
For example, 'ls paul.1' will display paul.1 on the screen.

b U: delete them?
S: 'rm <filename>' will remove filename from your current directory.
For example, 'rm paul.1' will remove paul.1 from your current directory.

(7) a U: How do I print my file on the printer?
S: 'lpr <filename>' will display filename on the printer.
For example, 'lpr paul.1' will display paul.1 on the printer.
b U: remove it from there?
S: 'lprm <jobname>' will remove jobname from the printer queue.
For example, 'lprm 100' will remove job 100 from the printer queue.

The referent of the pronoun "I" is usually considered to be the utterer of an utterance. The use of "he" is more difficult to resolve as the pronoun could be referring to anyone in the discourse under discussion. The speaker's intention can be useful in determining the correct referent. However, note that pronouns do not always have referents. For example, the referent of "his," is unclear in, "No man is older than his father." A referent is partially determined by the temporal ordering of utterances, but is not uniquely defined by this. Much pronoun reference is resolved by determining the beliefs and intentions of the speaker and hearer that enable the hearer to determine the correct referent.

The processing of anaphoric reference is not simply a problem of determining referents from their temporal ordering in a discourse. A complex feature of anaphoric reference is that of *long-distance* reference (see Dale 1988, 1989). In dialogue people sometimes use long-distance reference rather than referring to entities in the query just before. Here is an example of a dialogue which demonstrates long-distance anaphoric reference:

[8]U stands for the *user* of a computer system.
[9]S stands for the *system*.

(8) a U: How do I print the file paul.1 on the laser printer?

S: 'lpr paul.1' will print the file paul.1 on the printer.

.....

b U: Has it been printed yet?

S: The file paul.1 is in the printer queue.

In (8a) the user is asking how to print a file and the system answers how to do this. However, after some intervening dialogue the user asks in (8b) if the file is printed. This is an example of long-distance anaphoric reference where "it" refers back to the file 'paul.1' in (8a). The system should remember the file referent 'paul.1,' and know that it is probably in the printer queue.

Most speakers will use reference in such a way that there is likelihood that the intended referent can be correctly inferred. It is possible to think of anaphors, such as pronouns, as being dependent on linguistic context, but it is what the speaker intends to refer to, that determines better what the form refers to.

In discussing anaphoric reference, there is a distinction between the *referent* of a form and its *antecedent*. The referent of a form is what the form refers to, which is some entity in the world. The antecedent of a form is another linguistic expression which has the same referent as the form. All referring expressions have referents, but they do not always have antecedents in discourse. A pronoun does not refer to a noun phrase, or other linguistic expression, but to whatever object in the world its antecedent noun refers to. Pronouns are called *coreferential* if they, and their antecedents, refer to the same entity.

The determination of pronoun-antecedent relations using syntactic conditions has been one approach to solving the problem of anaphoric reference. Researchers such as Kuno (1972, 1975) sought discourse conditions to account for the possible relations of pronouns and antecedents within and across utterances. He concentrated on such variables as whether the noun phrase represented predictable information, or an exhaustive answer to a question. Kantor (1977) argued that the case of comprehending that a pronoun was intended to refer to the same entity as the closest preceding candidate noun phrase, depended on the probability that the interpreter expected a reference to some entity other than the referent of the pronoun. Kantor showed that this expectation could be manipulated by both syntactic structure and discourse structure, but there has been no comprehensive modelling of the interpreter's expectation.

Personal pronouns are the most stereotypical type of anaphoric expression but many other expressions pose the same sorts of problem. Pronouns can refer directly to a previous item referred to, or reference may be less direct, as in the situations where an anaphor is used to refer to an idea expressed in an entire clause, or an entire line of thought.

1.2.3 Natural-language generation

The context of an utterance will affect how the utterance should be generated[10] in a dialogue. For example, looking at the sample dialogue in (9), between a computer user and a system, it is noted that the system changes its behaviour if it remembers it has answered a related question.

(9) a U: How can I delete a file?
    S: 'rm <filename>' will remove filename from your current directory.
    For example, 'rm paul.1' will remove foo from your current directory.
  b U: How can I delete a directory?
    S: Use 'rmdir <directoryname>'.

For query (9a) the system gives an example of how to use the command in question. However, no example is given for query (9b). The user should infer from the context how to use the command. In fact, it may be quite boring for the user to have examples of the use of commands displayed all the time. The point here is that the system should be aware of what it has said to the user to reduce the redundancy of information presented. Certainly, if a user asks the same query one hundred times the system should notice this, and should not just blindly keep answering the same query.

1.2.4 User-modelling

Any useful natural-language interface must build a model of the user for the effective cooperative interaction. This is particularly important for the generation of useful natural-language responses. There has been much work in Artificial Intelligence (see Ballim and Wilks 1990, 1991, Carberry 1989, Chin 1988, Kobsa and Wahlster 1988, Mc Kevitt 1990a, and Wilensky et al. 1988), and human-computer interaction (see Benyon 1984, 1987, and Benyon and Murray 1988) on the modelling of users' levels of expertise. Below, there are two example dialogues exemplifying different degrees of user expertise.

(10) a U: How can I delete a file?
    S: Use 'rm'.
    For example, to remove the file named foo, type 'rm foo'.
  b U: What does ruptime do?
    S: ruptime is used to list the uptime of all machines
      on the network, list the number of all users on them,
      and list their load and average.

[10]By *generation* we refer to *natural-language generation*, or the generation of natural language from some internal representation. Such representations can take many forms.

(11) a U: How can I find out the inode of a file?
    S: Use ls -i.

  b U: What does uptime do?
   S: uptime is like ruptime, except ruptime is for all
     machines on the network.

These example dialogues taken from (Chin 1988) show how the Unix Consultant (UC) system behaves depending on the level of expertise of the user. When the user asks (10a) it would seem that he/she doesn't know very much about operating systems. The user is asking a query about a simple topic: how to delete a file. Therefore, the system answers with the right command and an example of its use. When the user asks about ruptime, the system gives a detailed answer as the user is less knowledgeable.

In the second dialogue, (11), the user is asking about finding the inode of a file, which is a more complex query. Chin (1988) argues that such a query indicates that the user might have some expert knowledge of operating systems. Therefore, the answer is given in a short format, which presumes that the user knows how to specify the command syntax. Also, the system assumes that the user is an expert. Then, in (11b) when the user asks about "uptime" the system assumes that "ruptime" is already known to the user, as he/she is considered somewhat of an expert. Hence, it is important to incorporate a user-modelling component into a system for dialogue processing so that the system has a good feel for the type of user being dealt with. Also, the type of user will affect the way the system generates answers to queries. The relation between a dialogue model and a user-model has been the subject of much debate (see Schuster et al. 1988, and Kobsa and Wahlster 1988). One of the main themes of the work here will be to emphasise the importance of user-modelling in natural-language dialogue processing. However, it will be argued that approaches such as Chin's concentrate too much on the domain specific knowledge of the user, while our approach will be to elevate the more universal aspects of user-models which occur across domains.

In summary, four of the most pertinent subproblems in relation to the problem of natural-language dialogue processing have been discussed: (1) the processing of verbs indicating temporality, (2) the determination of anaphora, (3) response generation, and (4) user-modelling. In the chapters to follow we claim that our theory of intention sequencing can be used to solve, in part, the above four problems. In particular we will show that the theory can solve, in part, the latter two problems. Let us now move on to discuss the background to existing research in natural-language processing.

1.3 Natural-language processing

Natural-language processing is an area of research concerned with the processing of natural languages, such as English, by computer. The field involves research on theories, designs, and implementations of techniques, for processing natural language.

There are two major application areas of research in natural-language processing: (1) machine translation, and (2) natural-language interfaces. Machine translation concerns itself with the development of systems for translating discourse in one language into one, or a set of, other languages (see Nirenberg 1987, Schank 1973, Schank 1975, Schank and Abelson 1977, and Wilks 1975a, 1975b, 1975c). Natural language interfaces involve the use of language as an interaction medium between people and computers. For example, one can envisage programs which answer English questions to databases (see Bates et al. 1986, Hendrix et al. 1978, Hendrix 1980, Martin et al. 1983, Slator et al. 1986, Winograd 1972, and Waltz 1975, 1978), act as a consultant for some task (see Billmers and Garifio 1985, Douglass and Hegner 1982, Finin 1983, Guthrie et al. 1989, Hecking et al. 1988, Hegner and Douglass 1984, Mc Kevitt 1986a, Mc Kevitt and Wilks 1987, and Wilensky et al. 1984, 1986, 1988), or even just carry on a conversation about some topic (see Weizenbaum 1965). Traditionally, most of the research in natural-language processing has concentrated on written text, rather than spoken. There are however, a number of researchers working on the problem of building computer programs which understand and generate speech (see Baker and Baker 1989, and Jack and Laver 1988). Also, these programs are being integrated with natural-language interfaces which understand text, to augment understanding of spoken information (see Young 1989).

Much of the work in natural-language processing has concentrated upon the processing of individual sentences, or utterances, rather than the processing of utterances with respect to their context. Although it is always difficult to rigidly segment a field of research into different areas, there have come about three subareas of interest within natural-language processing. First, there is an area concerned with processing the structure of utterances. or *how* utterances are formed, called *syntactic* processing. Second, is an area concerned with processing the meaning of utterances, or *what* an utterance is, called *semantic* processing. Third, is an area concerned with processing utterance usage, or why utterances are used, called *pragmatic* processing. Each will be discussed in turn.

1.3.1 Syntax

Syntactic analysis of natural-language input is mainly concerned with the structure of natural language, and any regularities that occur therein (see Gazdar and Mellish 1989, Pereira and Warren 1980, and Woods 1970). In determining the structure of an utterance, syntactic analysis identifies subjects and objects for verbs. This is usually done by assigning a labelled tree structure to the input in a processing strategy referred to as *parsing*.

Some researchers have suggested that natural-language processing can be done without syntactic analysis (see Burton 1976). This has come about because the limited semantic of some on-line natural-language systems restricts the user to relatively simple sentences, for which sophisticated syntactic processing is unnecessary.

1.3.2 Semantics

Semantic analysis of natural-language sentences centres on techniques for recognising and representing the meaning of sentences. There are many methods for conducting semantic processing, but most techniques represent semantics using some form of meaning representation. Formal languages for the representation of meaning of natural-language utterances include propositional logic, predicate calculus, semantic networks (see Quillian 1969), Conceptual Dependency (see Schank 1972, 1973, 1975, Schank and Abelson 1975), and Preference Semantics (see Wilks 1973, 1975a, 1975b, 1975c, and Fass 1988). All usages of the words *semantics* or *semantic* in our work will refer to the processing of the meaning of sentences or utterances.

1.3.3 Pragmatics

Even after studying the structure and meaning of sentences there is still more work to be done. For example, in studying, or processing, the utterance, "It's cold in here," its structure could be represented as shown in Figure 1.3 below, and its meaning represented as shown in Figure 1.4. However, neither of these structures tell us much about the case where the speaker may have uttered the sentence in order that the hearer do something about the fact that it's cold, rather than just to describe the situation. Therefore, people's intentions must be taken into account in any serious attempt at natural-language processing. A possible representation of the intention of the utterance would be as shown in Figure 1.5. The representation demonstrates the fact that there is an inference that the speaker wishes to inform the hearer that he/she is cold, and would like the hearer to do something about it.

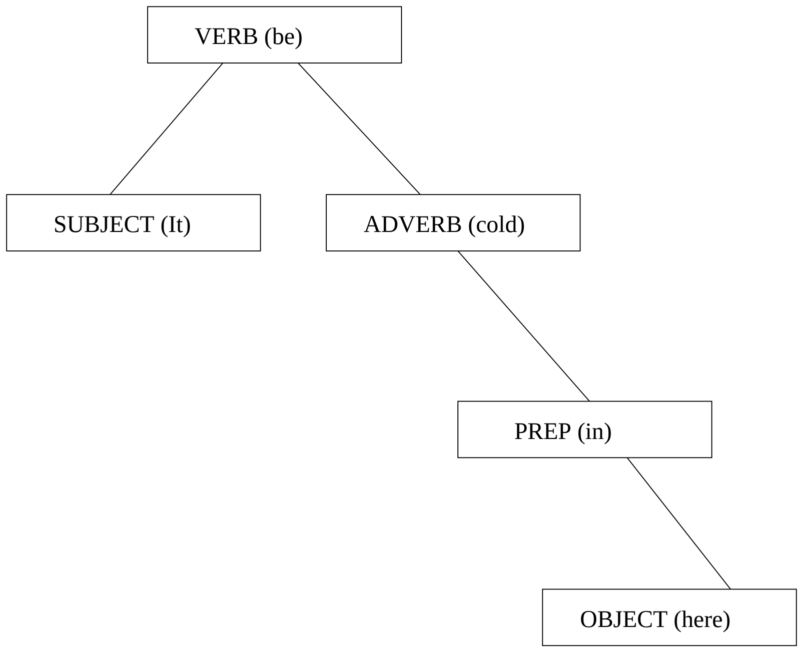


Figure 1.3: Example sentence structure representation

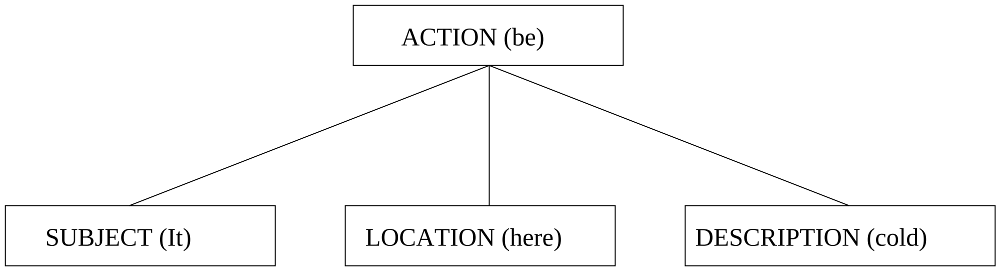


Figure 1.4: Example sentence meaning representation

```
INTENTION (inform (self cold))
      INFER (inform hearer)
               (reduce (self cold))
```

Figure 1.5: Example sentence usage representation

The analysis of utterances in their context, or the study of the use of language, is termed *pragmatics* (see Green 1990, and Levinson 1987), a major new area of research within natural-language processing. The study of pragmatics is of central concern to the work described here. However, it is important to bear in mind that although much of the work in natural-language processing has concentrated on the processing of syntax and semantics, these problems are not solved, by any means. Now, let us investigate what has been achieved in the application area of natural-language processing which we are more interested in here: natural-language interfaces.

1.3.4 Natural-language interfaces

Over the last twenty years most of the success in the application of natural-language processing has been in the area of natural-language interfaces. One of the first natural-language interfaces, called SHRDLU, was that built by Winograd (1972). Success with natural-language interfaces has occurred because the input to natural-language interfaces is usually simpler than the texts to be processed by machine translation, or information retrieval systems. Also, a natural-language interface will remain usable even if it rejects, or does not process some input. Hence, most of the work in natural-language processing since the 1970s has been within the area of natural-language interfaces.

So far, no one has been able to construct a computer program which will interact with some user in a realistic dialogue, over a reasonable period of time. Researchers have built many natural-language programs which will answer English questions, or help users to do some task, but few of these systems will work for every, or even a good percentage of, input (see Damerau 1981, Jarke et al. 1985, Krause 1980, and Tennant 1979).

Of course, there are many methods of building interfaces to computer programs rather than to have the luxury of natural language. Examples of alternatives are menu-based and touchscreen interfaces (see Shneiderman 1987). Some researchers would argue that people prefer menu-based systems (see Hauptmann and Green 1983) or formal query languages (see Jarke et al. 1985, Krause 1980, and Small and Weldon 1983) over natural-language interface systems. However, it has been shown recently that users do prefer natural language interfaces in certain situations (see Napier et al. 1989, and Walker and Whittaker 1989), and that previous experiments, which show the contrary, have been unfairly conducted (see Whittaker and Walker 1989). Our view is that the debate as to whether people prefer natural language, or menus, is a *straw man*, and there is no issue here because the usefulness of an interface depends on the application. For example, a secretary wishing to know, off hand, how to print a file, because her director has asked her to, would rather use a system where a sentence can be typed in English, and an answer given, than to have to 'wade' through a massive hierarchical menu-based system. Also, as you might imagine, any hierarchical menu-based system would get very large for domains such as operating systems, and there may be some slowdown if too much wading through the menu hierarchy is needed. Sometimes people find it difficult to interpret technically what they are looking for, and therefore may not know where to "click[11]" on the menu-based interface. It may be a lot easier to describe the goal through English, rather than having to wade. Therefore, there is good motivation for building natural-language interfaces. On the other hand, if the domain is quite small, the data is more limited, and a menu-based system may be very useful, and faster, than having to type English queries. Also, after learning a menu-based system, it may be faster to use than typing queries.

It is possible to build natural-language interfaces without any dialogue modelling component. However, this methodology does not produce good interfaces because users usually assume that the computer can behave like a human, remembering the preceding dialogue. Whittaker and Stenton (1989) have found that approximately 25% of the errors while users used a natural-language interface were caused by the program failing to have a good model of the ongoing dialogue. Also, it has been pointed out by Cohen et al. (1982), Grosz (1978b), Hendrix (1980) and Mann (1975) that users believe that they are in a conversation with a dialogue system, rather than just asking and answering queries out of context. Cohen et al. (1982) argue that users expect, (1) to engage in a conversation with a system, and (2) that the system incorporate its own responses into its analysis of user utterances. They say that users maintain these expectations even when it is obvious that the system is not a competent conversationalist.

The computer can be used to process dialogue in different ways. First, the computer can be used to monitor and understand dialogue between two, or more, agents, whether people or computers, and even answer questions about the conversation (see Ballim and Wilks 1990, Wilks and Bien 1983). Second, it may be possible for two computers to interact with each other in a dialogue[12]. Third, the computer itself can interact with a user and answer his/her questions (see Mc Kevitt

[11]It is common to use a device named a *a mouse* to achieve operations on computer systems. The mouse exists in the form of a small box that rests on a pad, which enables computer users to conduct operations on computer screens by clicking buttons on the box.

[12]Of course computers already have algorithms for communicating with each other. However, these algorithms are mainly concerned with communication on the level of passing data in the form of ASCII strings, which can be translated into analogue signals, rather than with natural language.

and Wilks 1987, Weizenbaum 1965, and Wilensky et al. 1988). It is the latter case that will be of interest here. This aspect of dialogue processing is important as it involves the user-computer interface which is one of the largest bottlenecks in information processing today.

Many of the computer systems which have been built to interact with computer users over the years, have been very limited with respect to processing pragmatics, or the usage of utterances (see Gazdar and Mellish 1989, p. 280, and Partridge 1991a, p. 336-339). Systems are, on the whole, concerned with the processing of utterances without taking context into account. Researchers have tended to brush such problems aside and to only worry about processing utterances one-by-one. Gazdar and Mellish (1989) say, "The pragmatics are everything to do with the context and the intentions of the speaker and addressee, and to a large extent this category is used to sweep up all the difficult aspects of meaning that researchers wish to postpone considering," (p. 280). Partridge (1991a) says, "AI-NLP work on pragmatics, what there is of it, is largely theoretical with sometimes a small-scale demonstration program," (p. 228). Taking the utterances in (12a) and (12b) below, and giving them as input to many of these computer systems, would result in the same question being answered twice as shown.

(12) a U: How do I get to Sainsbury's supermarket?
S: To get to Sainsbury's supermarket you must go to the City Centre.

b U: How do I get to the Sainsbury's supermarket?
S: To get to Sainsbury's supermarket you must go to the City Centre.

Anyone using such a computer system might find this alarming because the system should have noticed that the user asked the same question twice[13]. Also, there is probably a reason why the user has repeated the question. The user may not have been satisfied with the first answer and wanted more information about how to get to the supermarket. Certainly, if you asked a person the same question twice, you would expect a response like, "Why are you asking me again?," or "I just told you." The reason for failure within such computer programs is that they are, on the whole, concerned more with detecting the structure, and meaning, of utterances independent of their context. Hence, the work reported here will take a major departure from existing work in the field of natural-language interfaces because in this work we propose a theory and computational model which concentrate on the context of utterances.

The phenomenon just described is interesting because people expect systems, like other people, to be able to keep a history of the ongoing dialogue (see Cohen et al. 1982). In fact it is common for people who are communicating in dialogue to use techniques available from language to refer to previous utterances in the dialogue, or to refer to future utterances. When people ask questions, they will ask for elaborations of answers to those questions, explanations of answers, and confirmations of information they believe to be true. There will be some coherence of the intentions or

[13]We assume that the user is not told about the capabilities of the system beforehand.

goals and plans that people have in dialogue. People do not normally ask for explanations, or elaborations, of information which has not yet been introduced in the dialogue, nor do they normally introduce information into a dialogue without a reason. Note that we use the word, *normally* here, and this is significant, because throughout this work, we will assume that the person interacting with the system, and the system itself, are behaving rationally[14], or at least appear to be behaving so. Otherwise, the system in (12) above, could be construed as being totally intelligent, but behaving *irrationally*[15]. Allen (1983), Cohen et al. (1982), and Cohen and Levesque (1985, 1987), discuss the rationality assumption with respect to natural-language dialogue modelling.

There have been a number of computer systems developed which incorporate mechanisms for pragmatics processing. Examples are systems developed by Cohen et al. (1982), the UC system (see Chin 1988, Norvig et al. 1992, and Wilensky et al. 1984, 1986, 1988), the SC system (see Norvig et al. 1991, Hecking et al. 1988, and Kemke 1986, 1987), a system for giving advice on how to use electronic mail (see Pollack 1986), the TACITUS system (see Hobbs and Martin 1987), and the ViewGen belief processing system (see Ballim and Wilks 1990, 1991, Barnden et al. 1991, and Wilks and Ballim 1987). Many of these systems incorporate theories of discourse processing. As such theories are of interest we shall move on to introduce them.

1.4 Theories of discourse processing

There are a number of theories of natural-language discourse processing, and computational models of these theories exist (see, for example, Ballim and Wilks 1990, 1991, Barnden et al. 1991, Wilensky et al. 1984, 1986, 1988). Theories concentrate on the themes of semantics, structure, and intention. A common principle of all approaches is that they provide a model of the coherence of discourse. Semantic models argue that the coherence of a discourse is a feature of its meaning and that if you model the meaning, the coherence falls out of that. For example, there are theories regarding the coherence of semantics in discourse such as those proposed by Fass (1988), Schank (1972, 1973, 1975), Schank and Abelson (1975), and Wilks (1973, 1975a, 1975b, 1975c). Structure-based theories argue that a discourse can be modelled in terms of structural units which can be recognised and marked out in the discourse. These theories have given names like *topic* and *focus* to such units. Examples of such theories are proposed in Alshawi (1987), Grosz and Sidner (1986), Grosz (1983), Grosz et al. (1981), Sidner (1983, 1985), Webber (1978), and Dale (1988, 1989). Finally, other theories model the coherence of discourse from the point of view of intention, or the goals, plans and beliefs of the participants in the discourse. These approaches argue that people's intentions underlie their use of language, and that by modelling these intentions one can model language. The approaches do not argue that intentions in people's brains can be seen, but that people's intentions can be recognised, and inferred from the utterances they use. Examples of such approaches are given in Appelt (1981, 1985), Carberry (1989), Cohen et al. (1982), Hinkelman and

[14]For deeper discussions of rationality one can look at Dennett (1981), and Narayanan (1990a, 1990b).
[15]We do not make any specific claims here about the behaviour of irrational people, or systems.

Allen (1989), Hobbs (1979), Litman and Allen (1984), Schank and Abelson (1977), and Wilensky (1983). Ideas here emerge from claims made by philosophers of language such as Austin (1962) and Searle (1969). Such philosophers argue that the motivation for people to use language is to achieve their intentions. Various approaches to modelling the coherence of dialogue shall be investigated, and particularly the latter approach on the analysis of intentions, which is close to our own. Let us now investigate the concept of coherence of dialogue in greater depth as many approaches stress this concept.

1.5 Coherence of natural-language discourse

One of the features of natural-language discourse, when used by people, is that it is usually coherent to some extent. If you speak to your students in a classroom you will usually try to make your dialogue coherent, so that the students will understand what you are trying to convey. Of course, if the students do not understand the topic under discussion then the continuing coherence of the discourse may not be obvious. Hence, it seems that understanding is inextricably linked with coherence, and the less coherent a conversation, the harder it is to understand, and the less one understands a conversation the less it will appear to be coherent.

The idea of coherence in discourse is tied up with rationality. If we speak to someone we expect them to behave rationally, and we recognise that rationality by the coherence of what he/she say. So, if a policeman asks you, “Where were you on the 17th March 1991?,” and you reply to the policeman, “Where were you on 17th March 1991?,” it might seem rather odd to him, or when James Joyce uses the sentence,

> The great fall of the offwall entailed at such short notice the pftjschute of Finnegan, erse solid man, that the humptyhillhead of himself prumptly sends an unquiring one well to the west in quest of his tumptytumtoes: and their upturnpikepointandplace is at the knock out in the park where oranges have been laid to rust upon the green since devlinsfirst loved livvy.
> — Joyce (1939, p. 4)

in his book, Finnegans Wake, it seems ridiculous. However, the policeman will, most likely, not assume that you are behaving irrationally. He will first try to work out why you gave the answer that you did, and may decide that you did so because you wanted to avoid answering the question. The reader of Finnegans Wake does not assume that Joyce is irrational; Joyce probably intended to make a point while using such a strange *sentence.* Likewise, if someone says, “Mr. Brooke is going to solve the Northern Ireland problem,” and you reply, “And pigs will fly,” then he does not assume that you are irrational; he works out that you probably said something silly because you wanted to imply the silliness of his initial statement.

However, there seem to be certain rules about how far one can go, so as not to appear irrational. For example, if I uttered, “same please,” and I was first in queue at the restaurant, it would not make sense. Also, consider the situation where two colleagues step up to a bar counter, and one orders a rum and coke from one bartender, while the other says, "same please," to another bartender, where the second bartender has not overheard the first request. The second bartender will be understandably confused[16].

The point is that, under *normal* circumstances, people try as much as possible to assume a speaker is rational, until they cannot do so any more. We assume people are rational, and that they cooperate when they use language to communicate with us. This issue has received much attention in the philosophy of language, linguistics, and artificial intelligence. A philosopher of language, Grice (see Grice 1957, 1969, 1971, 1975), has termed this phenomenon, *The Cooperative Principle*[17].

As coherence of natural language is an indication of rationality, this coherence is very important. It is important because how we interact with people is dependent on their rationality, and if someone seems irrational then we may decide to terminate communication with them. Likewise, we may decide to terminate natural-language dialogue communication with a computer if the dialogue appears incoherent, and hence irrational. And the computer may do the same if we appear irrational. Hence, if a computer is to conduct natural-language dialogue communication then it is important that it have some sense of what it means for a dialogue to be coherent.

Consider the situation where a natural-language processor has the task of modelling coherence in dialogue. One of the problems of that processor will be to build the utterances of the dialogue into a coherent network. A major problem will be that utterances with the exact same syntax, will have different meanings in different contexts. For example, say a program exists which has the capability of conducting natural-language help on operating systems. Say, the dialogue in (13) below occurs between a user and the program.

(13) a U: How do I see my files on the screen?
b S: 'more <filename>' will display file contents on the screen.

In this sample dialogue the system answers a question about displaying files on the screen. Now, say (13) is followed by (14).

(14) U: What does more do?

The system should recognise that (14) is an elaboration of the reply given in (13b). However if (14) is followed by,

(15) U: How do I copy a file?

[16] I am indebted to Andreas Wehlau who demonstrated this example in the bar of the Northcott Theatre, at Exeter University, on March 8th 1991. I must admit that I did not think of this deviation from the first example.

[17] The Cooperative Principle has been formulated by the philosopher Grice (1975), where he argued that we can assume that people cooperate as much as possible in dialogue, in the endeavour to communicate effectively.

the system should recognise that (15) is not a request for elaboration, but a request for information. Hence, the context of an utterance in a dialogue will, to a large extent, determine its meaning, just as the context of words determines their meanings.

Halliday and Hasan (1976) have argued that coherence in discourse is a linguistic property of sentences. They point out that it is (1) anaphoric expressions, and (2) lexical items appearing across sentences, which establish coherence. In the former case, the links between references, and their previous occurrence, is considered, and in the latter case words are considered. Halliday and Hasan used the word *cohesion* to describe the linguistic properties of text that contribute to coherence, and argued that contributions to coherence are not a function of the reference of the words in a discourse, but of the forms themselves, and their histories of occurrence.

Another set of theories (Rumelhart 1975, and van Dijk 1972, 1977) treat coherence as reflecting a defining property for discourse, as comparable to grammars for sentences. They argue that coherence of discourse is due to grammars of coherence, and they have specified these grammars in detail. A good survey of such grammars occurs in Andersen and Slator (1990).

A more useful possibility of defining the coherence of discourse might be one which considers Grice's Cooperative Principle. Each utterance is considered to say something necessary, true, and relevant to accomplishing some objective, in which it is believed that the speaker and listeners are mutually interested. A coherent discourse would be one where an interpreter can reconstruct the speaker's plan from his/her utterances, and make inferences about them to understand the dialogue at hand. Hence, coherence here is defined not by the discourse components themselves, but by the effort needed to construct a plan to attribute to the speaker involved in producing the text. This will depend on how hard, or easy, it is to take each sentence as representing a true, necessary, and relevant contribution to the plan. It seems to be the case that the coherence of text is based, not only on properties of the text itself, but also on the inferences made by discourse hearers. As a central theme of our theory for solving, in part, the problem of discourse processing, is to model coherence through the analysis of intention, the meaning of intention shall now be discussed.

1.6 Intention in natural-language discourse

When you call your mother on the phone, or I write a letter to my father, we usually have a reason for doing do. Also, while using natural-language dialogue people utter requests, promises and orders. There are even certain types of requests, orders and promises, such as explanations, elaborations, and confirmations. It is usually the case that we use language to communicate our intentions[18] about some topic.

[18]The word *intention* is central to the work presented here. It is difficult to provide a definition of intention, because it means many different things to people from different research backgrounds. In this work, by intention, we mean the purpose of the speaker. This sense of intention will involve the goals, plans and beliefs of an agent who has an intention, and how they indicate such intentions, to a computer, by means of natural language. Such goals could be "to find out where Sainsbury's is", "to eradicate the Poll Tax" or "to print a file on the printer", with the respective plans, [use town-map], [Non-payment/obstruction], [use UNIX-command, 'lpr']. It turns out that,

It seems that the basic reason why people use language is to communicate some intention. The intention they communicate, of course, is not necessarily the one they wish to achieve. This is the basis for much work in the philosophy of language and the philosopher of language, Searle (see Searle 1969), argues that people are motivated to use language because of the intentions they wish to achieve in the world. Searle's predecessor, Austin (see Austin 1962) argued that people are motivated to use certain types of utterance to achieve intentions. Much research in linguistics and natural-language processing has taken up this claim, and have used it in the modelling of language.

The claim that intention analysis is necessary for effective natural-language dialogue between a person and a computer has been made by a number of philosophers of language including Grice (1969) and Dennett (1981). Also, this claim has been made by researchers in artificial intelligence such as Allen (1979, 1983), Allen and Perrault (1980), Cohen and Perrault (1987), Cohen et al. (1982, 1990), Perrault et al. (1978), Perrault and Allen (1980). From the point of view of psychology, Skinner (1957) discusses how people's intentions are manifested in their verbal behaviour. Hence, much research has concentrated on how intentions can be recognised in natural language, and how these intentions can be used in language understanding. If the motivation for the use of language is because of our intentions, then it would seem reasonable that the coherence of a discourse can be modelled from the point of view of analysing intentions. This is a major theme of the work presented here.

There are a number of forms of natural-language dialogue involving computers. For example, there is the case where two computers are communicating with each other. This is not of interest to us, as we are motivated by the ability of people to communicate with computers. Also, there are cases of n people communicating with m computers, where n and m are greater than one. However in this research, we are only interested in n and m, where n and m are equal to one, as this is the simplest case, and is very common in today's computing world. Yet, we do not rule out future studies concentrating on the case where n and m are greater than one. Natural-language discourse includes all forms of language communication whether written, or spoken, whether text, or dialogue. The interest here will be in natural-language dialogue.

An interesting phenomenon in natural-language dialogue is the way in which people interact, and the type of conversation that ensues. Different types of people conduct different types of dialogue. For example, nuclear physicists usually conduct a certain type of dialogue with each other, and it is different from the dialogues between farmers, or between fishermen. Hence, one of the phenomena of natural-language dialogue is the *type* of dialogue conducted by the participants. Of interest will be the fact that different types of people maintain different types of intention in their dialogues[19].

There are many types of people who interact with computer systems. These types vary from

as most of the work here is concerned with natural-language question answering, the intention types we discuss tend to be request categories such as *explanation*, *elaboration*, and *confirmation*. We are not interested here in any medical, or psychological notion of intention, or how intentions are represented, or formulated, in people's heads. For philosophical discussions on intention, see Dennett (1981), and Narayanan (1990a, 1990b).

[19]No claim is made here as to whether people are explicitly aware of the intentions that they are communicating in a dialogue, or whether those intentions are subconscious.

secretaries to systems' programmers. It is difficult to characterise the expertise of a person with respect to a computer system, as some people know more about some aspects of a system than others. For example, sometimes the best person to ask a question about word processing is not the system's programmer, but a secretary. However, assuming it is possible to characterise people by their degrees of expertise with respect to given topics, then we will choose *experts* as a term for those who know more than *novices*[20] about particular topics.

Types of intention can be characterised in many ways. One way to so this could be to give different names to the types of intention. For example, the names *explanation*, *confirmation*, and *elaboration* could represent the intentions of communicating explanations, confirmations and elaborations respectively. Of course, we already have words in natural languages such as English for intentions: we use words like *promise*, *hope*, *believe*, *learn*, and so on, to convey respective intentions. Utterances in natural-language dialogue will represent intentions. As natural-language dialogues consist of linear sequences of utterances, and as utterances represent intentions, natural-language dialogues will represent sequences of intentions. Hence, another way of characterising intentions, is their intention sequences. Also, it may be the case that different dialogues have different frequencies of intentions and intention sequences. Hence, *analysis of intention* could be characterised by types and sequences of intention. Now that we have characterised the meaning of *analysis of intention* it is possible to move on to describe a theory of intention analysis.

1.7 A theory of intention analysis

Our thesis is that a theory of intention analysis provides, in part, a solution to the problem of natural-language dialogue processing. A central principle of the theory is that coherence of natural-language dialogue can be modelled by analysing sequences of intention. We also wish to test the following hypotheses:

The Intention-Computer Hypothesis: The analysis of intention in natural-language dialogue facilitates effective natural-language dialogue between different types of people and a computer.

The Intention-Person Hypothesis: The analysis of intention indicates the degree of expertise of different types of people.

These first hypothesis says that it is possible to analyse the types and sequences of intention in natural-language dialogue and then to use that analysis to enable better natural-language dialogue between different types of people and a computer. It is claimed that certain types of intention will indicate to a computational model that a person is satisfied in a dialogue, while other types will indicate dissatisfaction. In turn, satisfaction and dissatisfaction will indicate whether a person

[20]There is no simple categorisation of degrees of expertise into such a binary set. In fact, it is more likely that there is a continuum of expertise, where people fall onto that continuum at different points.

is behaving, locally and globally, as an expert, or a novice. This information can be passed to a natural-language generator, which returns responses sensitive to such local and global behaviour.

The second hypothesis says that it is possible to analyse the types and sequences of intention in natural-language dialogue and then use that analysis to determine the level of expertise of different people. Certain types of intention will indicate that a person is behaving as an expert while others will indicate that a person is behaving as a novice.

The practical implications of the Intention-Computer hypothesis are that natural-language dialogue systems which incorporate intention analysis will be more accurate at responding to the specific intentions of specific types of users than those systems which don't. It is important to note that although the goal is to explore the validity of, or to test, the Intention-Computer hypothesis, any number of hypotheses incorporating intention analysis could be tested. For example, H1, H2, and H3 below could be tested. Indeed, we will see later that it has been argued by others such as Barnden et al. (1991), Farwell and Wilks (1990a, 1990b), and Green (1989) that intention is useful for solving many of the problems in natural-language discourse.

Hypothesis (H1)[21]: Natural-language dialogue can be segmented into discrete spaces, and the analysis of intention can facilitate effective determination of boundaries marking such spaces in natural-language dialogue.

Hypothesis (H2): Intentions have primacy[22]over topics in natural-language dialogue.

Hypothesis (H3): Topics have primacy over intentions in natural-language dialogue.

Hence, the thesis is to provide a theory of the analysis of intentions to solve, in part, the problem of natural-language discourse processing, through modelling discourse coherence, and to explore the validity of, and test two hypotheses. A theory of intention analysis, which can be incorporated within a computational model, is proposed.

In order to model intentions in natural-language dialogue we must have some idea of how that can be done. Each natural-language dialogue consists of a number of utterances in a temporal sequence. Each utterance by each speaker should convey some intention. Hence, there will be a mapping between utterances and intentions. However, it is most likely that the mapping will not be one-to-one from utterances to intentions. This is certainly the case with *indirect speech acts* (see Searle, 1969) like, "Can you pass the salt," which contain a number of possible meanings depending on the context in which they are used. It is likely that the coherence of a dialogue depends on the coherence of its intentions to some extent. A central principle of the theory of intention analysis is that a dialogue, which consists of sequences of utterances, can be modelled in terms of sequences of intention. There will be, at least, two properties of the theory. First, it must be possible to

[21]This hypothesis has been proposed by a number of researchers, most notably Grosz and Sidner (1986), and Reichman (1985).

[22]By *primacy* here, we mean that natural-language dialogue can be considered to have an architecture where the topic of the dialogue is considered to be subordinate to intention, and it is intentions that cause different topics to come into and out of focus.

recognise intentions from utterances, and second, it must be possible to build some representation of intention sequences. It is likely that there will be different types of intention represented by utterances, as people wish to accomplish a number of different types of activity in the world.

It must be the case that it is possible to recognise intentions in utterances as people are able to recognise each others' intentions to a great extent. Much of intention recognition by people will involve recognising facial expressions and voice intonations of speakers. However, it is also likely that the syntax, or structure, and semantics, or meaning, of utterances, and their context, will give some information about the intentions they represent. Hence, if you say, "Explain to me why your thesis is not done properly," your use of the word *explain*, as the first word of the utterance, indicates that your intention is to order someone to give an *explanation*. On the other hand, "on the other hand," might indicate that you have described some argument or point, and now you wish to describe the other side of it. Finally, the intention of the utterance "What do I do next?," will depend on the context of the utterance.

Hence, the recognition of intentions involves processing the syntax, semantics and pragmatics of utterances. We will see later that much work in natural-language processing has concentrated on the processing of syntax and semantics of utterances, and to a lesser extent the context of utterances.

Our central concern is to measure the coherence of dialogue by recognising and representing different types of intention. In this work intention types are used to determine the level of satisfaction of a speaker. In turn, *satisfaction* of the speaker is used to indicate level of user expertise. Following that, the level of expertise is used to generate context sensitive natural-language responses. It is argued that intention types can be placed in an ordering and that intention types on this ordering will represent lesser, or greater, satisfaction. For example, a number of *explanation* requests might indicate that a person is less satisfied, in a dialogue where help was being provided than if he/she uttered a number of basic requests. In this work we argue that the natural-language consultancy domain has at least nine principle types of intention which are: *information*, *description*, *instruction*, *elaboration*, *confirmation*, *explanation*, *guidance*, *repetition*, and *nointention*. The definitions for, and meanings of, these intention types are given in Chapter 5.

In contrast to much of the work on modelling the behaviour of users in dialogue the work here considers the local satisfaction of a user, whether he/she be a novice or expert, together with having a global notion of the level of expertise of the user. This approach is taken because people can be expert about a number of aspects of a domain, but novice about other aspects. Most work on user-modelling is only concerned with a global notion of expertise.

The requirements for a representation of intentions will depend on what we wish to represent. The theory and hypotheses require that we represent intention types and possibly sequences of those types. The theory could use a representation of intention sequences for modelling dialogue coherence. A minimal measure of intention sequence frequencies would be one where intention sequences incorporating one intention are counted. Of course, this would be no sequences at all. Such a count could be very useful as it would tell us information about the cumulative frequencies of various intention types in a dialogue. In turn, the cumulative frequencies can give a global picture of say, a user's expertise. However, this information does not give a measure of the local behaviour of a user. A simple frequency count of intentions does not enable a system to infer how many times an explanation intention has occurred in a row, but just how many explanation intentions have occurred.

Also, a measure of single-intention sequences would not be able to separate out the fact that, say, an elaboration of an explanation may indicate less satisfaction than an elaboration of an information request. It could be argued that a measure of *windows* of frequencies, counting the last, say, ten intention frequencies, could be considered but this is a move towards some form of sequencing anyway. In effect, it is argued that although a simple intention frequency count gives a global measure of the satisfaction of a user it does not give good decisions locally as to whether a person is satisfied or dissatisfied. However, a measure of intention pairs could be given different weights to model the performance of a user. For example an explanation followed by an explanation will get a higher dissatisfaction weighting than an information followed by an explanation.

Furthermore, there are more complex problems in natural-language dialogue which we will claim intention analysis can solve. A more complex representation of sequences of intention is needed to tackle these problems. For example, any model of the coherence of a dialogue is not possible without a check on intention sequences, rather than just intention frequencies. Also, it is our intention in future work to use the intention representation to investigate solutions to the problems of (1) ellipsis[23], (2) anaphoric reference resolution, and (3) the extended plans and goals of users. To solve such problems there must be some representation of intention sequences, rather than a simple frequency count. Hence, an assumption of our theory and hypotheses is that intention sequences of length two, or intention pairs, will be more useful in modelling intentions than a simple frequency count of intention types. This will involve an extra computational cost for a computational model of the theory, and there is always a tradeoff between computational cost and how powerful a mechanism we wish to have.

A representation can be used to store the frequencies of various intention pairs from a dialogue. Other representations could be used to represent intention triples, quadruples, and so on. However, in this work we shall concentrate on intention pairs, and leave other more elaborate representations to future work. Of course, rather than counting frequencies of intention pairs an actual representation of the dialogue itself, which represents the complete temporal ordering of all utterances, and hence all intentions, could be built. However, although possible, such an elaborate structure will not be necessary to provide evidence for our theory, and hypotheses. Hence, the requirements of an intention representation for our purposes will be that it can represent the cumulative frequencies of different intention pairs from a dialogue. A possible representation might be a picture showing intention pairs and links between them. The temporal ordering of intention pairs can be represented by a directed link, and the frequencies of various pairs can be represented as numbers on the links

[23]Ellipsis is where information is missing in a sentence, which hopefully can be inferred from context. Computational theories of ellipsis processing are provided in Carberry (1985, 1989) and Farwell (1990a, 1990b).

between them. The representation could be termed an *intention graph.* The intention graph can be used to represent phenomena relating to intention sequences in natural-language dialogue. As our theory and hypotheses involve interaction with a computer, a definition of how the theory is instantiated in computational terms must be given.

1.8 A computational model of intention analysis

A requirement of any computational model of the theory of intention analysis will be that the model, at least, be able to recognise and represent intentions. First, the problem of intention recognition will not be easy. The problem is that any given utterance will represent a number of different intentions, depending on its context. For example, the utterance, "What do I do next?," could represent a vast number of intentions, and sometimes the hearer may have to ask a question such as, "What are you trying to do?," to give an appropriate answer. The syntax and semantics of utterances will give information about the recognition of intention, as will the context of the utterance. If a natural-language parser which can use syntax, semantics, and pragmatics of natural-language utterances to recognise intentions, is constructed, that will get us well on the way to recognising intentions.

Second, there is a requirement of representing intentions, and intention sequences. The standard computer science data structure of a matrix can be used to represent such information. A two-dimensional matrix is sufficient for representing intention pairs[24]. Each cell of the matrix will represent cumulative frequencies of various intention pairs. The representation can be termed an *intention matrix.* To demonstrate that the theory of intention analysis can be used for effective natural-language dialogue between different types of people and computers it needs to be shown how this can be done using the intention matrix.

A computational model, called OSCON (Operating System CONsultant), written in Quintus Prolog, answers, in English, English queries about the UNIX and MS-DOS computer operating systems, and interacts with people through natural-language dialogue. With respect to intention analysis the program has four basic functions: (1) the recognition of intentions in natural-language dialogue, (2) the representation of intention sequences, (3) the application of the intention representation to user-modelling at local and global levels, and (4) the utilisation of the user-model to generate natural-language responses appropriate for the user and context at hand. The model uses a semantic grammar, in the Definite Clause Grammar (DCG)[25] formalism of Prolog, to recognise intentions in natural-language dialogue.

[24] If we wanted to represent intention triples a three-dimensional matrix would be used, and so on.

[25] Definite Clause Grammars (DCG's) were first developed by Pereira and Warren (1980) as a tool to be used in Prolog for natural-language processing.

1.9 Using intention analysis

As pointed out in the previous section certain intention types will indicate that a user is having trouble with the content in the dialogue, while others will indicate that the user is getting on fine. The Intention-Computer hypothesis requires that the computational model be able to analyse intentions, and use that information for determining the type of user who is interacting with the system. In turn, the user-model can be exploited to facilitate more effective dialogue with the user through the generation of user-sensitive natural-language responses.

The intention matrix, which represents the frequencies of various intention pairs, can be used to compute the level of expertise of a user. A function will be applied to the intention matrix and will weigh the frequencies of different intention pairs in different ways to determine the level of satisfaction of the user. Such a selective weighing function would not be possible if the theory, or computational model only considered a simple intention-frequency count rather than a intention-sequence count as there would be no consideration of the context of an utterance. The computational model can then decide to take appropriate action depending on its point of view of the user. For example, a natural-language dialogue program might decide to generate more elaborate answers for a user who is having trouble with understanding the content of the dialogue. Hence, this would give evidence for our theory and the Intention-Computer hypothesis.

1.10 Testing intention analysis

In AI it is common for researchers to have their views and theories of how, for example, computer programs should be constructed to handle problems such as natural-language dialogue processing, and yet not provide any data on where the views and theories come from, or whether they have been tested. It is all very well to have theories about how intelligent phenomena should be modelled but a question asked by many a researcher is: how close do these theories actually match the real world?

It has become common in AI to develop theories of phenomena without checking to see how those phenomena actually behave. This point has also been elaborated by Narayanan (1986), Sharkey and Brown (1986), Sutcliffe (1990), and Mc Kevitt (1990d, 1992) and Mc Kevitt and Partridge (1991). Many researchers build systems to tackle phenomena which are ill-defined and often the theory ends up being a researcher's *own* theory without any question as to whether it is a *valid* theory. An argument could be made that more experimentation should be done in AI. It is important to analyse the data resulting from empirical experiments on the AI problem at hand.

There are many ways to obtain the necessary data and some are closer than others to the real problem being attacked. For example, if the intention relations in natural-language dialogue are being studied it would be possible to obtain data from everyday conversations and make inferences about the intention relations therein. However, the problem with this is that the density of the phenomena looked for may be very low, and a lot of data would need to be collected and analysed. An investigation of a more limited domain which yields a higher density of intention relations will be more beneficial. Also, a more restrained domain provides a better mechanism for testing a program which constrains any hypothesis about intention relations. In this work we choose the limited domain of UNIX help, where people interact with the computer through the medium of natural-language dialogue, asking queries, and obtaining answers, about UNIX. Of course, it must always be kept in mind that care needs to be exercised in generalising empirical information gleaned from the limited UNIX domain to *any* natural-language dialogue.

Three experiments are conducted to test the Intention-Computer and Intention-Person hypotheses. The experiments provide positive results for the hypotheses. The first experiment demonstrates that it is possible for a computer program to analyse intentions, and that this can facilitate effective natural-language dialogue between different types of people and a computer. This experiment also demonstrates that the program, while analysing intention sequences, performs better natural-language dialogue processing, that when it does not analyse intention sequences. The other two experiments are conducted in the form of Wizard-of-Oz experiments where subjects are asked to conduct a number of tasks in the domain of computer operating systems. The subjects type their queries in English to the program, and a hidden person answers their queries. The subjects are not told that a person is answering their queries. The first experiment shows that there seems to be a frequency difference in the types and sequences of intentions for different subjects. The second experiment shows that there is a significant frequency difference in the types and sequences of intentions for different types of subjects.

This evidence provides support for the theory and two hypotheses. The results have a number of implications, both theoretical and practical, for a variety of research fields such as natural-language processing, human-computer interaction, artificial intelligence and philosophy.

1.11 Implications

Obviously, this research has implications for natural-language processing. It is possible that other problems in natural-language processing can be solved by modelling intention. We will show in later chapters that one of the problems of natural-language discourse, the problem of modelling the level of expertise of a user in the discourse, can be solved, in part, by intention analysis. Additionally, we will claim that the problems of ellipsis resolution and anaphoric reference resolution can be solved, in part, through intention analysis. In Chapter 9 we will discuss in detail the implications of intention analysis.

Hence, in summary the results here provide support for a theory of intention analysis and the Intention-Person and Intention-Computer hypotheses. The following chapters show how some initial evidence was obtained. The analysis of intention by computer, whether by natural language or otherwise will, hopefully, enable people to use computers more effectively, and computers to understand people better.

1.12 Overview

Part 2, consisting of Chapters 2, 3, and 4, discusses current approaches to discourse modelling. Here, the major theories are examined in detail, showing how they propose to solve the problem. Research falls within three main themes: modelling semantics, structure, or intention. Some of the approaches, while stressing their own theme, include elements of the themes of others. All of the approaches argue that there is some coherence to discourse, and that their particular approach captures that coherence.

Chapter 2 describes research which models natural-language discourse from the point of view of meaning, or semantics. Such research argues that the coherence of a discourse is inherent in its semantics, and that modelling the semantics of the discourse will capture that coherence.

Chapter 3 describes research which emphasises structure in the modelling of discourse. These approaches argue that the coherence of a discourse can be measured in terms of its structural properties. Structural properties include the topic and focus of the discourse.

Chapter 4 describes intention-based models of discourse processing. These approaches stress the recognition and representation of goals, plans or beliefs of participants in a discourse. The models are based on the assumption that it is the intentions of the participants in a discourse that make the discourse coherent. Hence, models described in this chapter have much in common with our theory. The chapter is introduced with a short survey of the philosophical roots of intention-based approaches to discourse. The chapter concludes with research by Pustejovsky which is an integrated approach to discourse modelling incorporating semantics, structure and intention.

Part 3, consisting of Chapters 5, 6, and 7, introduces the theory of intention analysis, describes a computational model of that theory, called Operating System CONsultant (OSCON), and describes an application of OSCON to the problem of user-modelling. In turn, the user-model can be used for generation of user-sensitive natural-language responses. In Chapter 5 the theory of intention analysis for natural-language dialogue is introduced. A central principle of the theory is that coherence of natural-language dialogue can be modelled from the point of view of analysis of sequences of intentions. It is argued that each utterance in a dialogue represents some intention, and that different types of utterance denote different types of intention. The theory centres on the principle that a dialogue, which consists of sequences of utterances, can be modelled in terms of sequences of intention. A *satisfaction* ordering of intentions is described, and it is pointed out that there may be a correlation between intention types and the satisfaction of the speaker uttering those intention types. In turn, we claim that the *satisfaction* of the speaker can be used to determine his/her degree of expertise. The motivation for an ordering of satisfaction is to investigate the Intention-Computer hypothesis. *Intention graphs* are introduced as pictorial representations of sequences of intention in dialogue. Intention graphs can be used to represent a number of interesting phenomena in natural-language dialogue. Finally, requirements are introduced for a computational model of dialogue modelling using intention analysis. The computational model needs to be able to recognise, and represent intentions.

Chapter 6 discusses a computational model for intention analysis in natural-language dialogue.

It is shown how the computer model can recognise different types of intention, and how counts of sequences of these intentions are represented using an *intention matrix*. The Operating System CONsultant (OSCON) program, which answers, in English, English questions about the UNIX and MS-DOS computer operating systems, incorporates this model of intention analysis.

In Chapter 7 it is shown how the theory of intention analysis, and its representative computer model, OSCON, can be used to model the level of expertise of a computer user. It is shown how a user-model, modelling the level of expertise of a user, can utilise the intention matrix. In turn, the user-model facilitates the generation of user-sensitive natural-language responses. The motivation for incorporation of such a user-model is to explore the validity of the Intention-Computer hypothesis.

Part 4, consisting of Chapter 8, describes three experiments which test the Intention-Computer and Intention-Person hypotheses. The first experiment shows that the OSCON computational model recognises and represents different intention sequences, and, by utilisation of a user model, caters its natural-language answers to those sequences. This experiment provides positive evidence for the Intention-Computer hypothesis. The other two experiments are in the *Wizard-of-Oz* form, where subjects are asked to conduct a number of tasks in the domain of computer operating systems. These experiments provide evidence for the Intention-Person hypothesis.

Finally, Part 5, containing Chapter 9, concludes with a summary of the work, and how it relates to existing theories in artificial intelligence, linguistics and philosophy. Implications of the work are discussed, and possibilities for future research are presented.

Appendix A provides sample traces which demonstrate the computational model, OSCON, answering English queries about the UNIX and MS-DOS operating systems. One sample traces demonstrates OSCON operating in dialogue modelling, and user-modelling, mode.

Appendix B, as part of Experiment I described in Chapter 8, gives sample traces of OSCON's behaviour while performing, and not performing, intention-sequence analysis. These sample traces also demonstrate that the OSCON system conducts intention analysis over a natural-language dialogue by the processes of intention recognition, intention-sequence recognition and intention representation. Appendices C and D give details of how the two Wizard-of-Oz experiments, Experiments II and III described in Chapter 8, were conducted.

Now, that the contents of the following chapters have been summarised we can begin with Chapter 2 which describes meaning-based approaches to modelling natural-language discourse.

Part II

Theories of discourse

Chapter 2

Discourse modelling through semantics

There has been much research conducted on developing theories, and designing computational models for understanding natural language discourse. Much of the work has been on developing models for text and speech processing. Some work has concentrated more specifically on processing natural language dialogue. The next three chapters serve as an introduction to the area of discourse modelling and the various points of view on how it should be accomplished. Many of the approaches have common themes, while there are differing points of view between them. The one common theme of all the models seems to be centred on the coherence of a discourse.

First, in this chapter we will concentrate on semantic-based approaches to modelling discourse. Semantic-based approaches argue that the coherence of a discourse can be analysed from the point of view of its meaning or semantics. Approaches centre on techniques for recognising and representing the semantics of the discourse. There are many methods for conducting semantic processing, but most techniques represent semantics using some form of meaning representation. Formal languages for the representation of meaning of natural language utterances include propositional logic, predicate calculus, semantic networks (see Quillian 1969), Conceptual Dependency (see Schank 1972 1973, 1975, Schank and Abelson 1975), and Preference Semantics (see Wilks 1973, 1975a, 1975b, 1975c, and Fass 1988).

There has been much debate with respect to how syntactic and semantic processing within natural language systems should be integrated. Some argue for systems where syntactic analysis is conducted first, and is then followed by semantic analysis. Some argue that syntax analysis should be conducted hand-in-hand with semantic analysis. Others argue that semantic analysis should come first, with syntactic analysis subservient. The latter approach has been taken most notably by Schank and Riesbeck (see Schank 1973, and Schank and Riesbeck 1981). The semantics-first camp argue that people still understand, and can build a representation of, ill-formed syntactic sentences like, “Paul Japan go to” with ease, even if it is a strange utterance. Hence, they argue, syntactic analysis is not central to natural-language processing. We will discuss the research of of Wilks and Schank here, as these are two of the major semantic-based approaches to natural-language discourse processing.

2.1 Wilks

Wilks (see Wilks 1973, Wilks 1975a, 1975b, 1975c) built a computer program for text processing which handled paragraph-length input. It used a system of preferential choice, or *Preference Semantics*, between deep semantic patterns based on *semantic density.* The program was applied to the problem of machine translation.

Wilks' technique uses a fragmentation routine to fragment text and then represent it in an interlingual structure called a *semantic block.* This block consists of *templates* bound together by *paraplates* and *common sense inferences.* Each of these three items consists of formulas (sic), with predicates and functions ranging over the formulas and subformulas. The formulas, in turn, consist of *elements.* Semantic items represent text items as shown in Table 2.1.

Items in semantic representation	*Corresponding text items*
formula	English word sense
template	English clause or simple surface item
semantic block	English paragraph or text

Table 2.1: Wilks' Semantic representation of text items

The paraplates and common-sense inferences combine to cause bound templates in the semantic block. Semantic elements exist as primitives out of which all the above complex items are made up.

Elements are primitive semantic units used to express the semantic entities, states, qualities, and actions about which humans speak and write. Examples are (a) entities: MAN(human being), (b) actions: FORCE(compels), and (c) cases: TO(direction). In addition to these primitive elements, there are *class* elements whose names begin with an asterisk. Examples are *ANI, for the class of animate elements, MAN, BEAST, and FOLK. *HUM is the class of human elements MAN and FOLK.

Semantic formulas are constructed from elements. They express the sense of English words. One formula exists for each sense. The most important element is always the rightmost, which is called the *head*, and it expresses the most general category under which the word sense in question falls. The formula for the action sense of *drink* is shown below, and the *cases* and *values* of the subformula shown in Table 2.2.

```
"drink" (action) –>
     ((*ANI SUBJ) (((FLOW STUFF) OBJE)
                         liquidity
                         ((SELF IN) (((*ANI THRU PART)) TO)
                                        aperture
```

(BE CAUSE)))))

Subformula	*Case(Act)*	*Value*
(*ANI SUBJ)	SUBJ	*ANI
((FLOW STUFF) OBJE)	OBJE	(FLOW stuff)
(SELF IN)	IN	SELF
(((*ANI (THRU PART)) TO)	TO	(*ANI (THRU PART))
(BE CAUSE)	CAUSE	BE

Table 2.2: *Cases* and *Values* for Wilks' subformula representation

The explanation of the subformula is that:

- the preferred agent is animate
- the preferred object is liquid
- the container is the self, the subject
- the direction of the action is a human aperture (i.e. the mouth)
- the action is of causing to be (somewhere else)

Drink is an action preferably done by animate things to liquids, or flowing substances, causing the liquid to be in the animate thing, and via a particular aperture of the animate thing, the mouth. The notion of *preferring* is important.

Templates are networks of formulas giving connectivity between agent, action, and object formulas. Each clause, phrase or primitive sentence of text (called fragments) is seen by a program as *senses* or strings of formulas drawn from a dictionary. There is one formula for each text word.

Bare templates exist as sequences of agent, action, and object heads into which semantic formulas can slot. If there exists a series of formulas whose heads are identical to a bare template of elements then the sequence of formulas is a template for that fragment together with any other formulas that depend on the main three. The program tries to locate one, or more, templates in each string of formulas by looking at head elements and seeking proper sequences of heads. It is an hypothesis of the work that an inventory of bare templates can be built up for the analysis of ordinary language.

In processing, each sentence is seen as a string of formulas, one for each of its words. The program will look only at the heads of the formulas. Templates are expanded by building formulas into a network and by strengthening bonds of formulas in a template. An algorithm is used to look into subparts of the formulas that express preferences to see if any preferences are satisfied. Hence one template network may be preferred over another.

A preference is between alternatives, and if the structure derivable does not satisfy preferences then it is accepted anyway. Paraplates are then used at another higher level to establish case ties between templates. Paraplates are of the form:

<list of predicates on mark-template>case
 <list of predicates on case-template>
  <generate stereotype>

A stereotype is a context sensitive generation pattern for generation of natural language output. Paraplates are patterns that span two templates: mark and case. *Mark* is the point of dependence of a phrase or clause. In the utterance, "He ran the mile, in four minutes," the second clause depends on the action "ran" which is its mark.

If all predicates are satisfied by the contents of the second template then the paraplate is considered to have matched the template and the case ambiguity of the preposition in the second template is solved. Taking the template for the second fragment "in 4 minutes," from the example, "He ran the mile in 4 minutes," all the predicates in the paraplate for TIMELOCATION case match onto the appropriate parts of the second template fragments. Then the case of the second template is TIMELOCATION, and not CONTAINMENT, as in, "He ran the mile in a plastic bag." Paraplates are attached to key words in English, such as prepositions and conjunctives, as lists in left-to-right order.

Wilks shows mark and case paraplates in (1) linear order as they really are, and (2) tabular form for ease of comprehension. We show the paraplate for *in* below, in both linear order, and as a tabular form schematic paraplate in Table 2.3.

PR MARK (MOVE CAUSE)) (2OBCAS INST GOAL)/
(TO into) (PROBJE (CONT THING))

(PR MARK (MOVE CAUSE))	FIRST AGENT
(2OBCAS INST GOAL)	FIRST ACTION
	FIRST OBJECT
(TO into)	SECOND AGENT
	SECOND ACTION
PROBJE (CONT THING))	SECOND OBJECT

Table 2.3: Wilks' schematic template for *in* in tabular form

With respect to linear structure there is a correspondence with the six major formulas of the mark and case templates. The predicates in the paraplates may refer to any, or all, of these. A '/' in the paraplate indicates the shift from mark template predicates to case template ones. Predicates like PRMARK have an argument that is a list which is a subformula that has to be located whole in the appropriate template formula so as to satisfy the predicate in question.

The linear form above shows a paraplate for *in* and where on the six formulas of the two templates above, predicates will match. In this form TO is a case marker. 2OBCAS is a predicate that looks at both the object (third) formulas of the current template (second) and of the preceding

```
(1) The soldiers fired at the women and we saw several of them fall

(2) (1 (THIS STRIK) (*ANI 2) <-> ((*ANI 2) (NOTUP BE) DTHIS)
                                                 dummy
(3) [1 strikes animate2] <-> [animate2 falls]
```

Figure 2.1: Inference rule

```
[(MAN)     inject   liquid  ]
[(MAN)     (USE)    #tube   ] insertion of fuel
[ICengine  (USE)    #liquid ] petrol moving engine
[*         (MOVE)   []      ] moves the car
```

Figure 2.2: A Pseudo-Text for car

template, i.e. at two objects. 2OBCAS is true iff they contain the same GOAL or the same INSTRUMENT subformula. PROBJE is a predicate on the semantic form of the object (third formula) of the current template and is satisfied if the predicate's argument is found in the formula. PRMARK is a predicate on the semantic form of the mark, or a word governing the fragment that the key begins.

Anaphora[1] resolution is conducted by computing semantic densities of alternative ties for references and getting the one with the denser representation. Semantic densities are numerical probability values. Also, common sense inference rules are used to understand more complex sentences like that shown in Figure 2.1. In (1) above we show an example utterance where the resolution of "them" can be dealt with through the inference rule in (2). The rule is made clearer in (3) by extending the informal '[]' notation to denote "template form". DTHIS is used as a dummy variable to fill out canonical form.

Wilks later introduced what he called *Pseudo-Texts* into his system (see Wilks 1975c). He points out that sentences which break preferences are the norm in language use, and not the exception. He notes that in the sentence, "my car drinks gasoline," none of the senses for car are animate, and so the system simply accepts what it is given. However, this is not enough as far as understanding the utterance goes. He discovered the need for representations of objects such as "car". The program should have access to a sufficiently rich knowledge structure for "car" and be able to notice that cars consume particular types of fluid, that is of the same semantic structure as the relation in which a drinker normally stands to a liquid to be drunk.

Pseudo-Texts are structures of factual and functional information about a concept, or item, and are similar to frames in the sense of Minsky (see Minsky 1975), Charniak (1977), and Schank (see Schank and Abelson 1977). An example of a Pseudo-Text for car is given below: in Figure 2.2. A Pseudo-Text is a semantic block consisting of a linear sequence of template forms tied by case

[1] *Anaphora* is explained in Chapter 1.

ties. The linear order of the templates is taken to indicate normal time sequence. The program accesses the Pseudo-Text for car and seeks the template in it with the closest match to the source form template. Wilks uses a notion of *projection* to describe the process of placing *use* in place of *drink* in the formula for drink.

In summary, Wilks' theory of Preference Semantics is one which assumes that the coherence of a discourse can be computed bottom up from the semantics of individual sentences. The coherence is computed in terms of the preferences of templates for various formulas. There is no notion of determining structural properties of a discourse, or of processing intention within Wilks' theory. Next, we look at another approach to representing discourse semantics which is similar to Wilks'.

2.2 Schank

Schank introduced a semantic representation of the meaning of sentences for natural language processing called Conceptual Dependency (CD) (see Schank 1972, 1973, 1975, and Schank and Abelson 1975). The basic axiom of Schank's CD theory is:

Axiom 1: For any two sentences that are identical in meaning, regardless of language, there should be only one representation.

The following corollary is derived from Axiom 1:

Corollary 1: Any information in a sentence that is implicit must be made explicit in the representation of meaning of that sentence.

An economical form for representing meaning is developed. The following additional axioms are used:

Axiom 2: The meaning propositions underlying language are called conceptualizations. A conceptualization can be active or stative.

Axiom 3: An active conceptualization has the form: Actor Action Object Direction (Instrument)

Axiom 4: A stative conceptualization has the form: Object (is in) State (with Value)

Axiom 1 forces the representation to include actions that seem similar to see if the basis of the similarity can be extracted. Axiom 2 requires that the differences between two actions should be made explicit. Two verbs in a language can share a primitive element but may also have differences. For example, TRANSFER OF POSSESSION is the primitive element shared by the verbs, "give," and, "take," but these actions also have differences. Schank defines eleven primitive acts of CD as follows:

ATRANS: The transfer of an abstract relationship such as possession, ownership or control. For example, "buy" is made up of two conceptualizations, one an ATRANS of the object money, the other an ATRANS of the object being bought.

PTRANS:	The transfer of the physical location of an object. For example, “put” is PTRANS of an object to a place.
PROPEL:	The application of a physical force to an object. PROPEL is used whenever any force is applied, regardless of whether a PTRANS took place. The verbs, “push,” “pull,” “throw,” and “kick” have PROPEL as part of them. For example, “John pushed the table to the wall,” is a PROPEL that causes a PTRANS.
MOVE:	The movement of a body part of an animal by the animal. MOVE is usually the ACT (act) in an instrumental conceptualization for other ACTs. For example, MOVE foot is the instrument of “kick,” and MOVE hand is often the instrument of the verb, “hand”.
GRASP:	The grasping of an object by an actor. The verbs “hold,” “grab,” and “throw,” involve GRASP or the ending of a GRASP.
INGEST:	The taking of an object by an animal to the inside of that animal. The most common objects for the semantics of INGEST are food, liquid and gas. “Eat,” “drink,” “smoke,” and “breathe” are common examples of INGEST.
EXPEL:	The expulsion of an object from the body of an animal into the physical world. Something previously INGESTed could be EXPELled. Words like “sweat,” “spit,” and “cry” are described by “expel”.
MTRANS:	The transfer of mental information between animals, or within an animal. Memory is partitioned into two pieces: CP (conscious processor), and LTM (long term memory). Things are thought of in the CP, and things are stored in the LTM. The various sense organs can also serve as the originators of an MTRANS. For example, “tell” is MTRANS between people, “look” is MTRANS from eyes to CP, and “learn” is the MTRANSing of new information to LTM.
MBUILD:	The construction by an animal of new information from old information. “decide,” “conclude,” and “imagine” are all examples of MBUILD.
SPEAK:	The actions of producing sounds. Only human objects usually speak as an instrument of MTRANSing. “Say,” “play music,” “purr”, and “scream” involve SPEAK.

ATTEND: The action of attending or focusing a sense organ towards a stimulus. "Listen" is ATTEND ear, and "see" is ATTEND eye. ATTEND is nearly always referred to in English as an instrument of MTRANS. In CD "see" is treated as MTRANS to CP from eye by instrument of ATTEND eye to object.

As sentences representing the same meanings are to be mapped into the same representation, the set of primitive ACTs (acts) is essential for representing meanings. The use of primitive ACTs reduces the inference problem as inference rules will only have to be written once for each ACT. For example, if you MTRANS something to your LTM, then it is located there. It does not matter whether MTRANSing was "see," "hear," "inform," "memorise," or whatever. The inference comes from the deeper representation of an ACT, rather than the verb.

A number of scales are used for conceptualizations that are attribute-value statements. The scales run between boundaries which are labelled -10 to 10. Scales indicate changes in state. For example, some of these scales and boundaries are shown below in Figure 2.3.

HEALTH (dead, diseased, under the weather, in the pink)
MENTAL STATE (broken, depressed, all right, happy, ecstatic)
AWARENESS (dead, unconscious, asleep, awake, keen)

Figure 2.3: *Scales* and *boundaries* in CD

One of the criticisms of Schank's theory of CD has been that emotional states cannot be measured in terms of numbers. Also, the set of primitive acts has been criticised for being too small a set.

A notion of causality is important to CD. In sentences such as the following we note that the second sentence must be in some way related to the first.

John came over yesterday. Boy, was he mad.

There are causal relationships of various types between sentences and rules for determining causal relationships between events are needed.

Schank points out that in the physical world the following causal syntax exists:

Causal Syntax 1: Actions can result in state changes.

Causal Syntax 2: States can enable actions.

However, not all actions can result in any state, and not any state can enable any action. For every primitive action, the set of states it can affect, and the set of states that are necessary in order to effect it, are associated with it. Schank pointed out that as there are only eleven primitive actions the boundaries of this world knowledge are finite. However, much of the criticism of Schank's work centres on the fact that eleven primitive acts are not enough to provide a basis for world knowledge. For acts such as PROPEL there are a list of states that can result from it, as well as the set of conditions under which those conditions can occur. The applicable rules are:

Rule 1: PROPEL can result in PHYSICAL CONTACT between the object of the PROPEL and any objects in the location specified in the Directive case.

Rule 2: PROPEL results in PHYS.ST (-) if one of the objects in the PROPEL is human, if it results in PHYSICAL CONTACT(PHYSCONT), and if the force of the PROPEL is great.

Thus, in the example, "John's leg was broken because Mary knocked over a pile of bricks," when these rules are applied to <Mary PROPEL Mary to bricks> we get <Mary is in PHYSCONT with bricks>. The causal rules say that Mary could be damaged but not John's leg. It is possible to use the causal rules backwards to get the following hypothetical event:

$Something_1$ PROPEL $something_2$ to leg(John)

This hypothetical event can be derived from some real world event as an inference in order to make sense of the causation stated. This can be accomplished if it is known that:

If a movable object is put in PHYS.CONT with the object of a PROPEL then it can become the object of the new PROPEL, where the actor of the new PROPEL is the same as object of the original PROPEL.

The following can be hypothesised as an inference:

Mary PROPEL bricks to leg(John)

The original rules for PROPEL can be used to prove the latter hypothetical event causally correct in the sense that the known event could cause it, and the known state could result from it. Hence, causal change inference rules are an important part of the understanding process. Other rules are shown below:

Rule 3: States can disable actions.

Rule 4: States (or acts) can initiate mental states.

Rule 5: Mental states can be reasons for actions.

The rules are useful for three purposes: (1) They can be used to decide what is, and is not, a causal chain, (2) the correct analysis of conceptual representations for words such as "prevent", "help", "allow", and (3) they provide a means of representing connected text. The latter point is important because this is where Schank's theory provides, in part, an analysis of discourse coherence. Connecting up causalities such as 'state enables action,' and 'action results in state,' is what makes sense of a text. The text will be disconnected and incoherent if a causal chain cannot be constructed to represent it. The basic philosophy of CD then, is the principle of how to make explicit that which is implicit in text. CD handles this problem at the single thought, or sentence, level. Causal chains handle the problem at the level of interconnected thought, or text, level.

In summary, Schank's theory of Conceptual Dependency (CD) argues that different natural language sentences can be mapped into an internal representation. He defined the internal representation, called Conceptual Dependency (CD), which utilises a set of eleven primitive acts into which all natural language actions can be mapped. He also, argues for a set of causal inference rules which enable the construction of CD representations for complete texts. The coherence of a text will be indicated by the coherence of the final CD representation.

2.3 Summary

The main theme of meaning-based approaches to modelling natural language discourse is that the coherence of discourse can be represented in terms of the coherence of representations of semantics of sentences or utterances in the discourse. We have shown two meaning-based approaches to representing natural-language discourse where both of them claim that natural-language words and sentences can be mapped into representations containing sets of primitives. The main theme of both approaches is that the coherence of the discourse will be represented, in part, by the coherence of semantic representations constructed from templates and inference rules. Both models argue that the difficulty, or ease, of template matching and causal chaining will determine the coherence of a discourse. Other approaches to modelling discourse argue that the coherence of the discourse will not fall out of semantics alone. They stress that the structure of the discourse is important. We can now discuss such structure-based approaches.

Chapter 3

Modelling discourse from structure

Structure-based approaches to discourse modelling tackle discourse coherence from the point of view of discourse structure. These approaches argue that there are structures in discourse, whether explicit, or implicit, and those structures can be recognised and represented. Much of the work centres on strategies for recognising and representing the *topic* and *focus* of the discourse. Examples of structure-based approaches are described in Alshawi (1987), Dale (1988, 1989), Grosz (1983), Grosz et al. (1983), Grosz and Sidner (1986), Sidner (1983, 1985), and Webber (1978). There are also formal structure-based approaches to discourse understanding discussed in Barwise and Perry (1983), Heim (1982), and Kamp (1981). While concentrating on the former approaches we will also discuss the formal approaches briefly. We close with a discussion of Pustejovsky's model (see Pustejovsky 1987) which models both structure and intention in discourse.

3.1 Grosz and Sidner

Grosz and Sidner (1986) propose a new theory of discourse structure, stressing the role of purpose and processing in discourse. Structure consists of three subcomponents (1) structure of sequence of utterances: linguistic structure, (2) structure of purposes: intentional structure, and (3) state of focus of attention: attentional state. It could be argued that Grosz and Sidner's approach should be taken as being general, but we do not do that because the approach stresses structure over intention. Linguistic structure is the natural aggregation of segments of discourse. Intentional structure captures discourse-relevant purposes expressed in the segments as well as relationships among them. Attentional state is focus of attention of participants as discourse unfolds. This records objects, and propositional relations that are important at each point in the discourse.

They point out that the subcomponent distinction is important for explaining cue phrases, referring expressions and interruptions. Their basic elements of a computational theory of discourse structure simplifies and expands upon existing previous work. The work involves the integration of two lines of research (1) focusing in discourse (Grosz 1978a, 1978b, 1981), and (2) intention recognition in discourse (Allen 1983, Sidner 1983, Sidner 1985, Litman 1985, and Pollack 1986). However, the former structure-based research is predominant in their theory. Grosz and Sidner concentrate on generalising the task-oriented elements that constitute discourses. Discourses are task-oriented but the tasks are varied. Tasks are physical, mental and linguistic. By distinguishing subcomponents they simplify computational mechanisms of related work. Intentional structure introduced depends on a small number of structural relations between intentions. They present an abstract model of discourse structure and intentionally do not recognise the need for the specifics of a computer program, or for a psychological model.

Utterances are the actual saying or writing of particular sequences of phrases and clauses and are the linguistic structure's basic elements. Intentions and relationships between them provide the basic elements of intentional structure. Attentional state contains the objects and properties or relations and discourse intentions that are salient at some point. Abstraction of the focus of attention of participants summarises information from previous utterances and hence there is no need for a complete history of the discourse. These three critical issues allow the Conversational Participant (CP) to determine how an individual utterance fits the rest of the discourse, i.e. why it was said, and what it means. Expectations of what is to come are also integrated in generation and interpretation.

With regard to *linguistic structure* utterances are aggregated into discourse segments. The utterances in a segment are relevant to that segment, and the segment is relevant to the discourse. Two consecutive utterances may be in different, or the same segments. Nonconsecutive utterances may be in the same segment. Segmentation of discourses has been proven by Grosz (1978b) for task-oriented dialogues, Linde (1979) for arguments and Reichman-Adar (1984) in informal debates, explanation, and therapeutic discourse. There has been a lot of debate on segmentation boundaries. There are few psychological studies although Mann (1975) has shown that when people segment dialogues they do so in similar ways.

Linguistic expressions are the primary indicators of discourse segment boundaries. Cues like "in the first place" are dividers that function to indicate these boundaries. Reichman calls these, *clue words* and Grosz and Sidner *cue phrases.* Boundaries can indicate changes in (1) attention, or (2) intention. Grosz and Sidner point out that discourse segmentation is useful in defining referring expression boundaries. The constraints for intra-segment reference are different from inter-segment reference.

With regard to *intentional structure* participants have purposes in dialogue. Some of the purposes will distinguish coherent, from incoherent, discourse. They define foundational purpose as discourse purpose (DP) which is the intention that underlies the discourse engagement. This component of Grosz and Sidner's work is very close to what we describe in future chapters. We also argue that the analysis of intention distinguishes coherent, from incoherent, discourse.

Discourse Segment Purpose (DSP) is the intention of the segment. Typically, the Initiating Conversational Participant (ICP) will have a number of different kinds of intentions associated with the use of discourse. The recognition of DP or DSP is essential to achieving purpose. ICP's use discourse for impressing or teaching. Private intentions are involved here too. Here is an example of a type of intention that can serve as DP/DSP which is intended to have some agent perform some physical task:

Intend that Ruth intend to fix the flat tire.

DP/DSP is similar to what we call intention. Grosz and Sidner define structural relations that play an important role in discourse structure. A dominance relation is defined as an action that satisfies one intention, e.g. DSP1 may be intended to provide part of the satisfaction of another, say e.g. DSP2.

DSP1 contributes to DSP2
DSP2 dominates DSP1

A dominance relation places a partial-ordering on DSPs called a *dominance hierarchy*. A second structural relation is *satisfaction-precedes* which defines the case where one DSP must be satisfied before another. For some discourses, including task-oriented ones, the order in which the DSPs are satisfied may be significant. For example, DSP1 satisfaction-precedes DSP2 means DSP1 must be satisfied before DSP2. The relationship between discourse purposes here is similar to what we will describe as intention sequencing in later chapters. The ordering of the intentions is important, and that will also become a recurring theme in later chapters.

Grosz and Sidner point out that the list of possible intentions that can serve as DPs or DSPs may be infinite and therefore it is not useful for determining discourse structure. They argue that what is essential for discourse structure is that intentions bear structural relations to one another. They distinguish between (1) determination of DSP: which is a semantic-like notion of what is intended by whom, and (2) recognition of DSP: which is a processing notion of the processing that leads a discourse participant to identify what the intention is. Recognition involves the processing notion of the discourse participant knowing the intention. Grosz and Sidner are more concerned with the relations between intentions rather than intentions themselves. We argue that to discover the relations between intentions it is important to determine the intentions themselves. Therefore, a distinction between our approach described in later chapters, and theirs, is that we emphasise the determination of intentions and relations between them, whereas they emphasise only determining the relations.

Attentional state is the abstraction of focus of attention and saliency at different points in the discourse. It is a property of the discourse and not the participants. It is inherently dynamic, recording objects, properties and relations that are salient at each point. The following structures are defined:

(1) Focus space: a set of these models attention.

(2) Transition rules: model changes in attention state. These specify conditions for adding/deleting focus spaces.

(3) Focusing structure: the collection of focus spaces available at any one time that are salient at each point.

(4) Focusing: the process of manipulating focus spaces.

The focusing process associates focus spaces with each discourse segment. The space contains entities that are salient because of (1) saliency in the segment, or (2) producing or comprehending utterances in the segment (see Grosz 1978b). Also the focus space includes the DSP. The fact that the focus space contains the DSP shows us that there is a commitment to structure/topic over intention.

While processing a discourse, focus spaces are created for each new DS and spaces are placed one on top of the other in a stack. The topmost elements of the stack are the most salient ones. Popping is done from the stack when the current focus space does not dominate the new one. This will give the dominating focus space. Focusing structure has the following properties:

(1) Focusing structure is parasitic on intentional structure, and the relationships among DSPs determine pushes and pops.
(2) Focusing structure evolves as the discourse proceeds. Three different types of information play a role in determining the DSP.

They give two examples of discourses processed by the theory: (1) an argument dialogue from a rhetoric text, and (2) a task-oriented dialogue. Grosz and Sidner also discuss processing issues and how to use the theory in constructing models where the model of the ICP is taken into account. Of interest are:

(1) How the ICP indicates, and the Other Conversational Participant (OCP) recognises the beginning and end of a discourse segment.
(2) How the OCP recognises the discourse segment purposes and

(3) How the focus space stack operates.

Processing issues involve intention recognition. One of the central problems is recognition of the DSP. This can be completed by the following methods:

(1) Hirschberg and Pierrehumbert (1986) suggest that intonation provides partial information about DSP relationships.
(2) Cue phrases partially specify a DSP but do limit the options.

(3) Utterance level intention of each discourse utterance. The OCP has this available.

(4) Shared knowledge on actions and objects in the domain of discourse.

The supports relations between propositions and dominance relation, which generates between actions, help in determining the interpretation of the following discourse. Intention is recognised as follows.

Recognition is not complete until the end of the DSP as any utterance in the segment may contribute. The use of an attentional state model enables certain processing decisions to be made locally. It limits the information that must be considered in (1) recognising the DSP, and (2) identifying referents for classes of definite noun phrases.

A term called centering (see Grosz et al. 1983, and Sidner 1979) is a more local phenomenon than focusing. A backward-looking centre (bl-c) is associated with each utterance in a discourse segment. bl-c is the focus-element that is central in that utterance. Syntactic, semantic and discourse information is used to identify the bl-c. This constrains the search for the referent of a pronoun in a subsequent utterance. Unlike the DSP the bl-c may shift where different entities become salient at different points. They make a conjecture that topic is a concept that is used ambiguously for both DSP and centre (intention and centering).

Grosz and Sidner extend Grice's (1969) theory of utterer's meaning to DP/DSPs. This involves recognising discourse-level intentions rather than just utterance-level.

(a) Specify the discourse-level intentions and attitudes that correspond to utterance-level intentions.
(b) Identify the kinds of features of utterance that contribute to determining discourse-level intentions.
(c) Identify modes of correlation between features of the discourse segments and types of discourse-level intentions.
(d) Specify how discourse-level intentions can be recognised by the OCP and argue that they have provided a partial answer to the problem.

In summary, Grosz and Sidner's conjectures are:

(1) Discourse is coherent only when its DP is shared by all participants and when each utterance of the discourse contributes to answering this purpose.
(2) Intuitions about topic correspond most closely to DP/DSP rather than to syntactic or attentional concepts.
(3) The same intention structure gives rise to different attention structures through different discourses.

They point out the following problems, open for further research include:

(1) Relationship between discourse level (DP/DSP) and utterance-level intentions.
(2) Identification of information that discourse participants use to recognise intentions and the ways in which they utilise it.
(3) Development of an adequate treatment of interaction among intentions of multiple participants.
(4) Investigation of the effect of multiple DSP on the theory.
(5) Investigation of alternative models of attentional state.

We now list some properties and problems that we believe exist with the work of Grosz and Sidner. First, their theory does not concern itself with meaning, but the role of DP/DSP in structure. Second, Grosz and Sidner's approach to dialogue modeling has information stored in a data structure called a 'stack'. A stack is a data structure where information is piled on top of previous information. The structure has the inherent quality that data first-in is last-out and data last-in is first-out. Terms like PUSH have been coined to describe placing data onto the top of the stack and POP for removing the data off the top of the stack.

The motivation for the use of a stack is not clear in the work of Grosz and Sidner. If the stack is motivated as a mechanically convenient, and simple method of representing a discourse then there are no arguments against it. However, if the stack structure is motivated by more theoretical considerations then we must object. The principle of a stack would mean that they believe dialogue to be inherently structured like a stack. They would be claiming that people in conversation bring up topics, discuss them, and then leave the topics again just like one would PUSH data and POP data from the stack. However, there are three main arguments against the use of a stack data structure for dialogue modeling, in principle: (1) it is based on the assumption that people bring up a topic, discuss it, and then leave it again; (2) it stores information based on the constraint that near-at-hand information is the most recently talked about; and (3) it says nothing about how references can be made to items no longer on the stack.

With respect to (1) we believe that the stack notion may not be well founded because dialogue is not always so well structured. People do not just open new focus spaces and close them again. We believe that dialogue is more random and that people forget to ask certain questions or make statements within various topics and do not close these spaces with any regularity. Also, because of this people are always referring back to previous topics of discussion with quite a lot of randomness. With respect to (2) we believe that contextual data storage should be more semantic based rather than structure-based. It is more efficient to store contextual information based on an importance principle rather than a recency principle. Therefore, the most important topic should be at the top of any data structure used rather than the most recent. Also, (3) is the most significant problem with the stack data structure. A system using a stack would have no means of working out references to items which have been POPped from the stack.

A data structure called a 'tree' is more appropriate for representing data context. Items at nodes in the tree would be context spaces or topics which were discussed by the user or system. These context spaces would never be closed but always remain open for further information coming in. The root node in the tree represents the most salient (or talked about) topic of the previous discourse. The tree is ordered so that nodes below other nodes contain less salient topics.

The rigidity of the stack-based model forces Grosz and Sidner into trying to make data fit their structure, rather than the structure fitting the data. They end up saying things like "if it [the discourse] completes normally" (Grosz and Sidner 1986, p. 180). We know of no notion of discourses completing normally and they seem to mean if the discourse terminates in a fashion where all elements are removed from the stack. Also, the stack leads to problems in the case of interruptions because the system must recognise when an interruption has occurred and treat this as a special case. Their stack algorithm must then treat interruptions as special cases.

In summary, Grosz and Sidner's approach to modelling the coherence of discourse is motivated by modelling the topic and focus of the discourse. This is integrated with models of the intentions and linguistic structure of the discourse. The main mechanism of their computational model is a stack which stores information about context by PUSHing focus spaces onto the stack. However, this has an inherent principle of recency about it. Also, the determining factor for focus spaces is syntactic, with the use of conversational cues, with no regard for the semantic content of utterances. Now, we move on to a theory and model based on the relationships between structures in discourse.

3.2 Reichman

Reichman (1985) proposes a theory of discourse which characterises a conversation as a hierarchical organisation of related utterances. She proposes a formal discourse grammar which can serve as a model of the interpretation and generation of "coherent" conversational speech. The grammar is based on the fact that an utterance is appropriate with regard to the discourse context that is currently "relevant". By this, she claims that Grice's general maxims of conversational cooperativeness (see Grice 1975) are turned into a well-defined set of operational rules.

Reichman describes two major discourse functions of an utterance, (1) a continuation of what has gone on before, and (2) a new communicative act serving a new discourse role. Utterances of type (2) cause a shift in the discourse which Reichman calls a *Conversational Move*.

Examples of conversational moves are explanations/clarifications, presenting claims, challenging claims, and shifting topic. Reichman points out that although Conversational Moves reflect speaker goals they can be identified without reference to a speaker's underlying intent for an utterance. We take issue with this point as it is hard to see how intention recognition has not got a role to play here.

Reichman defines a basic discourse constituent called a *Context Space* which exists as a fundamental concept of what she terms *Context Space Theory.* She describes how discourse participants package pieces of discourse into these spaces and bring them into and out of the forefront of attention. The structure of a discourse can be specified by the identification of context spaces and relationships between them. Reichman defines a set of different types of context space. There are four major types of context space:

Issue spaces: One of the most important types of context space is an issue space. These act as *topic setters.*

Non-issue spaces: These spaces are not topic setters and do not introduce propositions. There are two types of non-issue space: (1) comment or narrative, and (2) non-narrative or support.

- **Narrative:** These spaces are concerned with telling stories.
- **Non-narrative:** These spaces are concerned with describing states of the world.

Context spaces may be independent discourse constituents or may be dependent subconstituents of other context spaces. Spaces contain a fixed number of slots for holding various kinds of information. We show some definitions of various slot types below.

(1) Each space has a "type" slot for specifying the type of context space being activated. There are ten types of context space.
(2) All context spaces have a "derivation" slot which specifies the origin of claims within the context space origin, whether explicitly stated by the speaker, or inferred by the grammar.
(3) There is a "goal" slot, which specifies the Conversational Move to be fulfilled in developing a space. This designates the discourse function that the context space has been activated for (e.g. support, explanation, and clarification). It is noted here that the goal slot in Reichman's context space encodes information about what we call intention type in later chapters. Note that, unlike Pustejovsky, Reichman places 'support' on the same level as 'explanation.'
(4) A "method" slot specifies the method used to perform the conversational move (e.g. analogy, narrative).
(5) A "corelator" slot designates the preceding part of discourse to which this new space is related.
(6) A "speaker" slot contains the list of persons generating utterances.
(7) A "status" slot contains information about the foreground/ background role of the context space at a given point in the discourse.
(8) A "focus" slot specifies the influence of the elements within the context space. The focus slot has four subslots for degree of focus (1) high, (2) medium, (3) low, (4) zero.

Some context spaces, such as debative-issue ones, have a protagonist slot. Two problems which arise are (1) how to determine which context space is currently in focus, and (2) which context space an utterance is relevant to. Reichman's solution to this is to recognise that particular conversational moves have standard effects on their influence on preceding context spaces.

For Reichman, context spaces contain goals whereas we believe that goals and plans are not substructures but at least equivalent in priority, and the structure of discourse helps in disclosing this in some way. Also, Reichman is primarily concerned with conversational structure and conversational markers rather than the dialogue content. Reichman allows the conversational moves, their components, and the relations between them, to form the basis for the elaboration of her context space grammar. Again, we note that Reichman places intention in second place to topic, as Grosz and Sidner do.

A context space grammar patterned after Augmented Transition Networks[1] (ATN's) (see Woods

[1] *Augmented Transition Networks* are networks developed by Woods (1970) where grammar rules for natural language are represented by a network of *states* and *arcs* where states reflect the linguistic units identified thus far, and arcs reflect possible moves or transitions through the system.

1970) is used to model an ongoing discourse. Each unit of analysis is a context space. Each traversal of the discourse ATN entails the production or parsing of a single conversational move.

Reichman seems to rely on Conversational Moves for the opening and closing of context spaces. In everyday discourse it is probably the case that Conversational Moves are not that obviously marked with cues. Moreover, Reichman's context spaces are motivated by argument structure in structured argument conversations. Reichman does not deal with context spaces for question answering dialogues. In fact she says, "In addition to the context space types we have so far described, other discourse phenomena – such as descriptions, small talk, question-answer pairs – may ultimately be incorporated into context space structures for the purposes of discourse analysis," Reichman (1986, p. 62). However, it is not obvious how these discourse phenomena may be incorporated into context space structures.

In summary, Reichman's theory and computational model involve the analysis of discourse coherence from the point of view of a discourse grammar. She defines *Conversational Moves* as shifts in discourse intentions and basic discourse constituents as *Context Spaces*. She explains how context spaces can be used to package pieces of discourse and how they combine to represent a discourse. Reichman's theory and model then is similar to Grosz and Sidner's in that it incorporates a model of discourse coherence using the structure of the discourse, and the recognition of implicit structures within the discourse. We now move on to the structure-based theory and model of Alshawi (1987) who argues for a spreading-activation model of discourse coherence.

3.3 Alshawi

Alshawi (1987) proposes a unified framework for solving a class of interpretation problems. These include (1) reference resolution, (2) word-sense disambiguation, and (3) interpretation of implicit relationships. He proposes central mechanisms of (1) memory, and (2) context which are concerned with the salience of entities in the current context. He admits that (1) reasoning about the intentions of discourse participants, and (2) reasoning by cases, are explicitly excluded from his model. His model is a solution, rather, involving the application of linguistic and world knowledge.

In Alshawi's model memory mechanisms encode relationships between word senses and other semantic entities. Representations in memory include (1) specializations, and (2) correspondences. Specializations are encodings of specialisations of objects. For example (specialization: computer of machine) encodes that computers are special types of machines. Correspondences show correspondences between different objects. For example, (corresponds: data/processing to computer as machine/activity to machine).

He has developed a text processing system called Capture which incorporates an implementation of a model of contextual processing for natural language. Capture has a model of context represented by a series of *context factors*, which are collections of contextual entities. Each context factor contributes to the context activation of a particular set of memory entities. The set of entities is called the *scope* of the context factor. There is also a numerical value called *significance* *weight* associated with a context factor. The sum of current significance weights of context factors within the scope of an entity is called *context activation.* The weight of context factors changes during processing.

Alshawi defines seven context factor types which Capture incorporates: (1) recency, (2) emphasis, (3) processing-history, (4) deixis[2], (5) subject area, (6) association, and (7) task.

(1) *Recency* context factors include entities in sentences, paragraph and text whether they are mentioned explicitly or indexically in the text.

(2) *Emphasis* context factors concern single memory entities and Capture has two types of emphasis factor. (1) Syntactic-topic emphasis factors foreground topics of sentences in the passive voice (e.g. The referent of machine in "The machine is supplied by Smith"), and (2) Be-clause emphasis factors foreground the agents of be-clauses. (e.g. Plexir in Plexir is a manufacturer).

(3) *Processing-history* factors keep a trace of memory processing so that entities which take part in memory processing are increased in activation. Make1 would be the sense of make which has the entity manufacture as specialization. For example, a marker processing operation executed during the interpretation of an utterance with the verb "manufacture" might generate a processing-history context factor where make1 is in its scope.

(4) *Deixis* context factors are those where the sum of significant weights from recency of mention of deictic entities is higher than a system constant. Entities in the scope of such a factor will have their activations increased if they have been mentioned frequently and recently in the foregoing text.

(5) *Subject-area* factors increase the activation of entities which are considered to be related to a particular subject area. Taking the topic of "data processing manufacturers and suppliers" the set of core concepts includes: "machine" "supply1" and "manufacturer."

(6) *Association* factors increase the context activation of entities in memory which are closely associated with entities currently in focus. This is similar to Grosz's (1977a) use of *implicit focus* and Sidner's (1979) associated foci.

(7) *Task* factors are those which are very specific to some application task.

Alshawi proposes that the above factors are adequate for interpreting static descriptive texts which describe objects. More than one factor can be present at any given time during text interpretation.

However texts involving connected information like stories would also include the following context factor types: (1) time and space, (2) goals and discourse purposes, (3) agent associations, (4) event sequences, and (5) dialogue structure.

[2] *Deixis* is the where an agent uses pointing in conjunction with an utterance.

(1) *Time and space* factors are those where the spatial and temporal relationships affect foregrounding of entities. For example, the interpretation of texts describing sequences of instructions for robot tasks.

(2) *Goals and discourse purposes* are a type of context factor which increases the context activation of entities related to the participants goals or purpose of discourse segments. Alshawi points out that this factor type would be useful in a human-machine advice giving system. This compares with the discourse purpose work of Grosz and Sidner (1985).

(3) *Agent associations* factors are those foregrounding objects, plans and events related to agents involved in situations in texts.

(4) *Event sequences* involve factors which foreground entities used for representing generic events in stereotyped event sequences of the kind encoded as *scripts* and these have been implemented in a story understander called SAM by Schank (1975) and Schank and Abelson (1977).

(5) *Dialogue structure* factors which are generated to foreground entities which take part in structures of knowledge guiding the discourse. It has been shown by Grosz (1977a) that some discourses like task-oriented ones have a structure which mirrors the structure of other knowledge, e.g. knowledge of mechanical assembly tasks.

Alshawi categorises context factors into high-level and low-level. *Processing-history*, sentence *recency* and *association* are low-level factors and *subject area*, *event sequence* and *dialogue structure* are high-level. He proposes that low-level factors can be used in recognising high-level factors by bootstrapping. The context factors are used to operate (1) reference resolution, (2) word sense selection, and (3) to define focus spaces. This restricts reference resolution searches and finding references for plural noun-phrases. The lower-level factors are more useful in unstructured discourses whereas the high-level factors are useful in the opposite case.

He tried a number of mechanisms of applying context factors and finally decided that the best mechanism was to use all context information present at the time of application. Other methods tried were (1) where memory operations had only a specified set of context factors as parameters, and (2) where memory operations were parameterised by a set of context factors currently at the top of a "context factor stack". These methods were abandoned because they complicate the control flow in the program and make it very inefficient.

He uses a marker processing method (see Charniak 1983) within his program to implement memory and context mechanisms. The scope of a context factor is encoded as a marked set. The simplicity of marker processing facilitates implementations in parallel hardware.

Alshawi applies his model to three common problems in natural-language processing as mentioned above. We will only discuss the noun phrase reference resolution here, as this is the most significant of the three problems and gives a better insight into his processing methods. A solution to this problem is essential to any text processing system. Resolution is conducted by locating memory entities which correspond to noun phrases. The Capture system can handle pronouns (e.g.

It is made by Marconi), definite noun phrases (e.g. *The machines* are red), compound nominals (e.g. *beer* cask, *The terminal manufacture* makes P9920), possessives (*doctor's* house, The cost of *Mikota's peripheral* is 235), and noun phrases with restrictive adjectival (*liddled* pot, Smith supplies the *blue machine*), and relative clauses (e.g. Jones supplies *the machine that is manufactured by Plexir*). The context mechanism is called forth when the parsing structures do not disambiguate possible candidate referents.

Alshawi uses a reference resolution mechanism similar to Mellish's (1980) technique. Mellish's technique accumulates constraints on entities until they are strong enough to identify the referent uniquely. Alshawi's algorithm does not do this because inherent ambiguity may not produce a unique referent. Alshawi argues that Mellish's technique worked because in mechanics problem texts there is little or no inherent ambiguity. Alshawi's algorithm derives constraint markers from noun phrase construction types by a number of methods:

(1) If the noun-phrase head noun is a generic entity in memory then a constraint marker is generated by a marker which marks specializations.

(2) For pronouns the constraint marker is derived by marking the specializations of a generic memory entity which subsumes the entities that the pronoun can refer to.

(3) Verb cases (e.g. agent) are specialised to the arguments of the verb of a clause (e.g. the argument 'agent-of-murder' can generate a constraint).

(4) Restrictive relative clauses like "who loves Mary" provide predications from which constraints are derivable.

(5) Restrictive adjectives where the adjective is viewed as a property of other entities, using correspondences, or a class restriction, using specialisations.

(6) Constraint markers are generated as a result of relationship interpretation operations which are used to derive explicit relationships from language constructs where relationships are implicit or vague. Such language constructs include compound noun phrases, possessive noun phrases, prepositional phrases using *with* and some *have* clauses. These include constraints based on specialised relationships implicit in a wide range of constructions.

Context is applied to singular and plural definite references after the above constraints have been derived. A parser gives number expectations to noun phrases. If the number expectation specifies a single referent then an initial search request parameterised by the focus threshold is evaluated. Entities which satisfy constraints are ignored if they have context activations lower than the threshold. Also, entities which satisfy the threshold condition but not the constraints, are ignored. If the search fails then another search is made without the threshold condition. If more than one entity is located by either search, the entity with the highest context activation wins. When the two activations are equal a derived association factor is created. If there are no entities satisfying all the constraints the referents satisfying the highest number of constraints wins.

For plural referents the procedure depends on whether the number of entities referred to in the set is known. If the number is known, the correct number of referents is chosen from the entities satisfying the reference constraints by selecting those with the greatest activations. When the number is not known the following algorithm is used:

An initial search is made for a memory entity which satisfies the constraints and a focus threshold condition. Such an entity may have been the referent of another plural noun phrase or may have been created as a result of interpreting a conjunction. If the search locates many such entities then the highest activated one is chosen. If no such entity is found then a search is made for all the entities which satisfy the reference constraints. The set of referents located here is taken to be the referent of the plural noun phrase.

Alshawi introduces what he calls *database capture* which is the task of processing natural language texts to extract the propositional content which can be included in a structured database. He points to two example applications for such a system: (1) generation of relational databases from museum records, and (2) medical reports.

The Capture program performs the task of creating a database from the factual content of a series of English texts. The system processes collections of English paragraphs, for example museum records, and produces output which allows incorporation within an existing conventional database. Two example domains were used: (1) hypothetical museum artifacts and their collectors, and (2) retailers' records of data processing machines, suppliers and manufacturers. The target databases are fixed format tables, of the standard kind conforming to the Relational Data Model.

In summary, then, Alshawi's model of discourse processing centers on the use of *context factors* which are collections of contextual entities. The context factors are activated in memory with weights indicating their relative importance. Hence, Alshawi, like Grosz and Sidner and Reichman models the coherence of discourse from the point of view of recognising implicit structures in the discourse. Alshawi's computational model, Capture, is a useful one as it integrates existing theories and models but he does not seem to provide a central theory of his own on dialogue modeling; rather a combining of existing techniques. The relationship with our work is the stress on the intention component of his approach. Another model of discourse context has been provided by Dale (1988).

3.4 Dale

Dale (1988) proposes a means of generating subsequent referring expressions[3] in structured discourses for the purposes of natural-language generation. Subsequent referring expressions are expressions where there are references to entities which have already been introduced. Dale shows how discourse structure affects the generation of subsequent referring expressions.

Dale describes the generation process as one which generates clauses from specifications of eventualities. These specifications are called Eventuality Specifications or ES's. They have the form [Process, Arglist] where Arglist contains a set of arguments. So, the structure in Figure 3.1

[3] *Subsequent referring expressions* are expressions which will refer back to previously mentioned items. In Chapter 2, subsequent referring expressions were introduced as anaphoric references.

```
Process = contain$
Arglist =  Arg1 = entity001
           Arg2 = entity002
```

Figure 3.1: Eventuality specifications

```
for each ES in DS do
    CCWS:= ES
    generate˙clause (CCWS)
    PDR:= PDR + CM
    CM:= CCWS
next ES
```

Figure 3.2: Algorithm for updating discourse model

specifies the fact the entity whose internal name is entity001 contains the entity whose internal name is entity002. A Discourse Specification (DS) is used for the generation of extended discourse.

The discourse model contains three components, (1) Current-Clause Workspace (CCWS), (2) Cache Memory (CM), and (3) Previous Discourse Referents list (PDR). The CCWS contains the ES currently being processed while the CM contains the last ES supplied to the clause generator. The PDR contains the entities mentioned previously in the discourse.

Dale makes two assumptions: (1) Referring is the process of providing a description which is sufficient to distinguish an element from a set, and (2) Pronominal reference only takes place with respect to entities which are currently in cache memory or the current clause workspace. The first assumption, Dale admits, provides a simplification as it ignores the need to provide a *useful* description. The second assumption is controversial, again admitted by Dale, although pronominal reference is usually made to entities mentioned in previous sentences. Also, one-anaphora and verb-phrase ellipsis seem to obey this locality constraint.

Dale proposes a simple algorithm for updating the discourse model and for generating discourse from discourse specification shown in Figure 3.2.

The clause to be generated is developed in the current clause workspace. Cache memory contains syntactic detail for the previous clause, but only a subset of this is placed into the previous discourse reference list. Syntactic information beyond the immediately preceding main clause is not retained.

Dale points out that one solution to minimise computational cost is to order the entities in the previous discourse referents list in terms of salience. Then, the problem of reference determination becomes one of looking at the most salient entities. Dale says that it is not clear what the measure of saliency is but that *recency of mention* is commonly used in the literature. In constructing a referring expression the intended referent need only be distinguished from those entities which have been mentioned since the intended referent was last mentioned. He proposes a new means of cutting the cost of reference search by reducing context size.

Dale criticises one of the major structure-based discourse theories, that of Grosz and Sidner (1986) which has already been discussed here. Dale points out that Grosz (1977a) gives an explanation of reference in terms of structures called focus spaces which are related to the structure of the discourse. He argues that this could be done just as well with pragmatic factors denoting intentions. Also, Grosz and Sidner (1986) suggest that the reiteration of an indefinite noun phrase is caused by a structural change in the discourse where there are two different discourse segments. However Dale points out that Reichman (1978) shows that people frequently use full noun phrases in preference to pronouns.

Dale shows an example from Grosz and Sidner (1986), the “screw” example where straightforward recency of mention does just as well as focus space structure, and points to the fact that possible referents in a stack are not relevant. Finally, in Grosz (1977a) there is a “compressor” example where the implication is that entities in intervening discourse do not interfere because they have been popped off a stack. Dale says that this can be explained as deixis where the compressor is present when the reference is made. Therefore, Dale decides that the evidence given by Grosz and Sidner (1986) does not support the claims they make. The claim that the structure of discourse determines the context with respect to an intended referent is neither proven nor unproven. In our mind this is an important point as we are interested in showing discourse structure is not completely useful in modelling dialogue without taking discourse intentions into account. Dale goes on to show that discourse structure could be used to explain “long-distance” pronominalisation which is the use of a pronoun to refer to an entity last mentioned sometime before the preceding clause. A long-distance pronoun then, in Dale’s terms, is one which refers to an entity no longer in cache memory. Dale suggests that long-distance pronominalisation is rare and that there is a heuristic for determining when long-distance pronominalisation occurs. It is only possible with the first utterance immediately following the closure of a discourse segment. This happens as (1) it is not possible to use a pronoun for a closed focus space, and (2) Dale’s assumption of reference not being outside cache memory.

Dale describes generation in terms of the discourse structure model. He says that for natural language *understanding* this is difficult as there are no algorithms developed for determining the structure of discourse. This does not happen with generation. Dale discusses the generation of structured discourse from plans. Top-level goals in a plan may be decomposed into subgoals and this is comparable to discourse segments relating to each other. We will be describing some methods and algorithms for determining discourse structure in following chapters.

Dale modifies the previous described model so that the previous discourse referents list is now replaced by a stack of focus spaces. Each focus corresponds to a discourse segment. In generating referents the reference generation algorithm takes account of discourse structure. The algorithm (1) uses discourse structure to reduce the size of context to be considered, and (2) offers the possibility of explaining long-distance pronominalisation. His program is being implemented in Prolog (see Dale 1989).

Dale shows that there are a number of unsolved problems for his model of discourse generation:

(1) How do his predictions compare with other models?, (2) How adequate is a solely structural approach?, Do the semantics of the elements of the discourse have any effect?, (3) The notion of discourse model as a stack of open focus spaces is clearly inadequate. This says nothing about how reference can be made to entities no longer on the stack, (4) What are the consequences of only letting discourse segments contain single utterances? (5) Is discourse structure useful below the level of the sentence?, and (6) Can reference be determined by structure alone or is a notion of linear distance required?

In summary Dale's approach to discourse modelling relies on the structure of the discourse. He does not incorporate any model of the discourse semantics and uses an implicit structural entity called the *Current-Clause Workspace*. Although his approach is structural he does note the importance of what he calls pragmatics/intention in discourse processing. We move on now to discuss briefly two formal semantics approaches to discourse modelling.

3.5 Kamp

Kamp (1981) is concerned mostly with the correct interpretation and representation of discourse referents. Kamp argues that deictic and anaphoric occurrences of pronouns are identical. Identifying antecedents for these involves selection from specified sets of previous entities. For each utterance Kamp defines a Discourse Representation Structure (DRS) containing quantification over the entities in the proposition and the propositional content. An example is shown in Figure 3.3.

1a. Pedro owns a donkey.
1b. He beats it.

u v

Pedro owns a donkey
u = Pedro
u owns a donkey
donkey (v)
u owns v

Figure 3.3: Example discourse representation structure

Here, we show the DRS for a short dialogue. The novel piece of Kamp's theory comes from the DRS for 1b. There are no possible referents for the pronouns in 1b. This does not bring forth a new DRS but must be embedded within another structure as shown above. The DRS for the discourse pair is shown below in Figure 3.4 where the linking between the pronominals and their antecedents is now possible.

Hence, Kamp's approach to discourse modelling involves the incorporation of a formal structure-based representation of the discourse. There is no consideration of intention or semantics. Another formal approach is that of Heim.

u v

Pedro owns a donkey
u = Pedro
u owns a donkey
donkey (v)
u owns v
He beats it
u beats it
u beats v

Figure 3.4: Discourse representation structure for *discourse pair*

3.6 Heim

Heim's (1982) theory is similar in flavour to Kamp's. She is concerned with how to represent the presuppositions from utterances. The notion of a *file* is central to this theory. This is a record of descriptions of entities which is evaluated with rules of *familiarity* and *file-change*. Heim says that every sentence has file-change potential. This means that all utterances have potential to change the contexts for the utterance following it. Heim believes that the common ground between speaker and hearer, which is the set of presuppositions common to both, is stored in a file of a context. Hence, Heim's approach is a structure-based one which attempts to derive semantic presuppositions from utterances.

3.7 Summary

The structure-based theories and computational models to discourse processing model the coherence of a discourse from the point of view of implicit and explicit discourse structures. Most of the approaches argue that structures representing the topic and focus of a discourse can be recognised and represented. A number of different names are given to entities which represent discourse structure. Grosz and Sidner use the terms *topic* and *focus*, while Reichman uses the term *context factor*. Alshawi uses *focus spaces* for representing information in discourse. All of the approaches argue that discourse structure can be recognised by syntactic markers called *conversational cues* or *clue-words*. Although many of the structure-based approaches mention other elements of the discourse such as semantics and intention they consider them secondary to structure. For example, Grosz and Sidner and Reichman consider intention as being subordinate to topic.

Now that we have considered structure-based approaches to modelling discourse we can move on to discuss intention-based approaches. Such approaches argue that modelling the meaning, or structure, of a discourse isn't enough; a model of the intentions of the participants in the discourse is necessary.

Chapter 4

Modelling discourse using intention

Intention-based approaches to discourse modelling argue that the coherence of a discourse can be modelled from the point of view of the intentions or goals, plans, or beliefs of the discourse participants. The approaches centre on the recognition and representation of goals, plans or beliefs of participants in a discourse. The models developed here are based on the assumption that it is the intentions of the participants in a discourse that lend coherence to the discourse. To that extent much of the work described here will fall in line with our work described later.

Examples of research incorporating intention analysis include Appelt (1981, 1985), Carberry (1989), Cohen, et al. (1982), Hinkelman and Allen (1989), Hobbs (1979), Litman and Allen (1984), Schank and Abelson (1977), and Wilensky (1983). There are plan-based approaches to text processing such as those by Alterman (1985) and Lehnert (1978, 1982) but we shall not go into those here because we are more interested in the specific problem of modelling dialogue. First, we start with a discussion of some philosophical work on intention.

4.1 Philosophical views of intention

Speech acts, or communicative acts, are a means of describing intention in discourse or dialogue. Austin (1962) studied a class of natural language utterances which he called *performatives* that do not refer to states in the world, but to acts such as *promising*, *threatening* and *naming*. Austin argued that it does not make sense to take the traditional approach of understanding utterances like, “I name this ship the ‘Beer Engine’ ”, in terms of truth and falsity, but to consider whether they are *felicitious*, or whether they are appropriate in the context where they are used.

Austin described a number of acts which occur in normal discourse. First, we have the *phonetic* act which is the articulation of the jaw, diaphragm and larynx which results in connected speech sounds. The goal of this act is to produce a sound that the addressee will recognise as speech sounds. Austin termed *phatic* act the intonation of an utterance in a dialogue. He defines referring with indexicals, agreeing to the conventions of the language and culture of the community which the speaker shares with the addressee, *rhetic* acts. All of these acts are defined as being *locutionary* acts, or acts of saying. Searle (1969) uses *utterance* act for the union of phonetic and phatic acts, and *propositional* act for *rhetic* act.

An *illocutionary* act is the act of something being done by performing the *locutionary* act. An illocutionary act may be questioning, commanding or describing. A speaker will also, in performing a given illocutionary act, perform some actions. These actions which can be intended, or unintended, are called *perlocutionary* acts. If a person makes an utterance, issuing a warning, then the cause will probably be to warn the hearer. Likewise, if an utterance is made, issuing a request for a confirmation, the effect may be that the hearer will give that confirmation.

An utterance may contain linguistic expressions which serve to indicate the illocutionary force of the utterance. In (1a) below, "warn" is a *warning* that something has gone wrong, and in (1b) "order" indicates an *order* for something to be done.

(1) a I warn you, be careful.
b I order you to take charge.

Warning and *ordering* are caused by uttering specific words. Austin termed verbs like these *performatives*. Only certain verbs act as performatives. For example, in (2a) and (2b) nothing is performed by the verbs therein.

(2) a I know what I know.
b I love all things sweet.

The terms *speech act* or *communicative act* have become common in the literature as alternative terms for *illocutionary act*. However, traditionally speech act and communicative act cover a wider range of phenomena than illocutionary act. Explaining, intimating, confirming, and predicating are all examples of speech acts.

Austin pointed out that performative verbs only have performative force in the present tense. So, in examples (3a) and (3b) there is no warning, or ordering, but only descriptions of warning, or ordering. Also, subordinate clauses like (3c) are ruled out, as performatives must be in the main clause of the utterance.

(3) a I warned him just in time.
b He orders her to get to it.
c Everyone in the Party is hoping that I order Mr. Heseltine to resign.

Some linguists (see Lakoff 1968, and Ross 1970) argue that the underlying linguistic representation of every sentence contains, as its highest clause, a performative verb. This was called the *performative hypothesis*. So, the explicit representation of performative verbs is shown in (3), and the implicit form shown in (4a) and (4b) below.

(4) a Be careful.
b Take charge.

However, the performative hypothesis has been challenged on syntactic grounds by the linguists Anderson (1971) and Fraser (1971). They argued that utterances like (5) have performatives, with performative force embedded in subordinate clauses.

(5) I am sorry I order you to obey me.

Research interest in linguistics concentrated on utterances where the syntactic form did not match their illocutionary force. Examples are shown below in (6).

(6) Will you please open the window.
(= open the window)
It's cold in here.
(= close the window)

Stampe (1975) argued against the performative hypothesis saying that it was not linguistic rules that say orders are orders. He argued that it was the speaker's intention that makes an utterance what it is. Also, whether it will be understood, depends on the hearer's ability to infer that it was so intended. Stampe argues that conversational cues, such as *why don't you*, and *please*, will be present, and provide a clue to the speaker's intention. However, we argue that it is rare that this will be the case and Whittaker and Stenton (1988) have shown that such cues are not reliable. In most cases the speaker will have an estimate of the speaker's goals and plans on which to base inferences of intended illocutionary force. Cohen and Levesque (1985, 1987) have shown that the number of illocutionary forces will not be limited to say, four or five, or by the number of verbs that can be used performatively, but by the different kinds of intentions which a speaker might have, and by the hearer assuming the speaker is abiding by the Cooperative Principle[1]. Some illocutionary forces will have no names as shown in (7).

(7) Cut it out!

Searle (1969) extended Austin's work by formalising the structure of felicity conditions associated with a variety of speech acts. He classified all speech acts as incorporating one of five *illocutionary points*. The important distinction of Searle's work is that he covered *all* utterances, and not just those having explicit performative verbs such as, *promise*, or *declare*. The statement, "I'll be there," can be a promise event though it's in the form of a statement. There are five categories of illocutionary point:

[1]The Cooperative Principle was introduced in Chapter 1 and will be discussed in the next section

Assertives: commit the speaker (in varying degrees) to something's being the case – to the truth of the expressed proposition.

Directives: attempt (in varying degrees) to get the hearer to do something. These include both questions (which can direct the hearer to make an assertive speech act in response) and commands (which attempt to get the hearer to carry out some linguistic or non-linguistic act).

Commissives: commit the speaker (again in varying degrees) to some future course of action.

Expressives: express a psychological state about a state of affairs. This class includes acts such as apologising and praising.

Declarations: bring about the correspondence between the propositional content of the speech act and reality, as illustrated by the example of pronouncing a couple married.

Searle distinguished three components of an utterance: (1) *illocutionary point*, (2) *illocutionary force*, and (3) *propositional content*. The illocutionary point is one of the five categories above. The illocutionary force is the manner in which a proposition is expressed, e.g. *politely* or *impolitely*. A set of speech acts can have the same illocutionary point but different illocutionary forces. An example would be the different forces indicated by a polite request, and an interrogation, for obtaining information. Utterances involve propositions about topics, and therefore also have propositional content.

Hence, communicative acts are concerned with the space of possibilities of expressing intentions, in the world, through language. Each culture, or language, will have its own unique ways of expressing different communicative acts. There is an enormous amount of research in the area of illocutionary force, and Gazdar (1979) provides a useful bibliography. Now, we can move on to describe theories of cooperation, or the rules followed by agents while communicating various acts to each other.

In discourse people tend to follow certain pragmatic rules of behaviour of exchange. The philosopher of language, Grice (1975), proposed that conversational exchange and rational behaviour are governed by what he called the *Cooperative Principle* [2]:

> Make your conversational contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged. — Grice (1975, p. 45).

Grice described four categories, or special cases, of this principle, and gave examples of language and non-language applications. Grice called the four categories *maxims*, and pointed out that as long as participants in a mutual conversation assume that each other is adhering to the Cooperative Principle, meanings that are passed, without being said explicitly, follow as inferences from some maxim not being violated. The maxims are as follows:

[2] It is this principle, which enables us to explain the irrationality of the examples on restaurants, and bars, described in Chapter 1.

Quantity: (1) Make your contribution as informative as is required. (for the current purposes of the exchange)

(2) Do not make your contribution more informative than is required.

Quality: Try to make your contribution one that is true.

(1) Do not say what you believe to be false.

(2) Do not say that for which you lack adequate evidence.

Relation: Be relevant.

Manner: Be Perspicuous.

(1) Avoid obscurity of expression.

(2) Avoid ambiguity.

(3) Be brief (avoid unnecessary prolixity).

(4) Be orderly.

Grice pointed out that the maxim of Quality was much more important than other maxims where violating it is a moral offence, whereas with the others it is, at most, inconsiderate or rude.

It can be possible that situations arise where the speaker cannot conform to all the maxims at once. It may not be possible to say as much as is necessary in some situation, without saying things with no adequate evidence.

A speaker may make an utterance in discourse which disregards a maxim (e.g. Relation) even when the speaker wishes to convey a relevant statement. In situations where speakers do not obey the maxims, hearers are still prepared to believe that speakers are abiding by the Cooperative Principle. Otherwise, the hearer must assume that the speaker is behaving irrationally. Many people have misunderstood what Grice meant by his maxims. He did not strive to say that people do, or should, use discourse to conform to the maxims, but that assuming the maxims exist, we can explain utterances that seem illogical or irrational. This assumption, called the Cooperative Principle, demonstrates that people cooperate in dialogue as much as possible.

Some examples of the use of the conversational maxims are in conversations like the following shown in (8). In (8) the speaker uses rhetoric to show that what the speaker says is of course true. On the surface (8b) does not seem relevant, and therefore would break the maxim of relation. However, deeper insight shows that (8b) is meant to convey some obvious rhetorical fact that renders (8a) obviously true.

(8) a Speaker1: Is Margaret Thatcher in favour of military action?
b Speaker2: Is the Pope catholic?

The use of the Cooperative Principle can lead to irony and humour in discourse. Consider the dialogue in (9[3]). In (9a) Julie is walking through the cabin asking the passengers if they would like some tea. However, in (9b) the senior hostess indicates that Julie should/could be more polite about how she is doing the asking. This is indicated by speech intonation and stress. The senior hostess is using the Cooperative Principle to indicate her intention indirectly. In (9c) Julie turns the tables, pretends to take the senior hostess literally, and tells her that, yes, she would like a cup of tea.

(9) [The context here is an airplane where an Air Hostess (Steward) is asking
passengers whether they would like some tea as she walks through
the cabin]
a Air Hostess (Julie): Tea, Tea, Tea....
b Senior Air Hostess: Julie, ‘‘Would you like some tea?”
c Air Hostess (Julie): Yes, please.

As speakers expect hearers to accept the Cooperative Principle, the speaker can exploit it, and speak in a way that his/her behaviour must be interpreted according to the Principle. If the speaker's utterance seems irrelevant, the hearer will try to construct an inference sequence which will force the utterance to be relevant, or cooperative. The exploitation of the maxims is a technique by which utterances can be used to convey more than is literally indicated. Grice termed this phenomenon, *implicature*. For example, in the dialogue in (10) below, person2's utterance seems to violate the maxim of Relation.

(10) a Person1: Paul hasn't been around lately.
b Person2: His thesis is nearly complete.

However, if person2 assumes that person1 is obeying the Cooperative Principle, he must assume that person2's response is relevant, truthful, complete, and clear. Person2 may intend person1 to infer (i.e., Person2 may implicate) that (Person2 believes that) Paul has been working not as publicly busy as others, and/or in the evenings, and/or weekends, and/or some other explanation for Paul's apparent laziness which preserves the assumption that person2 is observing the maxim of relation.

Grice (1975, p. 49-50) characterised implicature more formally as follows. In, by, and when saying, “P,” a speaker implicates ‘q,’ provided that:

(1) the speaker is presumed to be observing the maxims, or at least the Cooperative
Principle

[3]This example is taken from a documentary about the life of Air Hostesses (Stewards) shown on British Television (Channel 4) during April 1991.

(2) in order to make the speaker's uttering "P" consistent with (1), it is necessary to assume that the speaker believes that 'q,'

(3) the speaker reflexively expects that the addressee is able to grasp intuitively, or actively infer (2).

Grice distinguished *conversational implicatures*, which must be capable of being inferred either intuitively, or by being worked out, and *conventional implicatures*, which can *only* be grasped intuitively, or by convention. Grice (1975) gave one example of conventional implicature, shown in (11) below.

(11) He is an Englishman; he is, therefore, brave.

There is an implicature of causality associated with "therefore" in 11. "Therefore" differs from "because" as "A because B" asserts that B is the cause of A, while "B, therefore A" only implicates, takes for granted, or presupposes, that B is the cause of A. Conventional implicatures also differ from conversational implicatures in that conventional implicatures do not crucially depend on assuming that the speaker is observing the Cooperative Principle.

Sperber and Wilson (1986) argue that the maxim of Relevance is central to the Cooperative Principle, and they tried to show that the implicatures Grice derived from violations of other maxims could be derived without the maxims. Following on from the work of Grice, it is possible to assume that if people are acting in accordance with the Cooperative Principle then that provides a basis for their dialogue being coherent.

Let us move on now to discuss the main theories of natural-language dialogue processing which utilise intention-based approaches to modelling dialogue.

4.2 Schank and Abelson

Schank and Abelson (1977) introduce a theory of the structure of knowledge and how it can be used for understanding. The main application of the theory is natural-language processing. First, they make the observation that people have the ability to understand many elements of stories without these elements being explicitly mentioned. Their theory is mainly concerned with text understanding and the representation of meaning in a text. They take the point of view that understanding a text is a problem of inference generation and control. Their theory can model attempts to discover implicit connections between sentences in text. Schank and Abelson (1977, p. 9) give the following example:

The policeman held up his hand and stopped the car.

They note that it is possible to create a picture in one's mind of a driver who steps on a brake in response to seeing the policeman's hand. None of these links are mentioned in the sentence. Hence, they infer that there must be knowledge other than that explicitly given, which we are able to bring forward to understand a situation or an utterance in a situation.

Scripts are introduced as a mechanism whereby knowledge is packaged for understanding so that people know correct behaviour in a particular context. Scripts are event sequences which store sequences of stereotypical events in certain situations. They provide connectivity of events. For example, scripts exist for situations such as restaurants. Each script has slots and requirements about what can fill slots. There are different tracks within scripts for specific manifestations of the script. For example, in the restaurant script there are three tracks: (1) fancy restaurant, (2) fast food restaurant, and (3) cafeteria. An example of a coffee shop track of the restaurant script is shown in Figure 4.1. During processing, script headers determine when a script comes into play,

Script:	Restaurant		
Track:	Coffee Shop		
Props:	Tables	Roles:	S-customer
	Menu		W-waiter
	F-food (Fast food)		Cook
	Check		Cashier
	Money		O-Owner
Scene 1:	Entering		
Scene 2:	Ordering		
Scene 3:	Eating		
Scene 4:	Exiting		
Entry Conditions:	S is hungry	Results:	S has less money
	S has money		O has more money
			S is not hungry
			S is pleased (optional)

Figure 4.1: An example restaurant script

and scripts are called up and then are instantiated from the input. Scripts are implemented within systems for understanding language. The system will have more confidence that it has invoked the right script when more input fits into slots in the script. Scripts are invoked on the basis of four headers:

Precondition Header (PH):	Script precondition mentioned in text (e.g. John was hungry).
Instrumental Header (IH):	Stronger predictions than a PH about associated context (e.g. John took the subway to the restaurant). Here the subway and restaurant are predicted.
Locale Header (LH):	This header refers to location of situations.

Internal Conceptualisation Header (ICH):	This header is relevant to cases where conceptualisations or roles in a script may occur in a text.

There are a number of types of script: (1) Situational Scripts, (2) Personal Scripts, and (3) Instrumental Scripts. Situational scripts are the general scripts we have been discussing before, such as going to a restaurant. Personal Scripts include scripts such as the customer in the restaurant wanting to make a date with the waitress. In effect, they are scripts in the mind of the actor. Other examples are flatterer, jealousy, pickpocket, and romancer. Instrumental scripts are those which are used in obtaining some goal. Examples are lighting-a-cigarette, starting-a-car, working-a-keypunch.

Schank and Abelson (1977) also discussed *plans* which can be in operation when scripts are not useful. Plans contain information about how actors achieve goals and they give relationships between events. Plans are useful for new or unexpected situations. A plan is intended to be a repository for general information that will connect events which cannot be connected by the use of available scripts, or by standard causal chain expansion. By finding a plan an understander can make guesses about the intentions of an action in an unfolding story, and use these guesses to make sense of the story.

The concept of plans is similar to one of intention sequences, describe in later chapters, which is a more flexible representation of sequences from which intentions of the agent/utterer can be inferred. The discussions in Chapters 5-8 will show that there is a strong link between Schank's work and ours. Scripts are event sequences which store sequences of stereotypical events and in future chapters we show that intentions are built into sequences in dialogue and that each intention could be considered an event. Scripts provide connectivity of events and sequences provide connectivity of utterances. The process of plan recognition leads to guesses about intentions of actors' actions. For example, in the dialogue below we can infer that the reason why Willa took out the Michelin Guide is because she wanted to locate good places to eat.

> Willa was hungry.
> She took out the Michelin Guide.

Plans are where scripts come from. They compete for the same role in the understanding process as scripts. They give explanations of sequences of actions intended to achieve goals. Sequences of scripts can be used to obtain *goals* as well as plans.

Chains of partial accomplishments along the path to the main goal are called *instrumental goals*. Some instrumental goals can be pursued without further planning. Eating is a goal and preparation-of-food is an instrumental goal to achieving this former goal. Such simple and stereotypical instrumental goals are called I-goals. Another type of instrumental goal called a Delta goal (or D-goal) is a building block in planning. The general goal of getting somewhere, or other, changes in other common states, are called D-goals. An example is where an agent is hungry and will go where there is food. Going to another location is a very general process. These types of goal usually tend to be subordinate to main goals. D-PROX is a change of proximity goal and D-CONT is a goal of gaining control.

According to Schank and Abelson's theory, there must be a set of methods, of which we are aware, for realising goals. Methods for realising goals almost always involve chains of instrumental goals, i.e. necessary partial accomplishments along the path to a main goal. An understander decomposes each main goal, heard, or inferred, into one or more I-goals and D-goals. Sets of possible actions called *Planboxes* are used to define goals. The process of understanding plan-based stories happens as follows:

(1) Determine goal

(2) Determine D-goals

(3) Analyse input conceptualisations for realisation of one of the planboxes that are called by one of the D-goals.

Although Schank and Abelson showed how their theory of understanding, using scripts, plans, and goals could be used in understanding story-based texts they did not show its use in other types of discourse such as natural language dialogue. However, later it will be shown how the theory described here can give useful information for dialogue modelling. In effect, the work in later chapters shows that there are scripts within natural-language dialogue and that these scripts are sequences of intentions. Schank and Abelson (1977) showed how scripts could be used for story understanding. A computer program called SAM was built to understand simple stories about script-based situations.

In summary, Schank and Abelson define a theory and computational model of discourse processing which recognises and represents goals and plans of people in the discourse. The theory utilises *scripts*, a representation of stereotypical sequences of actions by people. Now, we move on to discuss a theory of discourse processing centred wholly on the use of plans and goals.

4.3 Wilensky

Wilensky (1983) proposed a theory of plan-based understanding and plan-generation and developed a computer program called PANDORA (Plan ANalysis with Dynamic Organization Revision, and Application). Wilensky introduces his model with the following example:

> John was in a hurry to get to Las Vegas. However, he noticed that there were a lot of cops around, so he stuck to the speed limit.

John has the following goal, plan, danger, and modified plan on analysis of the dialogue example:

goal:	being at Las Vegas soon
plan:	high-speed driving
danger:	speeding ticket
modified-plan:	go more slowly

Wilensky noted some problems with Schank's approach to text processing. Not all stories, or texts, can be characterised as stereotypical sequences of events. In later chapters we take the same point of view. We argue that there are few stereotypical intention sequences but that a particular sequence, while recognised, will give us information about the dialogue. Wilensky's model of discourse involves text coherence based on goals and plans of participants. Wilensky outlines the requirements of a planner as follows:

(1) Plans are associated in memory with the goals to which they apply.

(2) Plans that are associated with a particular goal can be retrieved from memory by specifying that goal.

(3) The planner can project plausible, hypothetical futures from its knowledge of the present world together with its own tentative plans.

(4) Goals can be inferred based on the situation in which a planner finds itself.

(5) The planner must be capable of detecting the interactions between its goals.

(6) These interactions must be taken into account in its subsequent planning process.

(7) In addition to generating and modifying plans for goals the planner must be capable of evaluating alternative scenarios and abandoning some goals in order to secure others.

Wilensky argues that his planner model will function in complex, changing environments. The planner operation is as follows:

(1) The planner realises it should have a goal.

(2) A plan is suggested.

(3) The Planner checks if any new plans it has added interact with previous plans in a way relevant to the planner's goal structure.

(4) The Planner may realise its current plans. Additional goals may be encountered at this stage.

He proposes the following major components of a Planner:

Goal detector:	A mechanism responsible for determining that the planner has a goal. The goal detector notices situations that have arisen and that are relevant to the planner.
Plan proposer:	This component has the task of finding stored plans relevant to current goals.
Pragmatics:	Task is to test plans by building hypothetical world models of what it would be like to execute these plans.

Executor: The executor tries to carry out the sequence of intended actions described in the task network.

His computational model, Plan ANalysis with Dynamic Organization, Revision and Application (PANDORA) is a plan generation program that incorporates the model of planning described above to propose plans, and detect goals. PANDORA was applied in (1) the UNIX operating system domain in a program called UC which is reported to handle a wide variety of English requests for information about UNIX, and (2) A program called FAUSTUS (Frame Activated Unified Story Understanding System) is a frame-based (see Minsky 1974) implementation of Plan Applier Mechanism (PAM) (a story understanding program based on the model of explanatory reasoning given above), which looks for confirming and refuting evidence of frames indexed under input when input occurs. An example of a frame is hunger leading to the goal of satisfying hunger.

In summary, Wilensky's model of discourse processing emphasises the use of goals and plans to model the coherence of discourse. Although his work concentrated on goals and plans it makes no commitment to the generalities of natural-language discourse or dialogue. He concentrates more on specific plans and goals rather than recognising intentions and their sequences in general. Also, he considers plans and goals for their own end, having no consideration of the implications these may have for the dialogue being conducted. Next, we discuss an intention-based approach which uses the recognition and representation of intention for a number of purposes.

4.4 Cohen, Perrault and Allen

Cohen et al. (1982) view natural language question-answering as a special case of natural language dialogue, and they develop a theory of how the user's intent can be inferred. The work of Cohen, Perrault and Allen is the closest to ours and we believe it to be one of the most significant contributions to dialogue modelling. They observe that usually people do not say what they mean, and people expect hearers to infer intentions that they are meant to infer. On analysis of dialogues with a real natural-language system called PLANES they noted that (1) users expect the system to infer and respond to their apparent but unstated goals, (2) users' subsequent utterances depend upon the system's responses, and (3) the system should always keep track of what is communicated. They point out that users (1) expect more than just answers to isolated questions, (2) expect to engage in a conversation with a system where coherence is manifested in the interdependence of their unstated plans and goals, (3) expect that the system incorporate its own responses into its analysis of their utterances, (4) maintain these expectations even when it is obvious that the system is not a competent conversationalist, and (5) expect the system to react to unstated goals. Also, users expect the system to use indirect speech acts, and users make complicated requests using several utterances. Users expect the system to be aware of their reference failures, the failures of their presuppositions, and ensure that they are not misled. The system should expect that the users are dependent on it.

(D1-1) U: Which A7's has NORMU of greater than 100 hours
between april and may 1973?
2 S:

```
        ACTDATE   SERIAL#   PLNTYP
        304       4         AAF
        305       5         AAF
        305       9         AAF
```

```
3 U: No, How many NORMU hours greater than 100?
4 S: I could not understand the word NO, will skip it.
5 S: ((COUNT BUSER) (SUM NORMUNS)) = (4,644)
6 U: Which four?
7 S: [S translates utterance to an incorrect query. User does not allow
system to continue.]
8 U: Which 4 planes had NORMUs greater than 100 hours?
9 S: [program fails]
```

Figure 4.2: Example dialogue with the PLANES system

In effect, Cohen, Perrault and Allen have recognised that there is a coherence to utterance sequencing in dialogue. We discuss in future chapters the structure of such sequence sets and categories of utterance types. We argue that coherence of dialogue is determined by coherence of intention.

Examples of conversational activity in terms of (1) reaction to unstated goals, and (2) dependence on system responses are shown below in Figure 4.2.

In utterance D1-1, the user issued a typical "query" of the kind PLANES is equipped to analyse. The system replied with what seemed to be the correct answer, but did not tell the user what he/she wanted. The user wanted another column of output indicating the number of NORMU hours for each plane. The user tried to indicate in D1-3 that the system's response was somehow unsatisfactory, by typing "No."

In D1-5, PLANES printed the value of the function "number of planes" as four. In the next utterance the user asked for further information about the four planes. The system did not record that it had communicated the existence of four planes. This is an example of the dependence of user utterances on the system's responses. The same problem exists in the following fragment in Figure 4.3. User responses can depend on even the most standard phrases as the next PLANES interaction in Figure 4.4 illustrates.

The system's "message" in D3-2 indicates that it has deferred supplying the answer because too much information has been retrieved. The user is dissatisfied and requests further information. So, the user is indicating as in D3-2 that the system's response does not meet the goal of the previous question. In D3-2 it is noted that users' responses can depend on simple, standard phrases. In D3-1 the system did not reply as there were too many items retrieved.

Cohen et al. react against the standard analyser-retriever-generator architecture of language

D2-1 U: What parts were repaired on BUSER 48 between may 16 1969 and
may 17 1969?
2 S:

PARTNO:
522-0885-005

3 U: What was wrong with it?

Figure 4.3: Another sample dialogue with PLANES

D3-1 U: Give me the planes that flew more than 10 flights in 1970,
according to the number of flights.
2 S: (OUTPUT SCHEDULED, THERE WERE 60 ITEMS)
3 U: How many was that? How many flights did they each fly?

Figure 4.4: Standard phrases in PLANES

understanding systems. Conventionally, the analyser is used for translation into a query language; the retriever retrieves sets of records from a database, and the generator lists extracted records and information as natural language utterances. They propose a design centred on the basis of recognising the user's plans in dialogue and on planning helpful responses for those plans. They sketch a different picture by presenting dimensions against which language understanding systems can be compared. These are (1) versatility, (2) discrimination, (3) context dependence, and (4) single-mindedness and helpfulness.

(1) *Versatility* refers to a range of functions of systems, both linguistic and non-linguistic. For example, systems can have the following capabilities: answering questions from a static database, answering and asking questions, and giving commands to a robot.

For example, if a system which can only answer questions is given the utterance below it must interpret the utterance as a yes/no question, or reject it altogether.

There are 3 flights a day from Boston to Toronto

(2) Linguistic and non-linguistic *discrimination* is the degree to which a system can recognise what the user wants it to do from actions. It is the degree to which the system can recognise the intentions of users from their actions. For example, providing, accepting, correcting, and checking information are different types of intention. Discrimination is a major theme of our hypotheses and theory. We argue that it enables a system to better determine user intention and, in turn, discriminated intention sequences denote users' plans, goals and performance. Also, previous discriminated intentions denote context.

(3) Utterances may depend on previous utterances and are context dependent. Cohen et al. call "context" the shared beliefs of the system and user in the discourse, the intentions of the

participants, the medium of communication, the physical setting and general knowledge assumed by participants.

(4) With regard to single-mindedness and helpfulness a computer system should not always just say "yes" to yes/no questions but give some other information too. For example, if a system is given the utterance below, and it knows that there is such a record, but does not have access to it, and believes that the user wants to see it, a reply of "Yes" is undesirable.

U: Is there a record of AWM times for these POC's?
S: Yes.

U: Well, give it to me.
S: I don't have it.

A reply like the one below would have been better, and closer to what the user intended.

Yes, but you will have to do such and such to get it.

Cohen et al. recommend that the retrieval components and language generator of systems must be replaced by a process whereby the system determines actions based on user intentions and the intentions of others. The system should make decisions based on intentions not explicitly communicated by the user.

They argue that (1) intentions are identified with plans, and (2) that utterances are planned by speakers to achieve effects on hearers. To solve these problems Cohen et al. argue that the system must (1) engage in reasoning about how an utterance is being used (user's intentions), decide (2) what communication actions it should perform, and (3) how they should be performed. They point out that there is no direct mapping between utterance form and the action it is being used to perform. Also, they make the assumption that the speaker is a rational agent. There are two processes a system must embody to provide a basis for solutions to intention-discrimination problems: (1) Plan Construction, and (2) Plan Recognition.

Plan construction is a process where an agent can examine the consequences of sequences of future actions before executing them. Plan recognition is the observation of actions performed by an agent and the predicting of subsequent actions. Inferring a plan which the user may be following is Plan Recognition and involves beliefs about (1) the agents' beliefs, (2) conditions that are true at the end of an action, and (3) likely plans and goals of the agent.

They introduce the notion of Intended Plan Recognition which is where the system cannot simply infer and act upon what the user wants but must act/infer on what the user wants the system to "think" that the system wants. Intended Plan Recognition is conducted by the following rules:

SBUW(Do S A):	System believes that the user wants it to do A.
SBUW(SBUW(Do S A)):	System believes that the user wants it to think he/she wants it to do A.
SBUW(SBUW(A)) –> SBUW(SBUW(B)):	Inferring other goals such as B from A.

A means of controlling Intended Plan Recognition is based on (1) a heuristic that terminates inference chains that lead to mutually exclusive alternatives, and (2) assuming the speaker is a rational agent.

Indirect speech act recognition is the discovery of goals inferred during Intended Plan Recognition. Intended Plan Recognition produces goals the system is supposed to attribute to the user. For imperatives the hearer will believe that the speaker wants the hearer to do some act A. The system believes that the user wants it to do A. Plan recognition is the process of inferring the plan an agent may be following and comes from:

(1) Observing an utterance of a sentence.

(2) Assuming the agent wanted to do it.

(3) Inferring the agent wanted the typical act effect.

(4) Characterising the effects of uttering of sentences to be hearer beliefs about what the speaker wants.

For example, we can demonstrate plan recognition with the utterance, "Do you know where the Enterprise is?" (1) From syntax and semantics of the utterance the system recognises that the user intends it to believe that the user wants to know whether the system knows where the Enterprise is. (2) The system can infer that the user wants to know whether the system knows where the Enterprise is, then adopt the user's knowing whether the system knows as a goal. (3) Then the system can satisfy the goal by telling the user whether it knows or not.

Therefore, a system should (1) reason about others' actions, (2) plan utterances to achieve specific communicative goals from beliefs and intentions of the user, and (3) recognise the user's utterances as part of larger plans over several utterances. The recognition of user's utterances as intentions within sequences is a major goal of the work described in the following chapters. Thus versatility, discrimination and helpfulness can be obtained by an algorithm which follows the following cycle:

(1) Observe utterance.

(2) The effect of an act is a want of the user based on utterance mood.

(3) Intended plan recognition and shared beliefs give information on how actions fit into a plan achieving a user goal. If a plan cannot be uniquely satisfied create a system goal to see the user's goal.

(4) Create system goals for the user's intention.

(5) Use *private beliefs* to detect where the user's plan might fail.

(6) Adopt negation of some of the obstacles as goals for the system.

(7) Use *private beliefs* to construct a plan achieving the system's goals to overcome the user's obstacles; Depending on the goal the plan may include communicative acts such as questions to clarify users goals.

(8) Execute the resulting sequence.

(9) Go To step 1.

Cohen, Perrault and Allen discuss two implemented prototype systems: (1) a simple question-answering framework, and (2) a decision-support framework. The former system is a simple program for understanding, acting as an information clerk at a train station, and which expects users to want to board, or meet, trains. The latter system engages in dialogues about a display screen. Each system has the ability to (1) distinguish beliefs and wants of the user from its own, and (2) to process indirect speech acts. The program consists of the following components: (1) a parser of input using syntactic and semantic information to produce a literal interpretation, (2) a plan-recognition component having expectations about goals and when given a set of these, and observed actions from (1), infers a plan, (3) an obstacle-detection component which analyses a plan for steps that the user cannot perform easily, and (4) a plan construction component, which given a goal, plans a course of action. Plan recognition was effected by search through a space of pairs of plan fragments.

In summary, Cohen, Perrault and Allen's work argues that the coherence of a discourse can be modelled from the point of view of recognition of the intentions of the participants in the discourse. An important feature of their work is the principle that intention sequencing is a fundamental contributor to discourse coherence.

4.5 Allen

In Allen (1983), Allen revises and summarises his previous work. He argues that in order to build good Question Answering systems some of the characteristics of human conversation need to be built in. In following Grice's (1975) maxims, systems should (1) be able to provide responses that specify more information than is explicitly asked for, and (2) should not provide too much information, or provide information that is of no use. For example, in the dialogue below the clerk gives more information than is required by giving the departure location which was not requested.

(1) a patron: when does the Montreal train leave?
    b clerk: 3:25 at gate 7.

He points out that three major assumptions of his work are: (1) people are rational agents who are capable of forming and executing plans to achieve their goals; (2) they are often capable of inferring the plans of other agents from observing the agent perform some action; (3) they are capable of detecting obstacles on other agent's plans. Obstacles are situations which inhibit the goal achieving process of an agent.

He discusses the use of language itself as goal-oriented behaviour. Utterances are produced by speech acts or actions that are executed to have some effect on the hearer. A speech act, like any other action, may be observed by the hearer and may allow the hearer to infer what the speaker's plan is.

Often speech acts can explicitly describe obstacles to the hearer. In the previous example, the speaker, conveys to the hearer that the speaker needs to know the departure time of the train. There may be other obstacles in the plan such as: the speaker may also need to know the departure location of the train.

A clerk will have expectations that the patron has goals such as boarding or meeting trains. A query about train departure time, as opposed to train arrival, indicates that it is likely the patron has a plan of boarding the train. Assuming the clerk believes that the patron does not already know the departure location he believes that not knowing the location is also an obstacle in the plan. Thus a response is generated that overcomes both obstacles.

Allen shows that his point of view provides the basis for explaining the following aspects of helpful behaviour in dialogue:

(1) The generation of responses providing more information than required.

(2) The generation of responses to sentence fragments.

(3) The analysis of Indirect Speech Acts.

Allen shows how the model defined above can be used to understand the dialogues in Figure 4.5.

(2) a patron: The 3:15 train to Windsor?
b clerk: Gate 10.

(3) a patron: Do you know when the Windsor train leaves?
b clerk: yes/no

Figure 4.5: Train station dialogue

The model can explain how utterance fragments can be understood when the context is well-defined. In (1), neither the syntactic form, nor semantic meaning, indicates the intention of the utterance. Fragment information is sufficient to allow the hearer to infer what the speaker's plan is. (2a) is sufficient to identify that the speaker's goal is to board the 3:15 train to Windsor. An obstacle in the plan is not knowing the departure location and so (2b) is the response given.

In (3) we have a yes/no question about the hearer's knowledge. An answer of "yes" would be inappropriate. However, if a parent is seeing a child off at the station, and wanting to make sure that everything is arranged he/she might say (3a) intending to receive a yes/no answer. This would, then, be an example of an indirect speech act (see Searle 1969). His formulation of actions and plans is taken from existing work in problem solving (see Ernst and Newell 1969 and Fikes and Nilsson 1971). In these systems the world is modelled by a set of propositions that represent what is known about its characteristics. *Actions* which change the world can be viewed as parameterised procedures. Actions are described by (1) *preconditions*, which are conditions that must hold true before the execution of an action, and (2) *effects* which are changes the action makes on the world. If W is an initial world state, and G is a goal, then a Plan is a sequence of actions that transforms W to G.

Allen introduces the processes of Plan Construction and Plan Inference discussed in the previous section. If an agent A asks a question of another agent B which B answers then A has some goal. He/she creates a a plan (Plan Construction) which involves asking B a question whose answer will provide some information needed in order to achieve the goal. A then executes this plan, asking B the question. B interprets the question and attempts to infer what A's goals could be (Plan Inference). The inferred goals may include A's original goal as well as many subgoals. This is a description of the inference process between speaker and hearer in a dialogue. We argue that the sequencing of intentions in dialogue will portray the goals and subgoals in the dialogue.

Allen defines Plan construction as follows. Given some goal state two major tasks need to be done to produce a plan to achieve the goal.

Find a sequence of actions that will accomplish the transformation from an initial world state to a goal state.
Specifying the bindings for the parameters of actions in the constructed plan.

Hence, there is an interest in reasoning about the planning behaviour of other agents. In order to facilitate this reasoning about the planning process it is characterised by a set of planning rules and a control strategy. Planning rules are of the form:

> If agent A wants to achieve X, then he/she may want to achieve Y.

Allen introduces the following simple planning rule:

> If an agent wants to achieve a goal E, and ACT is an action that has E as effect, then the agent may want to execute ACT (i.e. achieve the execution of ACT)

One other rule of interest is used for reasoning about knowledge necessary to execute an action:

> If an agent wants to achieve P and does not know whether P is true, then that agent may want to achieve "agent knows whether P is true".

Plan inferring concerns the attempted (re)construction of some other agent's plan based on actions that agent was observed performing. Plan inference depends on the observer's knowledge of what constitutes a rational plan and his/her original beliefs about what goals the other agent is likely to have.

Possible candidates for the other agent's plan can be simulated as follows: (1) From expected goals the observer could simulate the other agent's planning process searching for a plan that includes observed action; (2) construct a plan from the observed action by applying the plan construction rules in reverse. The model uses the latter rule most of the time and sometimes the former rule.

The plan inference process is specified as a set of rules and a control strategy. Plan inference rules are of the form:

> If agent S believes agent A has goal X, then agent S may infer that agent A has goal Y.

Two example rules are:

(1) If S believes A has a goal of executing action ACT, and ACT has an effect E, then S may believe that A has a goal of achieving E.

(2) If S believes A has a goal of knowing whether a proposition P is true, then S may believe that A has a goal of achieving the content of P.

Plan inference process is a search through a set of partial plans. Each partial plan consists of two parts:

(1) Part 1 is constructed using the plan inference rules from the observed action and called an *alternative.*

(2) Part 2 is constructed using plan construction rules on an expected goal and called an *expectation.*

When mutually exclusive rules can be copied to part 1 or part 2 the plan is copied and one rule is applied to each copy. Each partial plan is rated as to how probable it is to be correct. The highest rated plan is selected for further expansion using inference rules. The rating is determined by using a set of heuristics that fall into two classes:

(1) those that evaluate how well-formed the plan is.

(2) those that evaluate how well the plan fits the expectations.

An example heuristic is:

(H1) Decrease the rating of a partial plan if it contains a goal that is already true in the present context

Allen goes on to build formulations of (1) belief, knowledge, and wants; (2) actions and plans, and (3) plan inference rules.

Allen uses a treatment of belief virtually identical to that of Hintikka (1963). Formalisations of belief exhibit one crucial property: what one agent S believes another agent A believes, has no logical relation to what S believes. Thus, S may believe A believes the world is flat, while personally believing that it is round.

They say that the belief operator allows us to consider actions and plans from another agent's point of view. This is approximated with the following axiom schema:

$$(BELIEVE(A, P \supset Q) \wedge BELIEVE(A, P)) \supset BELIEVE(A, Q)$$

Allen explains this formula is interpreted as, A infers some proposition Q, if it is believed that A believes there is sufficient evidence to infer Q. We find it easier to understand the above rule more clearly as follows:

> If A believes that $P \supset Q$
> and A believes P,
> then A will believe Q.

There are a number of other axioms which we shall not address here.

With respect to actions and plans, actions are grouped into families represented by action schemas. Action schemas consist of the structure shown in Figure 4.6.

name
set of parameters
sets of formulas{

preconditions:
  Conditions which should be true if the action's execution is to succeed.

effects:
  Conditions that should become true after the successful execution of the action.

body:
  A specification of the action at a more detailed level. May specify a sequence of actions to be performed, or a set of new goals to be achieved.
  }

Figure 4.6: Action schema

As with STRIPS (see Fikes and Nilsson 1971) many helpful responses arise because the hearer detects obstacles in the speaker's plan. The most obvious obstacles are those that the speaker specifically brings attention to by his/her utterance. Explicit obstacles are indicated by subgoals which are an essential part of the chain of inferences that the hearer makes when he/she infers the speaker's plan.

Obstacle detection is tackled by giving helpful responses to the user when the system detects obstacles in the user's plan. In the case of the utterance below the hearer must infer that the speaker has the goal of knowing when the train leaves.

When does the Windsor train leave?

Since the speaker does not know the departure time, 'hrs.', it is an explicit obstacle. Non-explicit obstacles also exist as in the following example:

A: Where is the nearest gas station?
S: on the next corner.

In this case A is carrying an empty gas can and asks S, who knows the station is closed, where S is not helpful. S did address the explicitly mentioned obstacle in the utterance.

In summary, Allen's modified description of his work serves as a clarification of the work by Cohen, Perrault and Allen described in the previous section. In this section Allen's description gives a clearer view of what planning a system would have to do to understand natural-language utterances. Allen's work has been updated in a number of ways. We can now go on to discuss the modifications.

4.5.1 Litman and Allen

Litman and Allen (1984) tackle the problem of modelling the plans of speakers in a task domain. They discuss clarification subdialogues and topic change. They note that models of plan-based analysis work well in task-oriented dialogues as long as the topic follows the task structure closely. A model is developed based on a hierarchy of plans and metaplans that accounts for clarification subdialogues. They point out that there is a large gap between Reichman's (1985) abstract model which addresses clarification subdialogues and topic switch and the actual processing of utterances. Their plan recognition model provides the link between processing of input and abstract discourse structure. One of the themes which we shall concentrate on later in Chapter 5 is the notion of clarification, or explanation subdialogues. We argue that the recognition of such subdialogues can be used to infer information about the status of the user.

Litman and Allen propose a new model of plan recognition which incorporates knowledge of both task and discourse structure. They show the use of a plan structure which stacks plans with more general ones piled on top of more domain specific ones. They claim this is similar to the manipulation of discourse topic hierarchies.

In addition to the standard domain-independent knowledge of task plans, they introduce knowledge about the planning process itself. For each dialogue a stack of plans is built where each plan on the stack refers to those below it with the domain-independent task plan at the bottom. For example, a clarification subdialogue is modelled by a plan structure that refers to the plan that is the topic of the clarification.

Plans are networks of actions and states connected by causality and subpart relationships. Plans are structured with *headers*, which are action descriptions that name the plan, *parameters of plan*, which are parameters of headers, *constraints*, which are assertions about the plan and its terms/parameters, *prerequisites/effects and decomposition.* The decomposition includes sequences of actions and sequences of subgoals or a mixture of both. For example, a simple plan schema for boarding trains is shown in Figure 4.7 below:

```
HEADER:       BOARD (agent, train)
STEPS:        do BUY-TICKET (agent, train)
              do GOTO (agent,depart-loc (train)
                             depart-time (train))
              do GETON (agent, train)
CONSTRAINTS: depart-station (train) = Toronto
---------------------------------------------
HEADER:       GOTO (agent, location, time)
EFFECT:       AT (agent, location, time)
---------------------------------------------
HEADER:       MEET (agent, train)
STEPS:        do GOTO (agent,arrive-location (train)
                             arrive-time (train))
CONSTRAINTS: arrive-station (train) = Toronto
---------------------------------------------
```

Figure 4.7: Sample plan schema for boarding trains

The constraint captures knowledge that the information booth is in the Toronto station. The plan consists of the three steps indicated. The prerequisites and effects are not shown. The second plan indicates a primitive action and its effect. Other plans needed would include plans to meet trains, plans to buy tickets, etc.

With regard to plan recognition a plan-recogniser recognises plans that lead to the production of the input utterance. An utterance either extends an existing plan on the stack, or introduces a metaplan to a plan on the stack. Otherwise the system tries to construct plans. Plan recognition follows the following algorithm in trying to incorporate observed actions into a plan:

(1) Direct match with a step in an existing plan on the stack.

(2) By introducing through inference processes a plausible subplan for a plan on the stack.

(3) By introducing through inference processes a metaplan to a plan on the stack.

(4) By constructing through inference rules, a plan, or stack of plans, that is plausible, incorporating domain-specific expectations about plausible speaker goals.

The central benefits of the approach are:

(1) A hierarchy of plans as opposed to single plans in previous plan recognition approaches to speech acts.

(2) Multiple approach provides insights into how much of the user's intentions must be recognised in order to respond appropriately. A central concern is recognising the user's intentions in full, as we will see later.

In summary, Litman and Allen's approach to discourse modelling incorporates a model of plan processing with a model of structure processing. Now, we can discuss another modification of Allen's work.

4.5.2 Hinkelman and Allen

Hinkelman and Allen (1989) discuss a new approach to processing speech acts and argue that current approaches do not account for the conventional aspects of speech acts. They use linguistic features (e.g. mood[4], verb form[5], and thematic roles[6]) to account for speech act interpretations of utterances. These are filtered using plan-based implicatures to eliminate inappropriate ones.

They critique three approaches to speech act processing: (1) The idiom approach is motivated by common phrases like:

(1) a Can you please X?
b May I X?
c I'd like X.

The approach does not decide whether an utterance has literal or non-literal intention in some context. It is too inflexible to choose the literal meaning. For example, in the case of the following utterance:

(2) Do you have a watch on?

it is not clear whether the intention is (1) what time is it?, or (2) a request to borrow a watch.

Hinkelman and Allen argue that the plan-based approach does not use language-specific information. It works by (1) modelling human problem solving, and (2) reasoning about other agents and inferring their intentions. It has nothing to say about the differences between these examples:

(1) a You want to cook dinner.
b You wanna toss your coats in there.

Hinkelman claims that (1a) and (1b) can be *requests* idiomatic to Hebrew and that American English and Hebrew have different meanings for both of these utterances. Neither is a request in British English, while (1b) is satisfactory in American English.

[4]Hinkelman uses 'Mood' to refer to the way a sentence is used. Examples are declarative, yes/no question or imperative (command)

[5]'Verb form' refers to the form of a verb, whether it is in the infinitive, simple present, simple past, etc.

[6]'Thematic roles' refer to components of representation structures for objects. For example, a structure for house may have roles such as kitchen, hallway, and front door, etc.

Hinkelman and Allen point out that descriptive approaches cover large sets of data but lack any theory. They do not handle language specific cases well. They claim to augment the plan-based approach with a linguistic component. Their algorithm works as follows: (1) Utterance interpretation by conventional rules gives candidate conventional interpretations; (2) interpretations are filtered by a plan-reasoner. They use compositional rules which associate linguistic features and partial speech act descriptions. Compositional rules consist of features (on the left hand side) and partial speech act descriptions (on the right hand side). Any structure matching the left hand side will be a speech act of a type on the right hand side. Each speech act is further defined in a knowledge base. We show below the proposed compositional rules of Hinkelman and Allen.

(1) General rule:

```
e.g. (? ADV please) =(1)=>

(DIRECTIVE-ACT)
```

The adverb "please" occurring in any syntactic unit signals a request/command, or other act in the directive class. In later chapters we will discuss this type of communicative act and call it a request for direction.

(2) Interrogative:

```
(S MOOD YES-NO-Q) =(2)=>

((ASK-ACT PROP V (REF))
 (SPEECH-ACT))
```

V returns the value of the specified slot of the input sentence in the proposition slot PROP filled with value of REF. The rule matches sentences whose mood is that of a yes/no question.

(3) Interrogative:

```
(S-MOOD YES-NO-Q  =(3)=>
    VOICE ACT
    SUBJ (NP PRO you)
    AUXS {can could will would might}
    MAIN-V +action)

((REQUEST-ACT ACTION V (ACTION REF))
 (SPEECH-ACT))
```

Interrogative sentences with modal verbs and a subject "you" are typically requests, or may be some other act: One of the members in the curly bracket set must be matched. Value function "V" follows a chain of slots to find a value. V (ACTION REF) is the value of the ACTION slot in the structure that is the value of the REF slot. So, "Can you ...? questions are interpreted as requests.

(4) Benefactive case:

```
(S MAIN-V +action
   SEM (? BENEF ?)) =(4)=>

((DIRECTIVE-ACT ACT V(REF))
 (SPEECH-ACT))
```

Benefactive sentences can be instantiated as requests, statements or questions.

Some rules are based on the semantic level. The semantic level is distinguished from the reference level which is the language of the knowledge base:

(5) Desire case:

```
(S-MOOD DECL  =(5)=>
      VOICE ACT
      TENSE PRES
      REF (WANT-ACT ACTOR !s))

((REQUEST-ACT
   ACT V (DESID WANT-ACT REF))
```

Any way of stating a want or desire of the agent. This rule will match any sentence that can be interpreted as asserting a want or desire. e.g. I need a napkin.

There are also a few general rules to fill in information about a conversation. These rules are given below for filling in information about the conversation. In applying the rules, interpretation of a sentence begins by finding rules to match it. The categories in each rule must match the syntactic structure given. If a rule matches, then structures on the right hand side of the rule are filled out, and become partial interpretations.

(6) General information filling rule:

```
(?) =(6)=> ((SPEECH-ACT AGENT !s))
```

Utterance of any syntactic category maps to speech act with agent specified by !s

```
(?) =(7)=> ((SPEECH-ACT HEARER !h))
```

This rule fills in the hearer.

```
(S MOOD DECL
   VOICE ACT
   SUBJ (NP HEAD i)
   MAIN-V +performative
   TENSE PRES)                 =(8)=>  V(REF)
```

Explicit performative utterances (Austin 1962) are declarative, active, utterances whose main verb identifies an action explicitly. The main verb identifies the action in explicit performative utterances.

The second constraint on speech act recognition is plan-based reasoning. Hinkelman and Allen justify the role of plan-based reasoning as follows:

(1) Eliminates speech act interpretations proposed by the linguistic mechanism if they contradict beliefs and intentions of the agent.

(2) Elaborates and makes inferences based on remaining interpretations.

(3) Proposes interpretations of its own when context indicates what the speaker may do next.

(4) Provides a competence theory motivating conventions described by them.

The entire processing cycle of their system is as follows:

(1) Build and merge parses of different speech act interpretations.

(2) Implicature module computes implicatures and discounting ones in conflict with contextual knowledge.

(3) Plan reasoning is invoked to identify speech acts only if remaining ambiguity interferes with planning or no completely plausible interpretations remain.

(4) Plan reasoning proceeds to plan a response or elaborate the interpretation in (3).

In summary, Hinkelman and Allen's approach to discourse modelling incorporates models of both linguistic and intention processing. One significant factor about their approach is that they operate linguistic processing first, and then operate plan-based processing. It might be a better approach to interleave these processes rather than to consider them in sequence. The problem with the approach as it stands is that there is no utilisation of the planning for parsing future utterances in the input. In later chapters we make the argument that it is very difficult to recognise intention on the surface, and that representations of intention sequences may be needed to recognise future intentions in the input. Now, we will discuss another plan-based approach to discourse modelling.

4.6 Appelt

Appelt (1985) introduces a theory of language generation and communication based on planning. He treats language as a set of actions available to agents that affects the mental states of other agents. Planning of actions appropriate for a given situation necessitates consideration of several crucial elements:

(1) different goals satisfied by utterances

(2) knowledge of the hearer

(3) general knowledge about the world

(4) constraints imposed by the syntax of language

He developed a planning system called KAMP (Knowledge And Modalities Planner) which is a hierarchical planning system that uses a procedural network (see Sacerdoti 1977). KAMP has a hierarchy of linguistic actions which is a hierarchical design for utterance planning. The hierarchical approach is selected because it provides for, (1) separating the planning of domain-level goals and actions, and planning of low-level linguistic actions, as well as for, (2) intermediate levels of abstraction that facilitate the integration of multiple goals into utterances. The hierarchy of linguistic actions used by KAMP is shown below:

Illocutionary Acts:	The highest level of linguistic actions are called illocutionary acts which are speech acts like *informing* or *requesting*.
Surface Speech Acts:	These are abstracted representations of sentences with particular syntactic structures. Specific linguistic knowledge is used here.
Concept Activation Actions:	These entail the planning of descriptions which are initially believed by speakers and hearers to refer to objects in the world.

Utterance Acts:	Here, specific words and syntactic structures are chosen to realise the descriptions chosen for the concept activation actions. These syntactic structures have to be compatible with the sentential syntactic structures selected when the surface speech act is planned.

Appelt accepts the hypothesis that agents have mental states that causally determine their actions. The research demonstrates how particular mental states account for particular actions.

He proceeds by constructing a formal, computational theory, along with a system embodying the theory, in which it is possible to represent mental states, and provides an explicit mechanism where these states are manifested in actions.

He uses a possible-world's-approach semantics for a modal logic to represent knowledge needed by a cooperative agent to participate in dialogues. The possible-world's semantics approach and its integration into a first-order logic system was developed by Moore (1980).

KAMP embodies an entire theory of communication and has been applied in the assembly and repair of complex elector-mechanical devices. The user of the system is a novice seeking assistance. The goal is to account for how agents manage to intentionally affect the beliefs, desires and intentions of other agents. Theories of language understanding make much use of the fact that the speaker is behaving according to a coherent plan.

In summary Appelt's approach is to model a discourse using a model of the coherence of a speaker's plan. Our approach is similar to Appelt's in that we also make use of the fact that a speaker is behaving in accordance with a coherent plan. The basis of our work on intention sequences in future chapters assumes that the speaker has a coherent plan and is asking questions about how to conduct that plan. We will now discuss another plan-based approach.

4.7 Carberry

Carberry (1989) provides what she calls a pragmatics-based approach to ellipsis resolution. The theory uses (1) discourse expectations, and (2) focusing heuristics to facilitate recognition of users' intent in elliptical fragments. This is done by recognition of aspect of task-related plan, and conversational discourse goal. She stresses discourse content, and goals, rather than just precise representation of the preceding utterance.

We agree with Carberry that it is goals and intentions which determine dialogue information rather than just straight syntactic and semantic information. Carberry says that (1) inferred task-related plans, and (2) discourse goals are more important than syntactic structure, or semantic content of preceding utterances. We hold a different view, as we believe that both are of equal importance in determining elliptical information and dialogue modelling. She shows the following two examples (see Carberry 1989, p. 76):

(D1)

Speaker1: "I want to cash this check. Small bills only, please."

(D2)

Speaker1: Who are the candidates for programming consultants?

Speaker2: May Smith, Bob Jones, Ann Doe have applied for the job.

Speaker1: Tom's recommendation?

Speaker2: He thinks Bob Jones and Ann Doe have the necessary background and should be invited for an interview.

Carberry says that in D1 the goal of speaker1 is cash-check and the plan involves the sub-plan of getting-nice-distribution of paper money. However, we believe that the syntax and semantics of sentences is just as important as discourse goals in determining the intention of these utterances. In D2 she says that the speaker's goal is hire-consultants and the plan contains a subplan of identify-best-applicant. Again, we believe that syntax and semantics is just as useful in showing intentions of speakers.

Carberry shows that previous work relied too much on intersentential ellipsis processing by syntactic and semantic strategies. She points out that is is not good enough just to determine task-elements but discourse goals must be recognised too. Mann et al. (1977), Reichman (1978) and Pollack et al. (1982) hold this view too. Discourse goals are indications of what the user wants, i.e requesting-information, or seeking clarification. Again, we agree with Carberry, that it is the intentions of speakers that should be recognised in dialogue, rather than syntactic, surface phenomena. Carberry's processing framework involves coordination of (1) discourse expectations, (2) inferred beliefs, (3) information seeker's inferred task-related plan, and (4) focusing strategies.

There are three approaches to ellipsis processing (1) syntactic representation, (2) semantic representation, and (3) expectations from the preceding discourse. Carberry points out that syntactic approaches as in Hendrix et al. (1978) and Weischedel and Sondheimer (1982) involve substitution and expansion where an elliptical fragment is substituted for a syntactic constituent in the preceding utterance. The syntactic representation of the preceding utterance is expected to accommodate the fragment. However this will not always work as in the example D2 above. Syntactic approaches can also involve transformation operations which map questions into answers and statements into questions.

Semantic approaches maintain a semantic representation of the previous utterance as a pattern that suggests slots where a fragment may provide a filler/substitution. Such strategies are described in Waltz (1978) and Carbonell (1983). The problem with these approaches is the extensive use of case frames and the approach was inadequate for handling elliptical fragments that rely on assumed communication of underlying tasks.

Pragmatic approaches involve (1) task plan, and (2) discourse goal recognition and processing. This approach is useful as it takes knowledge from the dialogue, rather than just relying on precise representations of preceding utterances. Allen and Perrault's (1980) work is an example of this approach.

While Carberry has mainly concentrated on dialogues where information-seekers are attempting to construct a plan for accomplishing a task, she argues that her principles can be extended to other kinds of dialogues. Carberry obtained dialogues from (1) a radio call-in show providing advice on investments, (2) interactions with the REL (see Thompson, 1980) natural language interface to a ship data base, and (3) student advice sessions, for her hypothesis and strategies.

She does not consider the use of clue words (e.g. "Now", "As I was saying"), that contain information about the structure of discourse. Clue words are a central part of the work by Reichman (1985). Carberry admits this, and says that they could be added.

Carberry says that (1) speaker's task-related plan, (2) the speaker's inferred beliefs, and (3) the anticipated discourse goals of the speaker are needed for ellipsis processing. A cooperative hearer should assimilate the preceding dialogue, infer the underlying task-related plan, and focus on the aspect of the task on which the information-seeker is centred. Carberry says that the task-related plan is an essential part of ellipsis interpretation. Again, we agree with Carberry's point of view that the determination of intention is useful for dialogue modelling. We shall see the use of intention sequencing for user modelling in later chapters.

With regard to shared (or mutual) beliefs she says these are required for processing intersentential elliptical fragments. Shared beliefs include (1) presumed a priori knowledge of the domain, and (2) beliefs derived from the dialogue itself. Shared beliefs can be useful in determining the intention of a user's utterance in context. For example, the same utterance may indicate surprise or a question as shown below

IS[7]: When does CS400 meet?
IP: CS400 meets on Monday from 7pm until 9pm.

Discourse expectations are regularities in dialogue that can be used for comprehending utterances. Existing discourse structure establishes expectations about appropriate next conversational moves. Carberry's analysis of naturally occurring dialogue indicates that expectations about next steps play a major role in comprehending elliptical fragments. Such expectations will be studied in future chapters and expectations of future communicative acts from previous ones is a major theme of our thesis here. Also, she notes that plan recognition techniques and focusing knowledge are very important in ellipsis processing. Focusing is necessary to determine which portion of a plan an utterance refers to.

At the beginning of processing, Carberry's system recognises fragments and begins ellipsis understanding. The processor has a model of the preceding dialogue containing (1) a context tree, (2) a stack containing the discourse expectations for speaker, and (3) a belief model representing the

[7]IP stands for Information Provider,
and IS for Information Seeker.

speaker's beliefs. The top element of the discourse stack represents the most immediate discourse expectation for the speaker and suggests potential discourse goals that the speaker may pursue.

Carberry proposes the hypothesis that shared beliefs are necessary for ellipsis processing. She shows the following example (Carberry 1989, p. 80):

IP: "Do you want to take CS360?"
Response 1: IS: "Are you asking me if I want to take it next
semester?"
Response 2: IS: "Will it be offered next semester?"
Response 3: IS: "Next semester?"

In Response 1 the speaker's utterance indicates that it is an attempt to clarify the question posed by the IP. In Response 2 the speaker's utterance requests information about CS360 in order to formulate an answer. In Response 3 the speaker's utterance is an elliptical fragment that could produce several different interpretations. It may be an attempt at clarification, it may be a request for information or it may be an expression of surprise. Therefore, the speaker's discourse goal is not indicated explicitly and the processing of shared beliefs is necessary.

Carberry defines discourse goals to be what the speaker is trying to do in making some utterance. The concept of discourse goal is close to the discourse segment purposes of Grosz and Sidner (1986). Carberry gives some examples of discourse goals found in analysing different dialogues. Here are some of them:

Provide-for-assimilation: Seeker provides information pertinent to formulation of his/her task plan.

e.g.
IS: I want to get a degree
CS Major

Obtain-information: Seeker requests information relevant to constructing the underlying task-related plan, or relevant to formulating an answer to a question posed by the IP.

e.g.
IS: Is CS360 being offered this fall?
IP: Yes
IS: The instructor?

Express-surprise-obtain-corroboration: Seeker expresses surprise regarding some proposition P and requests elaboration on, and justification of it.

e.g.

IS: What time does CS360 meet this fall?

IP: Monday, Wednesday, and Friday at 8am.

IS: Who is teaching it?

IP: Dr. Smith.

IS: At 8am?

Seek-identify: Seeker is unable to satisfactorily identify the referent of an item in IP's utterance and requests help from the IP in doing so.

e.g.

IS: What is Dr. Smith teaching this fall?

IP: CS360

IS: The course in architecture?

Seek-clarify-question: Seeker requests information relevant to clarifying a question posed by the provider.

e.g.

IP: Do you want to take CS105?

IS: Next semester?

Suggest-answer-own-question: Seeker suggests an answer to his/her question.

e.g.

IS: What course should I take during winter session?
CS370?

Answer-question-with-restrictions: seeker answers a yes-no question, providing restrictions on the relevant underlying task-related plan.

e.g.

IP: Would you like to take CS360?

IS: At night with Dr. White.

There are two other discourse goals not linked specifically with ellipsis but which are important in understanding elliptical utterances:

Accept-response: Provider has responded to seeker's request for information; seeker accepts the response.

Accept-question: Provider has posed a question to the seeker who accepts the question.

We shall see much similarity between these discourse goals and intention categories mention in our work later in Chapter 5.

Discourse goals also serve as discourse expectations. These expectations play a major role in comprehension of elliptical fragments. When the seeker makes an utterance he/she is attempting to accomplish a discourse goal. The goal may establish expectations about what the seeker will do next. If the seeker asks a question then he/she may want to expand that by clarifying the question. One of the central themes of our work is using communicative act sequences to determine expectations about follow-up utterances in a dialogue.

A discourse stack contains (1) expectations about the seeker's discourse behaviour, and (2) the semantic representation of the utterance. Carberry says that elliptical-fragment understanding relies on discourse expectations that are available on the stack. She proposes the following stack processing rules:

SP1: If the provider asks a question of the seeker with the discourse goal of getting information, Answer-question and Accept-question are pushed onto the discourse stack.

SP2: When the provider answers a question posed by the seeker Accept-response is pushed onto the discourse stack.

SP3: When the seeker actively pursues a discourse goal, the discourse goal is pushed onto the discourse stack.

SP4: When the seeker's utterance does not pursue a goal suggested by the top entry on the stack, this entry is popped from the stack.

If the provider asks, or answers, a question, SP1 or SP2 respectively apply. If the Seeker makes an utterance, then rule SP4 is applied until it fails, and then SP3 is applied.

Discourse expectation rules indicate suggested speaker discourse goals that the Seeker might pursue. Here are some examples of discourse expectation rules that Carberry gives:

DE1: The discourse expectation Accept-question suggests the following ordered set of discourse goals for the seeker:

(1) Seek-confirm. (Provider provides question where the Seeker may confirm question.)

(2) Seek-identify. (Seeker identifies referents in question.)

(3) Seek-clarify-question. (Clarification of the question.)

(4) Express-surprise-question. (Seeker may be surprised about the question.)

DE2: The discourse expectation Answer-question suggests the following partially ordered set of discourse goals for the seeker:

(1)

Answer-question.
(Answering the question directly or indirectly.)
Answer-question-with-restrictions.
Answer-question-suggest-alternative.

(1)

Suggest-answer-question.
(Seeker may suggest one/more possible answers.)

(1)

Obtain-information.
(Seeker may ask for information to provide an
answer.)

DE3: The discourse expectation Provide-for-assimilation suggests the following ordered set of discourse goals for the seeker:

(1) Provide-for-assimilation. (Seeker will elaborate on plan by providing more information.)

(2) Obtain-information. (Seeker will seek information in order to achieve his/her objective of completing his/her task.)

Of these three expectation rules we will discuss only the third with respect to our theory. The first case is not catered for as we are not concerned with cases where the system asks questions to clarify user questions. We are not interested in the second case where the user provides answers for his/her own questions. In fact, in experiments we conducted, described in Chapter 8, there were no cases of subjects providing answers for their own questions. Finally, the third expectation rule has more in common with what we will be discussing in future chapters. It includes cases where the user elaborates on his/her plan and where the user seeks more information with respect to his/her goal. We will see numerous examples of communicative act sequences to achieve goals in Chapter 8.

Associated with each discourse goal is one or more discourse rules. The rules are used to apply factual and processing knowledge to analyse a fragment and determine whether it can be understood as pursuing the goal that the rule is associated with. Here are some examples:

Rule DG-seek-identify-1: Check that the following conditions are satisfied:

(1) The seeker's fragment terminates in a "?".

(2) The fragment highlights a component of the seeker's underlying task-related plan and matches a description used in the utterance by the provider that is closest to the top of the discourse stack.

(3) It is mutually believed that the seeker might not know the referent of this description.

If these conditions are satisfied then interpret the fragment as seeking further identification of the high-lighted plan component.

Rule DG-seek-identify-2: Check that the following conditions are satisfied:

(1) The seeker's elliptical fragment terminates in a "?".

(2) The fragment highlights a component of the Seeker's underlying task-related plan and this component is referenced by a description D in the utterance by the provider that is closest to the top of the discourse stack.

(3) It is mutually believed that the Seeker might not know the referent of the description D.

(4) It is mutually believed that the seeker does not believe that the fragment and the description D refer to different entities.

For plan analysis a tree structure called a *context model* is used to represent task-related plans inferred for the Seeker from the preceding dialogue. Nodes in the tree represent a goal that the seeker has investigated achieving and is the descendant of a higher-level goal whose associated plan contains the goal represented by the child node. The current focus of attention marks one node in the context model. An active path marks the path from the root to the current focus of attention.

Carberry says that elliptical fragments from input utterances highlight propositions in the Seeker's plans. The matching of fragments with plan elements is called *association.* A constant fragment can only associate with terms whose semantic type is the same, or a superset of, the semantic type of the constant. Each term in a plan has a limited set of valid instantiations. A constant associates with a term only if the system's beliefs indicate the seeker might believe the constant is one of the term's valid instantiations.

With regard to retaining context Carberry notes that humans retain context for interpreting intersentential ellipsis. Carbonell (1983) conducted an informal poll where he found this was the case. Carberry points out that she has found the same results in other domains. When an elliptical fragment is associated with a component of the task-related plan, the context established by the dialogue should be used to replace information deleted from the fragmented utterance.

Carberry represents the underlying task-related plan inferred for the seeker in a context tree. The set of nodes along the path from the root of the tree to the current focus of attention form a stack of goals and plans. The top of the stack is the most recently considered subgoal in the current focused plan. *Active nodes* represent the established global context, and the propositions in these nodes have the effect of restricting the instantiation of variables in their ancestor nodes in the context tree. By using propositions along an active path restrictions on variables in ancestor nodes can be retained when the focus pops back to them. Also, the nodes on the path from the root of the context tree to the nodes at which fragment elements associate with plan elements represent the new context derived from the seeker's fragmentary utterance.

Within the seeker's task-related plan there are a number of components with which a fragment might associate. Carberry notes that participants in information-seeking dialogues move around in predictable ways in discussing aspects of plans. This structure has led to the development of focusing rules that predict how a speaker may shift his/her focus of attention. She points out that elliptical fragments in themselves contain little information. We agree with Carberry's point about seeker's moving around in predictable ways in dialogue. This is a central theme in Chapter 7.

Shifts in focus occur each time one moves from goals to subgoals but these can represent small and large shifts in attention. She introduces the notion of "focus domains" which group together goals with the same level of focus. Moving from a goal to another in one focus domain will be considered a smaller shift in attention than moving from a goal in one focus domain to one in a different focus domain. Carberry gives the following two rules for focus understanding:

(1) Rule-current-focus-domain:

> Associations with elements in the current focus domain, or within expansions of the plans associated with goals in the current focus domain, should be preferred.

(2) Rule-higher-plans:

Associations with elements in the focus domains of higher level plans whose expansion led to the current focus domain should be preferred over other associations with elements that do not occur in an expansion of a goal in the current focussed domain.

Carberry's ideas are implemented in a computer program which gives information about the courses, policies and requirements for students at a university. The following information is input to the system:

(1) A Semantic representation of an elliptical fragment.

(2) A Context tree representing system's beliefs about the speaker's plans.

(3) A Belief model representing the system's beliefs about the seeker's beliefs.

(4) Initial discourse stack containing the system's beliefs about the seeker's expected discourse behaviour.

In summary, Carberry's approach to discourse modelling is to recognise the intentions of participants within the discourse and to build a model of those. Her approach to building relations between intentions has much in common with what we will discuss in later chapters. The problem with Carberry's work is that a strong emphasis on complex plan-goal analysis places the system into too strong a depth-first mode. This causes problems because users can conduct business in breadth-first mode with an immediate new plan. The depth-first processor would be lost in this case. Also, although Carberry's work gives a good description of plan sequences in dialogue, it is not clear that there are any new ideas here, rather the application of older ideas.

4.8 Summary

In summary, intention-based approaches to discourse modelling concentrate on recognising and representing the intentions of participants in a dialogue. The theories and computational models argue that by recognising the intentions of participants it is possible to model discourse coherence. A number of the approaches have produced models of the existence of plans and goals in dialogue and how these plans and goals occur in sequences. Some of the approaches also incorporate semantics and structural information. Finally, we can discuss an approach which attempts to integrate semantics, structure, and intention.

4.9 Pustejovsky

Pustejovsky (1987) discusses an approach to discourse modelling from the point of view of modelling both structure and intention. He points out that much of the work in Artificial Intelligence with regard to discourse analysis seems to be reinventing the wheel failing to take note of work done in the past in linguistics, philosophy and by psychology. He does not mention which work but he is probably referring to some of the research we mention in Chapters 2, 3 and 4.. He says that some of the work has added new and complex dimensions to the study of discourse analysis including speech act theory. He points in particular to the work of Cohen et al. (1982) and Wilks and Bien (1983). Also, he mentions the work by Webber (1978), Grosz (1978a), Grosz (1981), Sidner (1985), and Reichman (1985). One common feature of these approaches is that they are process oriented models of discourse processing as opposed to competence models as in Chomsky (1965).

Pustejovsky provides a discussion of conversational implicature, and how it may be used in discourse analysis. He shows that the use of cue words like "but" indicate that there is some conversational implicature in place. In this case it is the conventional implication that there is some contrast between the elements each side of this lexical unit. He points to Grice's (1975) Cooperative Principle of conversation, governing a person's behaviour. This principle subsumes the Gricean maxims of quantity, quality, relation and manner.

Pustejovsky discusses three contemporary paradigm's which he calls (1) Structural Analysis, (2) Goal Recognition, and (3) Model Theory. Structural Analysis approaches seek to identify structural elements in discourse. Examples are *topic*, *focus*, *discourse moves* and *context spaces*. These approaches centre upon how structure influences interpretation. Proponents of this type of approach are Grosz (1978, 1981), Webber (1978) and Reichman (1985). Webber and Grosz aim at identifying contexts where discourse anaphora could be confined. Topic and focus denoted search spaces for anaphora. Reichman said that the purpose of discourse analysis is to identify "a conversations deep structure in terms of the structural relations between the discourse elements." She points to conversational moves or the representation of the various kind of semantics and logical relations that can hold between utterances of a discourse. For example, a conversational move could be 'support' or 'challenge'. A *context space* is defined by Reichman to be a constituent which is hierarchically related to other constituent in a discourse. A discourse is partitioned into a set of hierarchically related constituents which package pieces of discourse into separate units.

The Goal Recognition approach is one where the representation recovered from an utterance or text involves information about discourse and is deeper than models found in the structural paradigm. Narrative form, coherence, and story understanding (Schank and Abelson 1977, Hobbs 1982, and Wilensky 1983) and work on speech acts and intention (Cohen and Perrault 1979; Allen and Perrault 1980) all take this approach. This is the approach which is most related to what we will discuss in later chapters. However, although our work is concerned with intentions in discourse we do not argue its primacy, rather that it is a fundamental component, and does not rule out other components such as structural analysis.

The Model Theory paradigm is one where formal models are developed for discourse. There are discourse representation theories by Kamp (1981) and Heim (1982) and situation semantics by Barwise and Perry (1983).

Pustejovsky outlines what he believes to be problems with existing approaches. Some researchers such as Reichman (1985) have coined the phrase *moves* for the role of utterances in discourse whereas others, like Hobbs (1982), have termed these *coherence relations*. He points out that researchers such as Alterman (1985) ignore the role of discourse moves altogether. The point Pustejovsky has against this is that without structural cues provided by the discourse or text, like topic and focus, it is impossible to recover pronoun reference and deictic items.

Also, Pustejovsky points out that although Sidner (1985), Grosz (1983), Webber (1980), and Reichman (1985) discuss structural environments for discourse anaphora resolution there is no analysis of the deeper relations between discourse entities. On the other hand the model theoreticians and formal theoreticians like Kamp (1981) do not, according to Pustejovsky, deal with inferencing, goal recognition, planning or computer modelling. Kamp (1981) does not look at the structure of text and ask questions about coherence. Heim does not extend her model to the deep coherence relations addressed by Hobbs (1979) and others.

Pustejovsky provides an integrated theory of discourse analysis with different levels which he hopes will integrate structural, plan-based, and formal approaches. He proposes the following questions:

(1) What are the levels of analysis for Discourse Analysis?

(2) What is the unit of analysis for Discourse Analysis?

(3) How does Discourse Representation (DR) affect interpretation?

(4) If DR is not the final semantic interpretation, then what is?

Pustejovsky attacks the problem of building a structural component of a discourse model and he formulates the following structural relations:

(1) A conversational move (CM), e.g. support, interrupt, challenge, etc.

(2) The parameters that act to constrain the evaluation of a discourse object, e.g. topic, focus, theme, rheme.

(a) John can open Bill's safe.
(b) He has the combination,
(c) which he got from Mary.

Figure 4.8: Sample monologue

(3) A textual directive (cohesion relation), e.g. elaboration.

The first relation is one which will concern us most in this work. We will be interested in the recognition of conversational moves, which we call communicative acts, and the sequences in which they arise. He says that the complexity of a discourse can be characterised by the possible turns available at any stage. We agree with this claim which is a major theme of our thesis. The simplest structure in this view will then be a directed monologue where the goal of the speaker is detected by the manner in which the discourse is structured. This does not mean that directed monologues lack complexity. Several basic types can be distinguished, some of which are simpler than others. Here are some of the examples of directed monologue types that Pustejovsky gives:

(1) Enumeration

(2) Elaboration

(3) Definition

(4) Description

(5) Proof-form

(6) Narrative

He gives an example of an elaboration monologue as shown in Figure 4.8 below.

There are two elaborations here: (1) The explanation given in (b), and (2) the descriptive elaboration given in (c).

He moves on to discuss the more interpretative aspects of discourse structure. Relations such as enablement, causation, or explanation are not uniquely or deterministically recoverable from the syntactic structure alone. He says, the major contribution to the semantics of an utterance is the intention of the speaker in performing a speech act. This is the *speaker's goal* and is what Grosz and Sidner (1986) call Discourse Purpose or DP. In later chapters we shall call this intention.

Pustejovsky decides that the input to the discourse processing process should be a Logical Form (LF) rather than surface structures. He assumes that any adequate model of analysis should represent the distinctions between properties of the discourse. This level of discourse representation should be distinct from the purely syntactic or semantic interpretation of the utterance. We do not necessarily agree with Pustejovsky here, in principle. Many who take the connectionist approach to natural-language processing (see Barnden and Pollack 1990, and Sharkey and Sharkey 1987), would argue that there is a continuous representation across the properties of a discourse. However, in practice, in chapter 5 where we show the implementation of a discourse structure we represent the intention sequences in a separate structure. A discourse is described by:

LF -> $\{_{DR}$ $[_{M1}$ $CF_i]$ ...$[_{M2}$ $CF_j]\}$ IF

LF (Logical Form) represents the cohesion relationships between clauses, the domain of topic and focus, and moves associated with the utterance. LF is seen as feeding Discourse Representation (DR). Cohesion relations relate clausal relations (CF) which are then bound to a particular move (M). Discourse Representation (DR) is the level of representation of an utterance derived from logico-syntactic form (LF). A DR is associated with one or more moves in the larger discourse structure.

At this level the following are interpreted:

(1) The bindings between discourse anaphora and deictic terms and their antecedents, i.e. topic/focus mentioned above. For example, if an Noun Phrase is in focus, the system allows for pronominal reference within a wider textual context than if it is not.

(2) The relationship between moves in the context of higher order structures, i.e. how these moves combine to make up a story-level or narrative discourse. Many of the the following chapters in this thesis involve discussions of how intentions combine together to make up a dialogue.

From discourse structure a level called Intentional Form (IF) is derived. This is done by: (1) establishing deep coherence relations between clausal forms, and (2) recovering the speaker's goal associated with the discourse representation.

LF -> $\{_{2DR}$ $[_{M1}$ CF_i ...$[_{M2}$ $CF_j\}$ -> IF

Clausal forms can be connected by one of the following deep coherence relations:

(1) Causal (Occasioning, Enablement).

(2) Spatial.

(3) Temporal.

(4) Definition.

The above levels combine to form a model for Discourse Analysis (DA). Pustejovsky shows the representation for the following simple discourse:

A. The economy of Houston, where most US oil is refined, is rapidly
declining,
B. Because the price of oil is falling.

Assuming that the logical representation as input to the analysis in satisfactory then the DRs for A and B are given as follows:

DR_A [*_M* type:statement &
Exists (x) [economy(x) of (x,H & decline(x) &
ELABORATE(H,λ x(most-of-oil-refined-in(x)))]

DR_B [*_M* type:support &
BECAUSE(m,the(y)(oil-price(y) &
failing(y)))]

"BECAUSE" relates propositions in different moves and acts as a move directive. This is equivalent to clue words in Reichman's approach. Then, the IF associated with each utterance will (1) establish coherence relations between clauses, and (2) recover the speaker's goal. On (1) elaboration in A will will translate to what Pustejovsky calls a "definitional" relation. The representation seems rather complex to us in that there are two levels of representation utterance type and move directives. This could possibly be collapsed into utterance type alone without any major loss of semantics. The representation would then look like:

DR_A [*_M* type:ELABORATE &
Exists (x) [economy(x) of (x,H & decline(x) &
(H,λ x(most-of-oil-refined-in(x)))]

DR_B [*_M* type:BECAUSE &
(m,the(y)(oil-price(y) &
failing(y)))]

The representation, in this more parsimonious form, becomes more like the representations we shall discuss in future chapters.

The intentional form will represent, among other things, the plans and goals associated with the utterance. Pustejovsky distinguishes between what is presupposed and inferred by a listener:

(1) Asserted clauses.

(2) Clauses presupposed by the lexical structure of the utterance.

(3) Clauses presupposed on the basis of syntactic structure of the utterance.

(4) Clauses presupposed as a result of convention by which the mutual beliefs of the discourse participants are understood.

The levels at which the presuppositions are derived and computed are:

(1) the lexical presuppositions follow the LF structure into the DR which means they are computed already.

(2) Structural presuppositions are computed from LF and fed into DR. Conventional implicatures are computed from DR itself, making use of information associated with clue words and other ‘conventional implicature triggers’.

(3) The presuppositions associated with beliefs and common ground will be computed at IF. Inferencing occurs as a result of conventional inferences and IF feeds into itself. Inferencing is in respect to the intentions and goals of the speaker/hearer.

Pustejovsky’s theory is implemented as a computational model called CICERO which is a subprogram of a system called COUNSELOR, a natural language interface to a legal reasoning system. CICERO has two subsystems one of which tracks and predicts the structure of a discourse based on conversational moves interpreted by a discourse grammar and keywords, and the other manages and controls the representation of the deeper semantic relations between discourse entities.

The system contains: (1) A knowledge base defined with clusters, and (2) A best-first control generating and recognising speaker and hearer plans. A cluster is a particular way to represent plans, goals and objects in the world. It is a frame representation language with inheritance properties (see Minsky 1975, Bobrow and Winograd 1977).

At the beginning of processing the CICERO system expects a case facts summary from a layman or attorney. CICERO expects a particular type of speech act called *inform* which is one of five from a set which includes *request*, *promise*, *suggest*, etc. After the initial parse of a sentence CICERO’s task is to confirm its expectations concerning the speaker-goal as well as to form a coherence representation of the semantic content of the proposition. CICERO makes inferences with respect to the presupposition-set rather than the expert system. Also, the same representation is used for understanding and generating text.

Hence, Pustejovsky’s theory and computational model is an integration of semantic, structure, and intention-based approaches to discourse modelling. His theory presents a combination of existing techniques rather than positing a theory in its own right. He highlights the factors of (1) stereotype and (2) intention in his model and how they can be integrated. Pustejovsky includes a model of intention in his approach and weighs structure and intention equally in his model. We can now take a look at approaches to discourse modelling based mainly on the analysis of intention.

4.10 Summary of discourse theories

That completes our discussion of the various approaches to modelling natural language discourse. The central theme of much, if not all, of the work is that there is a notion of coherence in dialogue, where some dialogues can be considered coherent whereas others are not. The motivation for a concentration on discourse coherence is the fact that rational speakers usually use coherent discourse. The different approaches, emphasise some phenomena more than others, within the field of natural-language dialogue modelling. Also, there is a problem in that much of the work gives different names to the same phenomena. The three main themes of the different approaches are the modelling of semantics, structure and intention.

First, semantic-based processing involves recognising and representing the meaning of a discourse. This is completed by representing the semantics of individual utterances in the discourse and linking these representations together. The semantics-based approach attempts to infer relationships between utterances, a process called inferencing, so that implicit links between utterances can be discovered. Such inferencing attempts to maximise the coherence of the discourse. Semantics-based approaches argue that by modelling the semantics of a discourse other information will fall out of that.

Second, structure-based approaches model the discourse from the point of recognising and representing explicit and implicit structures in the discourse. Many of the theories define explicit spaces which represent implicit spaces in the discourse. Such spaces are called topic, context spaces or focus spaces. Much of the work involves recognising and representing these spaces and the relationships between them.

Third, intention-based approaches model the discourse from the point of view of the intentions, or plans and goals, of the participants in the discourse. The coherence of a discourse is determined by the coherence of the intentions of the participants. Much of the work here involves recognising and representing the plans and goals of the speaker and the relationships between them. Again, there are a number of names for structures representing intention. Finally, Pustejovsky claims to provide an integrated theory and computational model of discourse processing, which models semantics, structure and intention. This general approach would seem to be the best solution for modelling discourse.

Now, that we have described the existing work in discourse modelling, we can move on to discuss our own theory. Our theory concentrates on using the analysis of intentions to model, in part, coherence in natural-language discourse. We also wish to show that intention analysis is useful for effective natural language dialogue between different types of people and a computer.

Part III

Intention analysis

Chapter 5

A theory of intention analysis

So far our main concern has been to describe existing approaches for modelling natural-language discourse. The next step is to argue for our own theory of modelling natural-language dialogue through the analysis of intentions. The theory shows how the coherence of a dialogue can be modelled, in part, by analysing intentions. Hence, the theory will have much in common with approaches described in the previous chapter. The theory is concerned with links between utterances in a dialogue. Each utterance in a dialogue represents one or more intentions of a speaker. Intentions can also be called speech acts or communicative acts[1]. We show that different types of utterance can represent different types of intention. It is shown that intentions can be ordered on the basis of *satisfaction*, where some intentions will indicate more satisfaction than others. In turn, such an ordering can be used to determine the degree of local and global satisfaction of a speaker in a dialogue. A theory for modelling dialogue through the analysis of intentions will have at least two properties. First, it must be possible to recognise intentions in dialogue, and second, it must be possible to represent intention types.

We show that it is possible to recognise intentions by the analysis of the syntax, semantics and pragmatics of utterances. *Intention graphs* are introduced as pictorial representations for intention sequences. We show that intention graphs can reflect a number of interesting phenomena from natural-language dialogue. A notion of *balance* is introduced to reflect the balance of frequencies of different types of intention in a dialogue. We show that intention graphs will reflect the balance in a dialogue. Assuming an ordering of intention satisfaction exists, the ordering can be used in conjunction with intention graphs to interpret degrees of intention satisfaction of users. The requirements for a computational model of intention analysis are introduced. The computer model must be able to recognise and represent intention types.

[1]Communicative acts have received much attention in the literature. A philosopher of language, Searle, extending the work of his predecessor, Austin, founded the term *speech act* from which the term communicative act is derived. Austin (1962) and Searle (1969) provide more detailed discussions on speech acts.

5.1 Recognition of intention

As already mentioned, one of the properties of a theory of intention analysis is that intentions in dialogue can be recognised. Each natural-language dialogue usually consists of sequences of utterances in a specific temporal order. A typical dialogue can consist of two up to say, fifty utterances. Each utterance in a dialogue will reflect some intention. As dialogues consist of sequences of utterances, then sequences of intention can be used to represent those dialogues. The coherence of a dialogue will, to some extent, be reflected in the coherence of intentions. A central principle of our theory is that a dialogue, which consists of sequences of utterances, can be modelled in terms of sequences of intention.

We have already seen in Chapter 4 that there has been much work in the field of natural-language processing on the recognition of intention in natural language[2]. In order to recognise intention it may be useful to determine whether there are different types of it, and see whether these types will give information about intention recognition. It seems likely that there are a number of different types of intentions in natural-language dialogue. It would seem that there is no way of determining the exact nature of such intentions as they exist in people's heads. However, it may be the case that natural-language utterances reflect the different types in different ways. As natural-language utterances can be investigated in terms of their syntax, semantics, and pragmatics, it is possible that various types of intention can be investigated in the same way. Let's look at an example dialogue in Figure 5.1 to see what different intention types might look like.

1 (a) J: How do you know we are in a recession?
(b) E: The last three months have shown a downturn in GNP.
(c) J: What is GNP?
(d) E: GNP is the Gross National Product of a country.
It is the total value of all goods and services produced
by a country over a given period, usually annually.
(e) J: Tell me more about GNP.
(f) E: GNP is sometimes given in relation to GDP or Gross Domestic
Product.
(g) J: Explain the difference between GNP and GDP?
(h) E: GDP is the Gross National Product, less income
from foreign investments.

Figure 5.1: An *Economics* dialogue

This is a dialogue between two people, an Economist (E), and Journalist (J). We shall call the dialogue the *Economics*[3] dialogue. First, it is noted that the dialogue is of a consultancy, or question-answering, form where one person, the Journalist, is attempting to obtain information

[2] We refer particularly to the work of Schank and Abelson (1977) on scripts, plans and goals, the work of Cohen et al. (1982) on the analysis of intention in utterances, and the further extensions of that work in Allen and Perrault (1980), Litman and Allen (1984) and Hinkelman and Allen (1989).

[3] We choose an Economics dialogue as it is: (1) likely to be a dialogue encountered frequently in the real-world, (2) general, and (3) practical and relevant.

on some topic. If one wanted to categorise the utterances in the dialogue into different types, one might notice that there is a difference in the types of utterances being used. For example, in (1a) the Journalist is making a basic request about the topic of recession. The intention of J then, is to communicate a request; the intention could be called *request*. However, we can go further than that. This *request* intention is of a specific type, which we can call an *information request*. In (1c) J asks for an explanation of GDP, i.e. an *explanation request*. Next, J asks an *elaboration request* in (1e). Finally, there is another *explanation request* in (1g).

Already we can see that it is possible to categorise the intentions of a speaker into different types, which have different names. It will not always be so easy to determine the type of intention indicated by an utterance, as it has been for this dialogue. Much of the time the intention will only be apparent from context. One of the problems of categorisation is that it can be subjective in at least two ways. First, the names chosen for intention types will usually be arbitrary even if we refer to the same entities. You might have used the names *basic-question*, *extend-answer-question*, and *clarification-question*, where we choose the names *information*, *elaboration* and *explanation*. Second, the categories themselves can be chosen subjectively so that some categories may exist in one classification, but not in another. In this work we hope to obtain a set of objective categories, although, of course, it will be almost impossible to come up with a set of names for the categories and hope that everyone else will come up with the same set of names. In fact, it is noted that in Sarantinos and Johnson (1990) there is a classification of intention types in consultancy dialogues where the authors have come up with a set of categories similar to our own, but with different names!

Intuitively, it is easy to see that there must be a distinction in types of intention. Otherwise, when people talk, they would discuss the same propositional content, or topic, without any regard for why that information was being related. For example, the topic, GNP, would be discussed with no regard for whether it was being explained, elaborated upon, provided as information, or confirmed. Also, it is probably the case that there will be no discrete categorisation of utterance types into intention types. The sets of utterances representing different intention types will change continuously from one intention type to another. This is reflected by the fact that it can be very difficult to determine which intention type a particular utterance represents.

There will be a many-many relationship between utterances and intentions. A given utterance will represent many intentions and many utterances will represent single intentions. The context of an utterance will, in a large part, determine its intention type. For example, the exact intention of the utterances, "What do I do next?" or, "It's cold in here[4]," will, in a large part be determined by their context. This point is important, and it was brought up in Chapter 1, because it makes clear that it will be difficult, if not impossible, to determine the intention of an utterance on a syntactic basis alone. In fact, it's often the case that you and I have problems distinguishing whether someone wants information, wants something explained, or wants it confirmed. On occasion, you might find

[4]This is an example of an *indirect speech act*, where the speaker may intend the hearer to close a door or window, rather than to just describe the current state of events. It would be difficult to see how any syntactically-based method, alone, could determine the intention of such an utterance.

yourself saying to someone, or saying to yourself, “Why are you telling me this?” Likewise, it’s often the case that a number of different utterances can be used in the same context to denote the same intention. It seems elementary that there are a number of different ways of expressing the same intention in natural-language. For example, if one wanted the salt at a table then one could say either, “Could you pass the salt?” or “Pass the salt, please” and these utterances would, most likely, convey the same intention.

It is difficult to discern how many intention types there are in everyday natural-language. The number of intention types depends on how one classifies them. It seems to be the case that any intention type can be further broken down into other sub-intention types. This phenomenon has also arisen in research on the classification of utterances into speech act categories. Many researchers found that speech acts could be further and further broken down into other sub-types. Hence, it may be the case that there are thousands of intention types in everyday natural language[5]. If we wish to study these types then a reasonable approach would be to investigate the categories in a domain which exists as a subset of natural language. That will reduce the number of intentions considerably, but there will still be a large number of them. We can choose the set of language utterances in the domain of consultancy, or information retrieval, as this is a useful domain, and is one often studied in relation to natural-language dialogue programs. Let us try to formulate what sorts of intention types might be expected in such a domain where a speaker is attempting to perform some task.

First, it is noted that most of the intentions in a consultancy domain will be *requests*. However, that does not get us anywhere as we wish to uncover distinctions between the different types of intention in the domain. There must be a number of different types of *request* intentions. There might be an intention type called *information* for denoting basic queries about how to do operations in the domain. An intention called *description* which denotes queries about descriptions of objects, or concepts might exist. Then, there could be an intention called *instruction*. This would be defined as an intention type requesting the execution of some operation to achieve information, rather than asking about how to achieve that information. For example, if I asked you to look up the *Yellow Pages* to find the phone number for a freight company, rather than asking you the number, this would be an *instruction* rather than an *information* intention. Next, there might be one called *elaboration* concerned with the elaboration of information in relation to some goal of the speaker. Another typical intention type to be expected is *confirmation*, for confirming the speaker’s beliefs about some component of a domain. Next, there could be an *explanation* intention for cases where the speaker wants information explained. A *guidance* intention would be used to ask for guidance when the speaker has become *stuck* in trying to execute operations in a domain. There could be a *repetition* intention for denoting intentions which need to be repeated for some reason. For example, a *repetition* could be caused by a situation where the speaker may not be satisfied with an answer that has been given for some other intention. Finally, there could be an intention type called *nointention* for denoting cases where an intention is not immediately relevant to the domain,

[5]This point has been argued by Cohen and Levesque (1985), Grosz and Sidner (1986), and Wittgenstein (1963).

or not understood by the hearer. For example, in the *Economics* domain a query like, "Do you like beer from the can, or from bottles?" might result in a *nointention*.

Each of the above intention types could be defined in terms of their relationship to plans and goals which a speaker wishes to achieve. A *goal* is defined as some state of affairs which a user wants to achieve. For example in the *Economics* domain a goal might be to decide the country with the highest GNP in the world. A *plan* is defined as a set of actions which need to be executed in order to obtain some *goal*. Such a set may involve one or more actions. An example plan for the goal of deciding the country with the highest GNP in the world would be to determine the GNPs for all countries and then to compare them. Many definitions of plans and goals have been given in literature on artificial intelligence and natural-language processing. For example, in Chapter 4 we saw the work of Cohen, et al. (1982), Fikes and Nilsson (1971), Sacerdoti (1977), Schank and Abelson (1977), and Wilensky (1983), on plans and goals. Hence, we propose that the list of intentions in Table 5.1 are, at least, the principal ones in the natural-language dialogue consultancy domain. These types have not been developed totally by an a priori *Gedankenexperiment*. Conceptualisations of some of the types were developed by *Gedankenexperiment* and were further reinforced by Experiments II and III described in Chapter 8. This table includes definitions for each type of intention.

There is no reason to believe that by doing such a *Gedankenexperiment* one can come up with all the possible intention types in the domain of consultancy. In fact, it may never be possible to decide whether one can define all the intention types in a domain, just as it may never be possible to define all the utterances in a natural language. However, the important point is that one can come up with some intention types, and these will help in investigating the domain of consultancy in terms of analysis of intention. Now that we have defined what we believe to be the principal types of intention in the consultancy domain it will be easier to devise some means of recognising them.

The recognition of intention in utterances can be conducted by analysis of their syntax, semantics and pragmatics. With respect to syntax, words such as "explain," and phrases like "tell me more" can be used to indicate intention in utterances. However, such cues will not always be available, and can be unreliable (see Whittaker and Stenton 1988). Therefore, the task of recognising various types of intention will not be easy. Semantics will be useful for determining intention. For example, the semantic content for, "How do I find out the population of England?", will indicate an *instruction* request whereas the semantics for "What is England?" will indicate an *information* request. Finally, context, or pragmatics, is also important because to recognise that, "How do I look at a file on the screen?", following "How do I see my file?", is an example of a *repetition*, or that, "What are condominiums?", following a reply to, "Where do most Americans live in Buffalo, New York?", is an *elaboration* whereas, "What are condominiums?", following, "What are time shares", is a *description*, requires some representation of the context of utterances.

As there are a number of different types of intention in various domains it is possible that these types exist in some ordering. We are particularly interested in characterising that ordering because

Intention	*Definition*
information	An intention requesting a PLAN[a] to achieve a specific GOAL[b] where the GOAL is described. e.g. "How do I cook this dish?" [a] A *PLAN* is defined as a set of actions to achieve some goal. [b] A *GOAL* is defined as an operation a speaker wishes to achieve.
description	An intention requesting a description of a concept or object. e.g. "What is Persia?"
instruction	An intention acting as an instruction to achieve a GOAL, rather than how to achieve the PLAN to achieve that GOAL. e.g. "Can you find out how many foreign nationals now live in Kuwait?"
elaboration	An intention requesting more information on a PLAN or GOAL. e.g. "Could you tell me more about Iraq?" following "Where is Iraq?"
confirmation	An intention requesting confirmation of a belief, or some PLAN believed to execute some GOAL. e.g. "Will sanctions stop Saddam Hussain?"
explanation	An intention requesting explanation or clarification of an item which occurred during the execution of a PLAN for a GOAL. e.g. "Could you tell me what you mean by U.N. resolution 611?"
guidance	An intention requesting a PLAN for a GOAL where there is no explicit GOAL expressed. e.g. "What do I do next?"
repetition	An intention which is a repeated request. e.g. "How many people live in the Gulf?" followed by e.g. "What number of people live in the Gulf?"
nointention	An intention which is not immediately relevant to the domain, or not understood by the hearer as being relevant to the domain. e.g. "Where does Strider live?" in the domain of Economics.

Table 5.1: Definitions of principal intentions for the consultancy domain

we are interested in showing that there is a correlation between intention types and the satisfaction of a speaker. In turn, we are interested in doing that because a measure of the satisfaction of a speaker in a dialogue can be used to indicate the speaker's level of expertise and facilitate the type of information presented to the speaker. Let's move on to discuss a possible ordering of intentions.

5.2 Ordering intentions

As is the case with most objects in the world, if there are a number of different types of them, then it is likely that they can be arranged in various orderings. For example, if I asked you to order the set of cars, then you might decide to order them in terms of their cost, and you might order a set of musical records in terms of how much you like them. Of course, some orderings are more objective than others. Many people would order objects like cars objectively, in terms of cost, but subjectively in terms of beauty. There are a number of possible orderings of intentions. Whatever the ordering chosen might be, all intentions will fall onto some point on that ordering. It is difficult to decide what a useful ordering might be. The usefulness of any ordering will depend on what one wishes to do with it. Also, the ordering should be ordered in a way that would be objective.

As we are interested in dialogues between people and computers, where the computer is supplying information, then a useful ordering of intention might be *satisfaction* of the user. Hence, some speaker intentions will indicate more satisfaction than others. In such an ordering an intention would indicate the degree of satisfaction, i.e. ease, or difficulty, that a person may be experiencing in understanding the information presented in a dialogue. Of course, other factors such as *interest* of the user may influence the ordering and given intention types will not *always* indicate the same levels of satisfaction. The frequencies of different intentions will indicate the overall satisfaction of a speaker in a dialogue. It would be possible to count the frequencies of different types of intention to determine the degree of difficulty a person is having in understanding a topic under discussion. If we represent the first eight intention types defined in Table 5.1 in an ordering of *satisfaction* then it would look like that shown in Figure 5.2. Nointentions are not placed in the ordering because they are not an obvious measure of the level of speaker satisfaction as they are not relevant to the domain.

The reasoning for positioning the intention types in this particular ordering works as follows. If a speaker has no trouble with the topic under discussion he, or she, will simply ask straightforward *information* requests. However, if the understanding of concepts becomes more difficult the speaker will ask more *elaborations* and *confirmations*, and even *explanations*. One could argue that if the information provider was adequate, then the information seeker would not require *elaboration*, *explanation*, and *guidance* intentions at all!

Figure 5.2 is not meant to represent degrees of satisfaction but only the ordering of intentions under a satisfaction rating. It is likely that certain instances of intention types will not always indicate the same relative levels of satisfaction. For example, in some contexts *explanations* may indicate less satisfaction than *guidances*. However, in the absence of propositional content, the

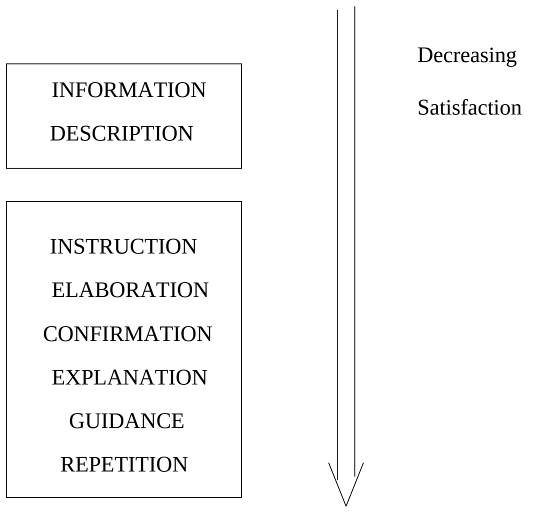


Figure 5.2: *Satisfaction* ordering of intention

ordering in Figure 5.2 would seem the most reasonable. Other work on the exact distributions of intention types with respect to satisfaction is beyond the scope of the work here.

So far, we have discussed the fact that there are a number of intentions in natural-language dialogue and that they, in principle, can be categorised into a number of different types. The names of the types are arbitrary, however we claim that the types can be an objective set. We have proposed nine intention types as the principal intentions to be expected in a domain of natural-language consultancy. We have tried to decide upon names which most people would agree on. We shall see later on in Chapter 8 whether this set is an objective one. Also, we have proposed that intentions can be ordered into a *satisfaction* ordering. Until now, intentions have been considered as primitives, derived from utterances within dialogues. Now, we can move on to take a look at how these primitives can be linked together to build representations of intention sequences in dialogues. In turn, such representations can be incorporated within computational models which model coherence in dialogue.

5.3 Representation of intention types

Now that a set of nine intention types have been defined for consultancy dialogues, they can be used to build a representation of the *Economics* dialogue given in Figure 5.1 above. The principal intention set which we have defined for the consultancy domain is appropriate for the *Economics* dialogue. The next question which must be asked is: what are the requirements of such a representation? Well, the representation should display intention sequences, because a central claim of our theory is that intention sequences can be used to represent the coherence of a discourse. What sort of sequences do we wish to represent? The smallest unit of intention sequencing is one containing a single intention. Such sequencing with a single intention will give useful information about a dialogue, such as the number of occurrences of certain types of intention, and such measures could then be used to give information about the global intention sets of a speaker. Even local measures of speaker intention frequencies could be determined by keeping a *window* of the last say, ten, utterances. For example, to determine local speaker *satisfaction* in a dialogue the satisfaction level of the last ten intentions could be determined.

It is difficult to see how such intention frequencies alone could give any measure of the coherence of a discourse. This is the case because there is no measure of which intentions follow others. In effect, to see an analogue of the danger of just counting frequencies, one can appreciate the difficulties of natural-language parsing where one would just count frequencies of words in utterances without any measure of which words came before other words. With respect to solving the problem of user-modelling, a simple frequency count of intentions does not enable a system to infer how many times an explanation intention has occurred in a row, but just how many explanation intentions have occurred. Also, a simple frequency measure would not be able to separate out the fact that, say, an *elaboration* of an *explanation* may indicate less satisfaction than an *elaboration* of an *information* request. However, a measure of intention pairs could be given different weights in any model of the performance of a user. For example, an *explanation* followed by an *explanation* will get a higher dissatisfaction weighting than an *information* followed by an *explanation*.

Hence, we argue that too much information is lost in using sequences with single intentions, although they may give *some* useful information. Also, the goal of the work here is not only to provide a theory of sequences of intentions in dialogue for solving problems such as user-modelling. The theory is proposed to solve, in part, the problem of natural-language dialogue processing by providing a measure of the coherence of intention in dialogue. We also wish to use the theory to solve other problems in natural-language dialogue such as ellipsis[6] and anaphoric reference resolution[7] A simple measure of frequencies of intention in dialogue would not be able to solve such problems, but a measure of intention pairs might help, because ellipsis and anaphoric reference can be solved by checking the preceding utterance of a given utterance. If a natural-language parser found it difficult to parse an elliptical utterance, because it had missing information, and a history of intention pairs was represented, the latter could be used to determine the most likely possible pair that is currently in question.

Also, we believe that the problem of anaphoric reference resolution could be solved in part by an intention representation. Again, we cannot see how a simple frequency count of intentions could be used to solve the problem of reference resolution. However, a representation, at least, of intention pairs could be used to determine the location of a reference in an utterance. For example, if an *information* request is followed by an *elaboration*, or an *explanation* request and the latter contains anaphoric references then it is likely that the referent can be found in the preceding utterance. However, this may not necessarily be likely if the preceding intention type is simply another *information* intention type. Although we will not discuss in this work how the latter two

[6]The problem of *ellipsis* is one where information is explicitly missing in natural-language utterances and must be filled in from context. This problem was introduced in Chapters 1 and 4.

[7]The problem of anaphoric reference resolution is one where words are used to refer to other words, which in turn, may refer to entities in the world. This problem was introduced in Chapter 1.

problems of ellipsis and anaphoric reference resolution can be solved in detail with an intention sequence representation it is of interest for future work. Here, we will be mostly be concerned with how the sequence representation can be used to solve, in part, the problem of user-modelling. Any arguments about the fact that an intention sequence representation where sequences of length two is not necessary for user modelling, and constitutes higher computational cost, are *straw men* as we will use the representation in future research to solve, in part, other problems in natural-language dialogue. In effect, the intention sequence representation is not an end in itself but a means towards an end where that end is to solve, in part, the problem of modelling natural-language dialogue.

It has been argued that to gain enough useful information from a dialogue the representation of intention types should, at least, contain representations of intention pairs. It could be argued that we must go further than intention pairs to gain full information about the coherence of dialogue. Such an argument is correct if one wishes to obtain information about the extended plans and goals of speakers in dialogue. However, such elaborate representations are beyond the scope of this research and may be investigated in future research. Hence, an assumption of the work here is that intention pairs will give at least enough information to solve, in part, the problems of (1) user modeling, (2) anaphoric resolution, and (3) ellipsis. Such an assumption will involve a means of counting intention pairs in dialogue[8]. The next question we ask is: what representation can be used to represent the complete set of intention pairs in a dialogue?

A possible representation could list, in sequence, all the intentions occurring in a dialogue, from start to end. However, such a representation would be too linear for long dialogues and would go off the page; what we need is a more compact representation. There must be some way of packing information about intentions in the dialogue into a more condensed representation, but in doing that there should not be a major loss of information. The representation, then, should represent all intention pairs for all intentions in the dialogue. How are intention pairs represented? This can be done by representing intention names and the links between them. How do we show which intention comes before another? The links can have directions. So, now a representation exists which shows all intention pairs in the dialogue, and the order in which they occur. However, if each intention type is only shown once in the representation, and there is no hint of temporal orderings of intentions, then there will be loss of information on intention pair counts, because some intention pairs may occur more than once. All intention pairs will be represented by showing the number of occurrences of intention pairs as integers on the links between those pairs. A frequency of "1" is assumed if no integer value exists on a link. Such representations can be called *intention graphs*. We will see in later chapters that intention graphs are useful representations of intention pairs in dialogue. An intention graph is useful because, at a glance, one can recognise the types of intention sequences which occur in a given dialogue. They will be particularly useful for viewing information

[8]We should point out here that we are not the first to conduct such counts. In the field of lexical analysis much work has been done on word association or *co-occurrence* frequencies in natural-language text. Such counts are termed *word digram* counts (see Jung 1968, Postman and Keppel 1970). More pertinent is the work in discourse analysis on analysing speaker-hearer utterance associations. Such associations are termed *adjacency pairs* (see Heritage 1986, 1988).

on frequencies of intention pairs obtained from the analysis of experimental data.

If you have a picture in your mind of what *intention graphs* might look like, then you might accept Figure 5.3 as an intention graph which represents the sequences of intentions in the *Economics* dialogue. The graph shows that an *information* intention is followed by an *explanation*

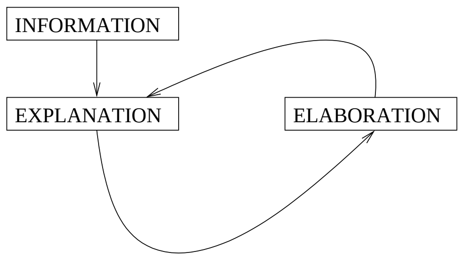


Figure 5.3: Intention graph for *Economics* dialogue

intention. An *explanation* intention is followed by an *elaboration* intention, and an *elaboration* is followed by an *explanation*. However, from the graph alone it is not possible to say which intention occurred first or last, or for that matter to decide any global temporal ordering, in the dialogue. This will be a general problem of intention graphs. There are no integers on the links and hence we assume a frequency of 1 for each intention pair.

It is noted that the representation does not include any information about the propositional content, or topic, of the dialogue, only about the pairs of intention sequences in the dialogue. If propositional information was also to be included it could be tagged as shown in Figure 5.4. The information is expressed in predicate calculus form, although many semantic representations could be used. A number of possible semantic representations were discussed in Chapter 2. There are two cases of *explanation* in the intention box for explanation, one for each explanation request in the dialogue. Such multiple occurrences of intentions can be shown within an intention box using temporal ordering, $t_0 ... t_n$, where t_0 is the first instance of the pertinent intention type, and t_n, the final instance. Figure 5.4 shows the sequences of intentions in the dialogue plus added

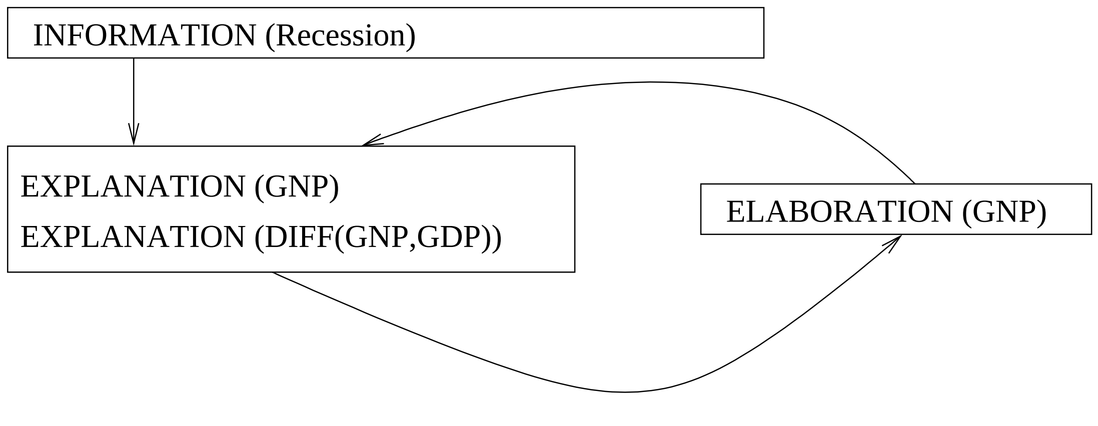


Figure 5.4: Intention graph incorporating propositional content

information about what the utterance intention concerns. A distinction is being made between *why* something is being uttered and *what* is being uttered. The reason something is being said is indicated by the intention type, whether it is an *explanation*, *instruction*, or *elaboration*. *What* is being said, is the subject being talked about. The distinction between the graph in Figure 5.3 and Figure 5.4 is that the former is only concerned with *why* a dialogue is being conducted, while the latter is concerned with *why* the dialogue is being conducted and *what* is being uttered. In other words Figure 5.3 only contains pragmatic information, while Figure 5.4 contains pragmatic and semantic information[9]. Hence, in effect, Figure 5.4 represents an integration of semantic- and intention-based approaches to dialogue modelling.

If intention graphs do not incorporate semantic information then they will be representations which are independent of any topic under discussion. In other words intention graphs, without semantic content, are topic independent; the dialogue could be about economics, elephants, Hopi Indians, or the UNIX operating system. Hence, the intention graph can be a universal representation. This means that intention graphs can be representations independent of specific domains. The only requirement will be that intention types incorporated within graphs circumscribe, at least, the principal intentions in respective domains. In later chapters it will be seen that this fact will be very useful for computational models of intention analysis. It is also possible to incorporate into the intention graph *who* is saying *what* is being said. Then, an intention graph will also denote the agent of communication, as shown in Figure 5.5.

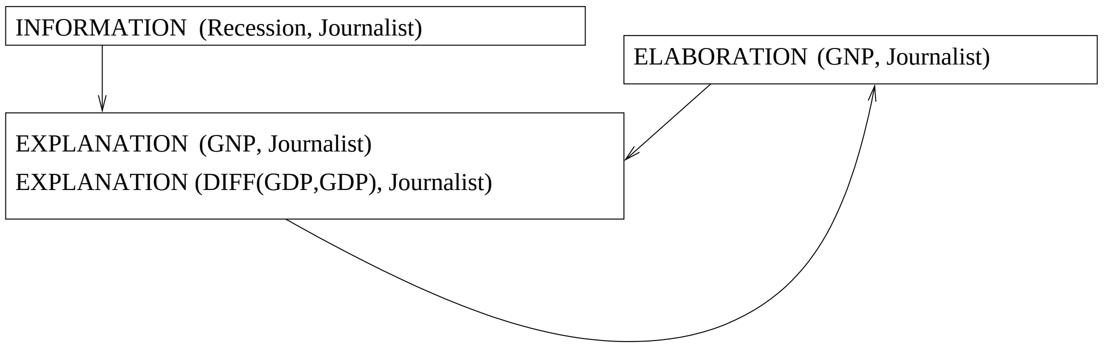

INFORMATION (Recession, Journalist)
ELABORATION (GNP, Journalist)
EXPLANATION (GNP, Journalist)
EXPLANATION (DIFF(GDP,GDP), Journalist)

Figure 5.5: *Agent* tagged intention graph

Note that, so far, we have only been concerned with the utterances of the Journalist. He/she has been of most interest because he/she is the one who is seeking information, and it is his/her intentions which are to be satisfied. However, the utterances of the Economist are important too, and the graph in Figure 5.6 shows his/her intentions.

In this figure the intentions of the Economist are represented in a similar fashion to those of the Journalist. We assume that the Economist has intentions compatible with the Journalist's although this will not always be the case. For example, if someone misunderstands your intentions he/she will not reply with similar intentions. Note that the index "A:" is used to reflect the fact that the Economist's intentions are replies or answers rather than requests. Looking back at Table 5.1 you will see that all the intentions there can be *answers* as well as *requests*.

[9]'Pragmatic' and 'semantic' are used here in the senses defined in Chapter 1.

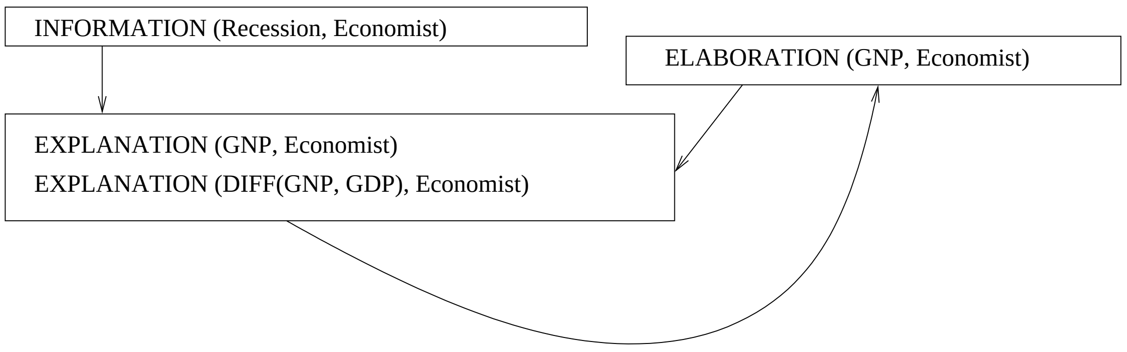

INFORMATION (Recession, Economist)
ELABORATION (GNP, Economist)
EXPLANATION (GNP, Economist)
EXPLANATION (DIFF(GNP, GDP), Economist)

Figure 5.6: Intention graph for Economist's intentions

2 (a) J: How do you know we are in a recession?
(b) E: The last three months have shown a downturn in GNP.
(c) J: What is GNP?
(d) E: Do you know what GDP is?
(e) J: No.
(f) E: GNP is the Gross National Product of a country.
It is the percentage increase in production of a country.
(g) J: Tell me more about GNP.
(h) E: GNP is the Gross National Product of a country.
It is the total value of all goods and services produced
by a country over a given period, usually annually.
(i) J: Explain the difference between GNP and GDP?
(j) E: GDP is the Gross National Product less income
from foreign investments.

Figure 5.7: An asynchronous Economics dialogue

It is important for a theory of intention analysis to represent the intentions of both the speaker and the hearer, rather than just using a single representation. This is the case because, as just mentioned, the intention representations for the speaker and hearer are not always the same. In an example dialogue the speaker may intend an utterance to be an *instruction* intention and the hearer may understand the utterance as an *information* intention. For example, this may be the case where I ask you, "Can you pass the salt?" and you say, "Yes" but you do not pass the salt.

In Figure 5.6 information on the structure of the Economist's dialogue is presented in the same order as that of the Journalist. This happens because the dialogue has followed synchronously from one speaker to the other. However, if the dialogue occurred as shown in Figure 5.7 this would result in an asynchronous representation. The representation of the dialogue for the Journalist for this case is shown in Figure 5.8. Note that one of the Journalist's intentions, *information*, is marked with an 'A:'. This is to indicate that this *information* intention is an answer rather than a request. An *answer* intention could have been added to the set in Table 5.1, but we do not do that because all the intentions in Table 5.1 can be types of answers as well as types of request. The intention graph for the Economist will be that shown in Figure 5.9. "R" is used to mark the Economist's request.

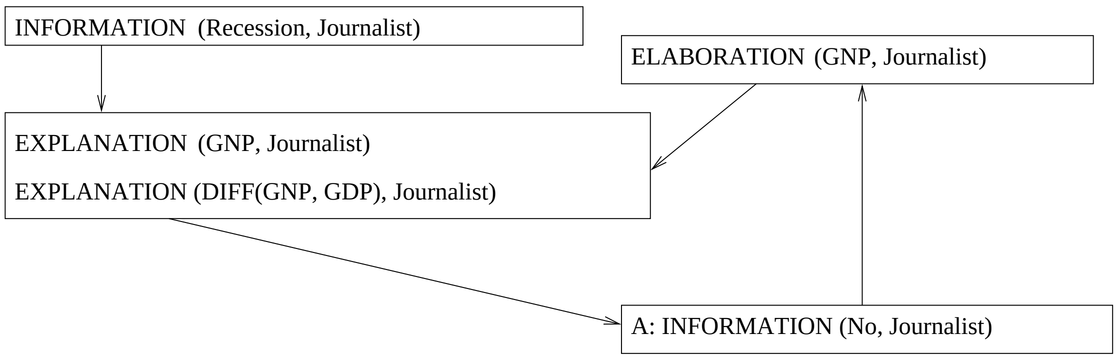

INFORMATION (Recession, Journalist)
ELABORATION (GNP, Journalist)
EXPLANATION (GNP, Journalist)
EXPLANATION (DIFF(GNP, GDP), Journalist)
A: INFORMATION (No, Journalist)

Figure 5.8: Journalist's intention graph for asynchronous Economics dialogue

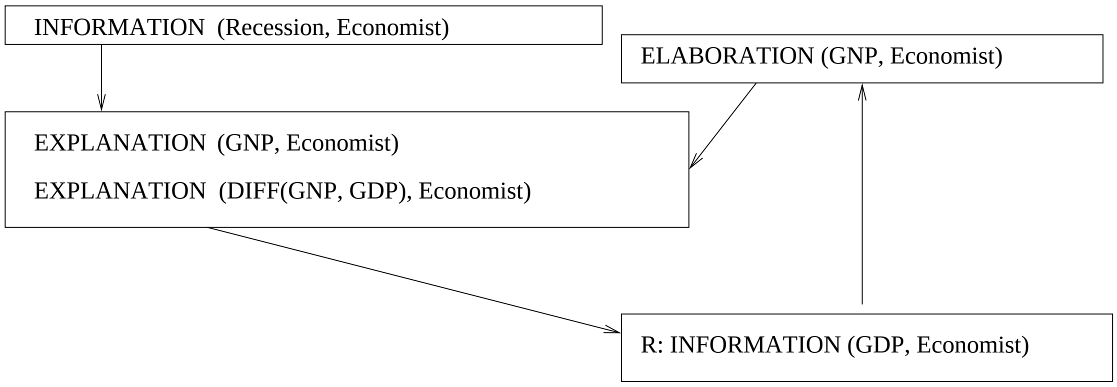

INFORMATION (Recession, Economist)
ELABORATION (GNP, Economist)
EXPLANATION (GNP, Economist)
EXPLANATION (DIFF(GNP, GDP), Economist)
R: INFORMATION (GDP, Economist)

Figure 5.9: Economist's intention graph for asynchronous Economics dialogue

To summarise, a representation called an *intention graph* has been developed for representing sequences of intention in natural-language dialogue. The motivation for developing intention graphs has been to build a representation of intention sequences in natural-language dialogue which (1) can be incorporated within a computational model, and (2) can facilitate the presentation of experimental data on intention sequences. Intention graphs can represent information about *why* something is being uttered, *what* is being uttered, and *who* is doing the uttering. The intention graph also represents the temporal orderings of intentions within intention pairs. The semantic content of multiple occurrences of a given intention pair can be represented in their order of occurrence. Intentions can be tagged in graphs for asynchronous dialogues. However, it is important to note that the intention graph does *not* represent a complete temporal ordering for a dialogue, as it is a condensed representation. In other words, by looking at an intention graph you can only infer the following information about the intentions in a dialogue:

(1) The complete set of types of intention modelled in a dialogue

(2) The local temporal relations of intention pairs

(3) The frequencies of occurrence of each of (2)

(4) The semantic/propositional content

(5) The speaker

In later chapters we shall only be concerned with intention graphs which can represent components (1), (2), and (3) of the above set. These components are all that are required for the purposes of providing evidence for our theory and hypotheses.

Now, that a representation of intention types in natural-language dialogue exists called *intention graphs* it is possible to move on to discuss how some interesting features of dialogue can be reflected with intention graphs. A feature of dialogue of interest to us will be the *balance* of a dialogue, or the frequencies of different intention pairs in the dialogue. This will be useful as it will reflect the degree to which a speaker is using certain intention types, and in conjunction with an ordering of satisfaction, the degree of satisfaction of the speaker. In the case where the hearer is a natural-language processor the processor would modify its responses to facilitate communication with the user.

5.3.1 Balance of intention

The frequencies of various intention pairs will indicate how balanced a dialogue is. If the frequency of a certain type of intention pair outweighs other intention-pairs then we say that the intentions are not balanced. A dialogue will be balanced if different intention-pairs in the dialogue are equal in frequency. The degree of imbalance between two or more intention pairs will be reflected by the differences in their respective frequencies. We will not go into any detail here about what functions could be used to weigh the different intention types to determine balance in a dialogue. However, in the next Chapter we will introduce a function which acts as an implicit measure of balance by applying weights to intentions pairs for measuring satisfaction.

In different dialogues, different frequencies of various intention pairs will occur. For example, if you have a dialogue with your spouse where you utter a large number of *explanation* requests, in comparison to other request types, the dialogue will be unbalanced, and this will have implications about how it will continue. If various intention types are ordered with respect to an ordering of satisfaction, as was done above, then the frequencies of intention types will indicate the degree of *satisfaction* of a speaker. If a computer program is interacting with a user in a consultancy dialogue, where the user utters a large number of *explanation* and *guidance* requests, then the computer may infer that it is being tested, or that the user is having trouble. One way in which balance may be manifested in a dialogue is where a speaker intersperses certain intention types with others.

The boomerang effect

Balance in a dialogue will be indicated by the readiness of a speaker to move from one intention type to another without staying on the same intention type for a long period. In these cases there will not be a large number of intentions of any particular type occurring. We can call this the *boomerang effect*. *Boomerang sequences*, then, will be those those where one intention is followed by another, and where there is a boomerang back, to the former intention. An example of the *boomerang effect* in Figure 5.10. Checks on whether the boomerang effect occurs in dialogue, or

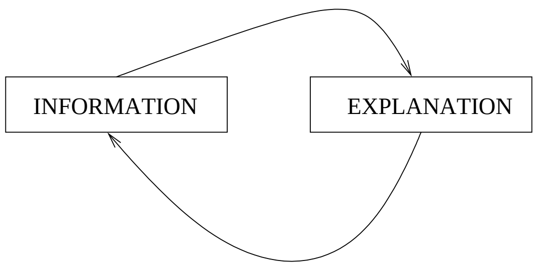


Figure 5.10: The *boomerang effect*

not, will be useful for determining, say, whether a speaker moves back for say *information* after asking an *explanation* intention or just keeps asking for *explanation* intentions. The latter activity might indicate to a natural-language processor that the user is having problems in understanding the type of answers that it is producing. Another indication of the balance in a dialogue will be the number of times a given intention type is repeated.

Loops as repetition of intention type

Assuming that the intentions in a dialogue are ordered in a satisfaction spectrum then, if a given intention is repeated a number of times, this will indicate the degree of satisfaction of a speaker, to an extreme. For example, Figure 5.11 below shows two explanation requests occurring in sequence.

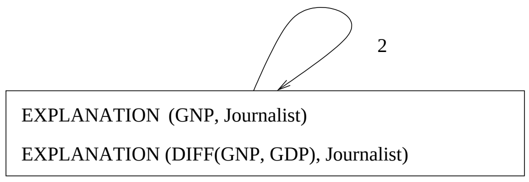

2
EXPLANATION (GNP, Journalist)
EXPLANATION (DIFF(GNP, GDP), Journalist)

Figure 5.11: Repeated intentions

As we can see from Figure 5.11 repetitions of intention will be represented in intention graphs as *loops*. The number of loops in an intention graph will be a very useful measure of the degree of satisfaction of a speaker. If a speaker performs intention loops it will be an indication of *extreme* satisfaction or dissatisfaction. For example, if a speaker loops ten times on explanation intentions, it is an indication that something is going seriously wrong.

At this point we must clarify the difference between (1) the repetition of an intention type, and (2) the *repetition* intention type. The former case is where an intention type has been repeated irrespective of the propositional content, or topic, of the utterance. Hence, repetition of intention only means that an intention *type* has been repeated; it does not necessarily mean that the *same* intention has been repeated. The latter case is where the propositional content, or topic of a particular intention type has been repeated, and this is marked as a *repetition* intention in an intention graph. Both the former and latter types of repetition will be indicators of the degree of balance in a dialogue.

It is important to point out here that a simple frequency count not counting intention pairs, would not give any information about whether repetitions, or the boomerang effect, exist in the dialogue. A simple frequency count would give at most a count of whether at a global level the intentions in a dialogue were balanced. Hence the factor of *balance* will be useful for modelling the degree of satisfaction of a speaker in a dialogue. Now, we can move on to compare some intention graphs representing various dialogues.

5.3.2 Comparing representations

In order to compare intention graphs it is necessary to take a look at some examples. Another example *Economics* dialogue might occur as shown in Figure 5.12: The intention graph for this

(a) J: How do you know we are in a recession?
(b) E: The last three months have shown a downturn in GNP.
(c) J: What is the current rate of GNP?
(d) E: GNP is currently at 0.9%

Figure 5.12: Another *Economics* dialogue

dialogue is shown below in Figure 5.13.

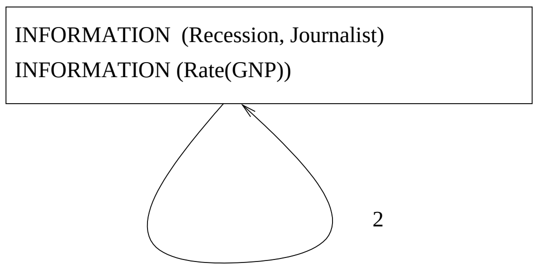


Figure 5.13: Intention graph for second *Economics* dialogue

If one compares the representation for this dialogue with the one in Figure 5.5, it is noted that the representations differ in terms of intention types and sequences. First, note that there are no *explanation* intentions in Figure 5.13, only *information* intentions. This is important as it is an indication that the Journalist in Figure 5.13 knows more about the topic under discussion as compared with the Journalist in Figure 5.5; assuming, of course, that the Journalists are being genuine in the expression of their intentions. Second, Figure 5.5 contains one *explanation* following an *information*, and another following an *elaboration*. Also an *elaboration* follows one of the explanations. Taking a look at the intention graphs in conjunction with the ordering of satisfaction of intention, it is possible to determine the degree of satisfaction of the Journalist. The explanations in Figure 5.5 indicate that the Journalist there is less satisfied than the one in Figure 5.13. Hence, it is concluded that, assuming a spectrum of intention-satisfaction exists, intention graphs will indicate the degree of satisfaction of speakers. Certain intention graphs will indicate dialogues with high satisfaction, while others will not.

It will be the case that various sequences of intentions can be ordered in different ways. Hence, taking Figure 5.5 above the intentions could be reorganised to give Figure 5.14 below. If intention

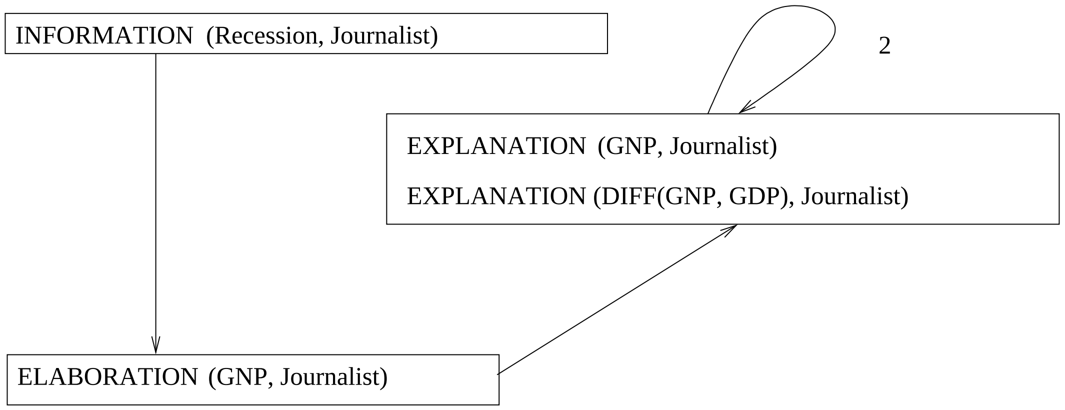

INFORMATION (Recession, Journalist)
2
EXPLANATION (GNP, Journalist)
EXPLANATION (DIFF(GNP, GDP), Journalist)
ELABORATION (GNP, Journalist)

Figure 5.14: Reorganised intention graph for Economics dialogue

graphs described complete temporal orderings of intentions in dialogue then it may be possible to map, or *decompile*, intention graphs into natural language again. Such intention graphs would need to have tags denoting temporal orderings of intentions. Such a process would be useful for natural-language generation from intention graphs[10]. Hence, this intention graph if tagged with time tags as shown in Figure 5.15, when *decompiled*, could represent the dialogue in Figure 5.16

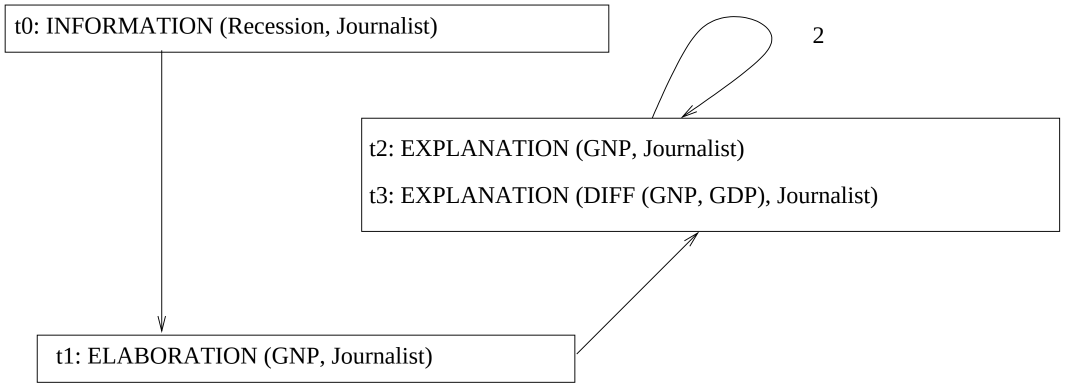

t0: INFORMATION (Recession, Journalist)
2
t2: EXPLANATION (GNP, Journalist)
t3: EXPLANATION (DIFF (GNP, GDP), Journalist)
t1: ELABORATION (GNP, Journalist)

Figure 5.15: Intention graph with temporal tags

below. It is noted that the dialogue seems strange as an *elaboration* of GNP is requested, and later there is a request for an *explanation* of GNP. The rules people follow which enable coherence in dialogue seem to have been violated. In other words one of Grice's maxims for conversational cooperation, which we discussed in Chapter 5, seems to have been violated. Let's investigate these coherence considerations in more detail.

[10]See Hovy (1988), McDonald et al. (1987), and Zock and Sabah (1988) for discussions of research on natural-language generation.

(a) J: How do you know we are in a recession?
(b) E: The last three months have shown a downturn in GNP.
(c) J: Tell me more about GNP.
(d) E: GNP is the Gross National Product of a country.
It is the percentage increase in production of a country.
(e) J: What is GNP?
(f) E: GNP is the Gross National Product of a country.
It is the total value of all goods and services produced
by a country over a given period, usually annually.
(g) J: Explain the difference between GNP and GDP?
(h) E: GDP is the Gross National Product, less income
from foreign investments.

Figure 5.16: Generating an *Economics* dialogue

5.3.3 Coherence of intention

In a dialogue it would seem that there will be expected orderings of intention pairs. If the set of rules defining that ordering is violated then the dialogue is considered incoherent, and the speaker irrational, if no obvious reason for the incoherence exists. This may be an important element of a theory of intention analysis. We argued that a central principle of our theory of intention analysis is that the coherence of a discourse can be determined by the sequencing on intentions within that discourse. If the sequencing involves intention sequences which violate rules of cooperative communication, such as Grice's, then it has been established that the discourse is incoherent, and in the absence of better explanations, we can assume that the speaker is irrational. Speakers seem to have the capability of checking for coherence of intention in dialogue and it is possible that people have some rules for doing so. It is difficult to see how the above dialogue can be considered to violate any hard and fast rule because the occurrence of the first *explanation* intention could be construed to be an example of a repetition of the *elaboration*. However a possible rule might be that given below:

- *elaboration*, *confirmation*, or *explanation* intentions would not be expected to occur as the first intention in a dialogue.

However, there are context where such intentions could begin a dialogue. For example, in an interrogation context, a dialogue could begin with an *explanation* intention. Similarly, it would be strange in certain situations for someone to begin a dialogue with, "Tell me more about the Ton Ton Macoute?" when the speaker and hearer had no cue for such a topic of discussion. However, there would also be counterexamples to any rule which encodes such behaviour.

The advantage of incorporating such rules in a theory or computational model of intention analysis is that they can be used to determine the level of coherence of a dialogue. Note that the above rule is not solely about the topic, or propositional content of a discourse, but about the intentions relaying the topic. A speaker who only maintains a model of the propositional content, or topic, of a dialogue would not notice that the dialogue was incoherent; in fact, in the example case of the *Economics* dialogue in Figure 5.16 it has just been demonstrated that, even though the topic of the discourse remains the same, the dialogue appears incoherent!

If we agree with certain philosophical schools of thought, then in order for verbal communication to occur between people and computers, a necessary condition is that people must ascribe intelligence to computers[11]. However, we would probably be reluctant to ascribe intelligence to computers if they do not notice an incoherent, and irrational dialogue! The implication is that if there is to be effective natural-language dialogue communication between people and computers then it may be necessary for computers to incorporate some model of the analysis of intentions, which incorporates rules such as the one above.

It is difficult to see how any theory, or computational model, which incorporates such rules could model dialogue without being very elaborate, and without conducting a substantial amount of contextual processing. We will not be concerned with such rules in this work as they are not required for testing our hypotheses. However, any complete theory and computational model of natural-language discourse processing would need to incorporate them.

5.4 Summary

We started by pointing out that any theory of intention analysis for natural-language discourse processing would have to satisfy at least two properties: (1) that intentions in natural-language dialogue can be recognised, and (2) that there is a representation of the intention sequences in natural-language dialogue. Next, it was noted that natural-language utterances represent intentions, and that there will be a many-many mapping between utterances and intentions. We noted that in order to recognise intentions a useful method might be to recognise the different types of them. We demonstrated that it is possible to define a minimal set of intention types for a domain of natural-language dialogue such as consultancy. The following nine types of intention were defined: *information*, *description*, *instruction*, *elaboration*, *confirmation*, *explanation*, *guidance*, *repetition*, and *nointention*. It was claimed that these nine intentions will be, at least, the principal intention set for consultancy dialogues. It was claimed that the names of such intention types are, of course, subjective, but that the set is an objective one. We do not claim that we have determined the complete set of intentions for the consultancy domain, or that the set of nine are in any way completely sufficient. It was pointed out that the recognition of intentions could be accomplished by the analysis of the syntax, semantics and pragmatics of utterances.

It was noted that intentions can be placed in an ordering and a sample ordering based on *satisfaction* was introduced. The motivation for such an ordering of satisfaction was that frequencies of the different types of intention in natural-language dialogue could be matched to the satisfaction ordering and, in turn, used to determine the local and global expertise of the speaker.

Next, it was shown that *intention graphs* could be used as pictorial representations of intention sequences in dialogue. Intention graphs have intention types as their primitive units. It was

[11]See Dennett (1981), and Grice (1969).

pointed out that intention graphs could be used (1) by computational models as a representation of intention sequences in discourse, and (2) for facilitating the presentation of experimental data on intention sequences. Each intention graph represents all intention pairs in a dialogue and the mutual ordering of those pairs. Intention graphs are condensed representations of dialogues and represent the frequencies of occurrence of intention pairs. We showed how intention graphs could also incorporate *what* is being said in a discourse, or semantic information, and information about the *speaker* or holder of an intention. However, intention graphs do *not* portray absolute sequences of intentions in dialogue, and hence it is, in general, impossible to infer more detailed information from them, such as the first intention in a dialogue, the last intention, or even intention sequences of length three. In this work we shall only be interested in intention graphs which represent intention types and pairs and this is all we need to test our theory and hypotheses.

We claimed that, assuming a *satisfaction* ordering of intention exists, interpretation of graphs would indicate the degree of satisfaction of a speaker in a dialogue. A notion of *balance* in dialogue was introduced, as a measure of the relative frequencies of different types of intention. We showed that the interpretation of intention graphs could give information about the balance in a dialogue. First, it was noted that *boomerang* intentions would indicate the interleaving of intentions in a dialogue, and a strong degree of balance should result if a dialogue included many of them. Second, it was noted that repetitions of intention type represent extremes in dialogue, and appear in intention graphs as *loops*. Such repetitions would tend to indicate *imbalance* in a dialogue. The distinction between repetition of intention-type and *repetition* of intention was pointed out. A fundamentally important fact about intention graphs was noted: they are universal representations which are independent of specific domains. For each domain, the intention graph will have a set of intentions which can be fleshed out with propositional content representative of that domain.

Finally, it was shown that the coherence of a dialogue can depend on the coherence of intentions, and if the ordering of intention pairs do not follow certain rules then a dialogue will be incoherent, and the speaker assumed irrational, unless there is other information to the contrary. The analysis of meaning, or topic, alone, in an incoherent dialogue would not reflect the incoherence of the dialogue. Hence, any theory of natural-language discourse analysis may need to incorporate a theory of intention analysis which will reflect the coherence of a discourse. A computer might be considered incoherent if it did not incorporate an ability to analyse intentions. It was pointed out that it was difficult to define hard and fast rules which could be used to define incoherent dialogues in terms of intention pairs. Also, any theory or model incorporating such rules would need to conduct an elaborate amount of processing. Such processing is beyond the scope of the work presented here.

As we are interested in the analysis of intentions in relation to computers, it is pertinent that a computational model of our theory can be developed. A requirement of any computational model will be that it can, al least, recognise and represent intentions in some manner. Our motivation is to analyse sequences of intention to facilitate more effective natural-language dialogue between people and computers and to do so by modelling the level of expertise of the user to effect context-sensitive and user-sensitive natural-language responses. First, the problem of intention recognition will not be easy, as we have seen already. The problem is that any given utterance will indicate a number of different intentions, depending on its context. For example, the utterance, "Can you pass the salt?," could be a request for *information* in the context where someone has a broken wrist, but in another context it might be an *instruction* request. To take an example from the natural-language dialogue consultancy domain for operating systems, a user may utter the query, "What is ls?," but it will only be the context of this utterance which will indicate that it is an *elaboration* rather than a *description*. Hence, any computational model of intention recognition will need to investigate the context of utterances. The syntax and semantics of utterances will give information about the recognition of intention, as will the context of the utterance. It might be possible to construct a natural-language parser which can use elements of syntax, semantics and pragmatics of natural-language utterances to recognise intentions and intention sequences.

Second, there is a requirement of finding a representation of intention graphs, or the sequences of intention contained within them. The most important feature of any representation will be that it can represent frequencies of intention pairs in the dialogue. We move on to describe a computational model which recognises and represents intention types and sequences, in the next chapter.

Chapter 6

A computational model of intention analysis

A theory of intention analysis for natural-language discourse processing has been described. The main principle of the theory is that the coherence of a dialogue can be modelled, in part, in terms of intention sequences. An intention representation can be used for modelling user satisfaction which can, in turn, be used for the generation of context-sensitive and user-sensitive natural-language responses. In order to test hypotheses about the use of this theory for natural-language processing we can incorporate the theory within a computational model. In turn, the computational model can be used to process natural-language dialogue. In the previous chapter at least two requirements for such a model were introduced. First, the model should have the ability to recognise intentions from natural-language utterances. Second, the model should have the ability to represent intentions and intention sequences. Now, we describe a computational model which is applied to the area of natural-language consultancy dialogues. The computational model uses a natural-language parser to recognise different intentions in natural-language utterances. A data structure, called an *intention matrix* is used to represent intention sequences. We apply the computational model in the domain of natural-language consultancy for computer operating systems. Hence, we choose a domain where people can type English questions to a computer program, and the program can answer these questions in English. The program, called OSCON (Operating System CONsultant), answers English questions, in English, about the UNIX and MS-DOS operating systems.

6.1 Recognition of intention

One of the main prerequisites of a theory of intention analysis is that intentions can be recognised in natural-language utterances. We saw in Chapter 4 that a great deal of work has been conducted on the recognition of intention in natural-language dialogue beginning with research by Cohen et al. (1982), right up to current research such as Hinkelman and Allen (1989). Now, we shall describe how intention categories are recognised within a computational model instantiated in the form of the OSCON operating system consultant program.

It was noted in Chapter 5 that the problem with recognising intentions is that there are many ways by which they are manifested in natural-language utterances. Any computational model must capture a range of possibilities of intention manifestation if it is to be effective. As discussed in Chapter 5, one way to conduct intention recognition is to examine the basic types of queries that people ask. By looking at the types, and their characteristics, a better idea of how to recognise them will be obtained.

In Chapter 5 we discussed the possibility of using natural-language consultancy as a useful domain for analysing intentions. We use the domain of computer operating systems, as it is a domain of natural-language consultancy which is well-defined, and has been used before by a number of researchers investigating theories in natural-language processing. A Unix Consultant (UC) (see Chin 1988, and Wilensky et al. 1984, 1986, 1988) has been implemented in Lisp, and acts as a natural-language consultant on the UNIX operating system. Another natural-language consultant, called the Sinix Consultant (SC) (see Hecking et al. 1988, and Kemke 1986, 1987) has been developed for the Sinix[1] operating system. Both of these systems are similar in scope and intent to the OSCON system. It is possible to do another *Gedankenexperiment*, like we did in Chapter 5, to investigate the types of intentions that might be expected in the computer operating systems' domain.

To start, there are at least four principal types of communicative act, or intention type, that people could use about computer operating systems. These are: (1) request-for-attribute of mentioned command (e.g. "What does rm do?"), (2) request-for-description of mentioned command (e.g. "What is more?"), (3) request-for-command for mentioned effect (e.g. "How do I see my file on the printer?"), and (4) request-for-description of mentioned concept (e.g. "What is a file?"). There are three cases of type (1): (1) request-for-effect (e.g. "What does rm do?"), (2) request-for-syntax (e.g. "What is the syntax of cp?"), and (3) request-for-precondition (e.g. "What is needed for rm?"). Each of these basic query types can also be asked in terms of options. Examples are, "What option of 'ls' shows the number of bytes in my files?" (request-for-option + mentioned command), "What does ls -l do?" (request-for-effect + option), "How do I rename a file without having reported errors?" (request-for-command + option), "What are the options on ls?" (request-for-options), "What does the -i option normally do?" (request-for-description of mentioned concept + option). Users could also ask queries involving command compositions. An example is, "How do I list my files and print them on the printer?" This sub-query involves a query about listing files (request-for-command) and then another sub-query about printing them on the printer (request-for-command). Now that we have formulated a set of principal types of query in the operating system consultancy domain they can be related to the consultancy domain intention categories that were defined by *Gedankenexperiment* in Chapter 5. These categories are: *information*, *description*, *instruction*, *elaboration*, *confirmation*, *explanation*, *guidance*, *repetition* and *nointention*. The result of such a correlation are shown in Table 6.1 below.

From Table 6.1 it is noted that a computational model, such as OSCON, should have the

[1]Sinix is a version of UNIX developed by Siemens AG in Germany.

information	*description*	*explanation*
request-for-command for mentioned effect	request-for-description of mentioned command	request-for-effect
		request-for-syntax
	request-for-description of mentioned object/concept	request-for-precondition

Table 6.1: Correlating query types with intention categories

capability of recognising at least three types of intention: *information*, *description*, and *explanation*. OSCON should also recognise *elaboration* intentions when an *information*, *description*, or *explanation* elaborates upon a previous query. Such elaborations can be recognised by checking equivalence of propositional content of the current and previous queries. In Chapter 5, *nointention* intentions were defined to be those where a speaker makes a statement irrelevant to a domain. When OSCON cannot parse an incoming query, because of its limited grammar and knowledge, such queries are, for all intents and purposes, *nointentions*. However, we will note that this is a limitation of OSCON, where OSCON does not recognise that the intention may be relevant to the domain, because of its limited grammar and knowledge. OSCON can also recognise *repetition* intentions, which leaves a capability of recognising, in real terms[2], five types of intention: *information*, *description*, *elaboration*, *explanation*, and *repetition*.

Looking at Table 6.1 we note that *information* intentions are those involving requests about operations in operating systems such as printing, copying and mailing files. It is predicted that these will be the most common type of query which users ask. Examples are, "How do I print my file on the printer?," "How do I delete my file?," and, "How do I copy my file to another directory?" *Description* intentions are those involving requests about commands, objects, or concepts. Examples are, "What are files?," "What is a pipe?," and "What is ls?". *Explanation* intentions are those involving explanations of objects, or concepts. Examples are, "What is the effect of ls?", "How do I use ls?," and, "What is the syntax of cp?".

In Chapter 5 it was proposed that the consultancy sublanguage of natural-language contained at least nine principal types of intention. The definitions of those types were given in general terms. Those types can now also be defined in terms of operating systems. The definitions might look like those in Table 6.2 below.

OSCON has currently got the capability of recognising six of the types shown in Table 6.2. OSCON does not currently have the capability of recognising *instruction*, *confirmation*, or *guidance* requests. *Confirmation* requests are not handled, as the capability of checking operating system actions against plans provided to execute them has not been built into OSCON. *Instruction* requests are not handled by OSCON, as the scope of OSCON does not cover executing commands in the operating system, but answering questions about how to execute commands[3]. Finally, *guidance*

[2]The recognition of nointentions is not counted as intention recognition as the system may recognise a valid intention as a nointention because of its lack of knowledge of the domain.

[3]*Instruction* requests were those which asked for the execution of a plan, or goal, rather than how to achieve the plan, or goal.

Intention	*General Definition*	*Operating System Domain Definition*
information	An intention requesting a PLAN[a] to achieve a specific GOAL[b] where the GOAL is described. e.g. "How do I cook this dish?" [a]A *PLAN* is defined as a set of actions to achieve some goal. [b]A *GOAL* is defined as an operation a speaker wishes to achieve.	An intention requesting an operating system command to achieve an operating system operation where the operation is described. e.g. "How do I print a file?"
description	An intention requesting a description of an object or concept. e.g. "What is Persia?"	An intention requesting the description of operating system concepts, objects, or commands. e.g. "what is UNIX?"
instruction	An intention acting as an instruction to achieve a GOAL, rather than how to achieve the PLAN, to achieve that GOAL. e.g. "Can you find out how many foreign nationals now live in Kuwait?"	An intention requesting the execution of an operating system command. e.g. "has my file been printed?"
elaboration	An intention requesting more information on a PLAN or GOAL. e.g. "Could you tell me more about Iraq?" following "Where is Iraq?"	An intention requesting more information on operating system commands, or the operating system itself. e.g. "how do i use more?" following "how do i see my file?"
confirmation	An intention requesting confirmation of a belief, or some PLAN believed to execute some GOAL. e.g. "Will sanctions stop Saddam Hussain?"	An intention requesting confirmation of a belief about the function of commands in the operating system, or the function of the operating system itself. e.g. "can i remove a directory with files in it?"
explanation	An intention requesting explanation or clarification of an item which occurred during the execution of a PLAN for a GOAL. e.g. "Could you tell me what you mean by U.N. resolution 611?"	An intention requesting explanation of a response from the operating system shell, or from the wizard. e.g. "What does cp -r mean?"
guidance	An intention requesting a PLAN for a GOAL where there is no explicit GOAL expressed. e.g. "What do I do next?"	An intention requesting help with operating system operations, or the operating system, where there is no operation described. e.g. "I don't understand what i'm supposed to do."
repetition	An intention which is a repeated request. e.g. "How many people live in the Gulf?" followed by e.g. "What number of people live in the Gulf?"	An intention repeating another intention. e.g. "How do I print a file?" followed by "How do I get a print out of my file?"
nointention	An intention which is not immediately relevant to the domain, or not understood by the hearer as being relevant to the domain. e.g. "Where does Strider live?" in the domain of Economics.	An intention not understood by the consultant system. e.g. "How do I eat a file?"

Table 6.2: Categorisation of intentions in terms of operating systems

intentions are beyond the current scope of OSCON, as these would require an enormous amount of processing to determine extended plans and goals of the user in a dialogue. For example, to answer the question, “What do I do next?,” the system would need to have extensive knowledge about the user’s goals and plans and his/her beliefs. In Chapter 5 it was proposed that a system should be able to discriminate at least nine intention types in order to adequately model the intentions in a natural-language consultancy dialogue. We shall show in later chapters that even by only recognising five types of intention OSCON has sufficient discriminatory power to demonstrate the benefits of intention analysis for natural-language dialogue modelling.

Now that we have described the types of intention recognised by OSCON we can go on to describe how the OSCON program conducts intention recognition in detail. In order to do that it will be appropriate to describe the computational model and its architecture. We show how OSCON recognises these intention types using syntactic, semantic and pragmatic processing. In order to discuss natural-language processing within OSCON we need to discuss the context of that processing within the system. Hence, we will give an overview of the system as a whole.

6.2 The Operating System CONsultant (OSCON)

The OSCON (Operating System CONsultant) program is a natural-language dialogue interface which answers English queries about computer operating systems (see Mc Kevitt 1986, 1988b, 1991a, and Mc Kevitt and Wilks 1987). OSCON enables a user to enter English queries and then answers them in English. The program is written in Quintus Prolog, and runs on a Sun-4 computer in real-time. OSCON can answer queries for over 30 commands from each of the UNIX and MS-DOS operating systems. OSCON handles four basic query types and five types of intention, as discussed in the previous section. OSCON can also answer queries about options on UNIX commands, and complex queries about command compositions. The system is intended to be used by varying types of users with different levels of expertise. The architecture of OSCON is modular so that it is easily updated and can be easily mapped over to other domains.

The OSCON program currently answers (1) the four basic query types, (2) queries about options, and (3) command composition queries, for both the UNIX and MS-DOS Operating Systems. The fact that queries are of a given type aids in understanding and generating answers to them. For example, queries of type (1) above will usually include a command name, and those of type (3) will usually not. Therefore, the parser for OSCON could check for command names and if it found them, predict that the query was of type (1). Also, the generator would generate an answer, in a particular format, depending on the type of query. Rules of thumb such as these also speed up the time it takes OSCON to answer queries. Although one can add such rules of thumb into the interface it does not reflect a short-cut to natural-language parsing. For example, it is unlikely that a short cut exists for understanding the query, “How do I print a file on the Imagen with no page burst?” Understanding queries is a combination of both (1) filtering the query type, and then (2) understanding the query. Examples of queries answered by OSCON are shown in Appendix A.

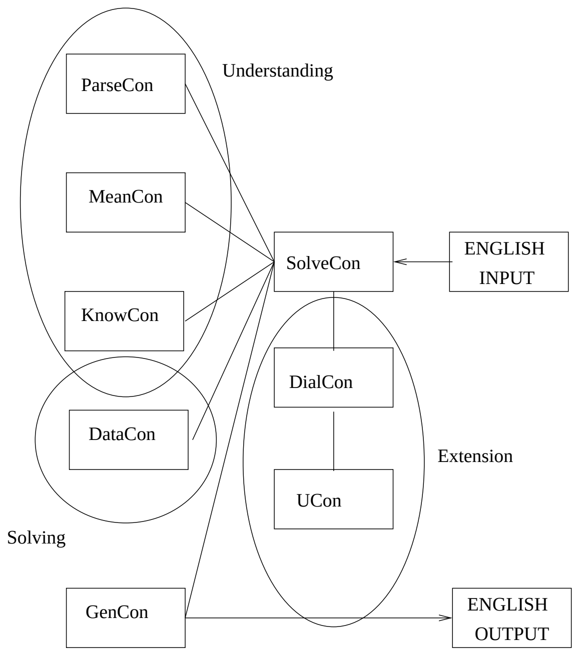


Figure 6.1: Architecture of the Operating System CONsultant (OSCON) system

6.2.1 Architecture

The architecture of the OSCON system consists of six basic modules and two extension modules. There are at least two arguments for modularising any system: (1) it is much easier to update the system at any point, and (2) it is easier to map the system over to another domain. The six basic modules in OSCON are as follows: (1) ParseCon: natural-language syntactic grammar parser which detects query-type, (2) MeanCon: a natural-language semantic grammar (see Brown et al. 1975, and Burton 1976) which determines query meaning, (3) KnowCon: a knowledge representation, containing information on natural-language verbs, for understanding, (4) DataCon: a knowledge representation for containing information about operating system commands, (5) SolveCon: a solver for resolving query representations against knowledge base representations, and (6) GenCon: a natural-language generator for generating answers in English. These six modules are satisfactory if user queries are treated independently, or in a context-free manner. However, the following two extension modules are necessary for dialogue-modelling and user-modelling: (1) DialCon: a dialogue modelling component which uses an intention matrix to track intention sequences in a dialogue, and (2) UCon: a user-modeller which computes levels of user-satisfaction from the intention matrix and provides information for context-sensitive and user-sensitive natural-language generation. A diagram of OSCON's architecture is shown in Figure 6.1.

ParseCon consists of a set of Prolog predicates which read natural-language input and determine the type of query being asked, or intention type presented, by the user. The four basic types of query, and five intention types recognised have already been discussed. For each type of query there are tests for characteristic ways that people might utter that query.

MeanCon consists of predicates which check queries for important information. There are predicates which check for mentioned (1) command names (e.g. “ls”, “more”), (2) command-effect specifications (e.g “see a file”), and (3) concepts, or objects (e.g. “file”, “directory”). In case (2) there are specific types of information searched for: (1) **verb** specifying action (e.g. “see”, “remove”), (2) **object** of action (e.g. “file”), (3) **modifier** of object (e.g. “contents”), and (4) **location** of object (e.g. “screen”). MeanCon also checks for option verbs (e.g “number”) and option verb objects (e.g. “lines”). MeanCon contains a dictionary of English words that define categories such as “person”, “modifier”, “article”, “quantifier” and “prepositions”.

KnowCon consists of a set of data files to represent knowledge about the domain language used for understanding English queries. Data files here contain information about English verbs which denote types of command or action. Examples of categories of action are: (1) creating, (2) screenlisting, (3) printerlisting, (4) sending, (5) transferring, and (6) removing. KnowCon also contains grammar rules for operating system objects like “date”, “file” and “directory”. The grammar rules encode characteristic ways in which people talk about the objects in English.

DataCon consists of a set of data files defining detailed information about operating system commands. This information is stored for the UNIX and MS-DOS Operating Systems. The data for UNIX is split among seven files about commands: (1) preconditions, (2) effects, (3) syntax, (4) names, (5) precondition options, (6) effect options, and (7) name options. The first four files contain basic data about commands while the last three contain data for options. For MS-DOS, data is contained in just four files which are similar, in spirit, to the first four here.

SolveCon is a solver which constructs and matches representations of user queries (called *Formal Queries*) against the knowledge base, DataCon, and produces an instantiated Formal Query which serves as an answer for the query. SolveCon is the heart, or driver, of the OSCON program because it contains the information for mapping English sentences into instantiated formal queries. It contains a set of complex rules which call other OSCON modules to determine (1) query type, (2) intention type, and (3) the instantiated Formal Query for that query. The determination of intention type is a two stage process where natural-language queries are first mapped into query types, and then into intention types. SolveCon also checks for repetitions by comparing the propositional content, or topic, of the current intention against that of the previous.

GenCon is the natural-language generator for OSCON and maps instantiated information from SolveCon into English answers. Here, there are algorithms for printing out (1) preconditions, (2) effects (or postconditions), and (3) syntax of commands. Also, there are predicates for printing out examples of the use of commands and command compositions. The type of query asked by the user determines the information presented in English to the user.

DialCon is the dialogue modeller for OSCON which updates the intention matrix, by locating the relevant cell in the matrix which needs to be updated, and increases its count by 1. DialCon indexes the cell in the matrix by pairing the current intention type with the previous.

UCon is the user-modelling component of OSCON. UCon derives a binary measure of user expertise, *expert* and *novice*. UCon applies a user-modelling function to the intention matrix to determine levels of user *satisfaction* and *dissatisfaction*. Initially, the user is assumed to be an expert. Subsequent changes in the levels of satisfaction and nonsatisfaction will result in changes in the level of user expertise. Such information is used by GenCon to generate context-sensitive and user-sensitive natural-language responses. UCon is explained in more detail in Chapter 7. The architecture of OSCON is now described in more detail.

6.2.2 Knowledge representation

One of the problems that crops up in building natural-language interfaces is the organisation of knowledge for the domain in some form so that it is effective. There are two types of knowledge stored in OSCON: (1) knowledge about natural-language, and (2) knowledge about operating systems.

The knowledge about language includes words used to refer to command actions. For example, a user may use the words "delete", "remove", "get rid of", "erase" and so on to ask a query about deleting files and directories. These words can be stored under the general category of remove. Also, there are many ways in which people ask queries about actions. For example, if you are asking about copying a file you will probably specify the file which you wish to copy. If you are asking about *displaying* you may specify what you wish to display and where you wish to display it. This type of knowledge is called "understanding knowledge" and is stored within a module of OSCON called KnowCon.

More detailed knowledge about operating systems is contained in a knowledge base, or database, called DataCon. This type of knowledge includes command preconditions, command effects, command syntax, and the names of commands. Also, stored here is (1) knowledge about options for commands, (2) English descriptions of operating system objects like "files" and "directories", and concepts like "pipe" and "filter", and (3) knowledge about plans, or possible command combinations (e.g. 'ls' can precede 'lpr' but the converse is not true). The knowledge stored here is for the UNIX and MS-DOS operating systems. The distinction between language knowledge and operating system knowledge is that one is language oriented, while the other is domain oriented. One type of knowledge is used for understanding queries, and the other for solving queries. This is the principle of separation of understanding and solving defined in Hegner (1988).

Knowledge for understanding queries (KnowCon)

There are two types of understanding knowledge stored in the KnowCon module: (1) data on operating system action reference, and (2) data on descriptions of operating system objects. The first type of knowledge includes sets of words or phrases that may refer to some operating system action or command. For example, the words, "print", "print out", and "get a copy" would indicate that the user was referring to printing something on the printer. Such words and phrases are stored as being associated with the general concept of printing. The second type of knowledge is used for defining the ways that users refer to operating system objects. For convenience the data has been split up into two types (a) data on files and directories, and (b) data on other operating system objects. In type (a) there are grammar rules specifying how users refer to files and directories, and in type (b) there are rules for how users refer to other operating system objects.

Referring to actions

While asking queries about operating systems users commonly use a limited set of verbs or verb phrases. For example, if a user wants to know about removing files or directories he/she will use, at least, the following verbs and phrases: "delete", "remove", "get rid of", "erase". Queries about copying may be referenced by: "copy", and "transfer." These phrases should be captured by the OSCON program. Phrases and words are stored under respective categories in the KnowCon module of the program.

It is possible to divide the set of operating systems commands into various categories. We have defined three major command categories and each of these has various subdivisions. The three categories are *listing*, *altering*, and *compiling* commands. Listing commands are those which display information about the state of files in an operating system. Altering commands are used to alter the state of files in the system. Compiling commands are used to compile files in the system rather than display/alter them. There are two types of listing command, (1) screenlisting, and (2) printerlisting. Screenlisting commands are those which allow the user to see information on the screen and printerlisting will do the same for the printer. There are three subtypes of screenlisting command, (1) display-file (e.g. *more*, *nroff*, *cat*), (2) display-file/directory-information (e.g. *ls*), and (3) display-system-information (e.g. *users*, *who*, *ps*, *jobs*). There is only one subtype of printerlisting command (e.g. *lpr*, *runoff*[4], *itroff*).

Altering commands are of three types, (1) creating (e.g. *gemacs*, *vi*, *mkdir*), (2) removing (e.g. *rm*, *rmdir*, *kill*), and (3) transferring (e.g. *mv*, *cp*). Each of these can be applied to either files or directories. A third type of command is compiling commands. These commands cannot be categorised under altering commands as they are really utilities which do not change files but use them. An example of such a command is *run*. The command-category hierarchy is shown in Figure 6.2.

In UNIX, files consist of other files and directories. It is important to point out that certain commands apply to files which are not directories, and others only apply to files which are directories. For example, "more" can only be used on files, not directories, and "rmdir" can only be used on directories, not files. Such information can be used to specify the preconditions for commands, and enables the system to detect errors in user queries, and inform the user about such errors. For example, if a user asked, "How do I use "more" to display a directory?" the system could locate the fact that the precondition for "more" is that it only works over files. Then the system could tell the user this information.

The rules listed in Figure 6.3 are examples of typical action rules in KnowCon. Rules [1] and [2] show typical verb phrasings used to denote the action of screenlisting. Rule [3] is for printerlisting,

[4] "runoff" is a command defined on the UNIX system at the Computing Research Laboratory, New Mexico State University, USA to load text formatting packages for word-processing a text file.

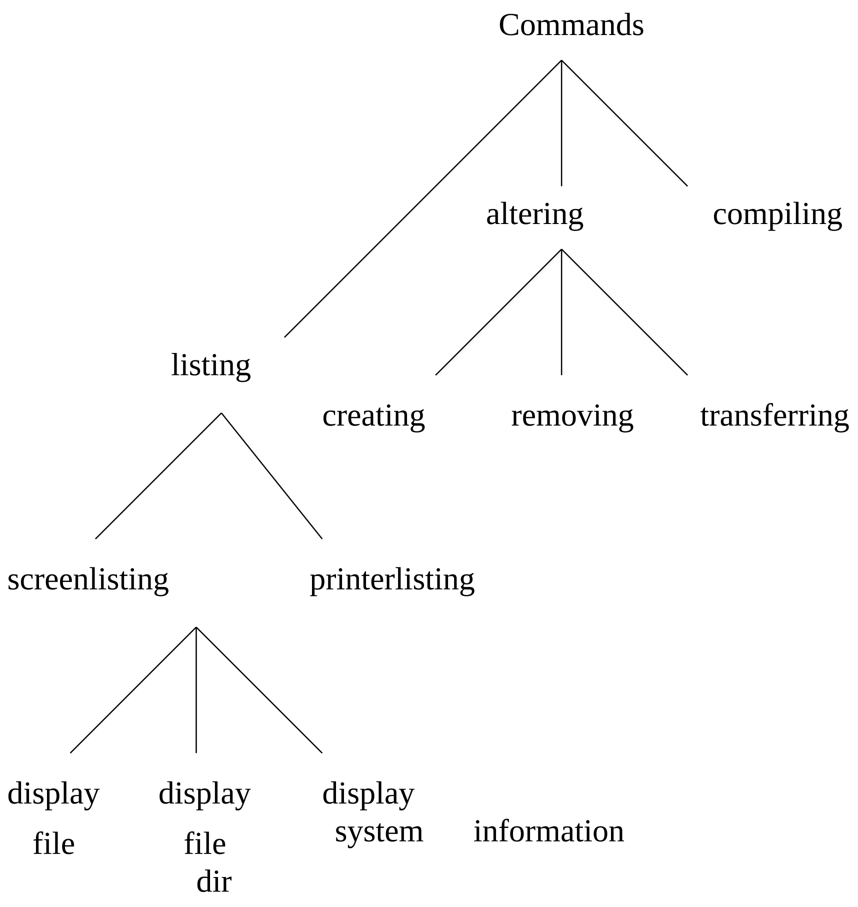


Figure 6.2: Hierarchy of categorised UNIX commands

and rules [4] and [5] for creating.

Referring to objects

Queries about operating systems often include reference to objects like files and directories. Typically, users will refer to the object which is operated over by some command. The phrasing of an English query will dictate the object present. KnowCon contains objects which are separated into two categories: (1) file/directory object specifications, and (2) other operating system object specifications.

The rules listed below in Figure 6.4 are examples of file/directory object specifications. File

```
[1] screenlist --> [see].
[2] screenlist --> [look, at].
[3] printerlist --> [print].
[4] create --> [edit].
[5] create --> [produce].
```

Figure 6.3: Rules for actions in KnowCon

```
[1] file --> mod, mod, filemod, [file].
[2] directory --> mod, mod, [directory].
[3] fileordir --> mod, mod, filemod, [files].
[4] location --> prep, mod, dir.
[5] slocation --> prep, mod, soutput.
[6] plocation --> prep, mod, poutput.
```

Figure 6.4: Rules for *file*, and *directory*, objects in KnowCon

object specifications include grammar rule definitions for files and directories. The first rule definition, given as [1] below, specifies that a file can be mentioned in a query by the word "file" preceded by three modifiers. The first modifier can be (1) a quantifier (e.g. "all", "some of"), and the second (2) a possessive (e.g. "my", "our"). Then *filemod* can be a modifier of type of file. Examples are "mail", "device", "plain", "executable" and so on. This grammar rule will capture many of the ways that a user might refer to a file.

The second rule, [2], shows that a directory may have two modifiers followed by the word referring to directory itself. Rule [3] shows that a file, or directory, could be referred to. This would happen when it is ambiguous as to whether the intended referent is a file or directory. The user, by using the plural for file may intend *directory*, as a set of files rather than just *files*. Rule [4] defines *location* to be a triple: (1) preposition (prep), (2) modifier (mod), and (3) directory (dir). Phrases like "..in my directory", "..in our directory", and "..in the directory" will match this rule. There are more specific definitions of location, for where the location is. Rule [5] is a definition of screen location in terms of (1) preposition (prep), (2) modifier (mod), and (3) screen-output (soutput). Preposition and modifier are self-explanatory and output is the location of output. This could be "terminal" or "screen". Rule [6] is the equivalent definition for a printer location. The output specification here is for a printer. Then *poutput* can be (1) "printer", (2) "imagen" or (3) "laser writer".

The second type of object definitions include objects other than those to do with files and directories. Some examples are shown below in Figure 6.5. Rule [1] shows the definition of a queue in terms of (1) a modifier, followed by (2) [printer, queue]. A query referring to "...the printer queue" would match here. There are definitions of users by rules [2] and [3]. Rules [4] and [5] define names. The predicates for other objects are simply definitions of the various ways by which users refer to such objects.

```
[1] queue --> mod, [printer, queue].
[2] users --> names, mod, [users], prep, mod, [system].
[3] users --> names, [system, users].
[4] names --> mod, [names, of].
[5] names --> [].
```

Figure 6.5: Rules for *other* objects in KnowCon

Knowledge for solving queries (DataCon)

The knowledge for solving, or matching query types against their answers, consists of files of data that describe detailed information about operating systems. In effect, these files contain the information for answering user queries. There are four types of knowledge requested: (1) Basic commands, (2) Options, (3) Concepts, and (4) Plans.

Basic command representation

There are four types of information about any command held in the database. These are (1) Preconditions, (2) Effects or Postconditions, (3) Syntax, and (4) Command Names. Preconditions are lists of objects that are constraints for a command to be executed. Some examples of preconditions for commands from UNIX are shown in Figure 6.6. Prolog facts [1] and [2] show that *more* and *cat* have the precondition, *file*. Hence, one can only *more* and *cat* files, but not directories. The command, *mkdir* has the precondition *directory* and *cp* has no precondition. Hence, one can only *mkdir* directories, and *cp* both files and directories.

```
[1] precon(more, [file]).
[2] precon(cat, [file]).
[3] precon(mkdir, [directory]).
[4] precon(cp, []).
```

Figure 6.6: Facts for *preconditions* of commands in DataCon

Effects, or postconditions, are definitions of the outcome of commands. The effect is defined by a predicate which has a name and three arguments. The predicate name is the action, and the arguments are (1) object, (2) object modifier, and (3) location. The facts below in Figure 6.7 show some effects for UNIX commands. Fact [1] shows the effect for the command *more.* The object for *more* is *file,* and its modifier *contents.* The location of output of *more* is the *screen.* One case of the *cat* command shown as [2] has the same effect as *more.* The other effect case of *cat,* in [3], is defined as concatenate, and describes the concatenation together of files. The command *ls* will either display directory contents, as in [4], or file information, as in [5], on the screen. The displaying-information command, *users,* will display usernames on the screen. Fact [7] describes the *gemacs* command which creates files, and rule [8], the *rm* command, which deletes them. Note that facts [6], [7], and [8] do not have object modifiers.

The syntax of commands is defined as a structure which contains the name of some command, and then its syntactic definition of use. Shown below in Figure 6.8 are some examples of syntax for UNIX commands. The syntax facts are three-place lists containing (1) Command name, (2) Optionname (filled in from context), and (3) Syntax description.

```
[1] comm(more, display(file, contents, screen)).
[2] comm(cat, display(file, contents, screen)).
[3] comm(cat, concat(file1, file2, file3)).
[4] comm(ls, display(directory, contents, screen)).
[5] comm(ls, display(file, info, screen)).
[6] comm(users, display(usernames, @, screen)).
[7] comm(gemacs, create(file, @, loc)).
[8] comm(rm, remove(file, @, loc)).
```

Figure 6.7: Facts for *postconditions*, or *effects* of commands, in DataCon

```
[1] syn(more, Optionname, ‘‘[more <filename>]’’).
[2] syn(cat, Optionname, ‘‘[cat <filename>]’’).
[3] syn(ls, Optionname, ‘‘[ls <directoryname>]’’).
[4] syn(users, Optionname, ‘‘[users]’’).
[5] syn(gemacs, Optionname, ‘‘[gemacs <filename>]’’).
```

Figure 6.8: Facts for *syntax* of commands in DataCon

Option representation

DataCon also contains information for option specifications of commands. DataCon has files for (1) Option Preconditions, (2) Option Effects, and (3) Option Names. There is no distinction for option syntax as syntax is modified in a consistent manner for commands with options.

Option Preconditions are defined as three-place predicates with the (1) Command Name, (2) Option name, and (3) Precondition List. Shown below in Figure 6.9 are a set of options for the various commands. The Precondition List contains the objects which must be present for the command to be executed. Facts [1], [2] and [3] show that each option for “cat” has “file” as precondition. Facts [4] and [5] show that for one option of “cp” (i) there is no precondition, and for the other option (r), “directory” is the precondition. Rules [6] and [7] show the preconditions for “ls”.

```
[1] opprecon(cat, n, [file]).
[2] opprecon(cat, s, [file]).
[3] opprecon(cat, v, [file]).
[4] opprecon(cp, i, []).
[5] opprecon(cp, r, [directory]).
[6] opprecon(ls, f, [directory]).
[7] opprecon(ls, l, []).
```

Figure 6.9: Facts for *option preconditions* in DataCon

The definitions of Option Effects are stored as strings of English words. This is merely a convenience for natural-language generation. The definitions enable the generator to give more detail about the specific effect of some option. Each fact here contains (1) Command Name, (2)

```
[1] opeffect(more,#,
        "set the window size to # lines").
[2] opeffect(more,c,
        "display each page after cleaning screen").
[3] opeffect(more,d,
        "prompt to hit space to continue at each screen").

[4] opeffect(cp,i,
        "prompt file name when overwriting").
[5] opeffect(cp,p,
        "preserve the modification information from
        the source").
[6] opeffect(cp,r,
        "copy each subtree rooted at that directory").
```

Figure 6.10: Rules for *option postconditions*, or *option effects*, in DataCon

```
[1] opcomm(number(lines), cat, n)
[2] opcomm(squeeze(blank-lines), cat, s)
[3] opcomm(display(non-printing-characters), cat, v)

[4] opcomm(include(hidden-files), ls, a).
[5] opcomm(display(directory-name), ls, d).
[6] opcomm(display(directory-content), ls, f).
[7] opcomm(display(group-ownership), ls, g).
[8] opcomm(display(long-listing), ls, l).
[9] opcomm(sort(file-ages), ls, t).
[10] opcomm(display(subdirectories), ls, R).

[11] opcomm(prompt(overwriting), cp, i).
[12] opcomm(preserve(modification-information), cp, p).
[13] opcomm(copy(subdirectories), cp, r).
```

Figure 6.11: Rules for *option names* in DataCon

Option Name, and (3) Option Effect with an English description. Some example of option effects are shown below in Figure 6.10.

Option Name definitions are similar in spirit to the Command Effect specifications defined in the previous section. However, the definitions here are for specific options. The option definitions have three arguments: (1) Option Effect defined as a structure in the form of, "action(object)", (2) the Command Name, and (3) Option Name definitions are shown below in Figure 6.11. The first three facts [1],[2] and [3] show the option variations on the UNIX command *cat.* The various options allow the displaying of file contents in a specific manner. The options for "ls" (facts [4]-[10]) and for "cp" (facts [11]-[13]) are also shown. Note that a characteristic of options is that sometimes they have the same action as the main action (e.g. display and display(non-printing-characters)), while other times they have a different action (e.g. display and squeeze(blanklines)).

```
[1] con([ada],[ada],
    ``Ada is developed on behalf of the U.S. Department of
    Defense for use in embedded systems.  Ada is the first
    practical language to bring together important features
    such as data abstraction, multitasking, exception
    handling, encapsulation and generics.'').

[2] con([working, directory],[working, directories],
    ``The directory you are working in.'').

[3] con([unix],[unix],
    ``The UNIX Operating System manages the resources of your
    computer system to perform useful work on your behalf.
    It is composed of three major parts: the kernel,
    the file system, and the shell.'').
```

Figure 6.12: Facts for *concepts* in DataCon

Concept representation

DataCon also contains a set of definitions of the possible objects or concepts which a user may wish to ask about. Objects include *files* and *directories*, and concepts include *piping* and *filtering*. Concept representations are three-place facts with (1) Concept name, (2) Concept name pluralised, and (3) English description of concept. The definitions for the objects, "Ada", "working directory" and "UNIX" are shown in Figure 6.12.

Plan representation

DataCon contains a list of possible Plans for action or command sequences which users may ask about. These command sequences are defined in terms of predicates which have five arguments. The first three arguments represent (1) an action, (2) an object, and (3) a location of output. The last two arguments define the second action and its output location. Two example facts are shown below in Figure 6.13. Fact [1] represents displaying directories on the screen, and then on the printer, and fact [2] likewise for files.

```
[1] plan(display,directory,screen,display,printer).
[2] plan(display,file,screen,display,printer).
```

Figure 6.13: Facts for *plans* in DataCon

Now that we have described the knowledge represented in OSCON, we can move on to describe how the system conducts natural-language processing.

6.2.3 Natural-language understanding

The most important part of any program which acts as a natural-language interface is the natural-language understander. There must be some means of mapping a user query, or intention, into a good meaning representation of that query; otherwise the user would be *dissatisfied*. The problem, of course, is that there are very many ways of specifying queries in natural languages like English. The first job of OSCON's natural-language interface is to use some type of syntactic filtering which determines the type of query being performed. The filter checks queries for certain objects, or phrases, and works as a fast mechanism for determining query type. The second job is that of determining the meaning of a query, or its content, once the type of query is determined. Let's take a look at the two components of OSCON which tackle these two problems. First, we will look at parsing.

Parsing natural-language queries (ParseCon)

The parser called ParseCon has the job of determining the type of query present. There are characteristic ways of asking certain types of queries, and the parser checks for these. The ParseCon module contains lists of the characteristic ways in which people ask queries about particular query types. The parser checks for the four basic types of query, and maps these into one of five intention types described in Section 6.1 above. Hence, parsing is a two stage process from English query into query type, and then from query-type into intention type.

One of the specific query types checked for is *request-for-description of mentioned concept or object* which denotes a *description* intention, as shown in Table 6.1. There are a number of characteristic phrases used to indicate this query type. Some of these are listed in Figure 6.14. The first set of facts [1-4] and facts [1a-1e] define the syntax of what would precede and succeed some concept respectively. The second set of rules [6-11] only define possible uses of syntax before the concept, but do not reflect any after-concept syntax.

Another query type checked for is *request-for-effect* which represents an *explanation* intention as shown in Table 6.1. Here, again there are a number of possible characteristic phrases. Some are listed in Figure 6.15. Similarly, there are facts for (i) request-for-precondition (*explanation* intentions), (ii) request-for-syntax (*explanation* intentions), (iii) request-for-command for mentioned effect (*information* intentions), and (iv) request-for-description of mentioned command (*description* intention) queries. Of course, there are syntax rules which will be compatible for all query types. However, this is not a problem, in principle, because there are other characteristics of query types which separate them.

Next, we describe processes in OSCON for determining the meaning of queries.

Determining query meaning (MeanCon)

The function of the MeanCon component of OSCON is to determine query meaning in terms of an internal language. This is obvious for most query types except for *request-for-command* queries

```
[1] firstphrase([what, does|X], o1).
[2] firstphrase([what, a|X],o2).
[3] firstphrase([what|X],o3).
[4] firstphrase([what|X],o4).

[1a] secphrase([mean|X],o1).
[2b] secphrase([is|X],o2).
[3c] secphrase([is|X],o3).
[4d] secphrase([are|X],o3).
[5e] secphrase([means|X],o4).

[6] wphrase([what, is, a]).
[7] wphrase([what, are]).
[8] wphrase([what, is]).
[9] wphrase([explain]).
[10] wphrase([describe]).
[11] wphrase([]).
```

Figure 6.14: Facts for *request-for mentioned concept* in ParseCon

```
[1] firstdesc([what|X],o1).
[2] firstdesc([does|X],o2).
[3] firstdesc([does|X],o3).

[1a] secdesc([does|X],o1).
[2b] secdesc([do|X],o2).
[3c] secdesc([have|X],o3).

[4] desc([what, happens, with|X]).
[5] desc([result|X]).
[6] desc([results|X]).
[7] desc([uses, of|X]).
[8] desc([effect, of|X]).
```

Figure 6.15: Facts for *request-for-effect* in ParseCon

(*information* intentions). These queries involve complex phrasings of English which describe the effects that a user wishes to execute. The MeanCon component of OSCON has the function of determining the occurrence of objects in user queries. There are seven types of object searched for: (1) command name, (2) option name, (3) verb reference, (4) object, (5) object modifier, (6) object location, and (7) concept.

MeanCon has a predicate called *findcmd* which searches for command names. The mention of command name is a good indicator of the type of query being asked. If a command name is present, this indicates that the query is probably about (1) command preconditions, (2) command effects, or (3) command syntax. MeanCon also has a predicate called *findopt* which searches for mention of option names in queries.

One of the most difficult types of user query to be understood by the system is *request-for-command for mentioned effect* queries. These are queries where the user knows what he/she wishes to accomplish, but does not know the command to do that. In these cases the user will specify, in English, some process, or effect, which he/she wants to be executed. The process/effect will usually be described with a verb which denotes the action/command to be accomplished. In some cases *request-for-command for mentioned effect* queries will involve action compositions. Examples exist in the queries, "How do I a copy a file and then print it?," and "How do I print a file having numbered lines?" In these cases (1) a primary verb will denote a main action/commend, and (2) a secondary verb denoting a secondary action, usually a restriction on the primary action. The primary verb will usually come before the secondary. Also, the restriction could be a definition of some option specification in conjunction with a request about some central action or command.

Therefore, MeanCon has algorithms which check for (i) Primary Verb, (ii) Primary Verb Object, (iii) Object Modifiers, and (iv) Locations. There are also algorithms which search for (i) Secondary Verb, and (ii) Secondary Objects which usually describe option effects.

While using *request-for-command* queries (or *information* intentions), the user will usually specify an action with a verb. This will be followed by the mention of an object such as a "file" or "directory". There may be a modifier of the object such as "contents". The location of the object may also be specified such as "printer," or "screen," or "directory".

There's a predicate in MeanCon called *findverb* which searches for verbs in user queries. When a verb is located this will determine the major category of action/command. Hence, the word "delete" will denote removing, "see" will denote displaying, and so on. MeanCon uses the stored verb-action structures in KnowCon to find verbs in queries, and their related action representations.

Another predicate called *findobj* searches for the object of a verb. Say, for example, the user had asked, "How do I see a file?" then the findverb predicate will locate the verb "see". The query is split up so that the segment after the verb is checked for an object. The object "file" is located and marked.

More complex queries may include modifiers and locations. Take the query, "How do I see my file contents on the screen?" In this case, the phrase "|..file contents on the screen?" is checked for an object by findobj. Then, "|..contents on the screen?" is checked for modifier by *findmod*.

Findmod locates "contents" as a modifier.

A predicate called *findloc* checks for locations in queries. Locations include "screen", "printer" and "directory". In our example, "|..on the screen?" is checked for location and "screen" is uncovered.

MeanCon has a predicate called *findcon* which checks for concepts in user queries. For example, the existence for the concept, "Ada" will reflect a query asking about the programming language, *Ada*.

Now that we have described the knowledge represented in OSCON, and how it understands natural-language queries, we can move on to describe the process by which OSCON generates natural-language answers.

6.2.4 Natural-Language Generation (GenCon)

The final task of the OSCON program is to map an instantiated Formal Query representation into an English answer. There are two types of answer which may be returned to the user: (1) stored English sentences describing concepts which are obtained from the DataCon knowledge base, and (2) English sentences mapped out from instantiated Formal Queries.

The natural-language generator for the OSCON system, called GenCon, is used to map instantiated formal queries into English answers. The generator has seven primary components as shown in Figure 6.16.

```
[1]  syntaxanswer: gives the syntax for a command
[2]  effectanswer: gives the effect of a command
[3]  preanswer: gives the preconditions for a command
[4]  commandanswer: gives the fact that some command is a command
[5]  objectanswer: gives a description of some object
[6]  exampleanswer: gives an example on the use of some command
[7]  pipeanswer: gives the commands involved in a
             piping example and an example of the piping
```

Figure 6.16: Rules for generation in GenCon

For each of the major query types various configurations of primary components are used. There are three types of request-for-attribute (*explanation*) query: (1) request-for-precondition, (2) request-for-effect, and (3) request-for-syntax. In these cases the components [3], [2] and [1] are printed out respectively. Printing out the syntax for some command is trivial. The syntax is already stored in the DataCon knowledge base. This is directly presented on the output. Displaying preconditions is quite trivial too, as all GenCon has to do is to print those preconditions retrieved from the DataCon precondition information.

Natural-language generation for *request-for-effect* queries is more complex. GenCon will print command effects by (1) checking to see if the output should be in plan/pipe form, and if it is, then generating the answer in plan/pipe form; (2) generating the (a) Command Syntax, and (b)

Effect for the command. The Effect is generated from the instantiated Formal Query produced by SolveCon which contains action, object, object modifier and object location. The latter information is generated in sentence form. Some interleaving information such as the output of prepositions between object modifier and location are handled too. For *request-for-command for effect* queries the latter algorithm is used.

Users can also ask queries about commands. For *request-for-description of mentioned command* queries (*description* intentions), information is output about the fact that the command in question is an operating system command. For *request-for-description of mentioned concept* (*description* intentions) queries, the answer is output from a pre-stored paragraph of text. The generation of the latter answers is simple, as the definitions of such concepts are just stored as English descriptions in the first place. Therefore, all GenCon has to do is to map the stored sentences into English answers. A simple algorithm has been developed to display English text on the screen. OSCON has the capability of presenting answers in a context-sensitive and user-sensitive form. The method for conducting this process is described in Chapter 7.

Now, all components of the OSCON system have been described except for the solver, or SolveCon component which matches queries to the database.

6.2.5 Processing natural-language queries within OSCON

Now that the data available to OSCON, in terms of both (1) understanding, and (2) operating systems knowledge, before the system begins to process a query, has been described, we move on to describe the rules which match user queries to database information. SolveCon determines the answer for a query through the following steps: (1) building an uninstantiated Formal Query from the query, (2) matching this structure to the DataCon database, (3) retrieving data from the database, and (4) using the data to build an instantiated Formal Query which is passed to the GenCon natural-language generator. We shall describe the solving process in three stages: (1) the algorithm used by SolveCon to specify the query and to retrieve data from the database, (2) the structure of instantiated Formal Queries returned by SolveCon to the GenCon generator, and (3) the determination of intention type.

The solving algorithm

The Solver basically searches queries for three types of information: (1) Command Names, (2) English Descriptions of command effects, and (3) Concepts. The search process in conducted in the following order.

[1] SolveCon checks to see if a command name is mentioned in the query. Then, (a) SolveCon checks if a description-of-option (e.g. -l) is mentioned. If (a) fails then (b) SolveCon checks if the query is request-for-description of option. This check is done by having ParseCon check the syntax of the query, having MeanCon check for an English Description of an option effect. If either (a) or (b) are satisfied SolveCon will retrieve from the database Option Preconditions, Option Effect, Option Syntax, and Option Name.

If (a) and (b) have both failed then (c) SolveCon checks if the query is a *request-for-precondition*, *request-for-effect* or *request-for-syntax* query. Here, SolveCon checks the syntax again using ParseCon. If (c) fails, then (d) SolveCon checks if the query is a *request-for-description of mentioned command*. ParseCon is involved here too. If either (c) or (d) are satisfied SolveCon will retrieve Command Preconditions, Command Effect, Command Syntax, and Command Name from the database. If (d) fails then SolveCon moves on to step [2].

[2] SolveCon checks the query semantics. In this case the user must have asked an English query with no command names. First, SolveCon has ParseCon check the syntax of the query. Second, SolveCon calls MeanCon to check for a Primary Verb, Verb Object, Modifier, and Location. SolveCon will retrieve Command Preconditions, Command Effect, Command Syntax, and Command Name from the database. Third, SolveCon has MeanCon check for a Secondary Verb (option action), and Secondary Verb Object. SolveCon will retrieve from the database Option Preconditions, Option Effect, and Option Syntax. If step [2] fails then SolveCon goes on to step [3].

[3] SolveCon checks the query semantics. In this case the user must have asked an English query involving no command names. Also, the query must be about command combinations, or pipes, otherwise step [2] would have passed. SolveCon checks for the existence of a command combination in the user query. SolveCon has MeanCon check for the existence of a sentence connector like "and". If this occurs then is it possible that the query involves command combination. SolveCon then calls the SolveCon algorithm again for (1) the segment of the query before the connector, and (2) the segment of the query after the connector. The data returned from (1) and (2) is integrated. If [3] fails then SolveCon tries step [4].

[4] SolveCon checks query syntax through ParseCon. Then MeanCon searches for objects or concepts mentioned in the query. Examples of an object is "Ada" and of a concept is "protection".

Structures returned

The step of SolveCon which succeeds will return an instantiated Formal Query to the generator. In step [1], if cases (a) or (b) succeed, an instantiated Formal Query will be returned containing the following: (1) Option Preconditions, (2) Option Effect, (3) Option Syntax, (4) Option Name, and (5) Query Type. In step [1], if cases (c) or (d) succeed, the instantiated Formal Query contains: (1) Command Preconditions, (2) Command Effect, (3) Command Syntax, (4) Command Name, and (5) Query Type.

In step [2] the Formal Query returned will contain slots for: (1) Command Preconditions, (2) Command Effect, (3) Command Syntax, (4) Option Preconditions, (5) Option Effect, (6) Option Syntax, and (7) Query Type. The complete structure will be instantiated when step [2] involves options. However, only parts (1), (2), (3), and (7) are instantiated when there is no mention of options.

With step [3] a list containing two instantiated Formal Queries is returned. Each Formal Query will contain: (1) Command Preconditions, (2) Command Effect, (3) Command Syntax, (4) Command Name, and (5) Query Type.

In step [4] a Formal Query with three pieces of information is returned. The structure contains the (1) Object or Concept Name, (2) Object or Concept Description, and (3) Query Type.

Now that we have described the processing of queries within OSCON it is possible to move on to describe how intention types are discriminated.

6.2.6 Determination of intention type

All the work for the determination of intention type has already been done by the solving algorithm. The intention type will be determined by the type of query returned from applying the solving algorithm. For example, the predicates for the determination of *description*, *explanation* and *information* intentions are shown below in Figure 6.17. Rules [1] and [2] show that a *request-for-description of concept/object* and a *request-for-description of command* will be recognised as *description* intentions. Rules [3], [4] and [5] show that *request-for-effect* and *request-for-syntax* queries will be recognised as *explanation* intentions, and *request-for-command for mentioned effect* as an *information* intention.

```
[1] intentiontype(Formalquery, Querytype, description) :-
                                         Querytype == oquery.

[2] intentiontype(Formalquery, Querytype, description) :-
                                         Querytype == coquery.

[3] intentiontype(Formalquery,Querytype, explanation) :-
                                         Querytype == squery.

[4] intentiontype(Formalquery,Querytype, explanation) :-
                                         Querytype == equery.

[5] intentiontype(Formalquery,Querytype, information) :-
                                         Querytype == cquery.
```

Figure 6.17: Rules for determination of intention type in SolveCon

However, the above predicates do not show how elaborations are recognised. To detect *elaboration* intentions it is necessary for OSCON to compare the topic of the current query to that of the previous. As was seen in previous sections the propositional content of English queries is represented by Instantiated Formal Queries. The Prolog predicates for selecting intentions in the case of elaborations are shown below in Figure 6.18. Rule [1] checks for cases where request-for-description of object/concept queries are elaborations of a previous query. The *checktopic* predicate matches the topic for the current query against that of the previous. A correct match will mark the query as an *elaboration* rather than as an *explanation.* Similarly, rules [2], [3], [4] and [5] check for *elaborations* of request-for-description of command, request-for-syntax, request-for-effect, and request-for-command for effect queries. In order to detect *repetition* intentions the rule in Figure

```
[1] intentiontype(Formalquery, Querytype, elaboration) :-
                                Querytype == oquery,
                                checktopic(Formalquery).

[2] intentiontype(Formalquery, Querytype, elaboration) :-
                                Querytype == coquery,
                                checktopic(Formalquery).

[3] intentiontype(Formalquery,Querytype, elaboration) :-
                                Querytype == squery,
                                checktopic(Formalquery).

[4] intentiontype(Formalquery,Querytype, elaboration) :-
                                Querytype == equery,
                                checktopic(Formalquery).

[5] intentiontype(Formalquery,Querytype, elaboration) :-
                                Querytype == cquery,
                                checktopic(Formalquery).
```

Figure 6.18: Rules for determination of *elaboration* intentions in SolveCon

```
[1] intentiontype(Formalquery, Querytype, repetition) :-
                          checkrepetition(Formalquery).
```

Figure 6.19: Rule for determination of *repetition* intentions in SolveCon

6.19 is invoked. This rule checks whether the propositional content of the current and previous queries are the same. Finally, if none of the above is matched then a *nointention* is assumed. In effect, this means that OSCON checks for five types of intention, plus *nointentions*.

Now that we have considered the recognition of intentions in natural-language dialogue we shall move on to discuss a second requirement of a computational model of intentional analysis, the representation of intentions.

6.3 Representation of intention

An important component of a computational model of intention analysis is the representation of intentions. OSCON represents two types of information on intentions: (1) Instantiated Formal Queries represent Intention type and propositional content of queries; and (2) frequencies of all intention pairs. With respect to (1), currently OSCON does not keep a history of all propositional contents. With respect to (2) it is noted that in Chapter 5 the theory of intention analysis calls for the modelling of dialogues through modelling intention sequences. We shall now consider how the computational model, OSCON, represents sequences of intention. A two-dimensional, six by six matrix within OSCON represents frequencies of intention pairs. The matrix is structured as shown below in Figure 6.20.

t_n

	INTENTION	In	De	El	Ex	Re	No
	Information	0	0	0	0	0	0
	Description	0	0	0	0	0	0
t_{n+1}	Elaboration	0	0	0	0	0	0
	Explanation	0	0	0	0	0	0
	Repetition	0	0	0	0	0	0
	Nointention	0	0	0	0	0	0

Figure 6.20: *Intention matrix* at rest

The matrix contains counts of intention pairs from natural-language dialogue. Various intention types are listed as rows and columns within the matrix. The matrix represents each intention pair by the horizontal axis indexing the first intention of the pair occurring at time, t_n, and the vertical axis the second intention at time, t_{n+1}. At the beginning of processing all intention-pair frequencies, and hence cells in the matrix are set to 0.

t_n

	INTENTION	In	De	El	Ex	Re	No
	Information	3	0	0	0	0	0
	Description	0	0	0	0	0	0
t_{n+1}	Elaboration	1	0	0	2	0	0
	Explanation	0	0	2	0	0	0
	Repetition	1	0	0	0	0	0
	Nointention	0	0	0	0	0	0

Figure 6.21: An example intention matrix

The matrix in Figure 6.21 is a snapshot of an example intention matrix which could occur during processing. The matrix indicates that there are three cases of an *information* $\rightarrow$[5] *information* intention pair, one case of an information $\rightarrow$ elaboration pair, two cases of elaboration $\rightarrow$ explanation pair, and two cases of an explanation $\rightarrow$ elaboration pair, in the dialogue. One case of

[5]We use the notation $X \rightarrow Y$ to denote the fact that intention X is followed by intention Y.

repetition occurs where an information intention has been repeated. There is a direct relationship between intention matrices and intention graphs. This intention matrix will represent the intention graph in Figure 6.22 below. However, as pointed out in Chapter 5, if an intention matrix does not maintain temporal tags with each intention then it will not be possible to determine an original dialogue structure from the intention matrix or graph.

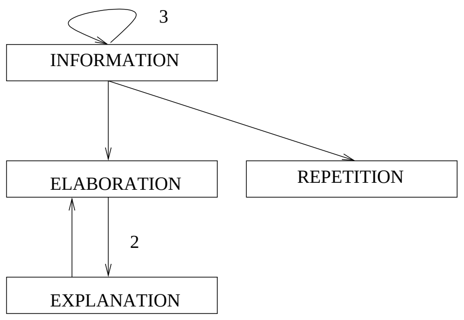


Figure 6.22: Intention graph represented by example intention matrix

The computational model, OSCON, fills up the intention matrix as follows. At the commencement of a dialogue, after the occurrence of the first utterance at time, t_0, there is no count assigned in the intention matrix. Each time an utterance is input to OSCON, at time t_n, its intention type is determined, and the frequency count of the intention pair $[t_{n-1} \rightarrow t_n]$ is updated. Hence the intention-sequence total count, if you sum all the cells of the intention matrix, will be one less than the intention count. DialCon is the component of OSCON concerned with updating the intention matrix. The predicate shown in Figure 6.23 below is the one concerned with updating the matrix. The previous intention row in the matrix which is salient is retracted, updated by a count of 1, and reinserted into the matrix again.

```
matrix(Newintention):-
                    retract(previousint(Intention)),
                    updateintention(Intention, Newintention),
                    asserta(previousint(Newintention)).
```

Figure 6.23: Rule in DialCon for updating the intention matrix

Hence, a computational model with the capability of recognising intentions and representing intention sequences now exists. Let's take a look at how an example query would be processed by the system.

6.4 An example

In this section we show an example of how the query "How do I see my files with numbered lines?" is understood, and answered, by OSCON. First, SolveCon will attempt to discriminate the intention type of the query. SolveCon attempts to find out the type of query being asked. Initially, SolveCon tries step [1] (see Section 6.2.5) to match the query as one mentioning a command and there is no match. Then step [2] (see Section 6.2.5) is tried and a match occurs. The query must be an English one, which mentions no command names. ParseCon is called forth, and a syntactic match is found. The fragment, "How do I," matches the syntactic form [how, do, i] for *request-for-command for effect queries* (*information* intentions).

Next, SolveCon calls MeanCon which analyses the meaning of the sentence. (i) Findverb checks for a verb and gets "see". From KnowCon, "display" is marked as the action. Then, (ii) findobj checks, "How do I," and, "my files with numbered lines," for objects. "Files" is matched as an object. Data from the query, in the form of uninstantiated formal query,

(i.e. **display(file,contents,*)**)

is matched against the DataCon database set of Effects and a match is found with

display(file,contents,screen)

This effect match from the query data to DataCon will allow SolveCon to retrieve (1) Command Preconditions ([file]), (2) Command Effect (display(file,contents,screen)), (3) Command Syntax (cat -n <filename>), and (4) Command Name (cat).

Next, the query is checked for existence of a secondary action. The findverb predicate retrieves "numbered" as a secondary action and its object is retrieved as "lines". This representation is matched against the option database and number(lines) from the query matches number(lines) in the Option Effect definition in the database. The DataCon database is referenced and (1) Option Preconditions, (2) Option Effect, and (3) Option Name are returned. The data retrieved is integrated to form an instantiated *Formal Query.* The instantiated formal query representation for this query is shown in Figure 6.24 below. Next, DialCon determines the intention type of the query, which is *information*, its sequence count is updated in the intention matrix, and the formal query is passed to GenCon.

```
cquery(display(file,contents,screen),       Command Effect
       ‘‘with numbered lines’’,             Option Effect
       [cat -n <filename>],                 Syntax
       [file]),                             Preconditions
       cat),                                Command Name
       -n).                                 Option Name
```

Figure 6.24: *Instantiated Formal Query* representation

The GenCon generator takes the Formal Query and maps it into an English answer. (i) effectanswer will print out "cat -n <filename> will display file contents on the screen", and (iii) optioneffect will display, "with numbered lines." The complete answer generated for this query is shown in Figure 6.25 below.

```
‘cat -n <filename>’ will display file contents
 on the screen with numbered lines.
```

Figure 6.25: Sample output generated by OSCON

6.5 Current state of the implementation

The OSCON program answers queries in real time when the Sun computer is at low load. OSCON answers four major types of query (1) request-for-attribute of mentioned command (*explanation* intention), (2) request-for-description of mentioned command (*description* intention), (3) request-for-command of mentioned effect (*information* intention), and (4) request-for-mentioned object or concept (*description* intention). There are three cases of type (1): (1) request-for-precondition (2) request-for-effect, and (3) request-for-syntax. In real terms this brings the query types covered up to six. The system answers the latter three query types with options. OSCON's database contains information on 30 UNIX and 30 MS-DOS commands incorporating their respective preconditions, effects, syntax and command names. OSCON has 20 grammar rules for understanding the ways that users ask queries about categories of commands. OSCON also contains 10 plan sets for possible combinations of commands.

6.6 Summary

We have described a computational model for the theory of intention analysis in natural-language dialogue. The computational model has the ability to fulfill two central requirements of a theory of intention sequencing. First, the model has the ability to recognise various types of intention in utterances which form a natural-language dialogue. Second, the model has the ability to represent intention sequences in a data structure called an *intention matrix*. The computational model has been applied to the problem of the analysis of intention types and sequences in the domain of natural-language consultancy or, more specifically, natural-language help on computer operating systems. The OSCON computational model can recognise, or discriminate, in real terms, five types of intention. The five intention types recognised are *information*, *description*, *elaboration*, *explanation*, and *repetition*. If none of these types is recognised then OSCON assumes a *nointention* has occurred. The OSCON computational model stores a representation of intention sequences from dialogue in an *intention matrix*. The intention matrix keeps a record of all intention pairs in a dialogue.

Of course, we cannot argue that just because the computational model, OSCON, can analyse intentions that that implies that the intentions in any natural-language dialogue can be modelled. However, we argue that we have shown that a computational model can be built to model intentions in an example case of natural-language dialogue. To model other types of dialogue specific knowledge about those domains would have to be incorporated into the computational model. At worst, a more elaborate computational model for all types of dialogue would be able to apply techniques similar to those provided here, and would only be limited by constraints of computational explosion in terms of time and space.

Now, it has been shown how a computational model can perform an analysis of intentions in natural-language dialogue, we can move on to show how such a mechanism can be used. Remember, our initial motivation was to use intention analysis to test hypotheses about natural-language dialogue processing, and in particular, to test the Intention-Computer hypothesis. That will require that the user can be modelled by the computational model in some way. Such a requirement involves the user-modelling component of OSCON, called UCon, which we introduced while describing the architecture of OSCON. In the next chapter the behaviour of UCon is described.

Chapter 7

Intention analysis for user-modelling

In the previous chapter a computational model of intention analysis, called OSCON, was presented. The next task is to show how that computational model can be used for applications such as user-modelling. Then the Intention-Computer hypothesis can be tested. It is possible to show how the computational model can be used to derive the level of expertise of a computer user from his/her intentions. It is shown that a representation of user behaviour is already implicitly available in the intention matrix. The satisfaction[1] of a user, and hence his/her level of expertise, can be derived from the intention matrix. It is shown that different regions of the intention matrix can be weighted to represent various degrees of user satisfaction. A user-modelling function is applied to the intention matrix to determine levels of satisfaction of users. Such a function will give information about user expertise levels, and that the responses of the dialogue component of the computational model can be modified to respond to these expertise levels.

There are at least three characteristics of a model of the level of expertise of a user. First, there will be some representation of user-behaviour to which a user-modelling function can be applied. Second, there must be a user-modelling function which determines the level of user expertise. Third, the system will take action, on application of that user-modelling function, modifying its behaviour. Each of these characteristics can be investigated in turn.

7.1 Representing user behaviour

In Chapter 5 it was shown that the intentions in a dialogue can be ordered into an ordering of satisfaction. One of the ways of determining the level of expertise of a user will be to investigate his/her level of satisfaction in the dialogue. If he/she is dissatisfied then that will indicate that he/she is having problems understanding the responses given by the system. Hence, the frequencies of different intention types in a dialogue will indicate the level of satisfaction of the user, and hence

[1]The meaning of *satisfaction* here is described in Chapter 5.

user expertise levels. Intention graphs were introduced in Chapter 5 as a means of representing intention pairs in a dialogue. In Chapter 6 the *intention matrix* was introduced as a computational model of the representation of intention sequences. Hence, as the intention matrix represents frequencies of intention it can be used to compute levels of user-satisfaction. It is possible to compute the level of user expertise by investigating the intention matrix. The intention matrix represents degrees of satisfaction, and any user-modelling function applied to the matrix needs to be applied selectively. The next step is to investigate how such a user-modelling function might be constituted.

7.2 Function of user expertise

In order to determine the level of user expertise a weighting can be applied to the intention matrix. An initial attempt might be to weight different types of intention pairs, and use that information to determine whether the level of user satisfaction, and hence expertise, is low or high. From Chapters 5 and 8 we proposed that intention sequences including *explanations*, *guidances*, and *repetitions* will indicate lower satisfaction than those incorporating *informations* and *descriptions*. Also, going back to Chapter 5, it was pointed out that extremes in dialogue would be represented by intention repetition in the dialogue, or as loops in an intention graph. It may be the case that because such sequences of intention are extremes, they should get a higher weighting than other sequences. Also, a better measure of satisfaction can be obtained if other indicators of dissatisfaction are given higher weighings. So, for example, sequences involving *elaborations* followed by *explanations*, and the complement of that, would get higher dissatisfaction weightings than say *informations* followed by *elaborations*, or *elaborations* followed by *informations*.

Various degrees of satisfaction will be indicated by different regions of the intention matrix. A user-modelling function will weight the intention matrix as follows. The regions indicated by "1" in Figure 7.1 will indicate higher levels of user satisfaction. In such cases users are going to, and from, the highest levels of intention satisfaction, i.e. *informations* and *descriptions*.

Next, the regions marked by "2" in Figure 7.2 indicate that user dissatisfaction is increasing. Users are moving to, and from, intentions indicating dissatisfaction. Such regions will be assigned a weight of "2".

Repetitions of intention, represented as loops in intention graphs, will occur along the centre diagonal of the intention matrix. This region of the matrix will indicate extremes of intention satisfaction or dissatisfaction. This region is marked by "3" in Figure 7.3. Note that *nointention* intentions are not incorporated in the weighting process as they do not reflect the satisfaction of the user, but inadequacies of the computational model.

It has now been shown how different regions of the intention matrix will represent different levels of user satisfaction. Each time an utterance is entered into the program, the user-modelling function can apply a selective weighting to the intention matrix which calculates the level of user expertise. At the commencement of processing the system can assume that a user is an expert, or

t_n

t_{n+1}

INTENTION	In	De	El	Ex	Re	No
Information	0	1	1	1	1	0
Description	1	0	1	1	1	0
Elaboration	1	1	0	0	0	0
Explanation	1	1	0	0	0	0
Repetition	1	1	0	0	0	0
Nointention	0	0	0	0	0	0

Figure 7.1: *Single weight* component of intention matrix

t_n

t_{n+1}

INTENTION	In	De	El	Ex	Re	No
Information	0	0	0	0	0	0
Description	0	0	0	0	0	0
Elaboration	0	0	0	2	2	0
Explanation	0	0	2	0	2	0
Repetition	0	0	2	2	0	0
Nointention	0	0	0	0	0	0

Figure 7.2: *Double weight* component of intention matrix

t_n

t_{n+1}	INTENTION	In	De	El	Ex	Re	No
	Information	3	0	0	0	0	0
	Description	0	3	0	0	0	0
	Elaboration	0	0	3	0	0	0
	Explanation	0	0	0	3	0	0
	Repetition	0	0	0	0	3	0
	Nointention	0	0	0	0	0	0

Figure 7.3: *Triple weight* component of intention matrix

that he/she knows as much as, or more, than itself. This is an example of use of the default *belief ascription* rule proposed by Wilks and Ballim (1987). The rule states:

Assume someone else believes the same as you believe unless you have explicit evidence to the contrary.

As each utterance is input, the program attempts to recognise the intention the utterance represents, and the respective intention sequence count is updated in the intention matrix. Hence, the level of user satisfaction can be computed dynamically as utterances appear in the input. A defence for maintaining the hypothesis that the user is an expert can be calculated by summing regions of relative satisfaction in the intention matrix. Hence, a formula for a measure of *satisfaction* is as follows (where [X → Y] represents the frequency of intention pairs from intention *X* to intention *Y*):

$$\begin{aligned} 3([I \to I]) + [I \to De] + [I \to El] + [I \to Ex] + [I \to Re] \\ +[De \to I] + [El \to I] + [Ex \to I] + [Re \to I] \\ +3([De \to De]) + [De \to El] + [De \to Ex] + [De \to Re] \\ +[El \to De] + [Ex \to De] + [Re \to De] \end{aligned}$$

The calculation of relative dissatisfaction can be calculated by summing regions of dissatisfaction in the intention matrix. The formula for a measure of *dissatisfaction* is:

$$\begin{aligned} 3([El \to El] + [Ex \to Ex] + [Re \to Re]) \\ +2([Ex \to El] + [Re \to El] + [Re \to Ex] + [El \to Ex] + [El \to Re] + [Ex \to Re]) \end{aligned}$$

It is noted that both the formula for *satisfaction* and *dissatisfaction* take weightings of respective

```
modeluser(Satisfaction, Dissatisfaction) :-
     retract(information(I, De, El, Ex, Re, No)),
     retract(description(I1, De1, El1, Ex1, Re1, No1)),
     retract(elaboration(I2, De2, El2, Ex2, Re2, No2)),
     retract(explanation(I3, De3, El3, Ex3, Re3, No3)),
     retract(repetition(I4, De4, El4, Ex4, Re4, No4)),
     asserta(information(I, De, El, Ex, Re, No)),
     asserta(description(I1, De1, El1, Ex1, Re1, No1)),
     asserta(elaboration(I2, De2, El2, Ex2, Re2, No2)),
     asserta(explanation(I3, De3, El3, Ex3, Re3, No3)),
     asserta(repetition(I4, De4, El4, Ex4, Re4, No4)),
     Satisfaction is 3*I + I1 + I2 + I3 + I4 + De + El + Ex + Re
                     + 3*De1 + De2 + De3 + De4 + El1 + Ex1 + Re1,
     Dissatisfaction is 3*El2 + 3*Ex3 + 3*Re4
                        + 2*Ex2 + 2*Re2 +2*Re3 + 2*El3 + 2*El4 + 2*Ex4.
```

Figure 7.4: user-modelling function

```
[1] updateuser(Satisfaction, Dissatisfaction) :-
              Satisfaction >= Dissatisfaction,
              retract(user(Level)),
              asserta(user(expert)).

[2] updateuser(Satisfaction, Dissatisfaction) :-
              retract(user(Level)),
              asserta(user(novice)).
```

Figure 7.5: Comparison of user satisfaction

regions of the matrix into account. Each time the user presents an utterance to the system the level of user satisfaction and dissatisfaction can be computed from the intention matrix. The component of OSCON which constructs the user-model is called UCon as was pointed out in Chapter 6. The Prolog predicate specifying the user-modelling function as shown below in Figure 7.4.

The levels of satisfaction and dissatisfaction are compared at each utterance cycle. If the dialogue is unbalanced[2], and there are too many dissatisfaction pairs, then the user will be assumed to be less satisfied, and hence a novice. However, if the user begins to use a number of *information* intentions, then the level of expertise will tilt back in favour of the user being an expert. This process is conducted by the Prolog predicates shown in Figure 7.5 below.

The computational model maintains a binary measure of expertise: *expert* and *novice*. A more sophisticated continuous measure of user expertise could be used but the binary measure will suffice for the current purposes of testing the application of the theory of intention sequencing to user-modelling. The degree of user satisfaction, and hence expertise, will determine the natural-language answers given back to the user in the dialogue. More information will be presented to

[2]The meaning of *balance* was defined in Chapter 5.

the user in cases where he/she is believed to be a novice, and less where the user is believed to be an expert.

The computational model's *beliefs* about the level of user satisfaction, and hence expertise, change constantly as a dialogue proceeds, and the model reacts dynamically to those levels of expertise.

It could be claimed that the complex apparatus of using separate measures for satisfaction and dissatisfaction is not required at all. One number could be used as a score of satisfaction and when dissatisfaction occurred a negative number could be added to the current score. Also, you could claim that instead of the complex weighing applied to the intention matrix a simple addition of different sizes of numbers could be used. However, we argue that an absolute measure of satisfaction and dissatisfaction may be needed for more complex processing of satisfaction. A single score only reflects relative satisfaction rather than the number of occurrences of satisfaction and dissatisfaction intentions. Also, a simple addition of different numbers only reflects relative differences in satisfaction and dissatisfaction. However, if computational cost became a problem, we could revert to using relative rather than absolute measures. In any event the *intention matrix* exists a priori as a representation of intention pairs in natural-language dialogue, and it will be used to solve other problems in natural-language dialogue processing. Now, it is possible to look at how the system will modify its responses depending on the level of user expertise.

7.3 Modification of system behaviour

A characteristic of any computational model of the level of user expertise is that it modifies its behaviour depending on the level. Otherwise, there would be little point in the system modelling such expertise.

The user expertise level will determine whether more, or less, information should be presented to the user. If the user is assumed to be an expert then less information would be provided than if he/she were a novice as it would be assumed that the expert has some knowledge already. This is a simplified heuristic and in future work we would hope to develop a more detailed answer generation system. Let us consider how the computational model, described in Chapter 6, would modify its behaviour based on levels of user expertise. Say, the user asks an *elaboration* request for information on command syntax: e.g. “What is the syntax for rm?” If the user is believed to be novice, then information about preconditions and effects for “rm” can also be given to the user.

Possibilities for the degree of information presented to a user depending on his/her level of expertise are shown in Table 7.1 below. The table shows that if the user is assumed to be an expert then the information returned to the user is selective with respect to the particular question he/she has asked. In most cases the syntax of the command will be returned. However, if the user is assumed to be a novice then all possible information is returned. Hence, the system modifies its generation of natural-language responses depending on the type of user it believes it is interacting with. The Prolog predicate which is part of the natural-language generator, GenCon, described in

Query Type	*Expert*	*Novice*
request-for-command for mentioned effect request-for-effect	effectanswer	syntaxanswer effectanswer preanswer commandanswer
request-for-syntax	syntaxanswer	
request-for-precondition	preanswer	
request-for-description of mentioned command	commandanswer	
request-for-mentioned concept/object	objectanswer	objectanswer

Table 7.1: *User-sensitive* natural-language responses

```
noviceanswer(Formalquery, Osys) :- commandanswer(Formalquery, Osys),
                                   syntaxanswer(Formalquery, Osys),
                                   effectanswer(Formalquery, Osys),
                                   preanswer(Formalquery, Osys).
```

Figure 7.6: Generation of novice responses

Chapter 5, and which provides novice answers, is given below in Figure 7.6.

That completes the discussion on how a computational model can determine the level of user expertise from an intention matrix, and how it modifies its behaviour in doing so. These achievements can be summarised as follows.

7.4 Summary

The intention matrix can be used to develop a computational model of the level of expertise of a user. There are three characteristics for such a model: (1) that there exists a representation of user behaviour, (2) that there exists a user-modelling function which determines the level of user satisfaction, (3) that the system takes action on application of the user-modelling function. The model uses the principle that intentions can be placed in an ordering of satisfaction. Hence, as an intention matrix represents the frequencies of different intention pairs in a dialogue it will also represent the degree of satisfaction of a user. A measure of the degree of satisfaction of the user is obtained by giving different weights to different regions of the intention matrix. Intention pairs occurring along the diagonal of the matrix are given the highest weighting as these pairs represent repetitions of intention types, or extremes, in dialogue. A user-modelling function which compares higher satisfaction intention types against lower satisfaction ones, and determines the balance of satisfaction in the dialogue, is presented. Finally, it is pointed out that the system can modify its behaviour by modifying its generation of natural language responses to give more, or less, information to the user, depending on his/her level of satisfaction. Example traces of OSCON's capability for conducting user-modelling are shown in Appendices A and B.

The motivation for developing a computational model for measuring user expertise levels is to demonstrate that the theory of intention sequencing can be used for effective natural-language dialogue between different types of people and a computer. This is the Intention-Computer hypothesis. It should now be apparent that a computational model can test this hypothesis. In the next chapter an experiment (Experiment I), which provides evidence for this hypothesis, is described. The experiment incorporates the computational model of intention analysis and user-modelling.

Part IV

Experiments

Chapter 8

Experimental results

The goal of the discussion in Parts 2 and 3 has been to describe existing research on discourse modelling, to present a theory of dialogue modelling using intention analysis, to describe how that theory can be incorporated into a computational model called OSCON, and to show how that theory can be used to model the satisfaction and hence levels of expertise of computer users. One of the motivations for establishing that goal has been the testing of the Intention-Computer hypothesis. Now, we describe three experiments, one of which tests the Intention-Computer hypothesis, and two of which test the Intention-Person hypothesis. The first experiment compares a version of OSCON which conducts intention-sequence analysis with a version that does not. This experiment tests the Intention-Computer hypothesis. The other two experiments are in the Wizard-of-Oz form where subjects are asked to conduct a number of tasks in the domain of computer operating systems. Each subject types his/her queries to the program, and a hidden person, called a Wizard, answers those queries. The subjects are not told that a person is answering their queries. These two experiments test the Intention-Person hypothesis. The data collected from each of these experiments, which will be analysed here, is available in Mc Kevitt and Ogden (1989a) and Mc Kevitt and Ogden (1989b) respectively.

The goals of the first experiment were twofold. First, we wanted to demonstrate that the OSCON system can conduct intention analysis over a natural-language dialogue by the processes of intention recognition, intention-sequence recognition, and intention-sequence representation, and, in turn, these processes can be used by OSCON to conduct user-modelling. Second, we wanted to compare three versions of OSCON, over the same natural-language dialogue, one conducting sequence analysis, the others iteratively less so. The first version, the *experimental*, has the capability of intention-sequence analysis, the second version, the *control*, does not have that capability, and the third version has the capability of counting only intention frequencies, or intention sequences with single intentions. We claim that the sample dialogue is representative of a typical natural-language dialogue. The results of the experiments show that the experimental performs much better than the first control with respect to intention discrimination and intention-sequence analysis. Also, the experimental performs much better than the second control with respect to user-modelling.

The first control obtains the wrong intention type for a number of queries. With respect to intention discrimination the experimental performs better approximately $\frac{1}{3}$ of the time, and with respect to intention-sequence analysis the experimental performs better approximately $\frac{4}{5}$ of the time. Also, the experimental modifies its natural-language responses, in relation to intention sequences, for user modelling, whereas the control does not modify its responses. The second control performs just as well as the experimental with respect to intention discrimination and intention-sequence analysis. With respect to user-modelling the experimental performs 34.8% better than the control. Hence, the first experiment provides a test of the Intention-Computer hypothesis and acts as an existence proof that a computational model can perform intention analysis and that intention analysis can be used for user-modelling.

The goal of the second experiment was to test whether there are recognisable types and sequences of communicative acts, and hence intention, in natural-language dialogue. The results show that it is possible to discriminate intention types and sequences, and furthermore, that there is a relationship between intention types and sequences and level of user expertise. The results show that intention type is a useful indicator of user expertise. The goal of the third experiment was to test whether there is a significant difference in the types of intentions used by subjects from different backgrounds of expertise. Hence the third experiment asks the question, "Is there a significant correlation between communicative act, and hence intention, frequencies and user expertise levels?" The experiment compared two groups of people, those who have taken a class in UNIX, the *experimental*[1], and those who have not, the *control*[2] The results show that the answer to the question is *yes*, and hence that intention recognition will be useful in building computer programs which interact with different types of computer users through natural-language dialogue.

Hence, the three experiments give positive evidence for the Intention-Person and Intention-Computer hypotheses.

In order to commence any experimental analysis one must ask oneself what sort of data one will collect. The answer to that question is provided by answering the question: what do we wish to do with that data? The data will be used to determine intention categories which will in turn be used to test predictions from hypotheses about intention in natural-language dialogue. It may be wise to look at a limited domain of natural language where the utterance types, and intention types, are more well-defined than in general natural language. While discussing our theory of intention analysis in Chapter 5 natural-language consultancy was selected as a domain of interest. In Chapter 6 a computational model of the theory was developed, and in Chapter 7 it was shown how the computational model could be applied to user-modelling. Again, the domain of natural language consultancy is used to conduct experiments. The experimental environment can be one where people type English questions to a computer program, and where the program answers those questions in English again. There are a number of forms by which experimentation could take place

[1]The term *experimental* is used to refer to an experiment which proposes to indicate a phenomena which the experimenter is attempting to demonstrate.

[2]The term *control* is used to refer to an experiment which proposes to indicate the state of the world when the phenomena from an *experimental* is not demonstrated.

in order to conduct experiments.

To test the Intention-Computer and Intention-Person hypotheses three experiments were conducted in the natural-language consultancy domain of help for the UNIX operating system. The first experiment to test the Intention-Computer hypothesis, involves a test as an existence proof using OSCON, a computational model of intention analysis. The second two experiments to test the Intention-Person hypothesis are two Wizard-of-Oz experiments. First, we shall discuss the experiment which tests the Intention-Computer hypothesis.

8.1 Experiment I

Experiment I acts as an existence proof of the viability of the theory of intention sequencing given in Chapter 5, by demonstrating that a computational model, incorporating a theory of intention sequencing, can conduct intention sequencing. Also, experiment I provides a test of the Intention-Computer hypothesis. Hence, intention analysis can be used to model the local[3] and global level of satisfaction, and hence expertise, of computer users. The goals of Experiment I were twofold. First, to test whether the OSCON system can conduct intention analysis over a sample natural-language dialogue by the processes of intention recognition, intention-sequence recognition, intention representation, and intention-sequence representation, which can then be used by OSCON to conduct user modelling. In turn, the user-model is used to generate natural-language responses which are sensitive to the *local* level of user expertise. Second, three versions of OSCON were tested over the same natural-language dialogue. The first version of OSCON had the capability of intention-pair sequence analysis. The second version had no ability to do sequence analysis at all. This version was not given the ability to detect elaboration or repetition intentions. The third version of OSCON was given the capability of detecting elaborations and repetitions, but additionally could only count intention frequencies, or intention sequences with single intentions. The sample natural-language dialogue was representative of a typical natural-language dialogue for UNIX natural-language consultancy.

First, as an *experimental*, the OSCON system was tested for its capability of intention-sequence analysis over a sample natural-language dialogue on UNIX help. This version of OSCON would have the capability of discriminating five types of intention: *information*, *description*, *elaboration*, *explanation*, and *repetition*. Then, one *control* was conducted to show the performance of OSCON over the *same* natural-language dialogue, when 'handicapped', with the capability of intention recognition, but without any capability of intention-sequence analysis. This control version of OSCON was not given the ability to detect *elaborations* or *repetitions*. This 'handicapped' version of OSCON would only be able to discriminate three types of intention: *information*, *description* and *explanation*. Finally, a second *control* test was conducted to show the performance of OSCON over the *same* natural-language dialogue, when 'handicapped,' with complete capability of intention

[3]By *local* satisfaction we wish to stress the fact that sometimes experts can act as novices on areas of a domain which they do not know well.

recognition, but without the capability of intention-pair analysis. Hence, this 'handicapped' version of OSCON would only be able to determine intention frequencies. Sample traces describing detailed results for the performance of the experimental and the two controls are shown in Appendix B.

It is argued that the natural-language consultancy dialogue on UNIX, for which the experiment comparing the experimental and controls was conducted, is representative of a typical natural-language consultancy dialogue. The dialogue is shown in detail in Appendix B. It contains examples of many of the intention pairs discovered in Experiments II and III. All intention pairs appearing in the sample dialogue are marked by 'X' in Figure 8.1. Looking at Figure 8.1 it is noted that the

t_n

	INTENTION	In	De	El	Ex	Re	No
	Information	X		X	X	X	X
	Description	X		X			
t_{n+1}	Elaboration	X	X	X	X		
	Explanation	X			X		
	Repetition	X				X	
	Nointention		X			X	

Figure 8.1: Intention pair sequences in sample natural-language dialogue

intention pairs existing in the sample dialogue act as a reasonable covering of the complete set of possible intention pairs. The data from experiments II and III had a large proportion of pairs with *information* intentions as sources and sinks. Many of the intention sequences in the sample dialogue exhibited similar intention pairs. The sample also contained queries about the topics of listing, displaying, removing, and copying files. Hence we argue that any results from tests conducted over the sample dialogue will be representative of the natural-language consultancy domain for UNIX, and in turn, the UNIX dialogue will be representative of a typical natural-language dialogue as was argued in Chapter 5.

It is possible to compare the behaviour of the experimental and the first control to determine respective performances. One criterion for comparison is the correctness of intention-type discrimination. The experimental discriminates all intention types for all 23 queries with no errors. The control has 9 intention discrimination errors out of the total of 23. Hence the experimental processes the selected dialogue 39.1% better.

This performance rate is obtained by finding the ratio of the number of intention-type discrimination errors for the control over the total number of queries and multiplying by 100:

$$= \frac{9}{23} \cdot 100 = 39.1\%.$$

The gap between the experimental and control is reflected even more when one looks at the differences in intention-sequence determination. The experimental can recognise the sequences shown in Figure 8.1 while the control can only recognise the sequences in Figure 8.2. The

t_n

	INTENTION	In	De	El	Ex	Re	No
	Information	X			X		X
	Description	X					
t_{n+1}	Elaboration						
	Explanation	X			X		
	Repetition						
	Nointention		X				

Figure 8.2: Intention pair sequences recognised by the control

experimental, which conducts intention-sequence analysis, processes the selected dialogue 81.8% better. This performance rate is obtained by finding the ratio of the differences in intention sequence totals between the experimental and control over the total number of possible sequences and multiplying by 100:

$$= \frac{(14-7)+(3-2)+(4-0)+(3-2)+(5-0)+(2-2)}{22} \;.\; 100$$

$$= \frac{18}{22} \;.\; 100$$

$$= 81.8\%.$$

It is noted that the levels of *satisfaction* and *dissatisfaction*, which act as a measure of user expertise, are dynamically changing for the experimental, but hardly change at all for the control. The experimental continuously modifies the character of its natural-language responses to the user, whereas the control never changes the character of its natural-language responses.

While comparing the experimental against the second control the control performs just as well as the experimental in terms of intention discrimination. With respect to user-modelling the control does not perform similar to the experimental. Out of 23 queries this control version of OSCON increases the wrong member of the set, {satisfaction, dissatisfaction}, 8 times. Hence the experimental performs user modelling on the selected dialogue 34.8% better. This performance rate is obtained by finding the ratio of the number of satisfaction update errors for the control over the total number of queries and multiplying by 100:

$$= \frac{8}{23} \;.\; 100 = 34.8\%.$$

It is noted that the levels of satisfaction and dissatisfaction, which act as a measure of user expertise, are dynamically changing for the experimental, but hardly change at all for this control. During most of the dialogue the control responds with too much information because of its eagerness to ascribe dissatisfaction to the user. The experimental continuously modifies the character of its natural-language responses to the user, whereas the control only changes once, the character of its natural-language responses. This phenomenon is caused by the fact that the control, because it is only conducting a simple frequency count, has lost the ability to determine when *elaborations*, *explanations*, and *repetitions* are indicating satisfaction rather than dissatisfaction.

Hence, the results act as existence proof of the viability of our theory as the computational model, OSCON, can conduct intention analysis, and additionally, can use that analysis to model the dialogue, and the user. Also the results show that OSCON performed better at intention discrimination and sequencing, and user-modelling, while having the capability of intention-sequence analysis. Note that the results in Experiment I are meant to be comparative in nature. There are no claims made about the proportion of intentions that will be discriminated by the computational model in an average natural-language UNIX consultancy dialogue. Also, there are no claims made here that the changes in system responses will actually increase the satisfaction of users. These types of analysis are beyond the scope of the work presented here.

There are a number of forms by which experimentation could take place in order to conduct experiments to test the Intention-Person hypothesis. We can move on to discuss these forms.

8.2 Forms of experimentation

There are a number of techniques by which data can be collected on natural-language dialogue between people and computers. Here are some of the ways in which this can be done:

(1) recording spoken conversations between subjects and experts

(2) recording batched, written-interactive[4]interactions between subjects and a computer operator[5]

(3) recording written-interactive interactions of subjects with a natural-language dialogue system

(4) recording written-interactive interactions of subjects with a computer program where a hidden person is supplying the return dialogue

While method (1) is very useful in providing data, it is tedious, and may provide subjective results as the interviewing expert may strongly bias the subject's response. Also, recent research has shown that human-human speech interaction provides quite different dialogue phenomena to

[5]The term *written-interactive* refers to the fact that while using a terminal, or console, people interact with the computer by typing using a keyboard.

[5]The term *computer operator* refers to the person who helps users with problems they have while using the operating system.

human-computer written-interactive dialogue (see Guindon 1988). If this is the case, and it is human-computer interaction which one wishes to model, then one is left with methods (2), (3) and (4). With (2) there is little simulation of a dialogue as queries are sent in batch[6] and replies are returned later on. Although this data would be useful for query, or intention, pairs it would not be useful for extended dialogue. In case (3) human-computer interactions are recorded. This is a very useful technique, except for the fact that today's dialogue systems are not robust, and tend to fail often. The problem with method (3) is that if the system fails continually then lots of data will be lost. This leaves method (4) which seems like a useful technique for obtaining data to design natural-language interfaces. Method (4) is one which is well known in the field of data acquisition, and is called the *Wizard-of-Oz* technique.

The Wizard-of-Oz technique applied to natural-language dialogue is one where subjects interact with a computer through written-interactive dialogue at a monitor, and are usually not informed that they are conversing with a hidden person. Subjects utterances are sent to another monitor where a hidden "Wizard" sends back a reply to the subject monitor. There have been many studies conducted on user-adviser dialogues using the Wizard-of-Oz experimental setup for domains such as statistical packages (see Guindon 1988, Guindon et al. 1986, and Slator et al. 1986), travel agent advice (see Brunner et al. 1989a, 1989b) and information retrieval (see Walker and Whittaker 1989, and Whittaker and Stenton 1989). The objective of many of these studies is to investigate written-interactive dialogue and use those investigations to help in building better and more robust natural-language interfaces.

The Wizard-of-Oz technique is beneficial as (1) it models human-computer written-interactive interaction, (2) there is no failure of the program in answering written utterances because there is no program, and (3) the setting is controlled as the subject has a set number of tasks to accomplish. The Wizard-of-Oz setting has been argued for strongly in the design and evaluation of natural-language interfaces (see Brunner et al. 1989a, Brunner et al. 1989b, and Guindon 1988). To test the Intention-Person hypothesis two Wizard-of-Oz experiments were conducted in the natural-language consultancy domain of help for the UNIX operating system.

8.3 Experiment II

The previous experiment gives empirical evidence which supports the Intention-Computer hypothesis proposed in Chapter 1. This next experiment was constituted so that real-world data could be collected in order to test the Intention-Person hypothesis. The second experiment was in the Wizard-of-Oz form and served as a pilot study to determine the types and sequences of communicative acts, and hence intentions, people use while interacting with a computer through typed written-interactive dialogue. The subjects were undergraduates from New Mexico State University (NMSU) in the state of New Mexico, USA, who were pursuing a variety of undergraduate degrees.

[6]The term *batch mode* is used to refer to computer processing which takes place over an extended period rather than in real-time.

There were 14 subjects in total. Details on how the experiment was conducted are given in Appendix C. The data collected during the experiment is available in Mc Kevitt and Ogden (1989a). The methodology of the experiment was as follows.

8.3.1 Methodology

The subjects sat down at a computer console and were read a set of instructions. The subjects had a number of simple tasks to complete using the UNIX operating system. Tasks involved typical and simple UNIX operations such as printing, listing, copying, and removing files. Each subject typed a number of questions in natural language about the UNIX operating system. A wizard sat approximately eight metres directly behind the subject in the same laboratory and answered the subject queries. All subject and wizard utterances were stored in a log file. The subjects were *not* told that the wizard would be answering their questions. It took each subject, on average, $1\frac{1}{2}$ hours to complete all the tasks in the experiment. There were, on average, six pages of log file data per subject.

Each subject's computer screen displayed a UNIX shell and a question and answer window. Tasks were specified in a task window. Each task was selected by the subject and the tasks were ordered randomly. The randomisation was done to stop wizards learning the sequence of tasks. When asking a question, the subject typed his/her question and sent it to the wizard's monitor. The wizard typed an answer in English, and, when sent, this arrived immediately in an ANSWER window on the subject's screen. Each wizard's screen showed the subject's questions in a QUESTION window, and the wizards typed and sent their replies in an ANSWER window. The subjects could not see the answers while they were being typed. Also, the wizard did not see the subject tasks although the wizard did see subject operations in the UNIX shell.

8.3.2 Subjects

The subjects were selected from a computer science undergraduate class (CS 110) which all students at the New Mexico State University (NMSU) must take. Each subject was asked to fill out a questionnaire which recorded demographic and historical data. The name, age, sex, and course of study of subjects were recorded. Also, the computer operating system's, and UNIX, experience of subjects was recorded. At the completion of the experiment each subject was asked whether he/she suspected a person was answering his/her questions. Table 8.1 below shows demographic data for each subject.

Of the 14 subjects 11 were female and three were male. Their ages ranged from 18-38 with an average age of 22 (22.14) years old. Many of the subjects were first year undergraduate students and this is indicated in Table 8.1 by the modal age, 18. The students had a wide range of courses of study. As indicated by "Y's" in the table, 4 subjects suspected that the wizard could be answering their questions though none of them were sure of this. The other 10 did not suspect this fact at all. One subject said that it was the time lag in answering, where the wizard was formulating his query, that caused her suspicions. Another subject had suspicions when the wizard over-helped

Subject	*Age*	*Sex*	*Course*	*Suspicions*
1	23	F	Marketing	
2	20	F	Computer Science	
3	21	F	Accounting/Business Computer Systems	
4	18	F	Accounting	
5	28	F	Dental Hygiene	Y
6	28	M	Computer Science	
7	23	F	Business Management	Y
8	20	M	Wildlife Science	Y
9	18	M	Business Computer Systems	
10	18	F	Psychology	
11	18	F	Accounting	
12	38	F	Community Health/ Nursing	
13	19	F	Accounting	
14	18	F	Psychology	Y

Table 8.1: Demographic data for subjects (Experiment II)

in answering her queries. The other two contemplated the possibility that the answers were being given by a wizard.

Table 8.2 below shows the list of subjects with their computer, and UNIX, experience. From the table we can see that three of the subjects reported that they had no computer experience, the rest reporting experience from less than 3 months to 1-2 years. The same three subjects reported having not used any operating system while the other subjects had used some operating system(s), mainly MS-DOS and PC-DOS. One subject (3) had used a number of different operating systems. Most of the subjects had not used UNIX although three subjects had between little (less than 6 months) and two years of UNIX. Thus, the subjects were, on the whole, what one might call novice, or *non-experienced*, UNIX users. It is important to note here that the number of years of computer, or operating systems, experience is not a strong criterion for determining the knowledge of a user. To have obtained other more complex information on subject histories the subjects would need to have been investigated in detail with respect to individual commands. For example, some users will have used UNIX for 1-2 years and will have learned much about the operating system while others will have used only a few commands. Hence, the best this information can give us is an indication of the level of knowledge of the user. The three wizards for this experiment had used the UNIX operating system before, and had adequate knowledge of the commands under analysis.

In analysing the data, of particular interest were the types and sequences of communicative acts, and hence intention types, in the dialogues. The 14 dialogues were investigated for these phenomena and the results are described below. It is important to point out that the analysis below relies on 14 subjects and stronger conclusions could only be drawn from larger scale studies.

Subject	*Computers*	*Operating Systems*						*Unix*
		MS-DOS	*UNIX*	*PC-DOS*	*CMS*	*VMS*	*MAC*	
1	Little	Y	Y					Little
2	1-2 Yrs	Y	Y					6 Mth-1 Yr
3	1-2 Yrs	Y	Y	Y	Y	Y		1-2 Yrs
4	1-2 Yrs	Y						None
5	1-2 Yrs			Y				”
6	$< 3Mth$			Y			Y	”
7	$< 3Mth$			Y				”
8	6 Mth-1 Yr			Y				”
9	6 Mth-1 Yr			Y				”
10	$< 3Mth$	Y						”
11	None							”
12	None							”
13	6 Mth-1 Yr			Y			Y	”
14	None							”

Table 8.2: Computer experience for subjects (Experiment II)

8.3.3 Intention categorisation

It was obvious from a first glance at the dialogues that subjects used a number of different types of utterance indicating different types of intention. Intention categories and sequences indicated the type of information they wished to be told.

On analysis of the dialogues, six types of intention were identified. This discovery was not a sudden process; it happened by an iterative analysis of the dialogues. The dialogues were judged, and it was noted that there were a number of commonalities in the types of utterances, and a representation of these was developed. It was also obvious that these intention types would not be the only ones to be expected in dialogue. On looking at different dialogues new types cropped up, and types changed their definition, along the way. In effect, the determination of intention types was a kind of bootstrapping process. Final decisions on the number of intention types were determined by how many types were needed to give expressive power to a person's intentions in the consultancy domain. The final categories decided upon, and their frequencies, are shown in Table 8.3 below. The table gives the intention name, and then a general definition of that intention, followed by a more specific definition for the UNIX domain.

One of the first things to be noticed in Table 8.3 is that some of the examples under certain categories could fit under others. For example, “How do I use more?” under *elaboration* could have been placed under *explanation*, as it could be a request for “an intention requesting explanation of a response from the UNIX shell, or from the wizard.” Also, “has oscon been printed” given as an example of an instruction request could be an indirect speech act[7] which really means, “Could you tell me how to find out whether oscon has been printed?” which would then fit under *information*. To emphasise the point further, “What does cp -r mean?,” could be an *elaboration* rather than an *explanation* as it is “An utterance requesting more information on UNIX commands, or UNIX

[7]An *indirect speech act* is an act where the intention of the utterance is not the same as the literal intention of the utterance. For example, the utterance, “It's cold in here,” can be used indirectly to mean, “Could you close the window?” Another example is, “Paul, will you be long at that terminal?”

Intention	*General Definition*	*UNIX Domain Definition*
information	An intention requesting a PLAN[a] to achieve a specific GOAL[b] where the GOAL is described. e.g. How do I cook this dish? [a]A *PLAN* is defined as a set of actions to achieve some goal. [b]A *GOAL* is defined as an operation a speaker wishes to achieve.	An intention requesting a UNIX command to achieve a UNIX operation where the operation is described. e.g. How do I print a file?
instruction	An intention acting as an instruction to achieve a GOAL, rather than how to achieve the PLAN, to achieve that GOAL. e.g. "Can you find out how many foreign nationals now live in Kuwait?"	An intention requesting the execution of a UNIX command. e.g. "has oscon been printed?"
elaboration	An intention requesting more information on a PLAN or GOAL. e.g. "Could you tell me more about Iraq?" following "Where is Iraq?"	An intention requesting more information on UNIX commands, or UNIX itself. e.g. "how do i use more?" following "how do i see my file?"
confirmation	An intention requesting confirmation of a belief, or some PLAN believed to execute some GOAL. e.g. "Will sanctions stop Saddam Hussain?"	An intention requesting confirmation of a belief about the function of commands in UNIX, or the function of UNIX itself. e.g. "can i remove a directory with files in it?"
explanation	An intention requesting explanation or clarification of an item which occurred during the execution of a PLAN for a GOAL. e.g. "Could you tell me what you mean by U.N. resolution 611?"	An intention requesting explanation of a response from the UNIX shell, or from the wizard. e.g. "What does cp -r mean?"
guidance	An intention requesting a PLAN for a GOAL where there is no explicit GOAL expressed. e.g. "What do I do next?"	An intention requesting help with UNIX operations, or UNIX, where there is no operation described. e.g. "I don't understand what i'm supposed to do."

Table 8.3: Categorisation of intention (Experiment II)

itself" This begs the question, "Is there any categorisation here at all?"

Second, it must be pointed out that there are further subcategorisations of the intention types shown in Table 8.3. For example, "How do i use more" could be called *form* where such an intention is:

form: An intention requesting the form of a PLAN to achieve a specific GOAL where the GOAL is described. e.g. How do I get to town by bus?

or more specifically:

form: An intention requesting the form of a UNIX command operation to achieve a UNIX operation where the operation is described.

And it is noted that this is now a more specific form of *elaboration.* The point here is that intentions can be further reduced and reduced in their definitions so that there are a number of subtypes of each type. It is also noted that there is some form of hierarchical ordering within the existing set and that explanations, instruction, confirmations, guidances and forms could all be types of elaboration. This is shown below in Figure 8.3. The goal here is not to delve into subtypes

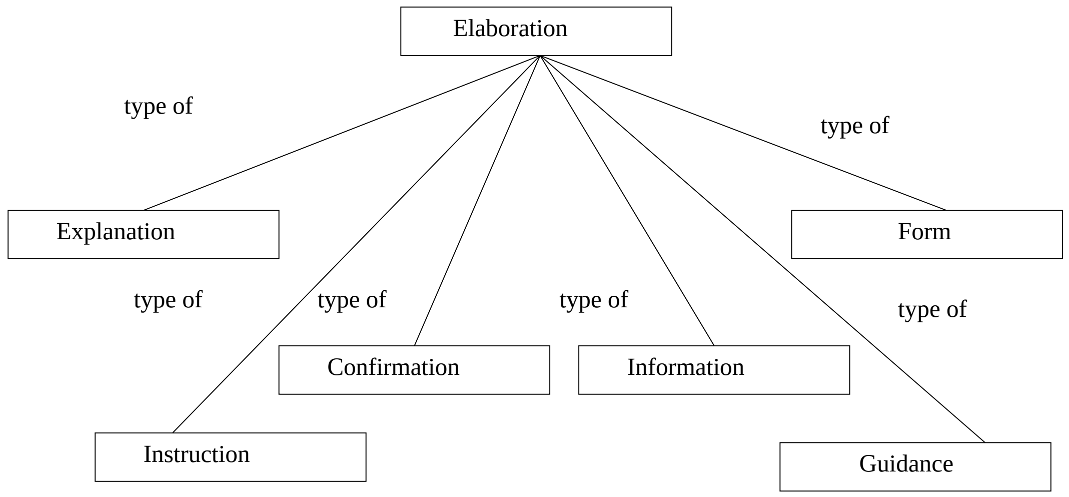


Figure 8.3: A hierarchy of intentions

although this is certainly an interesting avenue for further investigation.

The problem of categorisation is not unique to the consultancy domain as was pointed out in Chapter 5. In most categorisation tasks there will be example instances of entities where it will be difficult to decide which category is best for them. This problem comes from the fact that in many domains objects tend to differ from each other continuously rather than discretely. However, context is a very good indicator for the categorisation of utterances which may fall into a number of different intention categories. For example, when, "How do I use more?," comes after, "How do I look at a file?," then the former is obviously an *elaboration* of the latter. However, when, "How do I use more?," follows an incorrect entry by the subject causing a UNIX error message, and the subject has just asked about looking at files, then the intention is an *explanation.*

Hence, the context of an utterance is a major factor in determining the intention of an utterance. While analysing the data from Wizard-of-Oz experiments context was used to a great extent in determining intention types.

To solve the problem of selecting an intention category for an utterance, with respect to a subcategorisation hierarchy, the intention category which give the most specific fit, or the most specific definition for an utterance can be selected. Hence, "What does cp -r mean," is an *elaboration*, but more specifically, an *explanation*, and if the utterance preceding it is not related, the utterance is placed in the *explanation* category. Hence, only utterances which act as *explanations*, and do not elaborate upon a previous utterance, will be counted as *explanations*. Also, it is noted that *guidance* is mutually exclusive of *information*, i.e. *guidance* does not include a goal, or UNIX operation, whereas *information* does. Hence, there is a clear difference between *information* and *guidance*. We are interested in the differences because these differences will indicate differences in people's intentions.

It will be apparent by now that the intention categories uncovered in the UNIX operating system natural-language consultancy domain have much in common with the types generated in the *Gedankenexperiment* in Chapter 5. However, it must be admitted here that although the *Gedankenexperiment* in Chapter 5 was described before the experiments, full ideas on intention types were not formulated until after the the experiments were conducted. Hence, the processes of theory development and experimentation reinforced each other. It is noted by now that we seem to be interested in differences between intention types rather than their similarities. One might ask, what is the use of all these differences, and why cannot the intentions just be collapsed into one type? In order to answer that question, frequencies of various intention types are now discussed.

8.3.4 Intention frequencies

The frequencies of intention types are of particular interest in analysing the natural-language dialogues. The frequencies are of interest because it is believed that there will be a link between frequencies of intention type and user expertise. As described in the previous section the determination as to whether an utterance represented one type of intention, or another, was not a straightforward task. This determination involved looking carefully at the context of an utterance and making sure the utterance was placed in (1) the closest, and (2) the strictest defining, category of intention. The 14 dialogue log files were analysed for frequencies of each intention type. The frequencies of the intentions were determined by counting the quantity of each type, except *information* requests. *Information* requests were not counted as there were, as you might expect, such a large number of them, and it was the frequencies of other types which were more of interest. However, *information* intentions occurred on the order of 100's. The frequencies are shown in Table 8.4 below.

It is noted from the data that the most common type of intention was a request for *information* while the least common was an *elaboration* intention. Other intentions with high frequencies were *explanation*, *instruction*, and *guidance* requests. *Explanations* give the highest frequency, and

Intention	*Frequency*
information	$\sim$ 100′*s*
instruction	9
elaboration	4
confirmation	6
explanation	23
guidance	8

Table 8.4: Frequencies of intentions for all subjects (Experiment II)

the next highest is *instruction*, with *guidance* not falling too far behind. It may seem that the frequencies of intention types other than *information* are so small as to be insignificant for any practical natural-language dialogue system. However, it needs to be pointed out that if a natural-language system does not understand and answer a given utterance there will be a *domino* effect, and dialogue communication will break down. Hence, the failure of one utterance will cause other problems with respect to the dialogue. Also, it will be apparent that the types of intention which have low frequencies here tend to occur at very delicate points in the dialogue where problems are surfacing. Also, Whittaker and Stenton (1988, 1989) have shown that the failure of natural-language systems to analyse intention is one of the major reasons for failure of such systems.

To explain the above frequencies, with respect to user experience, it is postulated that inexperienced subjects approaching a new topic such as UNIX for the first time would be expected to need much guidance. It is to be expected that subjects would ask for many *explanations* of information given to them while learning a new subject. Likewise, they would ask for *instructions* and less *elaborations*. Of course, the task provided in the experiment will have influenced the communicative acts or intentions. It could be argued that subjects wish to learn enough information to complete the task, but do not want more information than required. This is a limitation of any experiment where subjects are asked to conduct a set number of tasks.

In a set of tables below (see Tables 8.5, 8.6, 8.7, 8.8, 8.9, 8.10), the complete set of utterances for each intention category are listed. In the case of *information* intentions we do not list all the utterances as there are a large number of them. The utterances are printed exactly as they were typed by subjects. As one looks at them one will wish to place some utterances into one category rather than another, and sometimes one would be right to do that, because, as was pointed out above, the boundaries between the categories are not clear. However, one can be satisfied with the categorisation here for two reasons: (1) the categorisation was determined with careful utilisation of the context of utterances, and, as it has been shown already, (2) it is the distinctions between these utterance types which is of importance, not their similarities. The distinctions will help in analysing the intention types present in a dialogue.

8.3.5 Intention sequences

As was pointed out in Chapter 5 another interesting feature of intentions is the sequences in which they come. The next data analysis involved an investigation of the intention sequencing in the

information
how do i copy a file
how do i remove a directory
how do i get the date
how do I know if it has produced a printed copy
what do i type to see who is using the computer?
⋮

Table 8.5: *Information* intentions (Experiment II)

instruction
is the computer hooked up to a printer
what other users are on the system today?
has oscon been printed
am i in the help directory
am i in oscon
is task complete
what other directory can I go to?
Does uswest now exist in your directory?
what is today's date

Table 8.6: *Instruction* intentions (Experiment II)

dialogues.

Most of the dialogues were saturated by *information* intentions. However, *instruction*, *elaboration*, *confirmation*, *explanation*, and *guidance* intentions did occur frequently in the dialogues. We call the set of the latter five intentions *interesting* as they seem to occur at interesting points in the dialogues where the dialogue breaks down, or a subject has problems. Uninteresting dialogues occur when subjects only ask *information* requests. In relation to the satisfaction spectrum, which we discussed in Chapter 5, *interesting* intentions will indicate degrees of *satisfaction* in a dialogue.

To derive data on intention sequences, the 14 dialogues were analysed to determine the sorts of intention sequences which contained the six component intentions. Naturally, of particular interest were *interesting* intentions as these would help to indicate where a subject was having problems in the dialogue. In fact, repeated *information* sequences were not counted. The analysis did not involve simply going through the dialogue log files and locating the point where an utterance indicating an *interesting* intention, and then determining the intention preceding it. The intention preceding each candidate intention had to be checked for coherence, i.e. the question was asked: did the later *interesting* intention follow coherently in a dialogue segment from the previous intention

elaboration
how do i use more?
how do i use logout?
can you give me more information on how... to find dir and change to dirl
how will "is" be entered?

Table 8.7: *Elaboration* intentions (Experiment II)

confirmation
to remove a directory do i need to remove all of the files
can i make a copy of a subdirectory?
can i remove a directory with files in it?
do I use the same method as the last task
DO I TYPE LS “MOVEMETO” “TOHERE”
is the directory named “rubbish” now gone?

Table 8.8: *Confirmation* intentions (Experiment II)

explanation
what does directory not empty mean
what does cp -r mean?
what now, I can’t seem to get it to go to the previous directory?
what do i do after i find out
explain more
where have all the files gone
if the file is not in that directory, how can I find out....
what directory it is in?
after I use cd how will I know if file1 is there?
if the file1 is not there where do I go?
what if there is no such file in diectory?
why is a directory not copied?
the rmdir rubbish doesn’t do the task, so what now
what do you mean by “trash”
when I command it to remove directory it says that
there is no such file or directory
what is oscon
why is my file movemeto not found?
What do I do if it is not empty?
What do I do if there is no such file or directory?
what if movemeto cannot access?
What do you do you do if you use the command rmdir and the file name but the directory doesn’t empty?
What do you do when there is no file or directory to remove with the name rubbish
What do you do if it still tells you there is no such file or directory?

Table 8.9: *Explanation* intentions (Experiment II)

guidance
how?
(how) will you help?
now what do i do next?
how do I get to the previous task
I don’t understand what I am suppose to do
Now what do I do?

Table 8.10: *Guidance* intentions (Experiment II)

with respect to the same topic?, or was the preceding intention part of another dialogue segment on another topic? As was pointed out in Chapter 5, intention pairs are the most useful primitive intention sequence for analysis of intention sequences in dialogue.

Hence, intention-pair sequences were counted in the dialogues. For any given *interesting* intention the question was asked, "what intention comes before this intention?" In Chapter 5 intention graphs were introduced as pictorial representations of sequences of intention in dialogue. An intention graph can now be used to show the results of the intention sequence count. The resultant graph is shown in Figure 8.4.

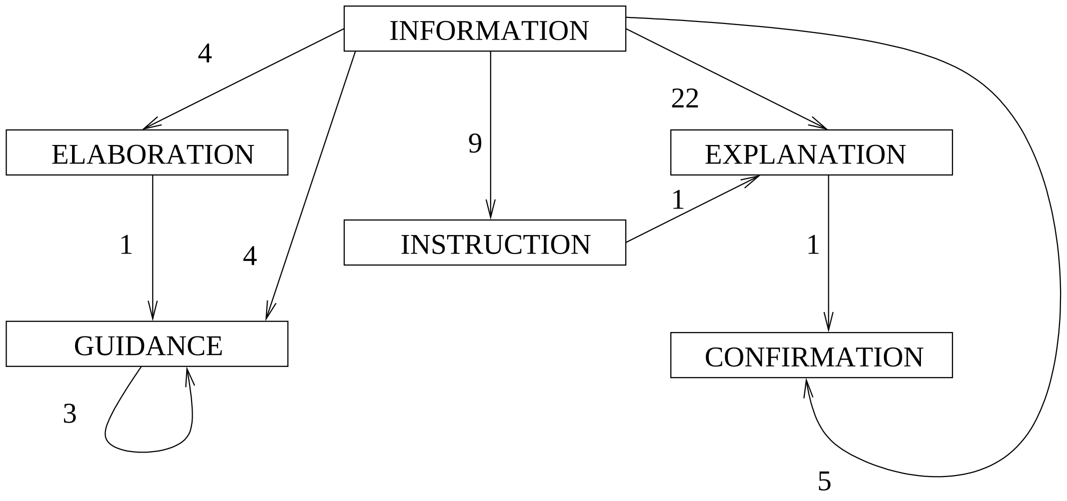


Figure 8.4: Intention graph (Experiment II)

This graph shows frequencies of intention pairs for the 14 subjects. Looking at the intention graph it is noted that *information* is the central node, i.e. this node has the most intentions following it; five in all. This is not surprising as *information* intentions are the most common in the dialogues. Also, the other nodes have only one, or no arc(s), leading from them. It is understandable that *information* intentions should be the most central as subjects will tend to ask queries about how to do something before they utter further intentions on continuation of their goals. It is noted that *information* intentions are prevalent in dialogue and other intention types are interspersed among these. Most dialogues begin with *information* intentions and it would be strange for a dialogue to begin right away with an *elaboration* or *explanation.*

It is also interesting to look at the arcs leading from other nodes. In one case a *guidance* request follows an *elaboration.* One could guess that this is expected if the subject has asked for an *elaboration* of something he/she does not fully understand, and even after the *elaboration*, needs further guidance. It is noted that we are touching on the notion of *satisfaction* of a subject, a feature introduced in Chapter 5.

On the right of the graph an *explanation* is followed by a *confirmation.* This is understandable as a subject may ask for confirmation after being given an explanation. Interestingly, only one subject asked for confirmation of an explanation although many subjects asked for explanations. This might indicate that subjects were happy with any explanations which they were given. However, if there were lots of confirmations following explanations this might indicate that subjects have problems with explanations given by the wizard. An *instruction* intention is followed by an *explanation* intention where again a subject is lost even after being given some information. It is noted that *guidance* has a loop which indicates repetition of an intention. Subjects who loop on this intention would be having considerable trouble. Also, there are four *guidance* intentions directly after *information* intentions. Note now what we are beginning to see: the sequences of intentions are telling us something about the people who are using them. It might be of interest to take a look at the subjects and categorise them somehow to see if there is a link between subject types and intention sequences.

8.3.6 Subjects and intentions

It is possible to categorise the subjects, to some degree, on the basis of their expertise with computers and operating systems. Remember, one of the pieces of information collected in the experiment was the level of expertise of the subjects. Again, however, care must be taken in counting intention types. It is not very easy to define the level of expertise of someone, as he/she may know a complex range of information which is difficult to define. For example, someone may not know much about UNIX, and a lot about other operating systems, and that would leave him/her in better stead than someone who knew nothing about any operating system. What we would like you to note here is that it is not obvious that expertise is an easy phenomenon to define. It is possible to segment the subjects on the basis of their experience with UNIX. A further segmentation can be exercised on the basis of whether the subjects had used computers before, or not. Three major categories of subjects seem to be evident. First, we shall name, *experienced*, those subjects who had experience with UNIX. Second, we shall name, *non-experienced*, subjects who had no experience with UNIX. Finally, *non-literate* subjects will be those who had little, or no, experience with computers. Table 8.11 shows the three categories of subjects, and their respective intention frequencies. As the number of subjects in each category is unequal, the data can be normalised, as if there were 10 subjects to get Table 8.12. The normalisation process is conducted by dividing each data item in a group by the number of the group and then multiplying by 10.

Intention	*Frequency*		
	Experienced (3)	*Non-experienced (8)*	*Non-literate (3)*
information	*	*	*
instruction	-	8	1
elaboration	2	2	-
confirmation	1	5	-
explanation	2	15	6
guidance	-	5	3

Table 8.11: Frequencies of intentions by expertise (Experiment II)

From Table 8.12 it is noted that experienced subjects asked very few types of intention other than information. There are 7 *explanations*, 7 *elaborations*, and 3 *confirmations*. More experienced

Intention	*Frequency*		
	Experienced	*Non-experienced*	*Non-literate*
information	*	*	*
instruction	-	10	3
elaboration	7	3	-
confirmation	3	6	-
explanation	7	19	20
guidance	-	6	10

Table 8.12: Normalised frequencies of intentions by expertise (Experiment II)

subjects would be expected to elaborate and confirm knowledge that they have. Also, if they were not extremely familiar with the domain they might be expected to ask for explanations, as they do. Non-experienced subjects like to ask a lot more *explanation* requests with *instruction* coming close behind. Note too that the non-literate subjects ask lots of *explanation* and *guidance* requests.

It is also possible to take a look at the intention sequences for various subject groups. Figures 8.5, 8.6, and 8.7 below show intentions for experienced, non-experienced and non-literate subjects respectively[8].

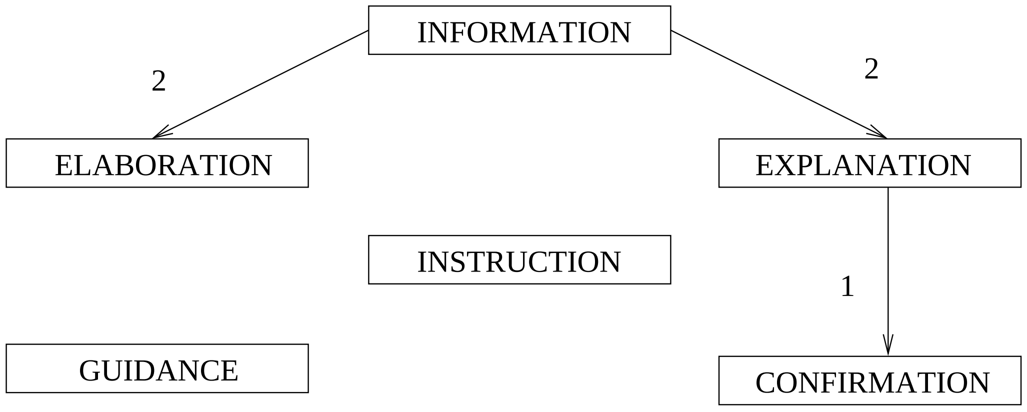


Figure 8.5: Intention graph for experienced subjects (Experiment II)

Taking a look at Figure 8.5 for *experienced* subjects, it is noted that a *confirmation* occurs after an *explanation* has occurred. There are no *confirmations* directly after *informations*. There are two elaborations after *informations* and no *guidance* or *instruction* intentions. This makes sense as more experienced subjects will want to experiment by asking for more information on a topic, and to confirm their beliefs about the topic. Also they will have less trouble with the topic. There will be a lesser tendency to utter many *interesting* intentions indicating strong degrees of *dissatisfaction*. Experienced subjects also have a better understanding of what a system can do, and so they will ask less *instruction* intentions due to this understanding.

In Figure 8.6 it is noted that the non-experienced subjects like to use *explanation* intentions following *information* intentions. Requests for instruction following informations are common

[8]Nodes representing intentions which have not occurred in the dialogues have not been omitted so one can observe which ones have not occurred.

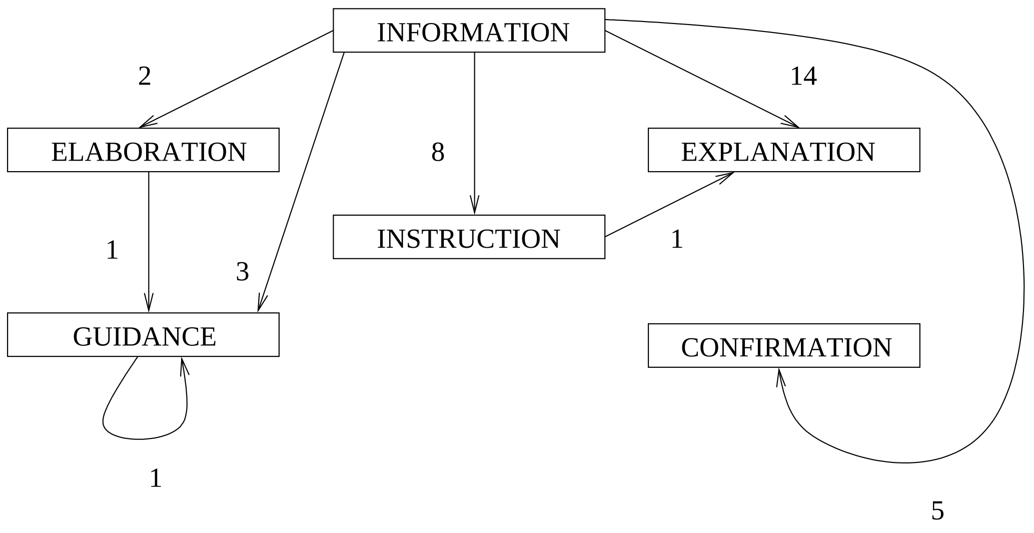


Figure 8.6: Intention graph for non-experienced subjects (Experiment II)

too. Note that one subject asks an *explanation* after an *instruction*. This suggests that the subject did not understand the information which was returned by the wizard. Looping occurs around *guidance*, a strong *dissatisfaction* intention, which indicates that a subject had trouble while uttering this request. Also, it is noted that there are three guidances from *information* and one from *elaboration*. There are five confirmations too. Hence, the intention graph for *non-experienced* subjects, is, on the whole, a lot more *interesting* than that for non-experienced subjects. The dialogues seem a lot more complex, and include more *interesting* intentions indicating increased dissatisfaction, than those of experienced subjects.

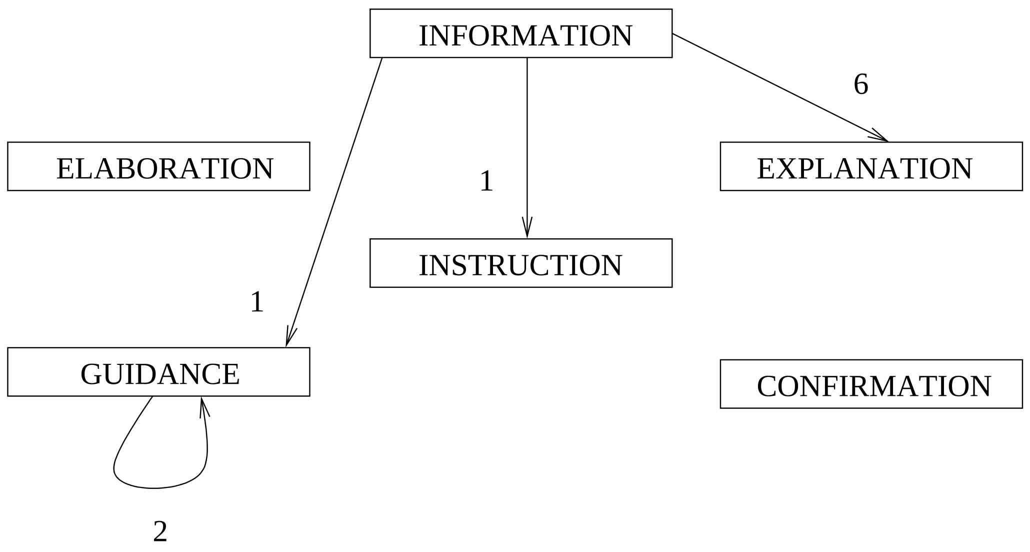


Figure 8.7: Intention graph for non-literate subjects (Experiment II)

Looking at Figure 8.7 it is noted that the non-literate subjects have many *explanation* intentions. An *instruction* request follows directly from an *information*, and is to be expected as

Subject	*Intention*				
	instruction	*elaboration*	*confirmation*	*explanation*	*guidance*
1			1	1	
2		2		1	
3					
4	1			1	
5	1		2	1	
6	4	1		3	3
7	1	1		5	
8			2	3	2
9				1	
10			1	1	
11	1			3	
12				3	
13	1				
14					3

Table 8.13: Frequencies of intentions for each subject (Experiment II)

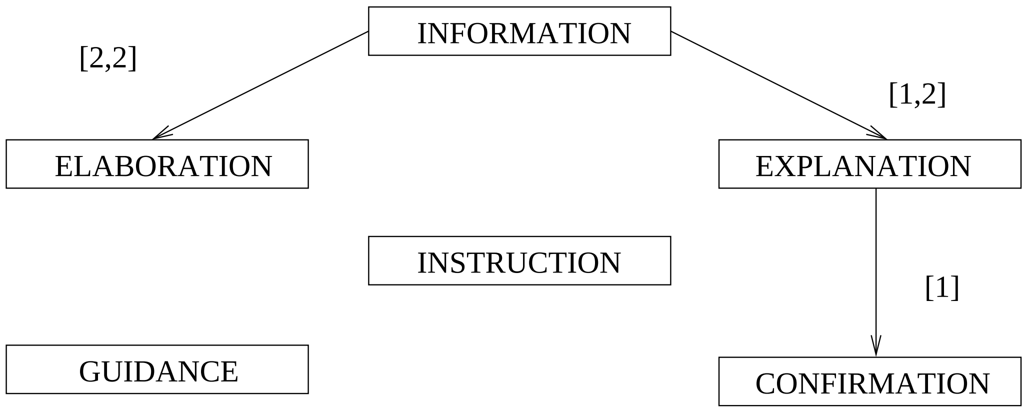


Figure 8.8: Intention graph, by subject, for experienced subjects (Experiment II)

non-literate subjects do not realise the capability of the wizard/system. Note that there are three guidances, two of which are loops. This is an indication of the subjects' having difficulty.

It is possible to develop a deeper analysis and determine which subjects, in particular, are using which intentions, in particular. In Table 8.13 the intentions for each subject are shown.

In Figure 8.8 the intention sequences for experienced subjects are shown. We introduce a new form of label on arcs in intention graphs to indicate the set of subjects who used a given intention pair. Subjects 1 and 2 are represented in this intention graph. From Table 8.2, shown earlier, it is noted that these subjects had at most 1 year of UNIX experience. Also, they had used only two operating systems. The other experienced subject, subject 3, is not represented at all as she did not use any *interesting* intentions. She had used a number of operating systems, and had 1-2 years of UNIX experience.

In Figure 8.9 we note that subjects 5 and 8 asked two confirmations each. From Table 8.1 shown earlier it is noted that these subjects studied subjects where they might have little interaction with

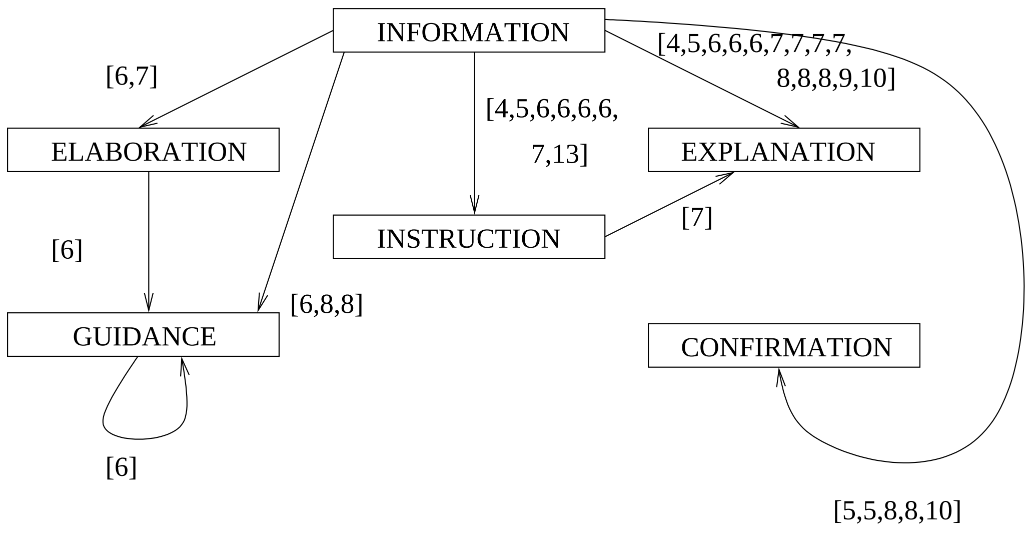


Figure 8.9: Intention graph, by subject, for non-experienced subjects (Experiment II)

computers and might have a greater tendency to confirm information they believed to be true. Also, subjects 6, 7 and 8 liked to ask a lot of explanations. From Table 8.2 we note that these subjects had mainly used the PC-DOS operating system. Also, they had less than 1 year of computer experience. Subject 6 asked four *instruction* requests and as he was a computer science undergraduate he might have expected the computer/wizard to do more than it actually could. Subjects 6 asked 3 guidance requests, looping on one. He had less than three months computer experience. Subject 8 asked two guidance requests and he had under one year of computer experience.

It is also noted here that of the intention pairs used by *non-experienced* subjects, 19 *interesting* intention pairs are used by males and 16 by females. Also, note that only 3 of the 14 subjects are males. This could indicate something about the hypothesis that females are better at linguistic reasoning as compared with males. However, more detailed experiments, comparing males and females, would need to be conducted to provide evidence for this hypothesis.

In Figure 8.10 subjects 11 and 12 asked three explanations each. These subjects had no computer experience at all. Also, of interest is the fact that subject 14, who had no computer experience asked three *guidances*, looping on two of them.

One of the things which will have become apparent by now is that in many of the intention graphs there are loops where subjects have repeated the same intention type. As was pointed out in Chapter 5 this phenomenon will be interesting as it would indicate that a subject has problems achieving his/her intention. Let's take a look at some of these loops.

8.3.7 Intention loops

As has been discussed before in Chapter 5, loops are representations of situations where subjects repeat intention types. If loops occur on *interesting* intentions it is obvious that subjects are having trouble understanding the domain, information presented about the domain. First, an inspection of the loops for experienced subjects can be conducted.

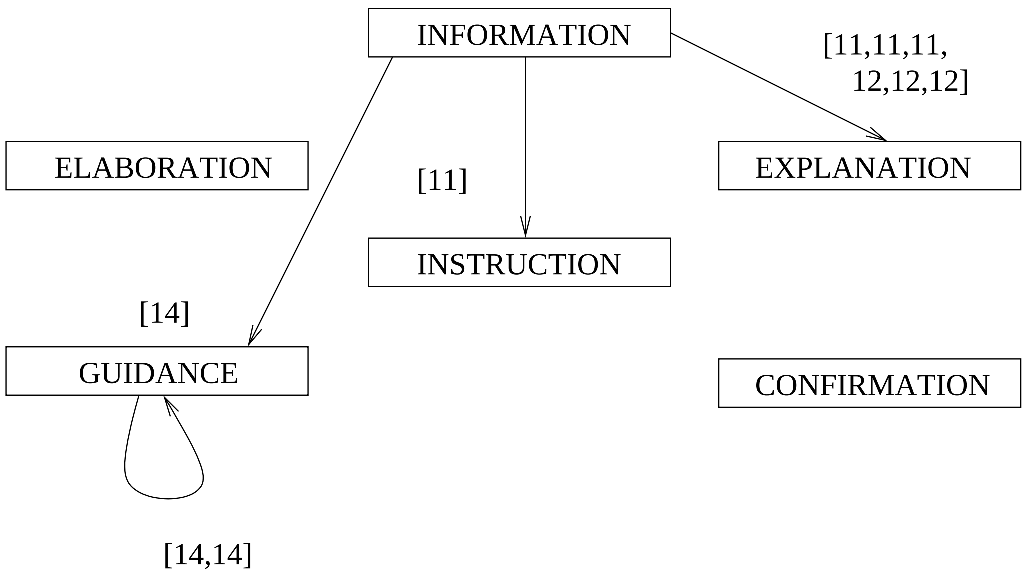


Figure 8.10: Intention graph, by subject, for non-literate subjects (Experiment II)

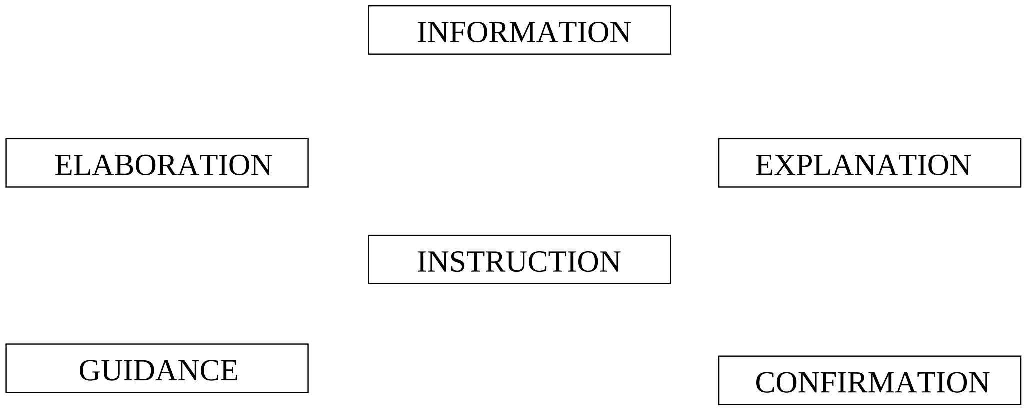


Figure 8.11: *Loop* intention graph for experienced subjects (Experiment II)

In Figure 8.11 it is noted that there are no loops. This is to be expected as experienced subjects should know more about the domain, and hence, have less problems.

Note that in Figure 8.12, for non-experienced subjects, there is one loop at guidance and it has been traversed by subject 6. Note from Table 8.2 that subject 6 had less than three months of computer experience. Such questions would indicate that a subject was non-experienced.

In Figure 8.13 for non-literate subjects there is a loop at guidance. Subject 14 performs a *guidance* intention twice. We note that this subject has no computer experience and is 18 years old.

In summary, experiment II shows that utterances representing intentions can be usefully categorised into different types. In this analysis six distinct types of intention were defined: *information*, *instruction*, *elaboration*, *confirmation*, *explanation*, and *guidance*. This primitive set of intention types seems to circumscribe the set of intentions required to get by in a consultancy domain. As was argued in Chapter 5 these intention types will be typical of most consultancy dialogues. Also,

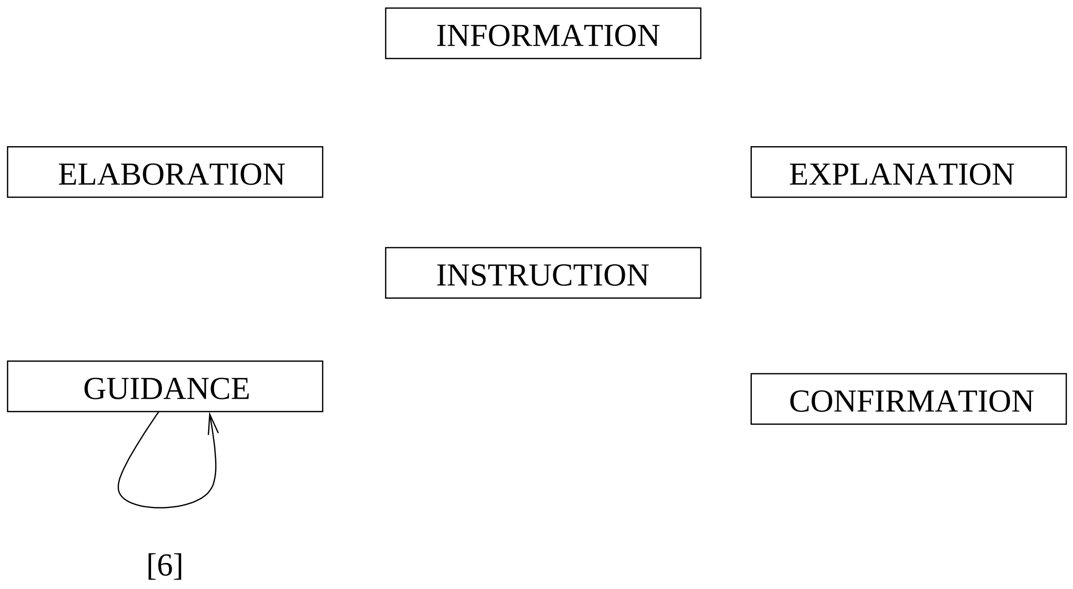


Figure 8.12: *Loop* intention graph for non-experienced subjects (Experiment II)

we have seen in the latter discussion that the intention types we have chosen seem to be useful in terms of how they can be used to indicate degrees of subject expertise. Hence, these intentions seem to provide a basis for a representation which can be used for user modelling. *Information* intentions are by far the most common type. The number of *explanation*, *instruction*, and *guidance* intentions are high too. Such intention types tend to indicate that subjects are *Non-experienced*.

When subjects are split into experienced and non-experienced it is noted that experienced subjects ask mainly *information* requests. Experienced subjects ask a larger number of elaborations too. Non-experienced subjects tend to ask many more explanation and guidance requests. In terms of a satisfaction spectrum, non-experienced subjects tend to produce intention sequence graphs involving increasing *interesting* intentions indicating lower degrees of *satisfaction*. They also loops on these intentions. It is also possible to see that there seems to be a link between each subject's background and the specific intention types they have. These results provide positive evidence for our theory of coherence of dialogue based on intention sequences and for the Intention-Person hypothesis. Intention types and sequences can be used to indicate the degree of expertise of a user. It is important to note here that we are claiming that high frequencies of certain intention types and sequences indicate the type of person using them, but not necessarily that a given type of person will always use the same intention types and sequences. Let's move on to experiment III which analyses these phenomena in more detail.

8.4 Experiment III

The third experiment was also a Wizard-of-Oz experiment, similar to the first. It had become apparent from Experiment II that there was a link between intention types and levels of subject expertise. One of the goals of the second experiment was to determine whether there is a difference

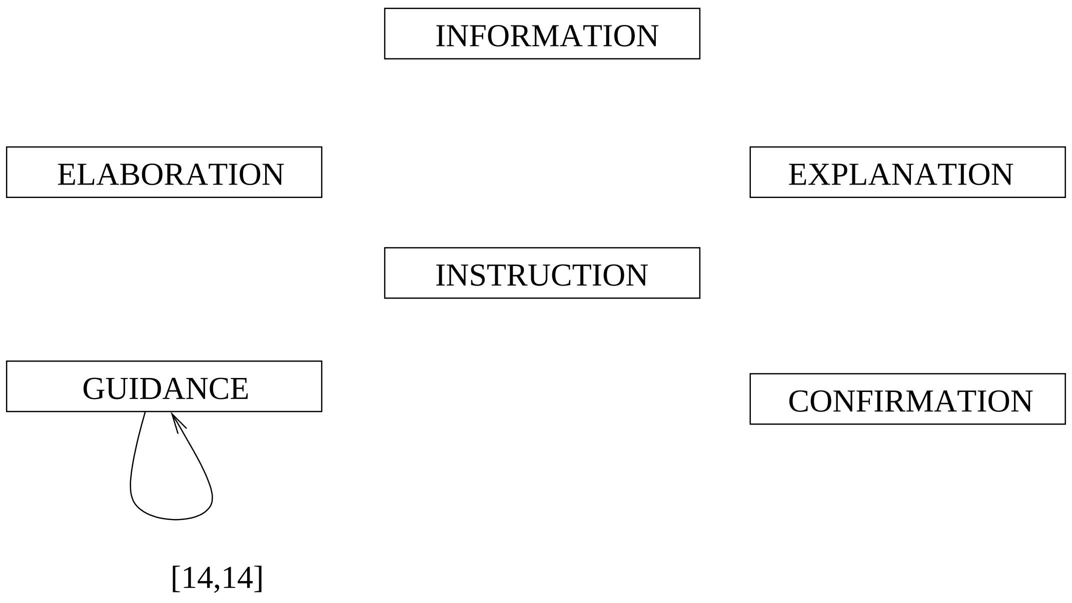


Figure 8.13: *Loop* intention graph for non-literate subjects (Experiment II)

between the sorts of intentions used by people with different levels of expertise. The subjects[9] were, on the whole, graduate students at New Mexico State University (NMSU) in Las Cruces, New Mexico, USA. Two groups of subjects were used. One group, called *experienced*, had taken a class on UNIX and acted as an *experimental*[10] while the other, called *non-experienced*, had not, and acted as the *control*. The experienced subjects were graduate students from NMSU who were pursuing graduate courses, mainly in Psychology. There were a total of 16 subjects. Details on how the experiment was conducted are given in Appendix D. The data collected during the experiment is available in Mc Kevitt and Ogden (1989b). The methodology of the experiment was as follows.

8.4.1 Methodology

As in the first experiment, subjects sat down at a computer terminal and were read a set of instructions. They were told that their operations would be monitored at another terminal nearby. However, this time the subjects were presented with two pictures which showed directory structures with files in them. One picture was a *source* directory structure, while the other was a *target* structure. Their task was to learn the UNIX commands to map the source directory structure into the target structure. Again, this would involve learning simple UNIX operations like printing, listing, copying, removing and locating files. The point of using pictures was to reduce bias in the types of questions asked by the subjects. In the first experiment it could be argued that the subject utterances were biased by the language of the tasks themselves. So, if the task was to "Print a file on the printer," then the subject might ask, "How do I print a file on the printer?" which

[9]Two subjects were staff at NMSU.
[10]It is common in scientific experimentation to compare an *experimental* group which demonstrates some phenomenon to a *control* which does not.

would not be very interesting as the subject has just repeated the task specification. Each subject asked a number of questions in natural language about the UNIX operating system. A wizard sat approximately eight metres directly behind the subject in the same laboratory and answered the subject queries. It took each subject, on average, one hour to complete the tasks in the experiment. There were, on average, eight pages of log file data per subject.

Each subject's screen contained a UNIX shell and a question and answer window. UNIX operations were completed in the UNIX shell. When asking a question, the subject typed his/her question in the question window and sent it to the wizard's monitor. The subject used CLEAR to clear successive questions. Answers, typed by the wizard, arrived in an ANSWER window on the subject's screen. The wizard's screen showed the subjects questions in a QUESTION window and the wizards could send their replies in an ANSWER window. However, this time, wizards also had a menu of pre-stored replies from which they could select their answer. The wizard could modify a canned reply in the reply window to suit the particular context if he/she so wished. The subjects could not see the answers as they were typed. Also, the wizard did not see the subject tasks although he/she did see subject operations in the UNIX shell.

8.4.2 Subjects

The subjects were selected, on the whole, from the graduate student population in the Psychology department at NMSU. The experienced subjects had taken a beginners' UNIX class while the non-experienced students has little, or no, UNIX experience, and had not taken the class. Each subject filled out questionnaires which recorded demographic and historical data. The same data was recorded as in Experiment II.

Table 8.14 below shows demographic data for the experienced subjects. Of the eight experienced subjects, six were male, and two, female. Their ages ranged from 23-28 with an average age of 26 (25.5) years old. All of the experienced subjects were graduate students in Psychology at NMSU. The "Y's" in Table 8.14 indicate that four of the eight subjects suspected that the wizard was answering their questions. The other four subjects did not suspect this at all. Some of the subjects had heard of Wizard-of-Oz experiments before, from courses they had taken, or papers they had read, while others had heard, or read, of the Turing Test[11].

Table 8.15 below shows demographic data for the non-experienced subjects. Of the eight non-experienced subjects five were male, and three, female. Their ages ranged from 27-34 which gives an average age of 31 (31.38) years old. Four of the non-experienced subjects were graduate students in Psychology at NMSU. Three were graduate students in other subjects at NMSU, and the other subject was a staff member at NMSU. Five of the eight subjects suspected that the wizard was answering their questions. The other three subjects did not suspect this at all.

[11]The Turing Test is a test proposed by Turing (1950) where a subject is asked to interact with a hidden agent through typed natural language on a computer. The subject is asked to determine whether the agent he/she is interacting with is a computer or a person. If he/she says a person, while the agent was really a computer program, then the program has passed the Turing Test, and can be said to embody AI.

Subject	*Age*	*Sex*	*Major*	*Suspicions*
1	28	M	Psychology	Y
2	28	M	Engineering Psy.	
3	23	F	Psychology	Y
4	25	M	Psychology	Y
5	26	M	Psychology	Y
6	25	M	Engineering Psy.	
7	24	F	Cognitive Psy.	
8	25	F	Psychology	

Table 8.14: Demographic data for experienced subjects (Experiment III)

Subject	*Age*	*Sex*	*Major*	*Suspicions*
1	28	F	Human Factors Psy.	Y
2	28	M	Psychology	Y
3	34	M	Engineering Psy.	
4	33	M	Mathematics	
5	26	M	Psychology	Y
6	46	F	History	
7	27	F	Non Student	Y
8	29	M	Physics/Optics	Y

Table 8.15: Demographic data for non-experienced subjects (Experiment III)

Table 8.16 below shows a list of the computer experience for the experienced subjects. The table shows that seven of the experienced subjects had 1-2 years of computer experience while the other subject had 6 months to 1 year. Most of the subjects had used a number of other operating systems. Five of the subjects had less than three months UNIX experience, and the other three had between three months and one year.

Subject	*Computers*	*Operating Systems*						*Unix*
		MS-DOS	*UNIX*	*PC-DOS*	*CMS*	*VMS*	*MAC*	
1	1-2 Yrs	Y	Y		Y		Y	$< 3Mth.$
2	6 Mth.-1 Yr.	Y	Y					”
3	1-2 Yrs	Y	Y				Y	”
4	”	Y	Y		Y		Y	”
5	”	Y	Y		Y		Y	”
6	”	Y	Y	Y	Y	Y	Y	6 Mth.-1Yr.
7	”		Y	Y	Y	Y	Y	1-2 Yrs.
8	”	Y	Y	Y	Y		Y	$< 6Mth.$

Table 8.16: Computer experience for experienced subjects (Experiment III)

Table 8.17 below shows the computer experience for non-experienced subjects. From this table it can be seen that although many of the non-experienced subjects had a number of years of computer experience, they did not have much UNIX experience. Six of the subjects had no UNIX experience, while two had less than 3 months. Some of the subjects had experience with other operating systems.

Again, in analysing the data, of most interest were the categories, and sequences of intentions in the dialogues. The 16 dialogues were investigated and the results are reported below. Again,

Subject	*Computers*	*Operating Systems*						*Unix*
		MS-DOS	*UNIX*	*PC-DOS*	*CMS*	*VMS*	*MAC*	
1	6 Mth-1 Yr.	Y			Y			None
2	1-2 Yrs.			Y	Y		Y	None
3	”	Y	Y				Y	$< 3Mth.$
4	None							None
5	6 Mth-1 Yr.	Y		Y				”
6	$< 3Mth.$							”
7	1-2 Yrs.		Y	Y	Y			$< 3Mth.$
8	”		Y		Y	Y		None

Table 8.17: Computer experience for non-experienced subjects (Experiment III)

it is important to point out that the analysis below relies on 16 subjects and stronger conclusions could only be drawn from larger scale studies.

8.4.3 Intention categorisation

The first experiment acted as an indicator of the principle types of intention to be expected in a consultancy domain. These were used in determining the intention types for the second experiment. In addition, two new categories were found on inspection of the dialogues. This demonstrates the point made earlier in Chapter 5 that it is difficult to argue one has selected the right set, or the only set, of intention types; only that one has selected *a* set. Hence the categories defined included the six in the first experiment plus the two new ones defined in Table 8.18 below.

Intention	*General Definition*	*UNIX Domain Definition*
repetition	An intention which is a repeated request. e.g. “How many people live in the Gulf?” followed by e.g. “What number of people live in the Gulf?”	An intention repeating another intention. e.g. “How do I print a file” followed by “How do I get a print out of my file?”
description	An intention requesting a description of an object or concept. e.g. “What is Persia?”	An intention requesting the description of UNIX concepts, objects, or commands. e.g. “what is UNIX?”

Table 8.18: Additional intentions (Experiment III)

Two new intentions are defined: one called *repetition*, the other called *description*. *Repetition* intentions occur when an utterance is repeated, although not necessarily through the same syntax. As was pointed out in Chapter 5, it is important to note that repetition intentions are not the same as repetitions of intention type, i.e. *repetition* intentions refer to repetitions of propositional content, whereas repetitions of intention type may not necessarily have the same propositional content. The uncovering of two new intention types is not to say that these did not exist in the dialogues for Experiment II; only to show that the set of intentions is not well defined, as was pointed out before, and that new ones can crop up all the time. Also, you will note that these two intentions can fall under the *elaboration* umbrella, shown in Figure 8.3, as both repetitions and descriptions can be elaborations of earlier items. Also, any intention can fall under the category of *repetition* because all utterances can be repeated. Repetition is a meta intention category, i.e. it is a category about utterances rather than within utterances.

8.4.4 Intention frequencies

Again, counting the frequencies of intentions needed to be done with care. This time two judges counted the frequencies of each type of intention and the results were compared. In 80% of the cases the judges agreed, and where they did not, they came to an agreement as to whether an utterance was one type of intention or another. The judges did not know whether a subject was in the experienced or non-experienced group. This time requests for information were counted too.

While inspecting the dialogues, in any situation where an utterance was a repetition of an *information* intention it was taken to be a repetition. All other *repetitions* were counted as the particular intention type in question. *Descriptions* could also be construed as *explanations*, yet, more specifically, *descriptions* are cases of explanation where the object has not been previously mentioned in the preceding discourse. The frequencies of intentions for the two groups of subjects are shown in Table 8.19 and Figure 8.14 below.

Intention	*Frequency*	
	Experienced	*Non-experienced*
information	91	160
instruction	4	22
description	-	2
elaboration	-	11
confirmation	9	18
explanation	6	30
guidance	1	4
repetition	1	18

Table 8.19: Frequencies of intentions (Experiment III)

In Table 8.19 it is noted that in all cases the number of utterances of each intention type is much higher for the non-experienced subjects. For the non-experienced subjects *information* utterances are the highest in frequency with descriptions the lowest. Again, not following far behind are explanations, repetitions and confirmations. Explanations are higher and the next highest type are repetitions with confirmations at the same level. Non-experienced subjects approaching a new subject for the first time would be expected to have many explanations and repetitions. Elaborations, guidance and description fall behind these. This is surprising as one would expect a larger number of guidances. However, this could be explained by the fact that the subjects were mature and would rather ask explanations which indicate lower degrees of *satisfaction* on an ordering of satisfaction. It is possible to compare the results here with those in Table 8.12. Combining the frequencies for non-experienced and non-literate subjects in Table 8.12 the same frequency spectrum is found for non-experienced subjects as in Table 8.19 except that guidance intentions are displaced higher in the spectrum of Table 8.12. This may be caused by the demographic factor of age just mentioned above. The spectrum similarity is not as close for experienced subjects. In

Table 8.12 there are a higher number of explanations and elaborations. This might be due again to the fact that the subjects in Experiment II had not taken a class in UNIX, and were much younger. We consider these results interesting as it indicates that there is, for non-experienced subjects at least, a link between experience with a domain, and an intention spectrum of satisfaction. Even more interesting is the fact that the subjects in Experiment III, who are graduate students at NMSU, usually come from outside the State of New Mexico, while most of the subjects in Experiment II are undergraduate students, who usually come from within the State. Hence, across background demographics virtually the same results are found except for a slight change in the spectrum probably due to a lower average age. Although beyond the scope of the present work, an interesting future experiment might be to determine if there is a link between age and intention spectrum. The experienced subjects had information intentions as their highest count and there were no elaborations or descriptions. The experienced subjects asked more confirmations than explanations which makes sense as they wish information which they believe to be true to be confirmed. Explanations are next highest with information coming after.

8.4.5 χ^2 and t-test statistical tests

It was decided that a further analysis of the difference between experienced and non-experienced subjects should be conducted. It became apparent from the dialogues that the non-experienced subjects used more *interesting* intentions, in the sense of *interesting* defined earlier, than the experienced subjects. Table 8.20 and Figure 8.15 show this distinction. However, the question that must be asked is whether this difference is statistically significant.

Intention	*Experienced*	*Non-experienced*
information	91	160
interesting	21	105

Table 8.20: Frequencies of *information* and *interesting* intentions (Experiment III)

It is possible to conduct a test of significance between the intention frequencies for experienced and non-experienced subjects. The most suitable test will be one which checks for significance between groups of frequency data. Such a test is termed *non-parametric* or *distribution-free*[12]. The appropriate type of non-parametric test will be a *chi-square* test, which usually asks one side, or other, of a two-sided question: (1) Do variables, or treatments between groups produce different results? or (2) Are variables between groups related, and if so, how highly related? In the test we wish to conduct, the variables are intention types, *information*, and *interesting*, and the groups are subjects, *experienced*, and *non-experienced*. Within the chi-square formalism tables such as that shown in Table 8.20 are referred to as *contingency tables*. Say, we have a 2 X 2 matrix which represents the contingency table for two groups and two variables. The table would look like that shown in Figure 8.16. The formula to compute the χ^2 value, for the Chi-square test, is:

[12]See, for example, Bruning and Kintz (1977) on the application of statistics to empirical data.

$$\chi^2 = \frac{N(AD-BC)^2}{(A+B)(C+D)(A+D)(B+D)}$$

where the numbers represented by the letters A, B, C, and D come from the contingency table in Figure 8.16. The degrees of freedom for the chi-square test are the number of rows minus 1, multiplied by the number of columns minus 1. A chi-square table is indexed by the degrees of freedom, and the χ^2 value, to determine significance. It is also possible to determine the degree of relationship between the two variables. This relationship is called the *phi* coefficient and is computed by the formula:

$$\phi = \sqrt{\frac{\chi^2}{N}}$$

where N = the total frequency in the entire contingency table. The *phi* coefficient gives a numerical value, ranging from 0 to +1, indicating the degree of relationship.

A chi-square analysis was conducted to show the degree of relationship between expertise and intention type. Under a chi-square analysis over the contingency table in Table 8.20 we get χ^2 (1) = 15.44, $p < 0.001$. Thus, with 1 degree of freedom this is a chi-square value which is significant at the 0.001 probability level. This means that the likelihood that the significant result happened by chance is 0.1%. It is concluded that expertise and intention type are significantly related. It is also possible to compute the degree of relationship between expertise and intention type. The ϕ coefficient is 0.20, which is the degree of relationship.

Next, it is possible to calculate the proportion of intentions out of total requests for experienced and non-experienced subjects. These are shown in Table 8.21. I marks information, while In marks interesting, requests. Another statistical test can be applied which checks proportional data for significance.

Experienced		*Non-Experienced*	
I	*In*	*I*	*In*
0.70	0.30	0.47	0.53
0.93	0.07	0.80	0.20
0.67	0.33	0.86	0.14
1.00	-	0.58	0.42
0.93	0.07	0.61	0.39
1.00	-	0.75	0.25
0.64	0.36	0.30	0.70
0.73	0.27	0.70	0.30

Table 8.21: Proportions of intentions (Experiment III)

One of the most commonly used tests to determine significance is the t-test. The most common use of the t-test is probably that of determining whether a behaviour difference between two groups of subjects is significant. In most experimental situations there are two groups, and the difference in behaviour of the two groups is analysed with respect to some phenomenon. The basic formula for the t-test of difference between two independent means is:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\left[\frac{\Sigma X_1{}^2 - \frac{(\Sigma X_1)^2}{N_1} + \Sigma X_2{}^2 - \frac{(\Sigma X_2)^2}{N_2}}{(N_1 - N_2) - 2}\right] \cdot \left[\frac{1}{N_1} + \frac{1}{N_2}\right]}}$$

where $\bar{X}_1$	= the mean of the first group of scores
$\bar{X}_2$	= the mean of the second group of scores
$\Sigma X_1{}^2$	= the sum of the squared score values of the first group
$\Sigma X_2{}^2$	= the sum of the squared score values of the second group
$(\Sigma X_1)^2$	= the square of the sum of the scores in the first group
$(\Sigma X_2)^2$	= the square of the sum of the scores in the second group
N_1	= the number of scores in the first group
N_2	= the number of scores in the second group

A t-table is indexed using the degrees of freedom (df), and the t value, to determine significance. For a test of significance between two means, the df are equal to $(n_1 + n_2) - 2$.

A t-test analysis of the differences between the proportion of *interesting* intentions of total requests for non-experienced versus experienced subjects showed up a difference of t(14)= 2.35, $p < 0.05$. This is a significant difference at the 0.05 probability level with 14 degrees of freedom which tells us that there is a significant difference between *interesting* intention types for non-experienced and experienced subjects for this particular set of subjects. Also, this result holds in spite of the fact that some of the non-experienced subjects had experience with other operating systems.

Experienced subjects have a liking for confirmations. A union of information and confirmation requests between the two groups can also be compared. Let's call that union of intentions, *happy*. It is also true that the non-experienced subjects asked more *happy* intentions than experienced subjects. Table 8.22 and Figure 8.17 show this.

Intention	*Experienced*	*Non-Experienced*
happy	100	178
interesting	12	87

Table 8.22: Frequencies of *information* and *happy* intentions (Experiment III)

8.4.6 Intention sequences

An analysis of intention sequences was completed on the 16 dialogues. The sequence analysis was conducted in the same manner as for Experiment II. The following sequence graph in Figure 8.18 shows the sequences for experienced subjects.

It is noted that there are a number of confirmations emerging from information intentions. This would be expected with experienced subjects. They will tend to want to confirm information they believe to be true from the domain. Also, note that instruction intentions are high on the frequency

Intention	*Subject*							
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
information	14	14	12	13	13	10	7	8
instruction		2						2
description								
elaboration								
confirmation	3	1	1				4	
explanation	3				1			
guidance	1		1					
repetition								1

Table 8.23: Frequencies of intentions for experienced subjects (Experiment III)

scale. Subjects wish to ask for information from the system as they are not used to its capabilities and believe it can do more than it actually can. Note, too that there are a number of loops in the graph, and yet there is no loop under guidance, but under confirmation and instruction.

In addition, it is possible to look at individual subjects to see specific correlations between subjects and intentions. Shown below in Table 8.23 are the intentions for each experienced subject.

From this data an intention graph, shown in Figure 8.19, is developed. First, it is noted that subject 7 has asked a lot of confirmations. From Table 8.16 it is noted that this subject had 1-2 years of computer experience but from Table 8.14 one can see that she was the youngest member of the group. Subject 8 asked for instruction by looping. Subject 8 has more than 3 months UNIX experience and hence will have higher expectations of what the system is capable of.

We note that of the intention pairs used by *experienced* subjects, 8 *interesting* intentions are used by males and 12 by females. Also, only 3 of the 8 subjects are females. This could indicate something about the hypothesis that males are better at conceptual reasoning than females and can therefore have a better grasp of a domain after taking a class explaining that domain. More detailed experiments comparing males and females would need to test this property.

Next, we move on to non-experienced subjects. The following intention graph in Figure 8.20 shows the sequences for the eight subjects. The first obvious fact is that there are a lot of explanations. Also, there are a number of loops on explanations. This is a strong indicator that the subjects are having problems with understanding the domain at hand. In one case there is a confirmation following an explanation. There are three explanations coming from confirmations. In other words the explanation intentions are becoming a *sink* within the graph. There are many repetitions and confirmations indicating that subjects need to confirm their beliefs, and repeat questions when they are not answered. Requests for instruction are high too. These are all indicators that non-experienced subjects have a more difficult time understanding the domain and how to conduct their dialogue with the computer. In Table 8.24 below we show the list of intentions by subject.

Drawing an intention graph of the data in Figure 8.21 it is noted that subject 1 asks a number of elaborations while from Tables 8.15 and 8.17 it is noted that she is 28 and had only used two

Intention	*Subject*							
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
information	18	20	25	21	17	15	14	30
instruction	9	1		1	3		8	
description	4	2				1	1	
elaboration					2		1	2
confirmation		1		3	1	1	1	7
explanation	1		2	9	5		10	3
guidance	1						5	1
repetition	5	1	2	2			1	

Table 8.24: Frequencies of intentions for non-experienced subjects (Experiment III)

operating systems. This may have led to an interest in learning, more and asking such elaborations. She also asked a number of instruction requests and even looped on instruction intentions. It is noted that subject 4 asked five explanations from informations and another three loop explanations. From Table 8.17 it is observed that this subject had no computer experience at all. This would explain this particular phenomenon. Note that subject 7 asked many confirmations as did subject 8. These subjects had very little UNIX experience although they had 1-2 years of computer experience. It makes sense for them to confirm beliefs they have about the UNIX operating system if they have used other operating systems before. Subject 4 asked a number of repetitions which makes sense as this subject had no computer experience. Note that subject 6 asks two guidance intentions. Looking at Tables 8.15 and 8.17 it is noted that this subject is a history student with less that 3 months computer experience and no UNIX experience.

We note that of the intention pairs used by *non-experienced* subjects, 44 *interesting* intentions are used by males and 42 by females. However, note that 5 of the subjects are males. It is possible that this indicates, like experiment I, something about the hypothesis that females are better at linguistic reasoning as compared with males. Again, more detailed experiments would need to be conducted.

8.4.7 Intention loops

Also, it will be interesting to take a look at the intention loops in the dialogues. First, taking a look at the loops for experienced subjects Figure 8.22 below is obtained. Note that subject 8 loops on instruction and, as pointed out already, this subject has more than 3 months UNIX experience, and hence will have higher expectations of what the system is capable of doing. Subject 1 is also involved in loops on explanation and confirmation. Subject 1 had less than three months UNIX experience.

Next, looking at the non-experienced subjects Figure 8.23 below is obtained. Note that there are five loops in all which is an immediate indication that these subjects are having trouble. In this figure it is observed that subjects 1 and 7 ask a number of instruction intentions. Subject 1 had used only two operating systems and had no UNIX experience while subject 7 had used three operating systems and had less than three months UNIX experience. Subject 8 loops on confirmations and this subject had no UNIX experience. Subject 4 had no computer experience and he looped a number of times on explanations.

In summary, Experiment III shows that there is a statistically significant difference between the sorts of intentions used by people with different levels of expertise. In comparison to Experiment II, Experiment III was modified in two ways. First, to reduce bias in subject responses, the subjects were presented with tasks in picture form rather than in natural language form. Second, to increase response speed, the wizard could modify a canned reply in his/her reply window to suit the particular context if he/she so wished. Two groups of subjects were compared, *experienced* and *non-experienced*, which provided an *experimental* and *control* respectively. The experienced subjects had taken a beginners' UNIX class, while the non-experienced subjects had little, or no, UNIX experience, and had not taken the class. The data shows that non-experienced subjects ask many more utterances. For non-experienced subjects *information* utterances are the highest in frequency with *descriptions* the lowest. Again, not far behind are explanations, repetitions and confirmations. Explanations are higher and the next highest type is repetitions with confirmations at the same level. Elaborations, guidance and descriptions fall behind these. The experienced subjects asked more confirmations than explanations which makes sense as they wish beliefs, which they hold true of the domain, to be confirmed. The set of eight intentions were split into two types, *interesting* and *information. Interesting* intentions consist of the set *instruction*, *elaboration*, *confirmation*, *explanation*, *guidance*, and *repetition.* Two statistical tests, one a Chi-square analysis, and the second a t-test analysis, show that there is a significant difference between *interesting* intention types for non-experienced and experienced subjects. With respect to intention sequencing, experienced subjects tend to loop or repeat, on higher *satisfaction* intentions, while non-experienced subjects tend to loop, or repeat, on lower *satisfaction* intentions. Also the intention graph for non-experienced subjects contains many more lower *satisfaction* intentions than the graph for experienced subjects. It is also apparent that there are specific links between intention types and subject backgrounds. Also, note that there now exists empirical evidence for the Intention-Person hypothesis. Again, it is important to note here that we are claiming that high frequencies of certain intention types and sequences indicate the type of person using them, but not necessarily that a given type of person will always use the same intention types and sequences. Let's move on to summarise the implications of all three experiments.

8.5 Summary

Three experiments have been used to test the Intention-Computer and Intention-Person hypotheses. An experiment using a computational model of intention analysis was used to test the Intention-Computer hypothesis. Two Wizard-of-Oz experiments were conducted to test the Intention-Person hypothesis. Experiment I provides positive evidence for the Intention-Computer hypothesis and acts as an existence proof of a theory of intention sequencing and shows that a computational model can recognise and represent intentions, can use the representation for user-modelling, and is less effective without the capability of intention-sequence analysis. The other two experiments provided positive evidence for the Intention-Person hypothesis, the second one giving statistically significant results indicating a correlation between intention-type frequencies and level of user expertise.

More specifically, with respect to experiment III, there is a statistically significant relationship between intention type and the level of expertise of subjects. This was evident from a Chi-squared analysis of the differences in intention types used by experienced and non-experienced subjects. This also showed up in a t-test where there was a statistically significant difference in the proportions of types of intentions used by experienced and non-experienced subjects. A number of intentions seem to occur at interesting points in the dialogues where the dialogue breaks down, or the subject has problems. Such intentions are termed *interesting*, where interesting is defined to be the set containing: *instruction*, *elaboration*, *confirmation*, *explanation*, and *guidance*. It seems to be the case that higher satisfaction intentions are used by people with more experience and lower satisfaction requests are used by those who have less experience with a domain.

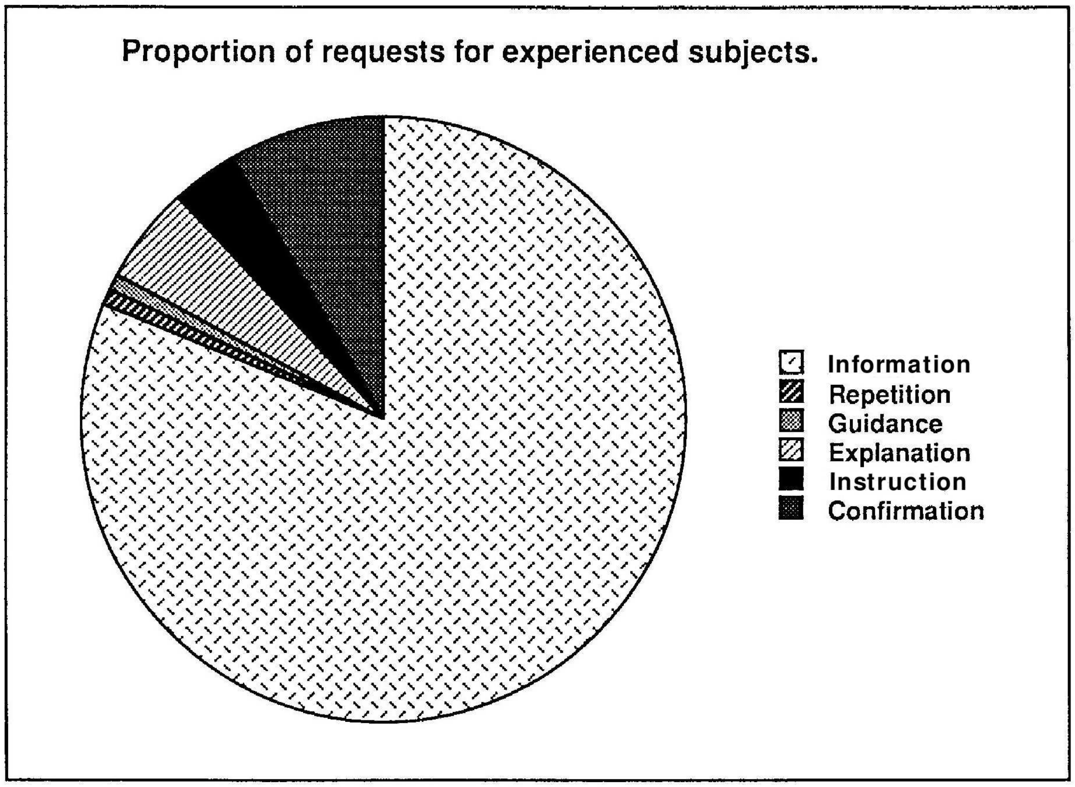
The experiments do not only provide a means of testing the Intention-Person and Intention-Computer hypotheses, but also provide evidence, in part, for the theory of intention analysis presented in Chapter 5. The first experiment shows that a computational model can conduct intention analysis and that when it does not do so it has limited abilities. Hence the first experiment acts as an existence proof of the viability of the theory. The other two experiments show that the theory is empirically valid with respect to real-world data.

More specifically, it has been shown that a number of phenomena exist in real-world data which give evidence for the theory in Chapter 5. First, it is evident that there is a distinct set of intention types in real-world natural-language dialogue and an of these can be linked to expertise levels. Eight of the principal intention types discussed in Chapter 5 were uncovered in the Wizard-of-Oz natural-language dialogues. Of course, the development of the theory of intention analysis was aided by information from Experiment I. Except for a slight change probably due to age, an ordering of intentions based on satisfaction was found to exist for experienced and non-experienced subjects, across demographics, in two Wizard-of-Oz experiments. Second, it is apparent from the data collected that specific types of subjects have tendencies to utter particular types of intentions. Hence, utterance type will be a strong indicator of the particular types of utterances a subject prefers. A history of particular subject intentions would give information about such preferences. Third, it seems that intention sequences in dialogue may be very useful in indicating the direction of the dialogue and whether it is progressing smoothly. In particular, if there are a number of sequences involving lower *satisfaction* then it should be obvious to a system that a user is having trouble. Fourth, it is noted that *loops* in intention graphs are strong indicators of intention extremes, demonstrating the fact that a subject is repeating an intention type, and these are usually used by people with lower degrees of expertise. Fifth, it seems that there is a strong link between subject demography and subject familiarity with the domain. There seemed to be a link between age and the intention spectrum of satisfaction.

One of the problems with experimentation is that it can be argued that experiments are so controlled that they are limited in what we can infer from them. In the case of the three experiments conducted here, it could be argued that they are too constrained in the sense that the environment is not like a real environment of human-computer natural-language consultancy in the sense that the interaction was conducted in a controlled way with specific tasks, and a limited set of UNIX commands[13]. However, it is claimed here that by restraining the domain a better handle on intention types and sequences in natural-language consultancy dialogue can be obtained. Also, natural-language is such a wide-ranging phenomena that, given current natural-language processing technology, the only experiments possible, are those in restricted domains. The UNIX domain, which seems restrictive in scope, does have the merit of being a real-world domain. Let's move on now to discuss the implications of the results from this chapter.

Now that empirical evidence for the hypothesis has been shown, with real-world data, an experiment can be conducted to demonstrate by existence proof that the computational model of intention analysis, described in Chapter 6, can recognise and represent sequences of intention. Also, it can be shown that the computational model can be used for user modelling. In turn, the user model can be used for effective natural-language generation.

[13]This is considered a problem with scientific experimentation in general. See Neisser (1987) for a discussion of the problem in Psychology.



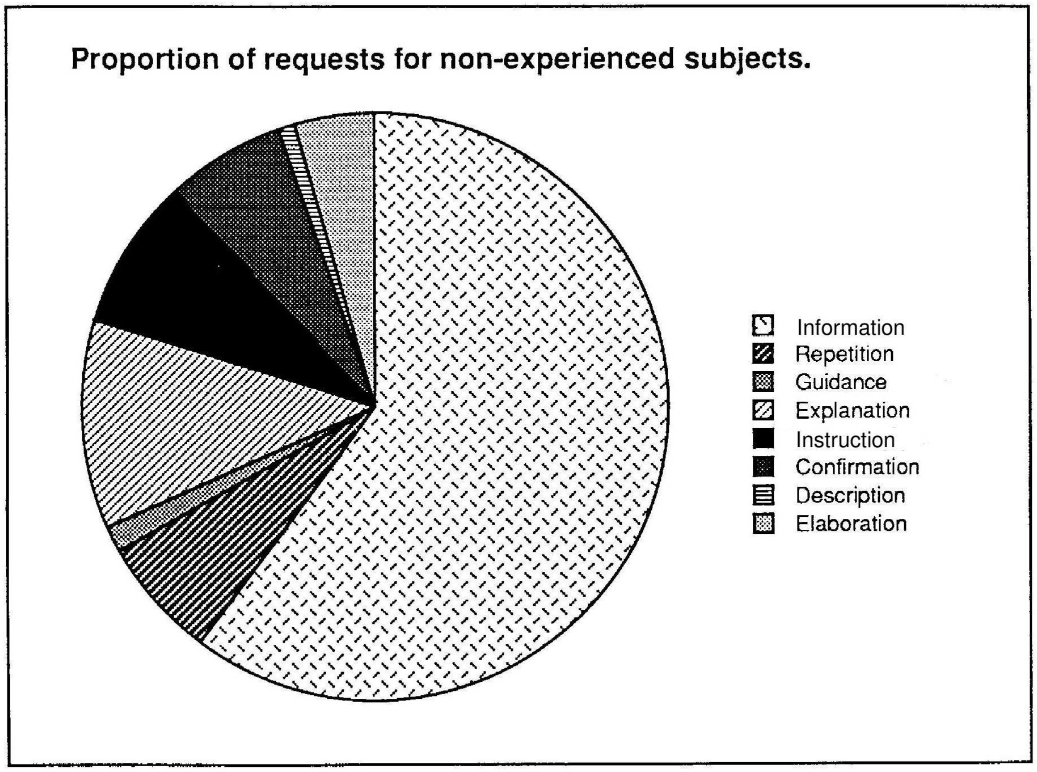


Figure 8.14: Piecharts of frequencies of intentions (Experiment III)

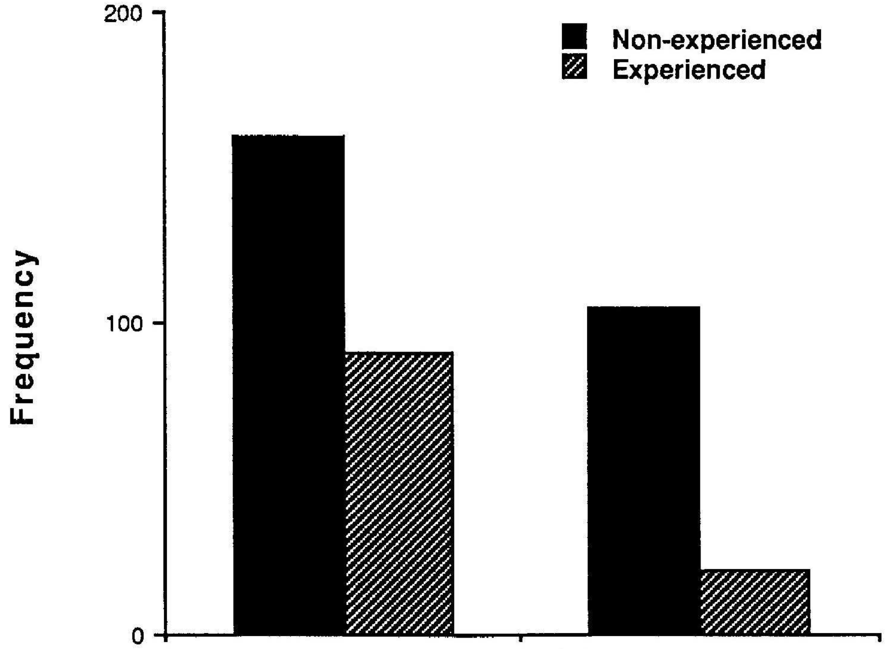


Figure 8.15: Histogram of *information* and *interesting* intentions (Experiment III)

A	B
C	D

Figure 8.16: Example contingency table for chi-square significance test

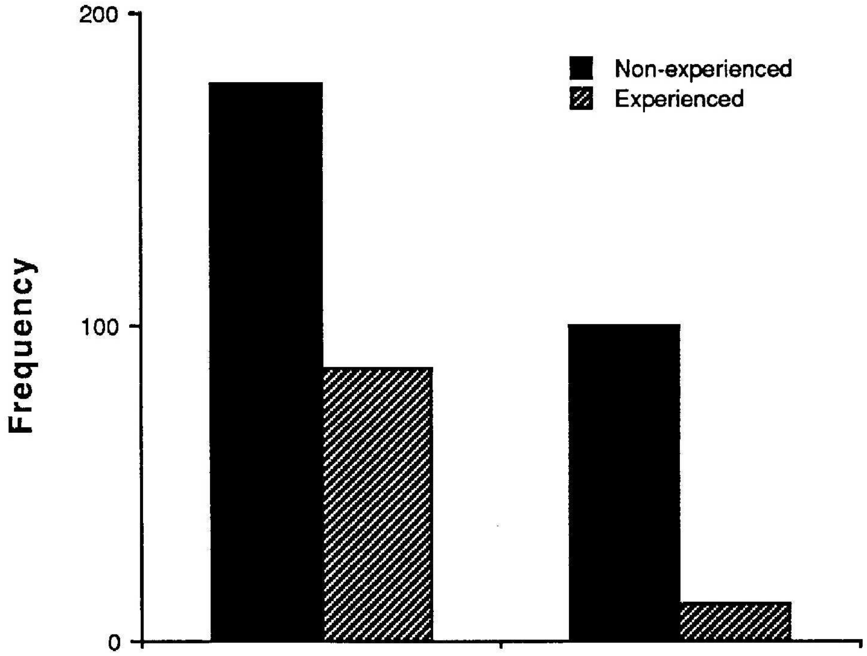


Figure 8.17: Histogram of *information* and *happy* intentions (Experiment III)

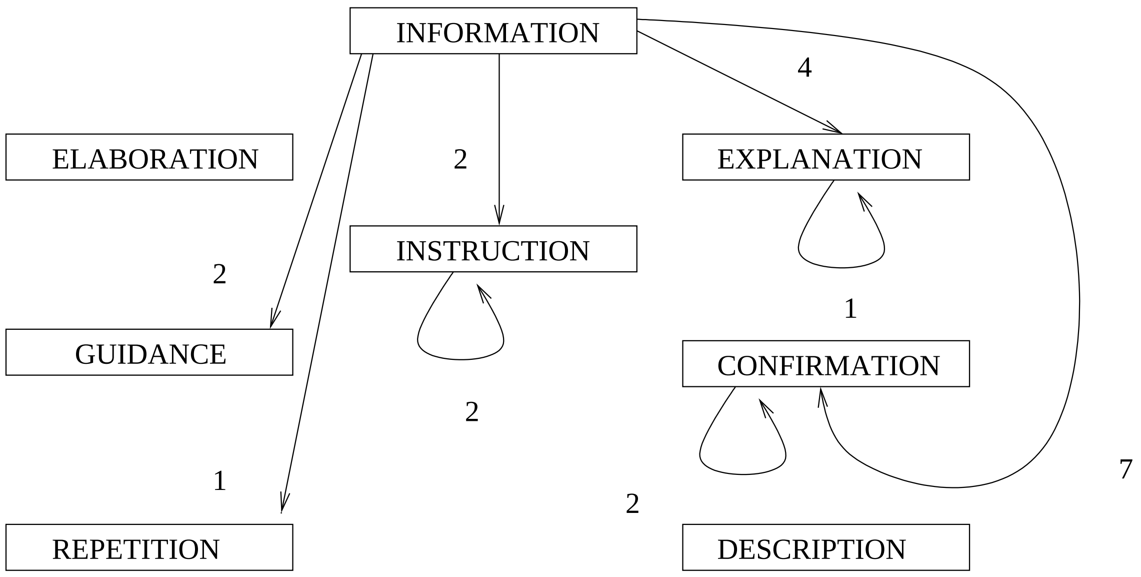


Figure 8.18: Intention graph for experienced subjects (Experiment III)

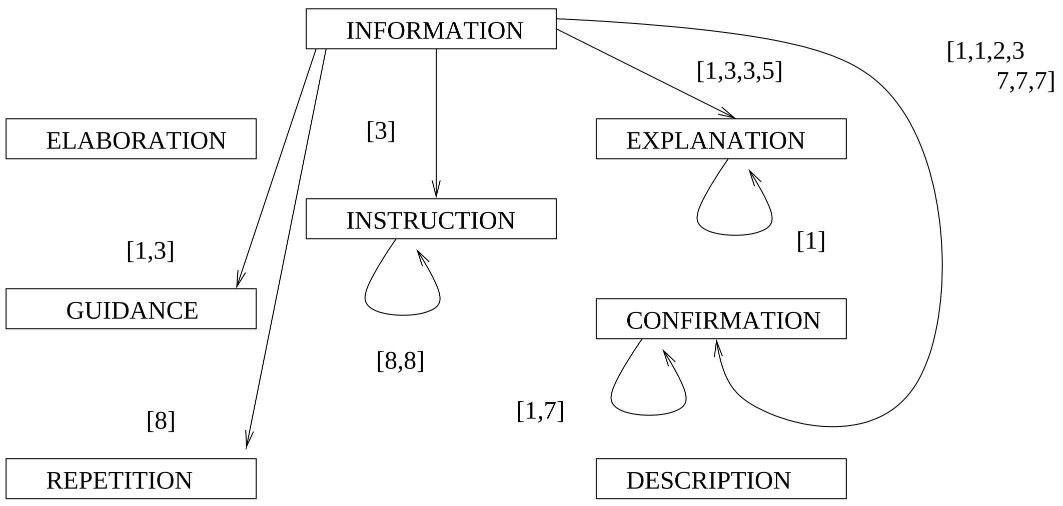


Figure 8.19: Intention graph, by subject, for experienced subjects (Experiment III)

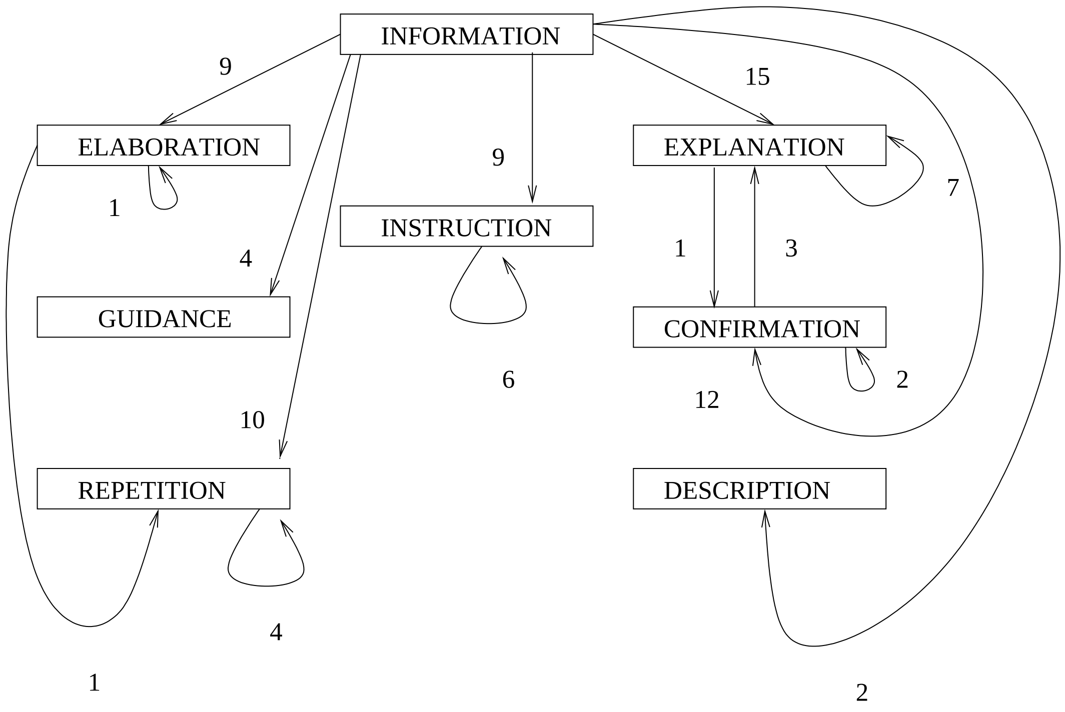


Figure 8.20: Intention graph for non-experienced subjects (Experiment III)

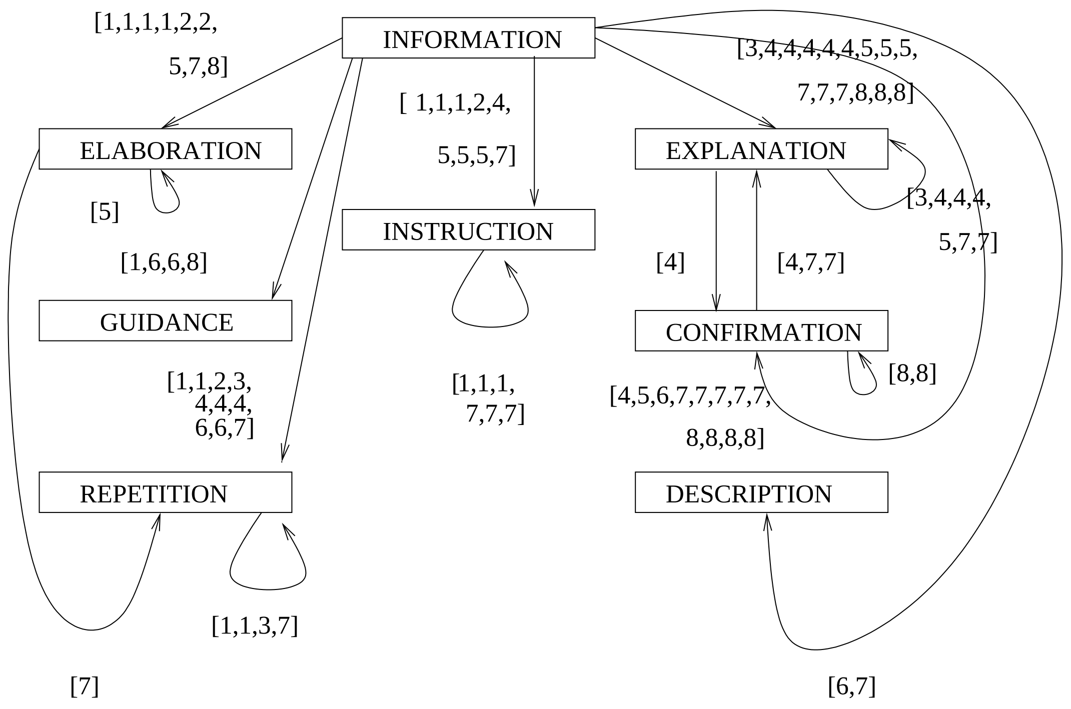


Figure 8.21: Intention graph, by subject, for non-experienced subjects (Experiment III)

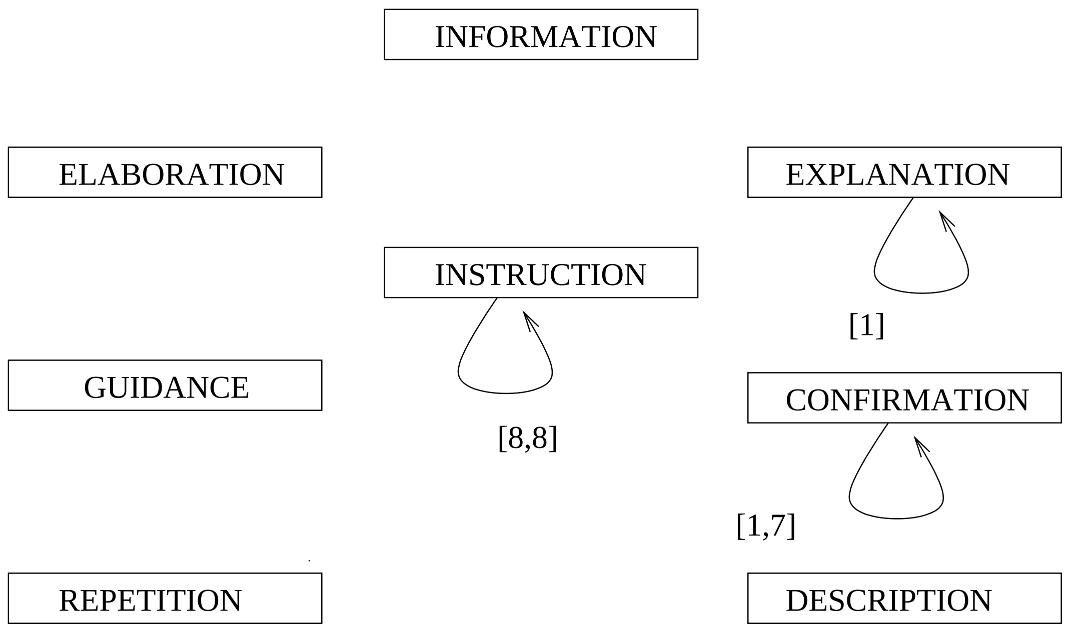


Figure 8.22: *Loop* intention graph for experienced subjects (Experiment III)

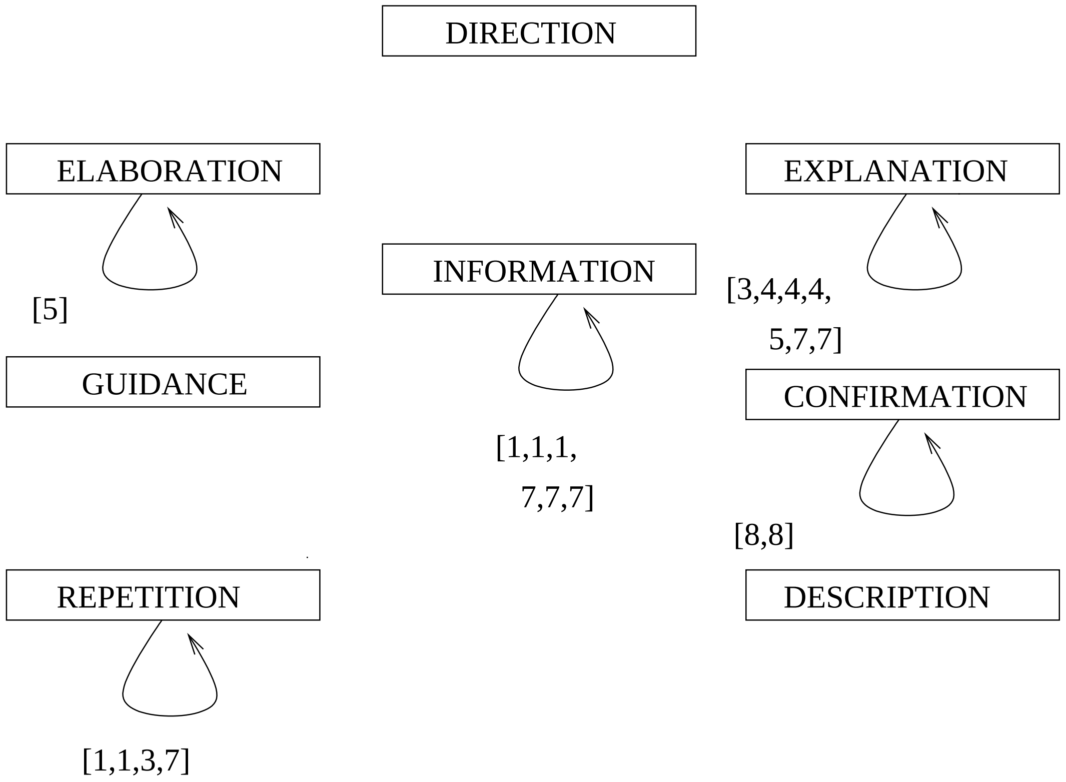


Figure 8.23: Loop intention graph for non-experienced subjects (Experiment III)

Part V

Epilogue

Chapter 9

Conclusion

Our goal has been to provide a theory of intention analysis for solving, in part, the problem of natural-language dialogue processing. A central principle of the theory is that coherence of natural-language dialogue can be modelled by analysing sequences of intention. In addition, it was hoped that the theory of intention analysis could be incorporated within a computational model, and that the computational model could be used for applications such as user-modelling. In turn, it was intended that user-modelling be used for generating natural-language responses which are sensitive to the level of user expertise. In particular, we hoped to test the Intention-Person and Intention-Computer hypotheses. In Part 2 a discussion of background research on the problem of natural-language discourse processing was provided. In Part 3 a theory, computational model, and application of intention analysis were proposed. In Part 4 three experiments were described. The first experiment acted as an existence proof that a computational model, embodying a theory of intention analysis, can conduct user modelling for user-sensitive and context-sensitive natural-language generation. The first experiment provided evidence for the Intention-Computer hypothesis. The other two experiments provided real-world empirical evidence for the Intention-Person hypothesis. Now it is time to summarise the main implications of the theory of intention analysis, its relationship to other work, and possibilities for future work.

9.1 Summary

A theory of intention analysis has been proposed and incorporated within a computational model in the form of a computer program called OSCON. The computational model has been applied to the problem of user-modelling. In turn, user-modelling is used to generate natural-language responses appropriate to the user. One experiment demonstrates that the computational model, OSCON, performs intention-analysis and user-modelling. Additionally, OSCON produces better natural-language dialogue while modelling intention sequences. Two other experiments show that users at different levels of expertise have different type and sequence frequencies of intention, and the analysis of type and sequence frequencies of intention is useful for effective natural-language dialogue between a person and a computer. The experiments provide an existence proof for the viability of our theory and the Intention-Computer and Intention-Person hypotheses.

In Part 1 existing theories of natural-language discourse processing were discussed. Existing research falls within three main approaches: modelling semantics, structure, or intention. Although these theories exhibit many differences they have one theme in common: they are all based on the principle that there is an underlying coherence in natural-language dialogue.

In Part 2 a theory of intention analysis was introduced. The central principle of the theory of intention analysis is that natural-language dialogue can be modelled from the point of view of coherence of intention. More specifically, the central principle is that a dialogue, which consists of sequences of utterances, can be modelled in terms of sequences of intention. The *analysis of intention* is defined by at least two properties: (1) that it is possible to recognise intention, and (2) that it is possible to represent intention. It is proposed that syntax, semantics and pragmatics of natural-language utterances can be used for intention recognition. It is proposed that the domain of natural-language consultancy has at least nine principal intentions which are: *information*, *description*, *instruction confirmation*, *elaboration*, *explanation*, *guidance*, *repetition*, and *nointention*. The first eight of the above set of intentions are ordered in terms of decreasing satisfaction. Intention sequences in natural-language dialogue can be described by what are called *intention graphs*. Intention graphs can be (1) incorporated within computational models and (2) used to determine, at a glance, the type of information present in experimental data. It is argued that an ordering of intention *satisfaction* exists and, when used in conjunction with intention graphs, indicates the *local* and *global* expertise of a speaker in a dialogue. It is argued that intention pairs will represent more information in a dialogue than a simple count of intention frequencies and could be used to solve, in part, the problems of user modelling, ellipsis resolution and anaphoric reference in natural-language dialogue. Here, we are only concerned with demonstrating the application to user modelling.

A computational model for intention analysis is developed and incorporated within a computer program, called OSCON, which answers, in English, English queries about computer operating systems. The computational model can successfully recognise and represent intentions and intention sequences. The model uses a semantic grammar, in the Definite Clause Grammar (DCG) formalism of Prolog, to recognise intentions in natural-language dialogue. Intention sequences are represented in a data structure called an *intention matrix*. The computational model of intention analysis is applied to the problem of modelling the level of expertise of users, or user-modelling. In turn, the user modelling component of OSCON facilitates the generation of user-sensitive natural-language responses. Example traces of the capability of the OSCON program for modelling natural-language dialogue, and user-modelling, are given in Appendices A and B.

Three experiments were conducted to test the Intention-Computer and Intention-Person hypotheses. The first experiment tests the Intention-Computer hypothesis and acts as an existence proof of the viability of our theory. The experiment demonstrates two results: first, that the OSCON system can conduct intention analysis over a natural-language dialogue by the processes of intention recognition, intention-sequence recognition, intention representation, and intention-sequence representation, and that the latter processes can be used by OSCON to conduct user-modelling; second, that OSCON performs better on modelling dialogue when three versions of OSCON are compared, over the same sample natural-language dialogue, one conducting sequence analysis, the other two not doing so. It is argued that the sample natural-language dialogue is representative of a typical dialogue expected in the natural-language consultancy domain for UNIX.

Two other experiments are in the Wizard-of-Oz form, where subjects are asked to conduct a number of tasks in the domain of computer operating systems. The subjects type their queries, in English, to the program, and a hidden person answers their queries. The subjects are not told that a person is answering their queries. These two experiments test the Intention-Person hypothesis. The first experiment tests whether there are recognisable types and sequences of communicative acts, and hence intention, in natural-language dialogue. The results show that there are recognisable intention types, and that at least six of the eight principal intention types proposed for the consultancy domain do exist. Sequences of intention pairs in the data analysed from the dialogues are represented in an *intention graph.* Support for the ordering of intention types is provided by the observation that there is a link between the level of user expertise and intention categories and sequences. Intention categories and sequences can be used to determine the degree of expertise of a user. The second experiment tests whether there is a significant difference in the types of intention used by subjects from different backgrounds of expertise. The results show that there is a significant difference. Hence, the latter two experiments provide evidence for the Intention-Person hypothesis. Now that the work which has been done has been described, we can move on to look at its implications.

9.2 Implications

The theory of intention analysis may have implications for a variety of research fields such as natural-language processing, human-computer interaction, artificial intelligence and philosophy.

Obviously, there are implications for natural-language processing. It may be the case that other problems in the field natural-language processing can be solved by modelling intention. In fact, we have shown that one of the problems of natural-language discourse, the problem of modelling the level of expertise of a user in the discourse, can be solved, in part, by intention analysis. Also we have claimed that two other problems in natural-language dialogue can be solved by intention-sequence analysis. These are the problems of ellipsis and anaphoric reference resolution. When a natural-language processor cannot parse an incoming utterance, or finds that utterance ambiguous, it may be able to determine the most likely possible meaning from a pre-stored set of possibilities of the types of utterance sequence that a particular user is likely to make. The problem of anaphoric reference could be tackled by checking the intention-pair relationship between the utterance with the anaphoric reference and the utterance previous to that. For example, if a reference occurs in an utterance which is an *elaboration* intention then it will be likely that the reference is in the previous utterance, or the reply to that utterance. Also, a more elaborate model of the dialogue could be constructed where the dialogue is segmented into spaces and where segment boundaries are determined by changes in intention type. Indeed, such a proposal has been sketched out in Mc Kevitt (1990b) and also, this links back to the proposals of Grosz and Sidner described in Chapter 3. Hence, there are many areas of natural-language processing which may be augmented by intention analysis.

Intention analysis has various implications for user-modelling. From the Wizard-of-Oz experiments it was apparent that particular types of intentions indicate specific types of subjects. Hence, intention type will be an indicator of the particular types of user interacting with the system. A history of user intentions could be stored for a given user-session with the computational model, and recovered when that user reuses the system at a future date. Intention analysis may also be useful for indicating the direction of a dialogue and whether it is progressing smoothly. In particular, intention loops are an indicator of the fact that a subject is behaving in an extreme manner. In conjunction with an intention ordering of *satisfaction*, loops indicate that the user is extremely satisfied or dissatisfied. Any system which detects that a user is "spiralling" into a set of loops, where the loops occur on a lower *satisfaction* intention, should strive to get the user out of the spiral. Also, if a user uses a number of sequences involving lower *satisfaction* intentions then it should be obvious to a system that a user is having difficulty.

As it is apparent that there is a link between subject demography, in terms of factors such as *age*, *sex*, *area of study*, and subject *familiarity* with a domain, such information could be determined from users before a session with a system. Then the system could align its behaviour to the user based on such information. The advantage of modelling the user from the point of view of intention sequences is that we have a less domain-dependent model of the user. Most existing user models are domain-dependent (see Chin 1988, and Kobsa and Wahlster 1988) and model the user in terms of how many times he/she referenced a particular entity in a domain.

The analysis of intentions has implications for information retrieval. An information retrieval system should be able to retrieve information in different ways depending on what the user's intention is at any particular moment. We believe that if a computer model gives a wrong answer at a given point in a dialogue, that will not facilitate communication, and the user may get frustrated. A good example of this phenomenon in the data we collected is where *repetition* intentions occurred. Here, subjects repeated the same intention because they did not get the answer they wanted.

Second, the analysis of intentions may have implications for the development of other forms of human-computer interface such as menu-based interfaces. In fact, most of the implications just discussed will also apply to forms of human-computer interface other than natural language. Any human-computer interface may be able to keep a history of sequences of user commands and use that history to determine user intentions at each stage. The analysis of users' intentions can be represented in intention graphs and such intention graphs could be used to design menu-based interfaces. In fact, Flores and Ludlow (1981) and Flores (1982) describe such an interface called a *coordinator*. Such menu-based interfaces could be linked to a database on say, travel advice, or UNIX, and by using a mouse to click on intention nodes in the graph, variations of information about the domain could be retrieved. The analysis of intention can be used to build better computer programs which can communicate with people through dialogue whether that dialogue be natural language, or otherwise. With such techniques people will be nearer to communicating with computers in their own natural language, or a language close to that, rather than having to learn some abstract computer language.

Third, the hypotheses have implications for artificial intelligence. If the analysis of intention is useful as a means towards providing coherent natural-language dialogue with a computer, then without any such analysis, while interacting with a computer, you may find it difficult to accept it as being rational, or intelligent in any way. Hence, you may no longer wish to communicate with the computer in natural language. The reason for failure of many of today's natural language interfaces might be that they appear irrational, as most of them do not incorporate any model of intention analysis or coherence. In fact, most current natural-language interface technology considers natural-language queries to be content-free and independent of each other. Indeed, Whittaker and Stenton (1989) have shown that pragmatics phenomena, which include intention, are the cause of approximately 25% of the failures in natural-language interfaces. The analysis of intentions will lead to more coherent dialogues between people and computers, better adherence to Grice's *Cooperative Principle* which was discussed in Chapter 1, and increased perceptions of rationality of computers. This leads us to implications for philosophy.

Finally, there may be implications for philosophy. If you are unsure about Searle's claim that people are motivated to use language because of their intentions, then you might be better convinced if it is shown that, even for computers, the analysis of intention is useful for understanding language. If there is such a strong link between language and intention then there would seem to be some truth in Searle's claim.

Also, if you accept the school of thought which argues that ascription[1] of rationality, and intentions, to computers is necessary for verbal communication between people and computers[2], then you may be more convinced of ascribing rationality and intentions to computers if it is shown that computers can analyse intentions, and by doing so behave coherently and rationally, and hence better convinced that computers will one day be able to conduct verbal communication. In other words the analysis of intention by a computer may be a necessary feature for you to be able to ascribe rationality, intentions and intelligence to it. Furthermore, such schools of thought make a stronger claim, in that they argue that it is impossible for people to communicate with computers through natural language, because a necessary precondition is that the computer *is* rational. Also, if a computer is to be ascribed rationality, then, according to Grice, it must obey his *Cooperative Principle*; therefore any natural-language dialogue with the computer should be coherent.

Although we do claim in this work that a computer can analyse people's intentions in natural-language we make no claim that the computer *is* rational. We are currently agnostic on that issue. Also, we make no claim that a computer has access to intentions in people's heads.

[1]Such schools do not require that computers *have* intentions but only that they be *ascribed* to them.

[2]See, for example, Dennett (1981, pp. 278-279), who argues for an *intentional stance*, where one can ascribe intentions to machines, but that alone will not instill consciousness or intelligence on the machine.

Now that we have defined the implications of our work, we can show how it relates to other work in the field.

9.3 Relation to other work

A number of relations exist between the work here and other work in the field of natural-language discourse processing. The work does not reflect so much on, semantics-, or structure- based approaches to discourse processing as it does on intention-based approaches. There is a relation, but more an analogous one, to the work of Schank (1972, 1973, 1975) and Wilks (1973, 1975a, 1975b, 1975c) described in Chapter 2. Both Schank and Wilks proposed theories of semantics for natural-language processing using primitive units as a basis for defining word meanings. Schank proposed eleven primitive acts as basic elements of a semantics, called Conceptual Dependency (CD), for natural-language processing. He proposed that the eleven acts were sufficient for natural-language processing. In a similar way we have proposed primitive intentions as the basic elements of a pragmatics for the processing of natural language consultancy dialogues. However, it is not argued here that these primitive intentions are sufficient for natural-language discourse processing. In fact, it has been pointed out explicitly that the set of intentions defined here, although being the principal set, may be a small fraction of the actual set of such intentions for the natural-language consultancy domain. Having said that, a fact in favour of the set of intentions is that the results show they are useful and effective for modelling, in part, natural-language consultancy dialogue.

The work here relates to structure-based approaches to discourse processing with respect to the fact that the structure of dialogue is modelled from the point of view of intention sequences and their frequencies. Intention graphs are condensed representations of intention structure in dialogue. Structural phenomena of intentions in discourse have been defined and described, such as intention *loops*, and the *boomerang* effect. Grosz and Sidner (1985) discuss discourse structure in terms of three components: *linguistic*, *intentional*, and *attentional*. *Linguistic* structure represents utterances which are aggregated into discourse segments. The utterances in a segment are relevant to that segment, and the segment is relevant to the discourse. An hypothesis that could possibly be tested would be to determine whether intention analysis gives information about where discourse segment boundaries lie. With respect to *intentional* structure, Grosz and Sidner argue that such a structure represents the purposes of participants in discourse. They argue that *purposes* distinguish coherent from incoherent discourse. In a similar way, intention graphs are a representation of the coherence of a discourse. Grosz and Sidner discuss the intentions, or purposes, of discourse segments, rather than the intentions of individual utterances. They point out that the ordering of purposes, or intentions, in a discourse is important, and that is also important in the work presented here, where our analogy is intention sequences. Grosz and Sidner point out that the list of possible intentions that can serve as discourse purposes may be infinite.

Reichman (1985) proposes a theory of discourse which characterises a conversation as a hierarchical organisation of related utterances. She describes two major functions of an utterance:

(1) a continuation of what has gone on before, and (2) a new communicative act serving a new discourse role. Utterances of type 1 cause a shift in the discourse which she calls a *Conversational Move*. Examples of conversational moves are explanations/clarifications, presenting claims, and so on. The intention types discussed in our work could be termed *Conversational Moves*.

With respect to intention-based approaches to discourse processing the work here has a number of relations to the work of Schank and Abelson (1977). As we discussed in Chapter 4, Schank and Abelson introduce a theory of the structure of knowledge and how it can be used for understanding. The main application of the theory is natural-language processing. Scripts are introduced as a mechanism whereby knowledge is packaged for understanding so that people know correct behaviour in particular contexts. Scripts provide connectivity of events and store sequences of stereotypical events. In a similar manner intention sequences can represent stereotypical sequences of intentions in natural-language discourse. Schank and Abelson also discuss plans and goals. Plans give information about how actors achieve goals and they give relationships between events. In a similar fashion intention sequences represent plans for obtaining, or presenting, information in order to achieve goals.

Our work has much in common with the work of Cohen et al. (1982) who present a theory of how the user's intent can be inferred, and argue how a theory of intention analysis can be used to model the coherence of a discourse. Litman and Allen (1984) develop a model based on a hierarchy of plans and metaplans to account for clarification subdialogues. Allen (1979, 1983), Litman and Allen (1984), and Hinkelman and Allen (1989) all discuss how intention recognition and representation can be conducted. Appelt (1981, 1985) discusses language generation and communication based on planning. Carberry (1989) provides a pragmatics-based approach to ellipsis resolution. The theory uses discourse expectations and focusing heuristics to facilitate recognition of users' intent in elliptical fragments. Carberry discusses a number of dialogue examples and the various ways by which follow-up utterances occur in a dialogue. Pustejovsky's (1987) general approach to discourse processing models both intention and structure. Pustejovsky points out that *textual directives* are parts of the structural component of his model of discourse. These *textual directives* are what we call intentions in preceding chapters.

In all of the work we have discussed in Chapter 4, there are few details of what intentions are, how they link together in sequences to represent discourse coherence, and how they can be utilised. Also there is little empirical work on what intentions look like in discourse, and what frequencies there are of them.

We are not the first to recognise the importance of sequencing of intention in dialogue. Winograd and Flores (1986) propose a tool for conversation called a *coordinator*[3] which is designed for constructing and controlling conversation networks in large distributed electronic communication systems. They say in Winograd and Flores (1986)

A coordinator is part of a computer communication network (which might be based

[3]A detailed description of the *coordinator* is given in Flores and Ludlow (1981) and Flores (1982).

on local networks, time-sharing, or advanced telephone exchanges) to which all of the participants have access through some kind of workstations.
– Winograd and Flores (1986, p. 159),

Its objective is to provide a tool to operate conversations for action. They point out that there are few basic conversational building-blocks (e.g. request/promise, offer/acceptance, report/acknowledge) that occur in conversations for action. The coordinator, which has a menu-based interface, supports the following:

Communicative act origination: An individual performs a communicative act using the coordinator by (a) selecting the illocutionary force[4] from a small set of alternatives (e.g. request/promise), (b) indicating the propositional content[5] in text, (c) explicitly entering temporal relationships to other past and anticipated acts, e.g. By specifying directly a *request* and a *date* the listening is more constrained than it would be for the English sentence, "Would you be able to...."

Monitoring completion: Much of the moment-by-moment concern of language is directed towards the completion of conversations for action. For example, "What do I have to do now?" and "What do I need to check up on?" are questions about sub-conversations being completed. The coordinator can keep track of where things stand and when they will change. This can be used to send reminders, alerts, or breakdowns. Note, that the two examples shown here by Winograd and Flores have a striking resemblance to what were called *guidance* intentions in Chapters 5 and 8.

Keeping temporal relations: The coordinator can track time relationships within the network and use them to help anticipate and cope with breakdowns. Time is a critical aspect of communicative acts and in future research we would hope to include temporal tags with intention sequences.

Examination of the network: An individual can display part of the conversation network showing conversations and their status. In effect this would be similar to displaying a form of intention graph.

Automated application of recurrence: Every organisation deals with situations that recur and are handled in a standard way. If some request (e.g. payment) has not been met within a certain time, other requests are made. The coordinator can be given this pattern and trigger acts without intervention. Hence the coordinator automatically triggers *repetitions* of intention if original intentions are not executed.

Recurrence of propositional content: A crucial dimension of conversation for the coordinator is the illocutionary content of speech acts and its temporal relations. For example, a pur-

[4]The concept of *illocutionary force* was introduced in Chapter 4.
[5]The term *propositional content* is used here in the sense we use *topic* in the preceding chapters.

> chase order or travel-agent advice request, or other such forms are designed to facilitate the generation of requests dealing with a particular content.

The existing coordinator system (see Cashman and Holt 1980) is part of distributed programming environment for maintaining large software collections. This application of the coordinator was motivated by the fact that people wanted to fix bugs, make improvements and produce updated versions of programs as they were distributed and used. The coordinator is a computer tool to maintain the structure of requests and commitments and greatly improves productivity. Other such systems are being developed at a number of different institutions (see Holt et al. 1983). The coordinator is not equivalent to other mediums of communication such as face-to-face conversations, telephone conversations, or electronic messages. Illocutionary forces and temporality are explicitly, rather than implicitly, specified. Hence, there are many overlaps between the proposals of Winograd and Flores and the work completed here.

Our work has much in common with the arguments provided in Suchman (1990). Suchman conducts a number of empirical studies to analyse plans, or sequences of actions, in tasks such as using a photocopier. She proposes a view of action sequences which is drawn from recent developments in the social sciences, principally anthropology and sociology. She says,

> "Finally, and perhaps most importantly, this approach assumes that the coherence of action is not adequately explained by either preconceived cognitive schemas or institutionalized social norms. Rather, the organization of situated action is an emergent property of moment-by-moment interactions between actors, and between actors and the environments of their action."
> — Suchman (1990, p. 179).

Although, it seems obvious that organisation must be emergent from action, Suchman is relating the fact that much of the work in artificial intelligence and cognitive science has attempted to define that organisation, a priori, without properly looking at the phenomenon. We would argue that the organisation of sequences of intention is an emergent property of the interaction between users and computers, and that it would be difficult to see such an emergent property without collecting and analysing data from the domain as we have done.

There is also a relationship between our work and that of Sarantinos and Johnson (1990). They conduct an empirical study and analysis of natural-language dialogues between experts, novices and partial experts and from this analysis a theory of explanation dialogues is developed. They point out that most existing expert systems produce explanations without taking the previous discourse into account, and additionally without tailoring responses to an individual user's knowledge of the domain, beliefs and attitudes. They define sets of question types, just like the intention types developed in our work, and build networks of relationships between question types, much in the spirit of intention graphs.

While there are a number of implications of our work, there is much future work to be conducted. Now that a summary of the work, and how it relates to previous research, has been given we can discuss how the work can be extended.

9.4 Further work

There are a number of theoretical, computational, and empirical aspects of this work which can be developed further. First, let us consider a number of theoretical issues. It was mentioned in the previous section that intention sequence analysis could be used to solve the problems of reference resolution and ellipsis. Research could be conducted to show how intention sequence analysis would be used to determine segment boundaries in discourse, and to help in solving the reference resolution problem. Work on intention sequencing could be integrated with the work of Carberry (1989) to help solve the problem of ellipsis. The relationship between intention and propositional content, or topic, has not been explored in our work to any depth. It will be of interest to study the relationships between topic, and intention, in various dialogues.

It has been demonstrated that the analysis of intentions can solve, in part, the problem of natural-language dialogue processing. A stronger question can be asked: is the analysis of intentions *necessary* for natural-language dialogue between people and computers? This is a claim made by philosophers of language such as Dennett (1981) and Grice (1969). Another question which could be asked is whether there is any universal set of intentions in natural language. Also, if there is a universal set, how close do the intention sets produced here for the natural-language consultancy domain come to the universal set for that domain? Intuitively, it would seem that there is some sort of universal set because people, at least, are usually able to determine other people's intentions in dialogue. For example, when you tell me that you have a plan to write a Ph.D. thesis or that you have a plan to take a trip to Tokyo, I know exactly what you mean, or at least I think I do.

The question of whether there are only nine principal types of intention in natural-language consultancy dialogues could be asked, i.e. the intention set: *information*, *instruction*, *description*, *elaboration*, *confirmation*, *explanation*, *repetition*, *guidance*, and *nointention.* It is unlikely, as (1) we have pointed out before that these types can be broken down into others, and (2) new types will crop up in new experiments, as happened in Experiment II, described in Chapter 8, where two more types were found. Also, Wittgenstein (1963), Cohen et al. (1982), and Grosz and Sidner (1986) claim that the number of intention types may be unlimited. This would make sense, as the set of natural-language utterances is assumed by many to be unlimited and there is a many-many mapping from natural-language utterances to intentions. It is also the case that language is continuously changing and people can always use existing, and new language, to produce new types. Hence, work will have to be done with intention sets where it is realised that they are not circumscribing the complete set of possible intentions. Such approaches are common in natural-language processing and AI research.

While conducting intention analysis the main concentration here has been with intention pairs. It would be interesting to investigate whether intention triples would provide any more power than intention pairs. An analysis of more extended intention sequences might be useful in determining the long term goals of users during a dialogue. Experiments could be conducted to compare the power of the use of intention pairs with intention triples.

The effect of answers on users' future behaviour within a dialogue could be investigated. Such research could investigate which answers result in the best user performance.

In this work only a binary measure of expertise has been considered. An ordering of expertise could be studied and could be correlated with an ordering of intention type. Also, other orderings of intention such as *politeness* and *effectiveness* could be studied. A politeness ordering would measure the degree of politeness incorporated in various intention sequences, whereas an effectiveness ordering would measure the degree of effectiveness of different intention types for obtaining users' goals. However, such orderings may need to be more elaborate as the orderings may need to involve sequences of intention greater than one. For example, an *explanation* intention could be expressed more, or less politely, but in certain situations a repeated sequence of *explanation* intentions might not be considered polite. There are no hard and fast rules about the links between intention and orderings such as these and, in conjunction, much elaborate processing of the context of intention sequences, and their propositional content, would need to be taken into account. An analysis of the link between intention and politeness in discourse has been provided by Hovy (1988).

All of the above theoretical issues could be investigated in the context of computational models and Wizard-of-Oz experiments. With respect to intention graphs formal methods in graph theory could be applied to them to see if any information could be obtained. Graph theoretical methods are described in, for example, Harary (1969). Partridge (1991a, p. 81) points out that there has been a scarcity of the application of formal graph theoretical techniques to knowledge representation schemes in AI. Studies could be conducted to determine the effectiveness of intention graphs as intention-sequence representation formalisms. A number of different forms of intention graphs could be compared, such as the forms we introduced in Chapter 5. Of course, we have used a more primitive form of intention graph in the work reported here.

Next, some computational issues are considered. There are a number of ways in which the computational model, OSCON, could be extended. With respect to intention representation it it possible that the intention matrix, within the computational model, OSCON, could be utilised by other computational models. Specifically, belief ascription mechanisms such as those incorporated within the ViewGen program for generating beliefs and points of view (see Ballim and Wilks 1990, 1991, Wilks and Ballim 1987), and planning systems such as those described in Cohen et al. (1982) could use the intention matrix. These computational models could use the intention matrix to determine knowledge of the user's beliefs, plans and goals.

As pointed out in the two previous sections other types of human-computer interface, such as menu-based systems could be developed from empirical data. Such interfaces would show intention types as icons on the screen and users could select information based on their intentions. Various intention names could be used to determine their usefulness. Such interfaces would be similar to that of the *coordinator* system mentioned in the previous section.

Finally, a number of empirical studies could be conducted with computational models, like OSCON, and Wizard-of-Oz programs. Studies could be conducted to determine whether it is the case that natural-language responses given by OSCON modify the behaviour of users to any important extent, i.e. do changes in responses modify the user's questions, and hence intentions, and make it easier for the user to understand the dialogue? Such experiments would be conducted to determine how a system, which returns answers with many degrees of specificity, affects user-behaviour. It has already shown that the OSCON system, by conducting user-modelling, can modify its responses depending on the intention sequences it receives. It could be determined how much this significantly improved the performance of subjects, or their perception of the system. In addition, subjects could be presented with tasks, manifesting different levels of difficulty within a domain, to determine how their intention sequences change within that domain.

Experiments could be conducted to determine the discriminatory power of the computational model in recognising intentions. Such experiments would determine what proportion of utterances in a dialogue the computational model can adequately map into correct intention types. Questions like the following could be answered: are there major problems with discriminating intention in cases of ambiguity of intention?, what level of sub-typing of intentions is needed for a given domain in order for the computational model to be effective?

An interesting experiment would be to incorporate within OSCON the capability of detecting the rationality of incoming dialogues. Most of today's natural-language programs have not got the capability of detecting the rationality of the users interacting with them. Programs should have the capability to detect irrational behaviour, and be able to react to it. Otherwise such computational models will leave themselves open to being considered irrational!

We can see that there are a number of interesting avenues for future research. Now, we move on to conclude with what has been learned from the research already completed.

9.5 Conclusion

Our final conclusion then, is that a theory of intention analysis provides, in part, a solution to the problem of natural-language discourse processing. A central principle of the theory is that coherence of natural-language dialogue can be modelled by analysing sequences of intention. Additionally, that theory can be incorporated within a computational model, and used for applications such as user-modelling, solving ellipsis, and anaphor resolution. We have shown that the problem of user-modelling can be solved, in part, by the analysis of intention sequences. In turn, user-modelling can be used in order to generate user-sensitive natural-language responses. The theory of intention analysis is instantiated in terms of intention recognition and representation. In particular, the Intention-Computer and Intention-Person hypotheses were tested. The results gave positive evidence for the hypotheses and theory. It has been demonstrated that a computational model of natural-language dialogue processing can recognise and represent intentions, that the analysis of types and sequences of intention is useful for effective natural-language dialogue between different types of people and computers, and that a natural-language dialogue system such as OSCON will be 'handicapped' without the capability of intention-sequence analysis.

We argue that computational models should, at least, have the capability of analysing humans intentions. Otherwise, achieving natural-language dialogue interaction between people and computers will be difficult. The analysis of intention by computer, whether by natural language or otherwise will, hopefully, enable people to understand computers better, and computers to understand people better. The evidence here supports the claim that intention analysis can be useful for effective natural-language dialogue between different types of people and computers.

The analysis of intention can be used to build better computer programs which can communicate with people through dialogue whether that dialogue be in natural language, or otherwise. With such techniques people will be nearer to communicating with computers in their own natural language, rather than having to learn some abstract computer language. The hope is that, if they are communicating in the same language, computers will be better able to understand people's intentions, and likewise, people will be able to use computers more effectively.

Part VI

Appendices

Appendix A

OSCON sample traces

This appendix demonstrates the OSCON program answering questions about the UNIX and MS-DOS operating systems respectively. The first sample trace shows OSCON answering English queries about the UNIX operating system. The second trace shows OSCON answering queries about the MS-DOS operating system. The third trace shows OSCON operating in dialogue modelling and user modelling mode.

```
Quintus Prolog Release 3.0 (Sun-4, SunOS 4.1)
Copyright (C) 1990, Quintus Computer Systems, Inc.  All rights reserved.
1310 Villa Street, Mountain View, California U.S.A. (415) 965-7700

| ?- %  loading file /usr/local/lib/quintas/generic/qplib3.0/library/lists.qof
%   loading file /usr/local/lib/quintas/generic/qplib3.0/library/basics.qof
%   basics.qof loaded in module user, 0.083 sec 1,264 bytes
%   loading file /usr/local/lib/quintas/generic/qplib3.0/library/types.qof
%   types.qof loaded in module user, 0.084 sec 8,012 bytes
%  lists.qof loaded in module user, 0.733 sec 23,904 bytes
%  loading file /usr/local/lib/quintas/generic/qplib3.0/library/readin.qof
%  readin.qof loaded in module user, 0.100 sec 1,916 bytes
%  loading file /usr/local/lib/quintas/generic/qplib3.0/library/lineio.qof
%  lineio.qof loaded in module user, 0.150 sec 2,316 bytes
%  loading file /usr/local/lib/quintas/generic/qplib3.0/library/strings.qof
%   foreign file /usr/local/lib/quintas/generic/qplib3.0/library/sun4-4/
strings.o loaded
%  strings.qof loaded in module user, 0.934 sec 29,884 bytes
%  loading file /usr/local/lib/quintas/generic/qplib3.0/library/ask.qof
%   loading file /usr/local/lib/quintas/generic/qplib3.0/library/ctypes.qof
%   ctypes.qof loaded in module user, 0.100 sec 13,400 bytes
%   loading file /usr/local/lib/quintas/generic/qplib3.0/library/prompt.qof
%   prompt.qof loaded in module user, 0.217 sec 2,152 bytes
%   loading file /usr/local/lib/quintas/generic/qplib3.0/library/files.qof
%    foreign file /usr/local/lib/quintas/generic/qplib3.0/library/sun4-4/
files.o loaded
%    foreign file /usr/local/lib/quintas/generic/qplib3.0/library/sun4-4/
eaccess.o loaded
%    loading file /usr/local/lib/quintas/generic/qplib3.0/library/errno.qof
%    errno.qof loaded in module user, 0.134 sec 8,604 bytes
%   files.qof loaded in module user, 0.633 sec 16,644 bytes
%  ask.qof loaded in module user, 1.467 sec 43,768 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/parsecon/means.pl
%  means.pl compiled in module user, 0.100 sec 708 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/parsecon/whcmd.pl
* Singleton variables, clause 1 of findwcmd/2: E
* Singleton variables, clause 2 of findwcmd/2: X
* Singleton variables, clause 1 of wcmd/1: X
* Singleton variables, clause 2 of wcmd/1: X
* Singleton variables, clause 3 of wcmd/1: X
* Singleton variables, clause 4 of wcmd/1: X
* Singleton variables, clause 5 of wcmd/1: X
* Singleton variables, clause 6 of wcmd/1: X
* Singleton variables, clause 7 of wcmd/1: X
* Singleton variables, clause 8 of wcmd/1: X
* Singleton variables, clause 9 of wcmd/1: X
%  whcmd.pl compiled in module user, 0.283 sec 804 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/parsecon/whcmdcon.pl
* Singleton variables, clause 1 of findwcmdcon/2: E
* Singleton variables, clause 2 of findwcmdcon/2: X
* Singleton variables, clause 2 of findfirst/2: W
* Singleton variables, clause 2 of findsec/2: W
* Singleton variables, clause 1 of firstphr/2: X
* Singleton variables, clause 2 of firstphr/2: X
* Singleton variables, clause 3 of firstphr/2: X
* Singleton variables, clause 4 of firstphr/2: X
* Singleton variables, clause 1 of secphr/2: X
* Singleton variables, clause 2 of secphr/2: X
* Singleton variables, clause 3 of secphr/2: X
* Singleton variables, clause 4 of secphr/2: X
* Singleton variables, clause 5 of secphr/2: X
%  whcmdcon.pl compiled in module user, 0.450 sec 1,464 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/parsecon/whcon.pl
* Singleton variables, clause 1 of findwcon/2: E
* Singleton variables, clause 2 of findwcon/2: X
* Singleton variables, clause 2 of findfirstphrase/2: W
* Singleton variables, clause 2 of findsecphrase/2: W
* Singleton variables, clause 1 of firstphrase/2: X
* Singleton variables, clause 2 of firstphrase/2: X
* Singleton variables, clause 3 of firstphrase/2: X
* Singleton variables, clause 4 of firstphrase/2: X
* Singleton variables, clause 1 of secphrase/2: X
* Singleton variables, clause 2 of secphrase/2: X
* Singleton variables, clause 3 of secphrase/2: X
* Singleton variables, clause 4 of secphrase/2: X
* Singleton variables, clause 5 of secphrase/2: X
%  whcon.pl compiled in module user, 0.400 sec 1,420 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/parsecon/wheft.pl
* Singleton variables, clause 1 of findweft/2: E
* Singleton variables, clause 2 of findweft/2: X
* Singleton variables, clause 2 of findfirstdesc/2: W
* Singleton variables, clause 2 of findsecdesc/2: W
* Singleton variables, clause 1 of firstdesc/2: X
* Singleton variables, clause 2 of firstdesc/2: X
* Singleton variables, clause 3 of firstdesc/2: X
* Singleton variables, clause 1 of secdesc/2: X
* Singleton variables, clause 2 of secdesc/2: X
* Singleton variables, clause 3 of secdesc/2: X
* Singleton variables, clause 1 of desc/1: X
* Singleton variables, clause 2 of desc/1: X
* Singleton variables, clause 3 of desc/1: X
* Singleton variables, clause 4 of desc/1: X
* Singleton variables, clause 5 of desc/1: X
```

```
%  wheft.pl compiled in module user, 0.416 sec 1,324 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/parsecon/whsyn.pl
* Singleton variables, clause 1 of findwsyn/2: E
* Singleton variables, clause 2 of findwsyn/2: X
* Singleton variables, clause 2 of findfirstsyn/2: W
* Singleton variables, clause 2 of findsecsyn/2: W
* Singleton variables, clause 1 of firstsyn/2: X
* Singleton variables, clause 2 of firstsyn/2: X
* Singleton variables, clause 1 of secsyn/2: X
* Singleton variables, clause 2 of secsyn/2: X
* Singleton variables, clause 3 of secsyn/2: X
* Singleton variables, clause 1 of wsyn/1: X
* Singleton variables, clause 2 of wsyn/1: X
* Singleton variables, clause 3 of wsyn/1: X
* Singleton variables, clause 4 of wsyn/1: X
* Singleton variables, clause 5 of wsyn/1: X
* Singleton variables, clause 6 of wsyn/1: X
* Singleton variables, clause 7 of wsyn/1: X
* Singleton variables, clause 8 of wsyn/1: X
* Singleton variables, clause 9 of wsyn/1: X
%  whsyn.pl compiled in module user, 0.500 sec 1,556 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/parsecon/input.pl
* Singleton variables, clause 1 of oscon/0: X
* Singleton variables, clause 1 of receive/1: Itype
* Singleton variables, clause 2 of receive/1: Osys
* Singleton variables, clause 1 of query/2: X
* Singleton variables, clause 1 of changepreviousfquery/1: Pfquery
%  input.pl compiled in module user, 1.383 sec 4,952 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/parsecon/setdata.pl
* Singleton variables, clause 1 of resetmatrix/0: De5, El5, Ex5, Re5, No5,
  De4, El4, Ex4, Re4, No4, De3, El3, Ex3, Re3, No3, De2, El2, Ex2, Re2, No2,
 De1, El1, Ex1, Re1, No1, I, De, El, Ex, Re, No, I1, I2, I3, I4, I5
* Singleton variables, clause 1 of resetpreviousint/0: Intention
* Singleton variables, clause 1 of resetuserlevel/0: Elevel
* Singleton variables, clause 1 of resetpreviousfquery/0: Pfquery
%  setdata.pl compiled in module user, 0.550 sec 4,116 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/meancon/findcmd.pl
%  findcmd.pl compiled in module user, 0.100 sec 472 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/meancon/findcon.pl
* Singleton variables, clause 1 of ucon/3: P
* Singleton variables, clause 2 of ucon/3: S
%  findcon.pl compiled in module user, 0.283 sec 776 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/meancon/findverb.pl
%  findverb.pl compiled in module user, 0.234 sec 948 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/meancon/findobj.pl
%  findobj.pl compiled in module user, 0.533 sec 2,044 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/meancon/findmod.pl
* Singleton variables, clause 1 of findmod/4: E
* Singleton variables, clause 2 of findmod/4: B
* Singleton variables, clause 3 of findmod/4: Obj, Mod, B, E
* Singleton variables, clause 1 of getmod/3: W
* Singleton variables, clause 2 of getmod/3: A
%  findmod.pl compiled in module user, 0.383 sec 1,208 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/meancon/findloc.pl
* Singleton variables, clause 1 of findloc/3: E
* Singleton variables, clause 2 of findloc/3: B
* Singleton variables, clause 3 of findloc/3: Loc, B, E
* Singleton variables, clause 1 of getloc/2: W
* Singleton variables, clause 2 of getloc/2: A
* Singleton variables, clause 15 of oloc/2: Loc
```

```
%  findloc.pl compiled in module user, 0.433 sec 1,844 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/meancon/findbreak.pl
* Singleton variables, clause 3 of findbreak/5: W
%  findbreak.pl compiled in module user, 0.166 sec 624 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/meancon/dict.pl
%  dict.pl compiled in module user, 0.533 sec 2,024 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/knowcon/screenlist.pl
%  screenlist.pl compiled in module user, 0.250 sec 876 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/knowcon/printerlist.pl
%  printerlist.pl compiled in module user, 0.100 sec 328 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/knowcon/transfer.pl
%  transfer.pl compiled in module user, 0.067 sec 288 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/knowcon/remove.pl
%  remove.pl compiled in module user, 0.117 sec 436 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/knowcon/create.pl
%  create.pl compiled in module user, 0.083 sec 396 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/knowcon/send.pl
%  send.pl compiled in module user, 0.067 sec 332 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/knowcon/fileobj.pl
%  fileobj.pl compiled in module user, 0.633 sec 2,228 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/knowcon/unixobj.pl
* Clauses for date/2 are not together in the source file
* Clauses for queue/2 are not together in the source file
* Clauses for users/2 are not together in the source file
* Clauses for who/2 are not together in the source file
* Clauses for names/2 are not together in the source file
* Clauses for nroff/2 are not together in the source file
* Clauses for news/2 are not together in the source file
* Clauses for user/2 are not together in the source file
%  unixobj.pl compiled in module user, 0.983 sec 2,296 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/datacon/unixeffect.pl
%  unixeffect.pl compiled in module user, 0.600 sec 3,936 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/datacon/unixopeffect.pl
%  unixopeffect.pl compiled in module user, 29.817 sec 72,280 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/datacon/unixprecon.pl
%  unixprecon.pl compiled in module user, 0.350 sec 1,636 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/datacon/unixopprecon.pl
%  unixopprecon.pl compiled in module user, 0.800 sec 3,936 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/datacon/unixcomm.pl
%  unixcomm.pl compiled in module user, 0.566 sec 3,192 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/datacon/unixopcomm.pl
* Singleton variables, clause 4 of opcomm/3: GMT
* Clauses for opcomm/3 are not together in the source file
%  unixopcomm.pl compiled in module user, 1.066 sec 5,808 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/datacon/unixsyn.pl
* Singleton variables, clause 1 of notin/2: C
%  unixsyn.pl compiled in module user, 2.800 sec 7,452 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/datacon/syn.pl
%  syn.pl compiled in module user, 1.133 sec 3,268 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/datacon/doseffect.pl
%  doseffect.pl compiled in module user, 0.267 sec 1,600 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/datacon/dosprecon.pl
%  dosprecon.pl compiled in module user, 0.167 sec 796 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/datacon/doscomm.pl
%  doscomm.pl compiled in module user, 0.283 sec 1,480 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/datacon/dossyn.pl
%  dossyn.pl compiled in module user, 1.083 sec 3,076 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/datacon/obj.pl
%  obj.pl compiled in module user, 4.500 sec 10,744 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/datacon/plans.pl
%  plans.pl compiled in module user, 0.067 sec 388 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/solvecon/solver.pl
* Singleton variables, clause 1 of s_or_e_convert/6: S
* Singleton variables, clause 2 of s_or_e_convert/6: S
* Singleton variables, clause 3 of s_or_e_convert/6: S
* Singleton variables, clause 2 of convert/3: Verb
* Singleton variables, clause 3 of convert/3: Verb
* Clauses for convert/3 are not together in the source file
* Singleton variables, clause 1 of convert/3: Osys, S
%  solver.pl compiled in module user, 1.017 sec 2,528 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/solvecon/selectintention.pl
* Singleton variables, clause 1 of selectintention/5: Osys
* Singleton variables, clause 1 of intentiontype/3: Querytype
* Singleton variables, clause 3 of intentiontype/3: Formalquery
* Singleton variables, clause 5 of intentiontype/3: Formalquery
* Singleton variables, clause 7 of intentiontype/3: Formalquery
* Singleton variables, clause 9 of intentiontype/3: Formalquery
* Singleton variables, clause 11 of intentiontype/3: Formalquery
* Singleton variables, clause 12 of intentiontype/3: Formalquery
%  selectintention.pl compiled in module user, 0.550 sec 1,224 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/gencon/generate.pl
* Singleton variables, clause 6 of generate/3: Qtype
%  generate.pl compiled in module user, 0.133 sec 444 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/gencon/answerlevel.pl
* Singleton variables, clause 4 of calculateresponse/3: Querytype
* Singleton variables, clause 6 of expertanswer/3: Osys, Formalquery
%  answerlevel.pl compiled in module user, 0.283 sec 1,020 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/gencon/response.pl
* Singleton variables, clause 1 of objectanswer/2: Osys
* Singleton variables, clause 1 of commandanswer/2: Osys
* Singleton variables, clause 1 of syntaxanswer/2: Osys
* Singleton variables, clause 1 of preanswer/2: Osys
* Singleton variables, clause 2 of preanswer/2: Osys, Formalquery
* Singleton variables, clause 1 of effect1answer/3: Osys, Precond,
 Precond1, Formalquery1, Formalquery2
* Singleton variables, clause 2 of ifprintpipe/6: Syntax, Command,
 Command1, Effect, Effect1, Type
* Singleton variables, clause 1 of effectanswer/2: Osys
%  response.pl compiled in module user, 0.650 sec 1,996 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/gencon/peffect.pl
* Singleton variables, clause 1 of printeff/3: Command
* Singleton variables, clause 2 of printart/2: Obj, Mod
* Singleton variables, clause 2 of printmod/1: Mod
* Singleton variables, clause 2 of printpreps/2: Action, Loc
* Singleton variables, clause 1 of printloc/2: Action
* Singleton variables, clause 2 of printloc/2: Action, Loc
* Singleton variables, clause 7 of prepfor/2: Prep, Act
%  peffect.pl compiled in module user, 0.750 sec 2,260 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/gencon/psyntax.pl
* Singleton variables, clause 1 of printsyn/2: Command
* Singleton variables, clause 2 of printsynsyn/1: Syn
%  psyntax.pl compiled in module user, 0.167 sec 628 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/gencon/ppre.pl
%  ppre.pl compiled in module user, 0.167 sec 556 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/gencon/ppipe.pl
* Singleton variables, clause 1 of printpipe/5: Obj1, Cmd, Mod1, Loc
%  ppipe.pl compiled in module user, 0.166 sec 492 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/gencon/pexample.pl
* Singleton variables, clause 2 of printexample/4: Command, Obj, Act, Loc
* Singleton variables, clause 1 of printeloc/3: Act, Loc
* Singleton variables, clause 2 of printeloc/3: Obj1
* Singleton variables, clause 7 of argsfor/2: Args, Act
* Singleton variables, clause 3 of objfor/2: Obj, Act
%  pexample.pl compiled in module user, 0.583 sec 1,808 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/gencon/ptools.pl
%  ptools.pl compiled in module user, 0.267 sec 1,408 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/gencon/pmatrix.pl
%  pmatrix.pl compiled in module user, 1.350 sec 4,584 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/ucon/usermodel.pl
* Singleton variables, clause 1 of updateuser/2: Level
* Singleton variables, clause 2 of updateuser/2: Level, Satisfaction,
 Dissatisfaction
%  usermodel.pl compiled in module user, 0.417 sec 1,368 bytes
%  compiling file /home/atlas2/phd/pmc/oscon/dialcon/intentionrep.pl
* Singleton variables, clause 1 of updatematrix/2: Formalquery
* Singleton variables, clause 7 of updateintention/2: Newintention, Intention
* Singleton variables, clause 1 of intloc/2: I, De, El, Ex, Re, No
* Singleton variables, clause 2 of intloc/2: I, De, El, Ex, Re, No
* Singleton variables, clause 3 of intloc/2: I, De, El, Ex, Re, No
* Singleton variables, clause 4 of intloc/2: I, De, El, Ex, Re, No
* Singleton variables, clause 5 of intloc/2: I, De, El, Ex, Re, No
* Singleton variables, clause 6 of intloc/2: I, De, El, Ex, Re, No
%  intentionrep.pl compiled in module user, 1.600 sec 3,944 bytes
% compilation completed, 67.233 sec 289,748 bytes
```

Reads, and compiles into, Quintus Prolog all the predicates for running the OSCON program. This version of the program does not conduct any dialogue modelling or user modelling.

```
| ?- oscon.
```

Type 'oscon.' to Quintus Prolog to call up the OSCON program.

```
OSCON Program 1.0 (Sun-4, SunOS Release 4.1)
Copyright (C) 1988, Computing Research Laboratory. All rights reserved.
Dept. 3CRL, Box 30001, NMSU, Las Cruces, NM (505) 646-5466, USA.

U S WEST Advances Technology

Please input your question ending with with a '?'
Write 'quit.' when you are finished
OSCON can answer queries for UNIX or MSDOS.
Type one of these to the prompt.
Do you want answers for UNIX or MSDOS?
The default is UNIX [unix]: unix
```

OSCON provides a choice of UNIX or MS-DOS as operating systems on which it will answer questions. Here, we choose UNIX.

```
How can I help you?

--> how do i see my file?

'more <filename>' will display file contents on the screen.
```

OSCON parses and answers 'information' intentions[1]. This is an example of a 'request-for-command' query.

```
--> how does he see my file?

‘more <filename>’ will display file contents on the screen.
```

Note the change in person.

```
--> how do they see my file?

‘more <filename>’ will display file contents on the screen.
```

Note another change in person.

```
--> how do i see my files?

‘more <filename>’ will display file contents on the screen.
```

Note the change in object.

```
--> how does she see my file on the screen?

‘more <filename>’ will display file contents on the screen.
```

Note the addition of a location.

```
--> how does she see my file on the printer?

‘lpr <filename>’ will display file contents on the printer.
```

The specification of a new location changes the system response.

```
--> how does he print my file?

‘more <filename>’ will display file contents on the screen.
```

Note that the system assumes a preference for printing on the screen if no location is specified.

```
--> how does he print my file on the screen?

‘more <filename>’ will display file contents on the screen.
```

The verb ‘print’ does not bias the system over the correct location.

[1]All references to intention types in this appendix will be in a context-free sense as queries are considered independent of each other when the dialogue modeller and user modeller are not activated. When the dialogue modeller is switched on, the assignment of intention types to utterances can change. For example, if the above ‘information’ intention was followed by an ‘elaboration’ if the query preceding it was ‘How do I use more?’.

```
--> how does he print my file on the printer?

‘lpr <filename>’ will display file contents on the printer.

--> how do i display a file?

‘more <filename>’ will display file contents on the screen.
```

A change in verb will still be understood as referring to the same operating system action.

```
--> how do i read a file?

‘more <filename>’ will display file contents on the screen.
```

Another change in verb.

```
--> how do i look at a file?

‘more <filename>’ will display file contents on the screen.

--> how do i view a file?

‘more <filename>’ will display file contents on the screen.

--> how do i output a file?

‘more <filename>’ will display file contents on the screen.

--> how do i examine a file?

‘more <filename>’ will display file contents on the screen.

--> how do i list a file?

‘more <filename>’ will display file contents on the screen.

--> how do i get a file?

‘more <filename>’ will display file contents on the screen.

--> What is the syntax of more?

It has the syntax ‘more <filename>’.
```

The OSCON system parses and answers ‘explanation’ intentions. This is an example of a ‘request-for-syntax’ query.

```
--> How do I use more?

It has the syntax ‘more <filename>’.
```

Another form of ‘explanation’ intention. This is another example of a ‘request-for-syntax’ query.

```
--> What does more do?

'more <filename>' will display file contents on the screen.
```

The OSCON system parses and answers another type of 'explanation' intention. This is an example of a 'request-for-effect' query.

```
--> What is the effect of lpr?

'lpr <filename>' will display file contents on the printer.
```

Another form of 'request-for-effect'.

```
--> How do i remove my file?

'rm <filename>' will remove a file from your current directory.
```

More examples of 'information' intentions ....

```
--> How do i remove my text file?

'rm <filename>' will remove a file from your current directory.

--> How do i get rid of my file?

'rm <filename>' will remove a file from your current directory.

--> How do i copy a file?

'cp <file1> <file2>' will copy a file to another directory.

--> how do i copy my file to another directory?

'cp <file1> <file2>' will copy a file to another directory.

--> how do i move a file?

'cp <file1> <file2>' will copy a file to another directory.

--> how do i create a file?

'emacs <filename>' will create a file in your current directory.

--> how do i create a text file?

'emacs <filename>' will create a file in your current directory.

--> how do i list my file?

'more <filename>' will display file contents on the screen.

--> how do i list all my files?

'more <filename>' will display file contents on the screen.
```

```
--> how do i list a directory?

‘ls <directoryname>’ will display directory contents on the screen.
```

Note the change in object to ‘directory’ changes the response given with the verb ‘list’.

```
--> how do i list a directory on the screen?

‘ls <directoryname>’ will display directory contents on the screen.

--> how do i find out the users on the system?

‘users’ will display usernames on the screen.

--> how do i find out the date?

‘date’ will display a date on the screen.

--> how do i get help?

‘man <commandname>’ will display command information on the screen.

--> how do i see file?

‘more <filename>’ will display file contents on the screen.
```

Note the elision of a modifier/article on ‘file’ does not stifle a response.

```
--> how do see file?

‘more <filename>’ will display file contents on the screen.
```

Note the elision of a person/subject does not stifle a response.

```
--> see file?

‘more <filename>’ will display file contents on the screen.
```

The elision of all but verb and object does not stifle a response.

```
--> see?

I don’t understand. Please rephrase your query.
```

Too much information has been elided. OSCON points out that it didn’t parse the query.

```
--> file?

I don’t understand. Please rephrase your query.
```

```
--> how do i move a directory?

‘cp <file1> <file2>’ will copy a directory to another directory.

--> what is ls?

‘ls’ is a command.
```

OSCON parses and answers a ‘description’ intention. This is an example of a ‘request-for-description of command’ query.

```
--> what is cp?

‘cp’ is a command.

--> what is c?

C is a general-purpose programming language which
features economy  of expression, modern control flow
and data structures, and a rich  set of operators.
```

OSCON parses and answers another type of ‘explanation’ intention. This is an example of a ‘request-for-description of object’ query.

```
--> what is ada?

Ada is developed on behalf of the U.S. Department
of Defense for  use in embedded systems.  Ada is the
first practical language to bring  together important
features such as data abstraction, multitasking,
exception handling, encapsulation and generic.
```

Some more ‘request-for-description of object’ queries.

```
--> what are aliases?

alias is the ability to establish shorthand names
for frequently used but long-winded commands.

--> what is arpanet?

ARPANET is the network created by ARPA,the Advanced
Research Project AGENCY of the U.S. Department of
Defense.

--> what does rd do?

I don’t understand. Please rephrase your query.

--> quit.
over
yes
```

Now, we show a sample trace of OSCON for the MS-DOS operating system.

```
| ?- oscon.
```

The OSCON program is called up again.

```
OSCON Program 1.0 (Sun-4, SunOS Release 4.1)
Copyright (C) 1988, Computing Research Laboratory. All rights reserved.
Dept. 3CRL, Box 30001, NMSU, Las Cruces, NM (505) 646-5466, USA.

U S WEST Advances Technology

Please input your question ending with with a '?'
Write 'quit.' when you are finished
OSCON can answer queries for UNIX or MSDOS.
Type one of these to the prompt.
Do you want answers for UNIX or MSDOS?
The default is UNIX [unix]: msdos
```

MS-DOS is selected this time.

```
How can I help you?

--> how do i see my file?

'type <filename>' will display file contents on the screen.
```

OSCON parses and answers 'information' intentions, or 'request-for-command for effect' queries, for the MS-DOS operating system. Note that the answer is different than that for UNIX.

```
--> how does he see my file?

'type <filename>' will display file contents on the screen.

--> how do they see my file?

'type <filename>' will display file contents on the screen.

--> how do i see my files?

'type <filename>' will display file contents on the screen.

--> how does she see my file on the screen?

'type <filename>' will display file contents on the screen.

--> how does she see my file on the printer?

'print <filename>' will display file contents on the printer.

--> how does he print my file?

'type <filename>' will display file contents on the screen.

--> how does he print my file on the screen?
```

```
‘type <filename>’ will display file contents on the screen.

--> how do i display a file?

‘type <filename>’ will display file contents on the screen.

--> how do i read a file?

‘type <filename>’ will display file contents on the screen.

--> how do i look at a file?

‘type <filename>’ will display file contents on the screen.

--> how do i view a file?

‘type <filename>’ will display file contents on the screen.

--> how do i output a file?

‘type <filename>’ will display file contents on the screen.

--> how do i examine a file?

‘type <filename>’ will display file contents on the screen.

--> how do i list a file?

‘type <filename>’ will display file contents on the screen.

--> how do i get a file?

‘type <filename>’ will display file contents on the screen.

--> what is the syntax of more?

It has the syntax ‘more <filename>’.

--> How do i remove my file?

‘del <file>’ will remove a file from your current directory.

--> How do i remove my text file?

‘del <file>’ will remove a file from your current directory.

--> How do i get rid of my file?

‘del <file>’ will remove a file from your current directory.

--> How do i copy a file?

‘copy <file1>, <file2>’ will copy a file to another directory.

--> how do i copy my file to another directory?

‘copy <file1>, <file2>’ will copy a file to another directory.

--> how do i move a file?
```

```
‘copy <file1>, <file2>’ will copy a file to another directory.

--> how do i list my file?

‘type <filename>’ will display file contents on the screen.

--> how do i list all my files?

‘type <filename>’ will display file contents on the screen.

--> how do i list a directory?

‘dir <directory>’ will display directory contents on the screen.

--> how do i list a directory on the screen?

‘dir <directory>’ will display directory contents on the screen.

--> how do i find out the date?

‘date’ will display a date on the screen.

--> how do i see file?

‘type <filename>’ will display file contents on the screen.

--> how do see file?

‘type <filename>’ will display file contents on the screen.

--> see file?

‘type <filename>’ will display file contents on the screen.

--> see?

I don’t understand. Please rephrase your query.

--> file?

I don’t understand. Please rephrase your query.

--> how do i move a directory?

‘copy <file1>, <file2>’ will copy a directory to another directory.

--> what is c?

C is a general-purpose programming language which
features economy  of expression, modern control flow
and data structures, and a rich  set of operators.


--> quit.
over
yes
```

The following sample trace demonstrates how the OSCON program operates while incorporating the capability of intention-sequence analysis and representation, and while conducting user modelling.

```
| ?- % compiling procedure oscon/0 in module user
* Singleton variables, clause 1 of oscon/0: X
% compiling procedure receive/1 in module user
* Singleton variables, clause 2 of receive/1: Osys
% compiling procedure check/1 in module user
% compiling procedure query/2 in module user
* Singleton variables, clause 1 of query/2: X
% compiling procedure changepreviousfquery/1 in module user
* Singleton variables, clause 1 of changepreviousfquery/1: Pfquery
% compilation completed, 1.467 sec 96 bytes
```

Prolog predicates are invoked for recognising and representing intention sequences and conducting user modelling.

```
| ?- oscon.
OSCON Program 1.0 (Sun-4, SunOS Release 4.1)
Copyright (C) 1988, Computing Research Laboratory. All rights reserved.
Dept. 3CRL, Box 30001, NMSU, Las Cruces, NM (505) 646-5466, USA.

U S WEST Advances Technology

Please input your question ending with with a '?'
Write 'quit.' when you are finished
OSCON can answer queries for UNIX or MSDOS.
Type one of these to the prompt.
Do you want answers for UNIX or MSDOS?
The default is UNIX [unix]: unix

How can I help you?

--> what does ls do?

'ls <directoryname>' will display directory contents on the screen.

--> how do i see my file?

'more <filename>' will display file contents on the screen.

--> how does he see my file?

'more <filename>' will display file contents on the screen.

--> how do they see my file?

'more' is a command.
It has the syntax 'more <filename>'.
'more <filename>' will display file contents on the screen.
'more' only takes the argument 'file'.
```

Note that unlike the previous sample trace a more detailed answer is now given to the user. This has happened because a 'repetition → repetition' intention loop has occurred, tilting the level of 'dissatisfaction' higher than 'satisfaction'. More information is returned to the user.

```
--> how do i remove a file?

'rm <filename>' will remove a file from your current directory.
```

A 'repetition → information' intention has occurred tilting the level of 'satisfaction' back to equal that of 'dissatisfaction'.

```
--> how do i copy a file?

‘cp <file1> <file2>’ will copy a file to another directory.

--> how do i copy a file?

‘cp <file1> <file2>’ will copy a file to another directory.

--> how do i copy a file?

‘cp <file1> <file2>’ will copy a file to another directory.

--> how do i copy a file?

‘cp’ is a command.
It has the syntax ‘cp <file1> <file2>’.
‘cp <file1> <file2>’ will copy a file to another directory.
```

A set of three 'repetition → repetition' intention loops has occurred tilting the level of 'dissatisfaction' higher than 'satisfaction'. More information is returned to the user.

```
--> what does cf do?

I don’t understand. Please rephrase your query.

--> quit.
over
yes
```

Appendix B

Experiment I

The goals of Experiment I were twofold. First, to demonstrate that the OSCON system can conduct intention analysis over a sample natural-language dialogue by the processes of intention recognition, intention-sequence recognition, intention representation, and intention-sequence representation which can then be used by OSCON to conduct user-modelling. In turn, the user-model is used to generate natural-language responses which are sensitive to the level of user expertise. Second, three versions of OSCON were tested over the same natural-language dialogue. The first version of OSCON, the *experimental* has the full capability of intention sequence analysis. The second version of OSCON, a *control*, has no capability of sequence analysis. The third version, another *control*, has the capability of recognising sequences with only single intentions.

Sample runs of the OSCON system demonstrating the experimental and controls are shown below. Comments are supplied in boxes following each salient stage of processing.

```
| ?- oscon.
OSCON Program 1.0 (Sun-4, SunOS Release 4.1)
Copyright (C) 1988, Computing Research Laboratory. All rights reserved.
Dept. 3CRL, Box 30001, NMSU, Las Cruces, NM (505) 646-5466, USA.

U S WEST Advances Technology

Please input your question ending with with a '?'
Write 'quit.' when you are finished
OSCON can answer queries for UNIX or MSDOS.
Type one of these to the prompt.
Do you want answers for UNIX or MSDOS?
The default is UNIX [unix]: unix

How can I help you?

--> what does ls do?

'ls <directoryname>' will display directory contents on the screen.

information  : 0  0  0  0  0   0  =  0
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 0
```

```
explanation  : 0  0  0  0  0   0  =  0         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 0  0  0  0  0   0  =  0

                                    == 0
```

OSCON displays the 'intention matrix' and levels of 'satisfaction' and 'dissatisfaction'. Note that OSCON caters for 5 'real' types of intention, and nointentions. The 'intention matrix' is a 2-dimensional, 6 X 6 matrix. The 'intention matrix' shows all intention-sequence counts to be 0 as only one intention has been entered into the program. Totals for each intention type are also shown. 'Satisfaction' and 'dissatisfaction' levels, computed from the matrix, using the user modelling function, are also 0.

```
--> how do i see my file?

'more <filename>' will display file contents on the screen.

information  : 0  0  0  1  0   0  =  1
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 1
explanation  : 0  0  0  0  0   0  =  0         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 0  0  0  0  0   0  =  0

                                    == 1
```

This 'intention matrix' shows a total intention sequence count of 1, and the intention count for the sequence 'explanation → information' is set to 1. 'Satisfaction' also totals to 1.

```
--> how does he see my file?

'more <filename>' will display file contents on the screen.

information  : 0  0  0  1  0   0  =  1
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 2
explanation  : 0  0  0  0  0   0  =  0         Dissatisfaction = 0
repetition   : 1  0  0  0  0   0  =  1

nointention  : 0  0  0  0  0   0  =  0

                                    == 2
```

This 'intention matrix' shows a total sequence count of 2. Now, a 'information → repetition' sequence has occurred. Note that the syntax of this query does not have to be exactly the same as the previous query to denote a repetition. 'Satisfaction' increases to 2.

```
--> how do they see my file?

'more' is a command.
It has the syntax 'more <filename>'.
'more <filename>' will display file contents on the screen.
'more' only takes the argument 'file'.
```

```
information  : 0  0  0  1  0   0  =  1
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 2
explanation  : 0  0  0  0  0   0  =  0        Dissatisfaction = 3
repetition   : 1  0  0  0  1   0  =  2

nointention  : 0  0  0  0  0   0  =  0

                                  == 3
```

A ‘repetition → repetition’ intention loop has occurred tilting the level of ‘dissatisfaction’ higher than ‘satisfaction’. As a reaction, more information is returned to the user. Remember that the user modelling function gives intention loops which are along diagonals a weight of 3 (see Chapter 7).

```
--> how do i see my files?

‘more’ is a command.
It has the syntax ‘more <filename>’.
‘more <filename>’ will display file contents on the screen.
‘more’ only takes the argument ‘file’.

information  : 0  0  0  1  0   0  =  1
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 2
explanation  : 0  0  0  0  0   0  =  0        Dissatisfaction = 6
repetition   : 1  0  0  0  2   0  =  3

nointention  : 0  0  0  0  0   0  =  0

                                  == 4
```

Another ‘repetition → repetition’ intention loop occurs tilting the level of ‘dissatisfaction’ even higher.

```
--> how do i remove a file?

‘rm’ is a command.
It has the syntax ‘rm <filename>’.
‘rm <filename>’ will remove a file from your current directory.

information  : 0  0  0  1  1   0  =  2
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 3
explanation  : 0  0  0  0  0   0  =  0        Dissatisfaction = 6
repetition   : 1  0  0  0  2   0  =  3

nointention  : 0  0  0  0  0   0  =  0

                                  == 5
```

A ‘repetition → information’ intention sequence occurs tilting the level of ‘satisfaction’ up to 3.

```
--> how do i copy a file?
```

```
‘cp <file1> <file2>’ will copy a file to another directory.

information  : 1  0  0  1  1   0  =  3
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 6
explanation  : 0  0  0  0  0   0  =  0        Dissatisfaction = 6
repetition   : 1  0  0  0  2   0  =  3

nointention  : 0  0  0  0  0   0  =  0

                                    == 6
```

A ‘information → information’ intention loop occurs tilting the level of ‘satisfaction’ back to equal the level of ‘dissatisfaction’.

```
--> how do i copy a file?

‘cp <file1> <file2>’ will copy a file to another directory.

information  : 1  0  0  1  1   0  =  3
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 7
explanation  : 0  0  0  0  0   0  =  0        Dissatisfaction = 6
repetition   : 2  0  0  0  2   0  =  4

nointention  : 0  0  0  0  0   0  =  0

                                    == 7
```

A ‘information → repetition’ sequence occurs tilting the level of ‘satisfaction’ higher than the level of ‘dissatisfaction’.

```
--> how do i copy a file?

‘cp’ is a command.
It has the syntax ‘cp <file1> <file2>’.
‘cp <file1> <file2>’ will copy a file to another directory.

information  : 1  0  0  1  1   0  =  3
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 7
explanation  : 0  0  0  0  0   0  =  0        Dissatisfaction = 9
repetition   : 2  0  0  0  3   0  =  5

nointention  : 0  0  0  0  0   0  =  0

                                    == 8
```

A ‘repetition → repetition’ loop occurs tilting the level of ‘dissatisfaction’ higher than the level of ‘satisfaction’.

```
--> what does cf do?

I don’t understand. Please rephrase your query.

information  : 1  0  0  1  1   0  =  3
```

```
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 7
explanation  : 0  0  0  0  0   0  =  0        Dissatisfaction = 9
repetition   : 2  0  0  0  3   0  =  5

nointention  : 0  0  0  0  1   0  =  1

                                 == 9
```

OSCON could not understand this query and hence, a 'repetition → nointention' sequence occurs. Note that the levels of 'satisfaction' and 'dissatisfaction' do not change as 'nointentions' are not counted by the user modelling function.

```
--> how do i see a file?

'more' is a command.
It has the syntax 'more <filename>'.
'more <filename>' will display file contents on the screen.
'more' only takes the argument 'file'.

information  : 1  0  0  1  1   1  =  4
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 7
explanation  : 0  0  0  0  0   0  =  0        Dissatisfaction = 9
repetition   : 2  0  0  0  3   0  =  5

nointention  : 0  0  0  0  1   0  =  1

                                 == 10
```

A 'information' intention follows a 'nointention'. Again, note that the 'satisfaction' levels do not change as a 'nointention' is involved.

```
--> what does more do?

'more' is a command.
It has the syntax 'more <filename>'.
'more <filename>' will display file contents on the screen.
'more' only takes the argument 'file'.

information  : 1  0  0  1  1   1  =  4
description  : 0  0  0  0  0   0  =  0
elaboration  : 1  0  0  0  0   0  =  1        Satisfaction    = 8
explanation  : 0  0  0  0  0   0  =  0        Dissatisfaction = 9
repetition   : 2  0  0  0  3   0  =  5

nointention  : 0  0  0  0  1   0  =  1

                                 == 11
```

A 'information → elaboration' sequence occurs as the user has asked for an elaboration of a information. Note that the latter intention was not counted as an 'explanation' as it was elaborating a previous intention.

```
--> how do i copy a file?

'cp <file1> <file2>' will copy a file to another directory.
```

```
information  : 1  0  1  1  1   1  =  5
description  : 0  0  0  0  0   0  =  0
elaboration  : 1  0  0  0  0   0  =  1         Satisfaction    = 9
explanation  : 0  0  0  0  0   0  =  0         Dissatisfaction = 9
repetition   : 2  0  0  0  3   0  =  5

nointention  : 0  0  0  0  1   0  =  1

                                  == 12

--> what does more do?

'more <filename>' will display file contents on the screen.

information  : 1  0  1  1  1   1  =  5
description  : 0  0  0  0  0   0  =  0
elaboration  : 1  0  0  0  0   0  =  1         Satisfaction    = 10
explanation  : 1  0  0  0  0   0  =  1         Dissatisfaction = 9
repetition   : 2  0  0  0  3   0  =  5

nointention  : 0  0  0  0  1   0  =  1

                                  == 13
```

A 'information → explanation' sequence occurs. Note that this query is an exact copy of the last but 1; yet, because of a different context, it has now been counted as an 'explanation' rather than as an 'elaboration'.

```
--> how do i move a file?

'cp <file1> <file2>' will copy a file to another directory.

information  : 1  0  1  2  1   1  =  6
description  : 0  0  0  0  0   0  =  0
elaboration  : 1  0  0  0  0   0  =  1         Satisfaction    = 11
explanation  : 1  0  0  0  0   0  =  1         Dissatisfaction = 9
repetition   : 2  0  0  0  3   0  =  5

nointention  : 0  0  0  0  1   0  =  1

                                  == 14

--> what is the syntax of more?

It has the syntax 'more <filename>'.

information  : 1  0  1  2  1   1  =  6
description  : 0  0  0  0  0   0  =  0
elaboration  : 1  0  0  0  0   0  =  1         Satisfaction    = 12
explanation  : 2  0  0  0  0   0  =  2         Dissatisfaction = 9
repetition   : 2  0  0  0  3   0  =  5

nointention  : 0  0  0  0  1   0  =  1

                                  == 15

--> how do i see a file?
```

```
‘more <filename>’ will display file contents on the screen.

information  : 1  0  1  2  1   1  =  6
description  : 0  0  0  0  0   0  =  0
elaboration  : 1  0  0  1  0   0  =  2        Satisfaction    = 12
explanation  : 2  0  0  0  0   0  =  2        Dissatisfaction = 11
repetition   : 2  0  0  0  3   0  =  5

nointention  : 0  0  0  0  1   0  =  1

                                  == 16

--> how do i remove a file?

‘rm <filename>’ will remove a file from your current directory.

information  : 1  0  2  2  1   1  =  7
description  : 0  0  0  0  0   0  =  0
elaboration  : 1  0  0  1  0   0  =  2        Satisfaction    = 13
explanation  : 2  0  0  0  0   0  =  2        Dissatisfaction = 11
repetition   : 2  0  0  0  3   0  =  5

nointention  : 0  0  0  0  1   0  =  1

                                  == 17

--> what is more?

‘more’ is a command.

information  : 1  0  2  2  1   1  =  7
description  : 1  0  0  0  0   0  =  1
elaboration  : 1  0  0  1  0   0  =  2        Satisfaction    = 14
explanation  : 2  0  0  0  0   0  =  2        Dissatisfaction = 11
repetition   : 2  0  0  0  3   0  =  5

nointention  : 0  0  0  0  1   0  =  1

                                  == 18
```

An ‘information → description’ sequence occurs.

```
--> how do i see a file?

‘more <filename>’ will display file contents on the screen.

information  : 1  0  2  2  1   1  =  7
description  : 1  0  0  0  0   0  =  1
elaboration  : 1  1  0  1  0   0  =  3        Satisfaction    = 15
explanation  : 2  0  0  0  0   0  =  2        Dissatisfaction = 11
repetition   : 2  0  0  0  3   0  =  5

nointention  : 0  0  0  0  1   0  =  1

                                  == 19
```

A ‘description → elaboration’ sequence occurs. Note that the latter intention, because of its context, is counted as an ‘elaboration’ rather than the usual ‘information’.

```
--> what is more?

'more' is a command.

information  : 1  0  2  2  1   1  =  7
description  : 1  0  0  0  0   0  =  1
elaboration  : 1  1  1  1  0   0  =  4        Satisfaction    = 15
explanation  : 2  0  0  0  0   0  =  2        Dissatisfaction = 14
repetition   : 2  0  0  0  3   0  =  5

nointention  : 0  0  0  0  1   0  =  1

                                  == 20
```

This time 'what is more?' is counted as an 'elaboration' intention rather than a 'description' intention. Hence, an 'elaboration → elaboration' intention loop occurs. Again, the difference in context affects how the exact same utterance is interpreted. This causes 'dissatisfaction' to increase, whereas a 'description' would have caused 'satisfaction' to increase.

```
--> what is ada?

Ada is developed on behalf of the U.S. Department
of Defense for use in embedded systems.  Ada is the
first practical language to bring together important
features such as data abstraction, multitasking, exception
handling, encapsulation and generic.

information  : 1  0  2  2  1   1  =  7
description  : 1  0  1  0  0   0  =  2
elaboration  : 1  1  1  1  0   0  =  4        Satisfaction    = 16
explanation  : 2  0  0  0  0   0  =  2        Dissatisfaction = 14
repetition   : 2  0  0  0  3   0  =  5

nointention  : 0  0  0  0  1   0  =  1

                                  == 21
```

An 'elaboration → description' intention sequence has occurred.

```
--> how do i eat a file?

I don't understand. Please rephrase your query.

information  : 1  0  2  2  1   1  =  7
description  : 1  0  1  0  0   0  =  2
elaboration  : 1  1  1  1  0   0  =  4        Satisfaction    = 16
explanation  : 2  0  0  0  0   0  =  2        Dissatisfaction = 14
repetition   : 2  0  0  0  3   0  =  5

nointention  : 0  1  0  0  1   0  =  2

                                  == 22

--> quit.
over
yes
```

The OSCON program is exited.

In order to provide a control to demonstrate the advantage of intention-sequence analysis we can ‘handicap’ the OSCON program so that it does not have the capability of sequence analysis.

```
| ?- % compiling procedure selectintention/5 in module user
* Singleton variables, clause 1 of selectintention/5: Osys
% compiling procedure querytype/2 in module user
% compiling procedure intentiontype/3 in module user
* Singleton variables, clause 1 of intentiontype/3: Formalquery
* Singleton variables, clause 2 of intentiontype/3: Formalquery
* Singleton variables, clause 3 of intentiontype/3: Formalquery
* Singleton variables, clause 4 of intentiontype/3: Formalquery
* Singleton variables, clause 5 of intentiontype/3: Formalquery
* Singleton variables, clause 6 of intentiontype/3: Formalquery
% compiling procedure checktopic/1 in module user
% compiling procedure checkrepetition/1 in module user
% compilation completed, 0.483 sec -268 bytes
```

The ability of the OSCON program to recognise intention sequences and repetitions is removed.

```
| ?- reset.

yes
```

The ‘reset’ predicate resets the ‘intention matrix’ to contain all 0’s. Hence, processing begins with all intention counts set to 0.

```
| ?- oscon.
OSCON Program 1.0 (Sun-4, SunOS Release 4.1)
Copyright (C) 1988, Computing Research Laboratory. All rights reserved.
Dept. 3CRL, Box 30001, NMSU, Las Cruces, NM (505) 646-5466, USA.

U S WEST Advances Technology

Please input your question ending with with a ‘?’
Write ‘quit.’ when you are finished
OSCON can answer queries for UNIX or MSDOS.
Type one of these to the prompt.
Do you want answers for UNIX or MSDOS?
The default is UNIX [unix]: unix

How can I help you?

--> what does ls do?

‘ls <directoryname>’ will display directory contents on the screen.

information  : 0  0  0  0  0   0  =  0
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 0
explanation  : 0  0  0  0  0   0  =  0        Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 0  0  0  0  0   0  =  0
```

```
                                    == 0
```

As before, the 'intention matrix' shows all intention sequence counts to be 0 as only one intention has been entered into the program. 'Satisfaction' and 'dissatisfaction' levels, calculated from the matrix, are also 0.

```
--> how do i see my file?

'more <filename>' will display file contents on the screen.

information  : 0  0  0  1  0   0  =  1
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 1
explanation  : 0  0  0  0  0   0  =  0         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 0  0  0  0  0   0  =  0

                                    == 1
```

As before, the 'intention matrix' shows a total intention sequence count of 1 and the intention count for 'explanation → information' is set to 1. 'Satisfaction' also totals to 1.

```
--> how does he see my file?

'more <filename>' will display file contents on the screen.

information  : 1  0  0  1  0   0  =  2
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 4
explanation  : 0  0  0  0  0   0  =  0         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 0  0  0  0  0   0  =  0

                                    == 2
```

As before, the intention matrix shows a total sequence count of 2. Unlike before, a 'information → information' loop has occurred. This results in the 'satisfaction' count being increased to 4. Before, a 'information → repetition' sequence was detected and a satisfaction of 2 was computed. The OSCON program has lost its ability to detect repetitions of intention and hence the ability to correctly determine the level of user satisfaction. In fact, this constitutes a contradiction, as OSCON should not *increase satisfaction when a repetition of an intention has occurred!*

```
--> how do they see my file?

'more <filename>' will display file contents on the screen.

information  : 2  0  0  1  0   0  =  3
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 7
explanation  : 0  0  0  0  0   0  =  0         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 0  0  0  0  0   0  =  0
```

```
                                  == 3
```

Again, OSCON has failed to detect another 'repetition' and has increased the level of satisfaction.

```
--> how do i see my files?

'more <filename>' will display file contents on the screen.

information  : 3  0  0  1  0   0  =  4
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 10
explanation  : 0  0  0  0  0   0  =  0        Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 0  0  0  0  0   0  =  0

                                  == 4

--> how do i remove a file?

'rm <filename>' will remove a file from your current directory.

information  : 4  0  0  1  0   0  =  5
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 13
explanation  : 0  0  0  0  0   0  =  0        Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 0  0  0  0  0   0  =  0

                                  == 5

--> how do i copy a file?

'cp <file1> <file2>' will copy a file to another directory.

information  : 5  0  0  1  0   0  =  6
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 16
explanation  : 0  0  0  0  0   0  =  0        Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 0  0  0  0  0   0  =  0

                                  == 6

--> how do i copy a file?

'cp <file1> <file2>' will copy a file to another directory.

information  : 6  0  0  1  0   0  =  7
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 19
explanation  : 0  0  0  0  0   0  =  0        Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 0  0  0  0  0   0  =  0
```

```
                                  == 7

--> how do i copy a file?

`cp <file1> <file2>' will copy a file to another directory.

information  : 7  0  0  1  0   0  =  8
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 22
explanation  : 0  0  0  0  0   0  =  0         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 0  0  0  0  0   0  =  0

                                  == 8

--> what does cf do?

I don't understand. Please rephrase your query.

information  : 7  0  0  1  0   0  =  8
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 22
explanation  : 0  0  0  0  0   0  =  0         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 1  0  0  0  0   0  =  1

                                  == 9

--> how do i see a file?

`more <filename>' will display file contents on the screen.

information  : 7  0  0  1  0   1  =  9
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 22
explanation  : 0  0  0  0  0   0  =  0         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 1  0  0  0  0   0  =  1

                                  == 10

--> what does more do?

`more <filename>' will display file contents on the screen.

information  : 7  0  0  1  0   1  =  9
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 23
explanation  : 1  0  0  0  0   0  =  1         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 1  0  0  0  0   0  =  1

                                  == 11
```

OSCON has lost its ability to detect intention sequences and hence believes the latter intention to be an 'explanation', whereas it should be an elaboration of the previous intention: 'information → elaboration'.

```
--> how do i copy a file?

'cp <file1> <file2>' will copy a file to another directory.

information  : 7  0  0  2  0   1  =  10
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 24
explanation  : 1  0  0  0  0   0  =  1         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 1  0  0  0  0   0  =  1

                                  == 12

--> what does more do?

'more <filename>' will display file contents on the screen.

information  : 7  0  0  2  0   1  =  10
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 25
explanation  : 2  0  0  0  0   0  =  2         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 1  0  0  0  0   0  =  1

                                  == 13

--> how do i move a file?

'cp <file1> <file2>' will copy a file to another directory.

information  : 7  0  0  3  0   1  =  11
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 26
explanation  : 2  0  0  0  0   0  =  2         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 1  0  0  0  0   0  =  1

                                  == 14

--> what is the syntax of more?

It has the syntax 'more <filename>'.

information  : 7  0  0  3  0   1  =  11
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 27
explanation  : 3  0  0  0  0   0  =  3         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 1  0  0  0  0   0  =  1

                                  == 15
```

```
--> how do i see a file?

'more <filename>' will display file contents on the screen.

information  : 7  0  0  4  0   1  =  12
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 28
explanation  : 3  0  0  0  0   0  =  3         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 1  0  0  0  0   0  =  1

                                  == 16
```

Again, OSCON believes the latter intention to be a 'information' whereas it should be an elaboration of the previous intention: 'explanation → elaboration'.

```
--> how do i remove a file?

'rm <filename>' will remove a file from your current directory.

information  : 8  0  0  4  0   1  =  13
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 31
explanation  : 3  0  0  0  0   0  =  3         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 1  0  0  0  0   0  =  1

                                  == 17

--> what is more?

'more' is a command.

information  : 8  0  0  4  0   1  =  13
description  : 1  0  0  0  0   0  =  1
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 32
explanation  : 3  0  0  0  0   0  =  3         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 1  0  0  0  0   0  =  1

                                  == 18

--> how do i see a file?

'more <filename>' will display file contents on the screen.

information  : 8  1  0  4  0   1  =  14
description  : 1  0  0  0  0   0  =  1
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 33
explanation  : 3  0  0  0  0   0  =  3         Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 1  0  0  0  0   0  =  1

                                  == 19
```

OSCON believes the latter intention to be a 'information' whereas it should be an 'elaboration' of the previous intention: 'description → elaboration'.

```
--> what is more?

'more' is a command.

information  : 8  1  0  4  0   1  =  14
description  : 2  0  0  0  0   0  =  2
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 34
explanation  : 3  0  0  0  0   0  =  3        Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 1  0  0  0  0   0  =  1

                                  == 20
```

Again, OSCON believes the latter intention to be a 'description' whereas it should be an elaboration of the previous intention: elaboration → elaboration.

```
--> what is ada?

Ada is developed on behalf of the U.S. Department
of Defense for use in embedded systems.  Ada is the
first practical language to bring together important
features such as data abstraction, multitasking, exception
handling, encapsulation and generic.

information  : 8  1  0  4  0   1  =  14
description  : 2  1  0  0  0   0  =  3
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 37
explanation  : 3  0  0  0  0   0  =  3        Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 1  0  0  0  0   0  =  1

                                  == 21

--> how do i eat a file?

I don't understand. Please rephrase your query.

information  : 8  1  0  4  0   1  =  14
description  : 2  1  0  0  0   0  =  3
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 37
explanation  : 3  0  0  0  0   0  =  3        Dissatisfaction = 0
repetition   : 0  0  0  0  0   0  =  0

nointention  : 1  1  0  0  0   0  =  2

                                  == 22

--> quit.
over
yes
| ?-
```

The OSCON program is exited.

In order to provide a control to demonstrate the advantage of intention-pair sequence analysis over single-intention sequence analysis we can ‘handicap’ the OSCON program so that it only has the capability of single-intention sequence analysis.

```
| ?- Procedure usermodel/2 is being redefined in a different file -
    Previous file: /home/atlas2/phd/pmc/oscon/ucon/usermodel.pl
    New file:      /home/atlas2/phd/pmc/oscon/ucon/usermodel1.pl
Do you really want to redefine it? (y,n,p, or ? for help) y
% compiling procedure usermodel/2 in module user
Procedure modeluser/2 is being redefined in a different file -
    Previous file: /home/atlas2/phd/pmc/oscon/ucon/usermodel.pl
    New file:      /home/atlas2/phd/pmc/oscon/ucon/usermodel1.pl
Do you really want to redefine it? (y,n,p, or ? for help) y
% compiling procedure modeluser/2 in module user
Procedure updateuser/2 is being redefined in a different file -
    Previous file: /home/atlas2/phd/pmc/oscon/ucon/usermodel.pl
    New file:      /home/atlas2/phd/pmc/oscon/ucon/usermodel1.pl
Do you really want to redefine it? (y,n,p, or ? for help) y
% compiling procedure updateuser/2 in module user
* Singleton variables, clause 1 of updateuser/2: Level
* Singleton variables, clause 2 of updateuser/2: Level, Satisfaction,
Dissatisfaction
% compilation completed, 0.583 sec 96 bytes
```

The user-modelling function is changed so that it compares intention frequency counts, or single-intention sequences, rather than intention-pair sequences.

```
| ?- Procedure updatematrix/2 is being redefined in a different file -
    Previous file: /home/atlas2/phd/pmc/oscon/dialcon/intentionrep.pl
    New file:      /home/atlas2/phd/pmc/oscon/dialcon/intentionrep1.pl
Do you really want to redefine it? (y,n,p, or ? for help) y
% compiling procedure updatematrix/2 in module user
* Singleton variables, clause 1 of updatematrix/2: Formalquery
Procedure matrix/1 is being redefined in a different file -
    Previous file: /home/atlas2/phd/pmc/oscon/dialcon/intentionrep.pl
    New file:      /home/atlas2/phd/pmc/oscon/dialcon/intentionrep1.pl
Do you really want to redefine it? (y,n,p, or ? for help) y
% compiling procedure matrix/1 in module user
% compiling procedure updateintention/1 in module user
Procedure computeinttotals/6 is being redefined in a different file -
    Previous file: /home/atlas2/phd/pmc/oscon/dialcon/intentionrep.pl
    New file:      /home/atlas2/phd/pmc/oscon/dialcon/intentionrep1.pl
Do you really want to redefine it? (y,n,p, or ? for help) y
% compiling procedure computeinttotals/6 in module user
% compilation completed, 0.916 sec 896 bytes
```

The ability of the OSCON program to represent intention-pair sequences is removed. Hence, OSCON can only represent intention frequencies, or single-intention sequences. Hence, the intention matrix for OSCON will only display absolute intention frequency counts.

```
| ?- reset.

yes
```

The ‘reset’ predicate resets the ‘intention matrix’ to contain all 0’s. Hence, processing begins with all intention counts set to 0.

```
| ?- oscon.
OSCON Program 1.0 (Sun-4, SunOS Release 4.1)
Copyright (C) 1988, Computing Research Laboratory. All rights reserved.
Dept. 3CRL, Box 30001, NMSU, Las Cruces, NM (505) 646-5466, USA.

U S WEST Advances Technology

Please input your question ending with with a ‘?’
Write ‘quit.’ when you are finished
OSCON can answer queries for UNIX or MSDOS.
Type one of these to the prompt.
Do you want answers for UNIX or MSDOS?
The default is UNIX [unix]: unix

How can I help you?

--> what does ls do?

‘ls’ is a command.
It has the syntax ‘ls <directoryname>’.
‘ls <directoryname>’ will display directory contents on the screen.

information  : 0  0  0  0  0   0  =  0
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 0
explanation  : 1  0  0  0  0   0  =  1        Dissatisfaction = 1
repetition   : 0  0  0  0  0   0  =  0

nointention  : 0  0  0  0  0   0  =  0

                                  == 1
```

This time the numbers in the first column only represent intention frequencies. Unlike before, where intention-pair sequences were represented, now the intention matrix shows a count of 1 for an explanation intention. ‘Dissatisfaction’ is already increased to 1. However, OSCON has reacted too quickly as there is no real evidence as yet to believe that the user is unsatisfied. In the experimental *no satisfaction measure was determined. The total set of intentions is also 1 as now OSCON is counting intentions rather than intention sequences.*

```
--> how do i see my file?

‘more <filename>’ will display file contents on the screen.

information  : 1  0  0  0  0   0  =  1
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0        Satisfaction    = 1
explanation  : 1  0  0  0  0   0  =  1        Dissatisfaction = 1
repetition   : 0  0  0  0  0   0  =  0

nointention  : 0  0  0  0  0   0  =  0

                                  == 2
```

OSCON recognises an information intention and the satisfaction count is set to 1.

```
--> how does he see my file?

'more' is a command.
It has the syntax 'more <filename>'.
'more <filename>' will display file contents on the screen.
'more' only takes the argument 'file'.

information  : 1  0  0  0  0   0  =  1
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 1
explanation  : 1  0  0  0  0   0  =  1         Dissatisfaction = 2
repetition   : 1  0  0  0  0   0  =  1

nointention  : 0  0  0  0  0   0  =  0

                                  == 3
```

The intention matrix shows a total frequency count of 3. The dissatisfaction count has been set to 2, whereas in the first sample run the satisfaction count was increased. Unlike in the first sample run OSCON's user model has lost its ability to detect satisfaction *and* dissatisfaction *repetitions. It must now assume that all repetitions are an indication of dissatisfaction. However, in the first sample run OSCON was able to detect that the repetition was part of a direction → repetition pair, and hence that the satisfaction count should be increased. Now, OSCON has responded with more information when it didn't need to!*

```
--> how do they see my file?

'more' is a command.
It has the syntax 'more <filename>'.
'more <filename>' will display file contents on the screen.
'more' only takes the argument 'file'.

information  : 1  0  0  0  0   0  =  1
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 1
explanation  : 1  0  0  0  0   0  =  1         Dissatisfaction = 3
repetition   : 2  0  0  0  0   0  =  2

nointention  : 0  0  0  0  0   0  =  0

                                  == 4
```

OSCON's user model has increased the dissatisfaction count. This time OSCON is correct as a repetition has been repeated.

```
--> how do i see my files?

'more' is a command.
It has the syntax 'more <filename>'.
'more <filename>' will display file contents on the screen.
'more' only takes the argument 'file'.

information  : 1  0  0  0  0   0  =  1
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 1
explanation  : 1  0  0  0  0   0  =  1         Dissatisfaction = 4
repetition   : 3  0  0  0  0   0  =  3
```

```
nointention  : 0  0  0  0  0   0  =  0

                                  == 5
```

OSCON increases the dissatisfaction measure once again as another repetition has occurred.

```
--> how do i remove a file?

‘rm’ is a command.
It has the syntax ‘rm <filename>’.
‘rm <filename>’ will remove a file from your current directory.

information  : 2  0  0  0  0   0  =  2
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 2
explanation  : 1  0  0  0  0   0  =  1         Dissatisfaction = 4
repetition   : 3  0  0  0  0   0  =  3

nointention  : 0  0  0  0  0   0  =  0

                                  == 6
```

As an information intention has occurred OSCON increases the satisfaction level.

```
--> how do i copy a file?

‘cp’ is a command.
It has the syntax ‘cp <file1> <file2>’.
‘cp <file1> <file2>’ will copy a file to another directory.

information  : 3  0  0  0  0   0  =  3
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 3
explanation  : 1  0  0  0  0   0  =  1         Dissatisfaction = 4
repetition   : 3  0  0  0  0   0  =  3

nointention  : 0  0  0  0  0   0  =  0

                                  == 7
```

As another information intention has occurred OSCON has increased the level of satisfaction again.

```
--> how do i copy a file?

‘cp’ is a command.
It has the syntax ‘cp <file1> <file2>’.
‘cp <file1> <file2>’ will copy a file to another directory.

information  : 3  0  0  0  0   0  =  3
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 3
explanation  : 1  0  0  0  0   0  =  1         Dissatisfaction = 5
repetition   : 4  0  0  0  0   0  =  4
```

```
nointention  : 0  0  0  0  0   0  =  0

                                  == 8
```

Now another repetition has occurred and OSCON has increased the level of dissatisfaction. However, there is no justification for this and OSCON has done so because it could not determine the fact that the previous intention was one of satisfaction.

```
--> how do i copy a file?

‘cp’ is a command.
It has the syntax ‘cp <file1> <file2>’.
‘cp <file1> <file2>’ will copy a file to another directory.

information  : 3  0  0  0  0   0  =  3
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 3
explanation  : 1  0  0  0  0   0  =  1         Dissatisfaction = 6
repetition   : 5  0  0  0  0   0  =  5

nointention  : 0  0  0  0  0   0  =  0

                                  == 9
```

OSCON is justified in increasing the level of dissatisfaction this time.

```
--> what does cf do?

I don’t understand. Please rephrase your query.

information  : 3  0  0  0  0   0  =  3
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 3
explanation  : 1  0  0  0  0   0  =  1         Dissatisfaction = 6
repetition   : 5  0  0  0  0   0  =  5

nointention  : 1  0  0  0  0   0  =  1

                                  == 10
```

There is no change in satisfaction or dissatisfaction.

```
--> how do i see a file?

‘more’ is a command.
It has the syntax ‘more <filename>’.
‘more <filename>’ will display file contents on the screen.
‘more’ only takes the argument ‘file’.

information  : 4  0  0  0  0   0  =  4
description  : 0  0  0  0  0   0  =  0
elaboration  : 0  0  0  0  0   0  =  0         Satisfaction    = 4
explanation  : 1  0  0  0  0   0  =  1         Dissatisfaction = 6
repetition   : 5  0  0  0  0   0  =  5
```

```
nointention  : 1  0  0  0  0   0  =  1

                                 == 11

--> what does more do?

‘more’ is a command.
It has the syntax ‘more <filename>’.
‘more <filename>’ will display file contents on the screen.
‘more’ only takes the argument ‘file’.

information  : 4  0  0  0  0   0  =  4
description  : 0  0  0  0  0   0  =  0
elaboration  : 1  0  0  0  0   0  =  1        Satisfaction    = 4
explanation  : 1  0  0  0  0   0  =  1        Dissatisfaction = 7
repetition   : 5  0  0  0  0   0  =  5

nointention  : 1  0  0  0  0   0  =  1

                                 == 12
```

OSCON has detected an elaboration intention. However, OSCON has increased the level of dissatisfaction where it should have increased the satisfaction level. OSCON has lost its ability to detect that an elaboration of an information request is not an indication of dissatisfaction. OSCON cannot distinguish between satisfaction and dissatisfaction elaborations indicating satisfaction and dissatisfaction.

```
--> how do i copy a file?

‘cp’ is a command.
It has the syntax ‘cp <file1> <file2>’.
‘cp <file1> <file2>’ will copy a file to another directory.

information  : 5  0  0  0  0   0  =  5
description  : 0  0  0  0  0   0  =  0
elaboration  : 1  0  0  0  0   0  =  1        Satisfaction    = 5
explanation  : 1  0  0  0  0   0  =  1        Dissatisfaction = 7
repetition   : 5  0  0  0  0   0  =  5

nointention  : 1  0  0  0  0   0  =  1

                                 == 13

--> what does more do?

‘more’ is a command.
It has the syntax ‘more <filename>’.
‘more <filename>’ will display file contents on the screen.
‘more’ only takes the argument ‘file’.

information  : 5  0  0  0  0   0  =  5
description  : 0  0  0  0  0   0  =  0
elaboration  : 1  0  0  0  0   0  =  1        Satisfaction    = 5
explanation  : 2  0  0  0  0   0  =  2        Dissatisfaction = 8
repetition   : 5  0  0  0  0   0  =  5

nointention  : 1  0  0  0  0   0  =  1

                                 == 14
```

OSCON has detected an explanation intention and dissatisfaction has been increased. However, OSCON should not have reacted so quickly as the explanation only followed an information intention. However, OSCON has lost its ability to see which intention came before the information intention.

```
--> how do i move a file?

‘cp’ is a command.
It has the syntax ‘cp <file1> <file2>’.
‘cp <file1> <file2>’ will copy a file to another directory.

information  : 6  0  0  0  0   0  =  6
description  : 0  0  0  0  0   0  =  0
elaboration  : 1  0  0  0  0   0  =  1        Satisfaction    = 6
explanation  : 2  0  0  0  0   0  =  2        Dissatisfaction = 8
repetition   : 5  0  0  0  0   0  =  5

nointention  : 1  0  0  0  0   0  =  1

                                  == 15

--> what is the syntax of more?

‘more’ is a command.
It has the syntax ‘more <filename>’.
‘more <filename>’ will display file contents on the screen.
‘more’ only takes the argument ‘file’.

information  : 6  0  0  0  0   0  =  6
description  : 0  0  0  0  0   0  =  0
elaboration  : 1  0  0  0  0   0  =  1        Satisfaction    = 6
explanation  : 3  0  0  0  0   0  =  3        Dissatisfaction = 9
repetition   : 5  0  0  0  0   0  =  5

nointention  : 1  0  0  0  0   0  =  1

                                  == 16

--> how do i see a file?

‘more’ is a command.
It has the syntax ‘more <filename>’.
‘more <filename>’ will display file contents on the screen.
‘more’ only takes the argument ‘file’.

information  : 6  0  0  0  0   0  =  6
description  : 0  0  0  0  0   0  =  0
elaboration  : 2  0  0  0  0   0  =  2        Satisfaction    = 6
explanation  : 3  0  0  0  0   0  =  3        Dissatisfaction = 10
repetition   : 5  0  0  0  0   0  =  5

nointention  : 1  0  0  0  0   0  =  1

                                  == 17

--> how do i remove a file?

‘rm’ is a command.
```

```
It has the syntax ‘rm <filename>’.
‘rm <filename>’ will remove a file from your current directory.

information  : 7  0  0  0  0   0  =  7
description  : 0  0  0  0  0   0  =  0
elaboration  : 2  0  0  0  0   0  =  2         Satisfaction    = 7
explanation  : 3  0  0  0  0   0  =  3         Dissatisfaction = 10
repetition   : 5  0  0  0  0   0  =  5

nointention  : 1  0  0  0  0   0  =  1

                                  == 18

--> what is more?

‘more’ is a command.
It has the syntax ‘more <filename>’.
‘more <filename>’ will display file contents on the screen.
‘more’ only takes the argument ‘file’.

information  : 7  0  0  0  0   0  =  7
description  : 1  0  0  0  0   0  =  1
elaboration  : 2  0  0  0  0   0  =  2         Satisfaction    = 8
explanation  : 3  0  0  0  0   0  =  3         Dissatisfaction = 10
repetition   : 5  0  0  0  0   0  =  5

nointention  : 1  0  0  0  0   0  =  1

                                  == 19

--> how do i see a file?

‘more’ is a command.
It has the syntax ‘more <filename>’.
‘more <filename>’ will display file contents on the screen.
‘more’ only takes the argument ‘file’.

information  : 7  0  0  0  0   0  =  7
description  : 1  0  0  0  0   0  =  1
elaboration  : 3  0  0  0  0   0  =  3         Satisfaction    = 8
explanation  : 3  0  0  0  0   0  =  3         Dissatisfaction = 11
repetition   : 5  0  0  0  0   0  =  5

nointention  : 1  0  0  0  0   0  =  1

                                  == 20

--> what is more?

‘more’ is a command.
It has the syntax ‘more <filename>’.
‘more <filename>’ will display file contents on the screen.
‘more’ only takes the argument ‘file’.

information  : 7  0  0  0  0   0  =  7
description  : 1  0  0  0  0   0  =  1
elaboration  : 4  0  0  0  0   0  =  4         Satisfaction    = 8
explanation  : 3  0  0  0  0   0  =  3         Dissatisfaction = 12
repetition   : 5  0  0  0  0   0  =  5
```

```
nointention  : 1  0  0  0  0   0  =  1

                                  == 21
```

OSCON recognises an elaboration measure. However, OSCON has increased the dissatisfaction measure, whereas it should have increased the satisfaction measure. OSCON has lost its ability to determine when elaborations are ones of dissatisfaction rather than dissatisfaction.

```
--> what is ada?

Ada is developed on behalf of the U.S. Department
of Defense for use in embedded systems.  Ada is the
first practical language to bring together important
features such as data abstraction, multitasking, exception
handling, encapsulation and generic.

information  : 7  0  0  0  0   0  =  7
description  : 2  0  0  0  0   0  =  2
elaboration  : 4  0  0  0  0   0  =  4        Satisfaction    = 9
explanation  : 3  0  0  0  0   0  =  3        Dissatisfaction = 12
repetition   : 5  0  0  0  0   0  =  5

nointention  : 1  0  0  0  0   0  =  1

                                  == 22

--> how do i eat a file?

I don't understand. Please rephrase your query.

information  : 7  0  0  0  0   0  =  7
description  : 2  0  0  0  0   0  =  2
elaboration  : 4  0  0  0  0   0  =  4        Satisfaction    = 9
explanation  : 3  0  0  0  0   0  =  3        Dissatisfaction = 12
repetition   : 5  0  0  0  0   0  =  5

nointention  : 2  0  0  0  0   0  =  2

                                  == 23

--> quit.
over
yes
```

The OSCON program is exited. Now, note that the OSCON program has responded with full information for the complete dialogue except for the first utterance because of its over eagerness to attribute dissatisfaction to the user. Such eagerness arises because of OSCON's inability to detect when elaborations, explanations, and repetitions are indicating satisfaction rather than dissatisfaction.

Appendix C

Experiment II

The following appendix describes a Wizard-of-Oz experiment, conducted in the form of a written/interactive natural language dialogue, between *subjects* and *wizards*. The domain chosen was that of natural language consultancy on the UNIX operating system. The experiment enabled the study of a set of subjects typing English questions, to a computer, about the UNIX operating system. These questions were answered by a wizard at another monitor. The results of the experiment, and their implications, are described in Chapter 8. This appendix serves as a more detailed description of how the experiment was conducted.

C.1 Environment

The experiment utilised a set of 14 subjects The subjects were undergraduates at New Mexico State University (NMSU) in the state of New Mexico, USA, who were pursuing a variation of undergraduate degrees. The subjects were introduced to the experiment by being read a set of instructions which are shown below in Figure C.1. The subjects were told that a person would be monitoring their operations and they could call upon the person for help at any time. This was done to stop people just sitting at the terminal and doing nothing when a problem arose. This phenomenon happens quite frequently with this sort of experiment. The subjects were *not* told that a person would be answering their questions. Each subject sat at a Sun-3 monitor, in a large laboratory, with many people entering and leaving to do lab work, or to access their offices. The wizard sat at another Sun-3 monitor, in the same laboratory, with his/her back to the subject. The wizard was situated approximately eight meters directly behind the subject. The subject could not see what was on the wizard's screen, and likewise, the wizard could not see the subject's screen. An effort was made to help the subjects feel at home with the experiment, by telling them that it was not their performance with UNIX which was being tested. Each subject took about $1\frac{1}{2}$ hours to complete the tasks.

USER INSTRUCTIONS

The function of this test is to familiarize you with some of the basic commands and their options in the UNIX operating system. However, rather than learning UNIX in a more traditional way, we want you to obtain your knowledge about UNIX from the computer itself. We are testing an approach that will allow you to do this. You will be able to type questions about UNIX into a special window on the computer screen. The system will attempt to understand your question and will return an answer in another window on the screen. In a third window you will be able to try out what you learn by using UNIX commands. We will be giving you several tasks to complete with the UNIX system. These will appear in yet another window on your screen.

We are primarily concerned with how you obtain information from the computer, so if you have any questions, please ask the computer. In other words, we are mostly interested in how the question-answer system works. We are NOT testing YOUR ability to learn UNIX and we are not testing YOU. We are testing the system, so please help us out and ask lots of questions. The setup of the experiment is as follows. You have four windows on the screen:

WINDOW 1: (Top Right)

In this window you will be receiving instructions. These will instruct you to accomplish work using UNIX. Most of this work will be involve finding, looking at, printing, or moving text information that is already stored in the UNIX system. UNIX stores information in things called files. In order to group information together, Unix uses things called directories. Directories can contain files or other directories. As you use UNIX you can think of yourself as being in a directory called the "current directory." [EXPLAIN MORE IF NECESSARY]

When you have completed an instruction, move the mouse until the arrow pointer is pointing to the *TASK* button, and press the left button on the mouse. Instructions for the next task will then appear in the window. You will be given 16 tasks to complete. Please try to complete each one before going on to the next. If you need help, ask the computer. If you really get stuck I will be working nearby [DESCRIBE WHERE]. After you get the next instruction you will then need to move the mouse so that the pointer is in one of the other windows on your screen. Try moving the mouse around now and bring up the first instruction.

WINDOWS 2 & 3: (Bottom Right)

Another window is a question window in which you can type any questions you have about how to do things with UNIX. To ask a question, move your mouse until you see the cursor in the "Question" window. Then type your question. Then move the mouse to the ASK box and press the left button on the mouse. The system may take a while to respond but eventually you will see a response to your question in the "Answer" window. If the response is what you need, move your mouse to the UNIX window and use UNIX. If the response does not quite answer your question, or if you have another question, you can move your mouse to the CLEAR button, press the left mouse button, move your mouse back to the question window and type in a new question.

Try it now, type in a question about how you could get a file printed on the printer.

WINDOW 4: (Top Left)

You can type UNIX commands in the other window in order to accomplish the tasks that make up this test. Remember to type the `<RETURN>` key at the end of each UNIX command.

Figure C.1: Instructions read to subjects

C.2 Subject's screen layout

There were three windows displayed on the subject's monitor. A picture of the subject's monitor is shown in Figure C.2 below. One window, in the upper left of the monitor, contained a reduced version of a UNIX operating system shell. This reduced version did not contain all operating system commands, as only a limited set of commands were necessary for the experiment. The subject typed all commands to UNIX in this window.

Another window to the right of this contained three frames[1]: (1) *task* frame, (2) subject *question* frame, and (3) wizard *answer* frame. The task frame was on top, the subject query frame in the middle, and the wizard answer frame below that. The task frame displayed 17 tasks in turn, each one randomly selected from a computer file of 17. The subject selected tasks by clicking on a button, marked *TASK*. A small window in the task frame showed the number of tasks to be completed. The question frame allowed a subject to type queries, and then send them to the wizard, by clicking on an button marked *QUESTION*. There was also a button marked CLEAR which the subject used to clear the question buffer, before asking a new question. The answer frame, marked *ANSWER*, displayed the wizard's answer. This frame was not cleared, but scrolled up, as the wizard's answers filled up the frame.

Below the answer frame there were three buttons. One button, marked *RESET*, was used for resetting the Wizard-of-Oz program for a new user. Another button, marked QUIT, was used for quitting the program. The third button, marked *Printscreen*, allowed one to print the screen, on a laserwriter, at any given moment during the experiment.

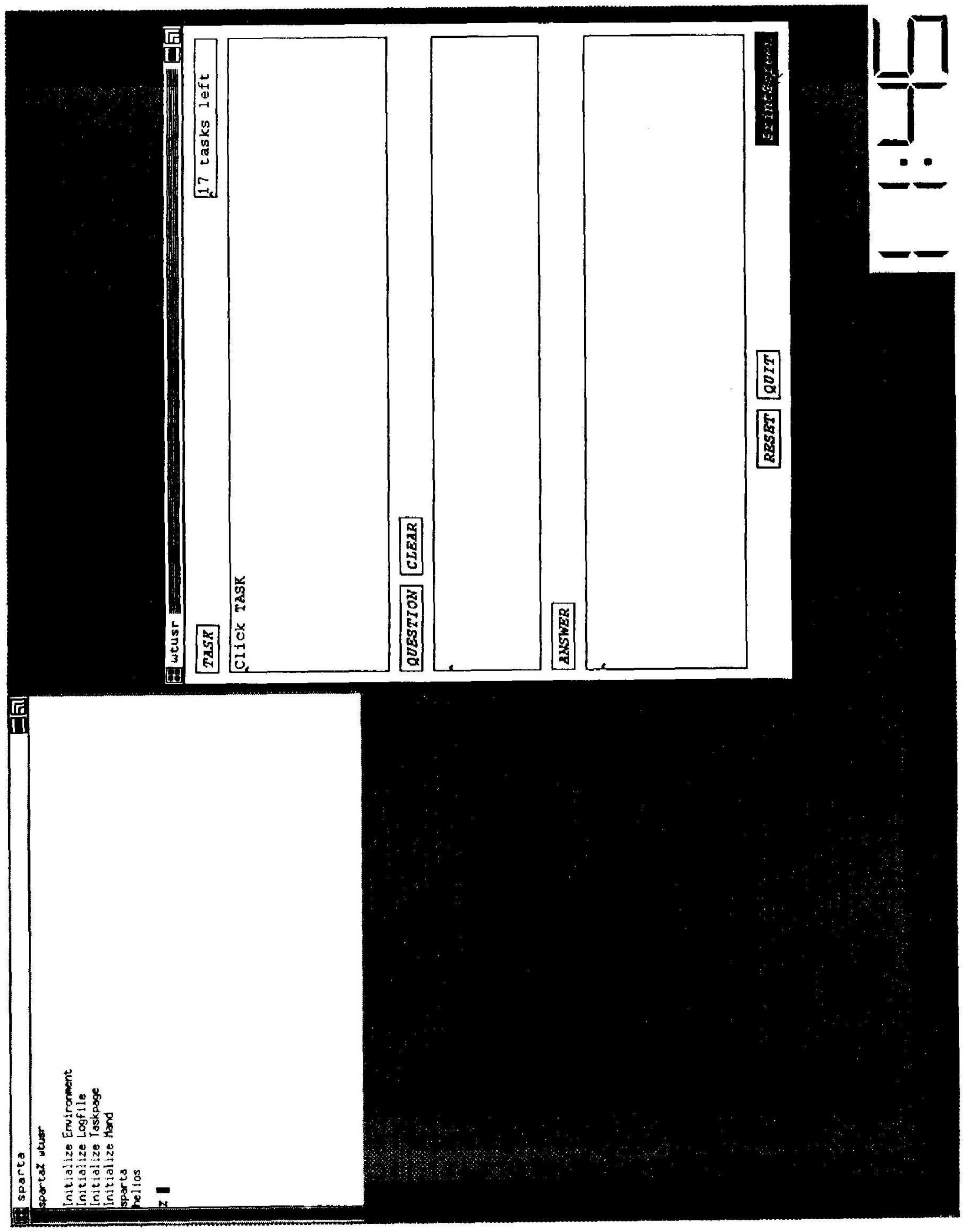
C.3 Wizard's screen layout

The wizard's screen layout, shown in Figure C.3 below, was virtually the same as the subject's. The top left of the screen contained a reduced UNIX shell, where the wizard could see what the subject was typing to his/her shell. On the right of that were frames for the subject's question, and the wizard's answer. Below that, was a frame where the wizard could type a note, or comment, which would be placed in a log file. These were displayed in the same way as for the subject. A small window showed the number of subject tasks to be completed, but the wizard did not see the tasks themselves.

C.4 Conducting the experiment

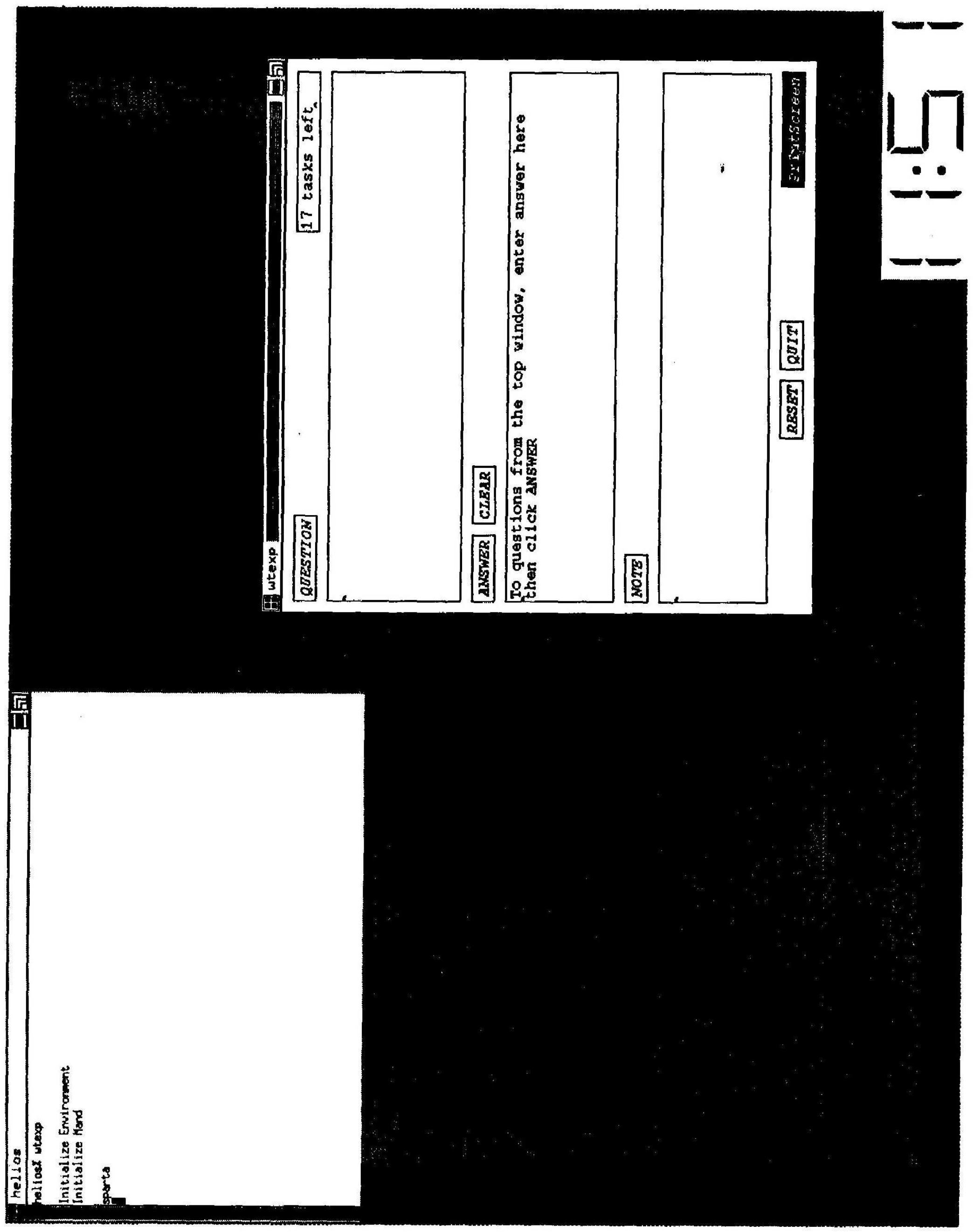
During the experiment the subject had to know about moving the mouse to place the cursor into the correct windows and on the right buttons. For example, the experiment involved placing the cursor in the TASK box, and clicking, when asking for a new task. Also, to ask a question, a subject had to move the cursor into the QUESTION frame. In cases where the subject cleared

[1]Note that we told the subjects, for ease of comprehension, that these frames were individual windows although, in fact, they were not.



```
sparta

sparta2 wtusr

Initialize Environment
Initialize Logfile
Initialize Taskpage
Initialize Hand
sparta
helios

z
```

Figure C.2: *Subject* screen layout



To questions from the top window, enter answer here
then click ANSWER

```
helios% wtexp
Initialize Environment
Initialize Hand
sparta
```

Figure C.3: *Wizard* screen layout

the QUESTION frame, by clicking on CLEAR, the cursor would automatically reset to the correct position in the QUESTION frame where a subject would type a question.

At the commencement of the experiment subjects were instructed to try a sample question by clicking on TASK. The sample question was the same for each subject, and involved printing a file on the printer. The complete set of tasks are shown in Figure C.4 below. The program mapped a prestored answer, about printing files on the printer, to the ANSWER window. Also, the subject was instructed to try the answer given by the computer, in the UNIX window. From then on the subject was on his/her own for the rest of the experiment, unless of course he/she called the wizard for help. The final task the subject had to do was to *logout* of the system.

Each time the subject, or the wizard, sent a message there was a bell beep from the wizard's, or subject's, console respectively. Also, when the subject typed a command to the UNIX shell, or moved on to click a new task, there was a bell beep from the wizard's console. Information from the screens was stored in log files. The subject and wizard did not see keystrokes, or mistyping, as only messages actually sent by the subject, or wizard, were shown on the screens. Subjects were not given a pen and paper, but it was noted that some of them jotted down information they were given through responses from the computer.

C.4.1 Tasks

The tasks that subjects had to execute were simple operating system tasks. Some of the tasks were specific to UNIX, others possible in any operating system. The tasks involved simple operating basics such as creating, displaying, printing, copying, moving, and removing files and directories. Some tasks involved displaying information about the system too. The tasks are shown in Figure C.4 below.

C.4.2 Logging the interaction

During each subject session data was collected and placed in a log file. A number of codes were used to mark each item of data. D marked the date and time that a session commenced. T marked the current task. U marked the subject's query, and W the wizard's response. C signaled any commands that the user typed to the UNIX shell. N marked any notes that the wizard made on the session. Each of T, U, W, C, and N were time-stamped with minutes and seconds. At the beginning of a session the date, and time, of the session were logged. Only sent messages were recorded in the log files, and keystrokes were not recorded. A sample piece of log data is shown in Figure C.5 below.

C.4.3 Questionnaire

Each subject was asked to fill out a questionnaire which recorded demographic data about the subject. The name, age, sex, and study major of subjects were recorded. Also, the computer,

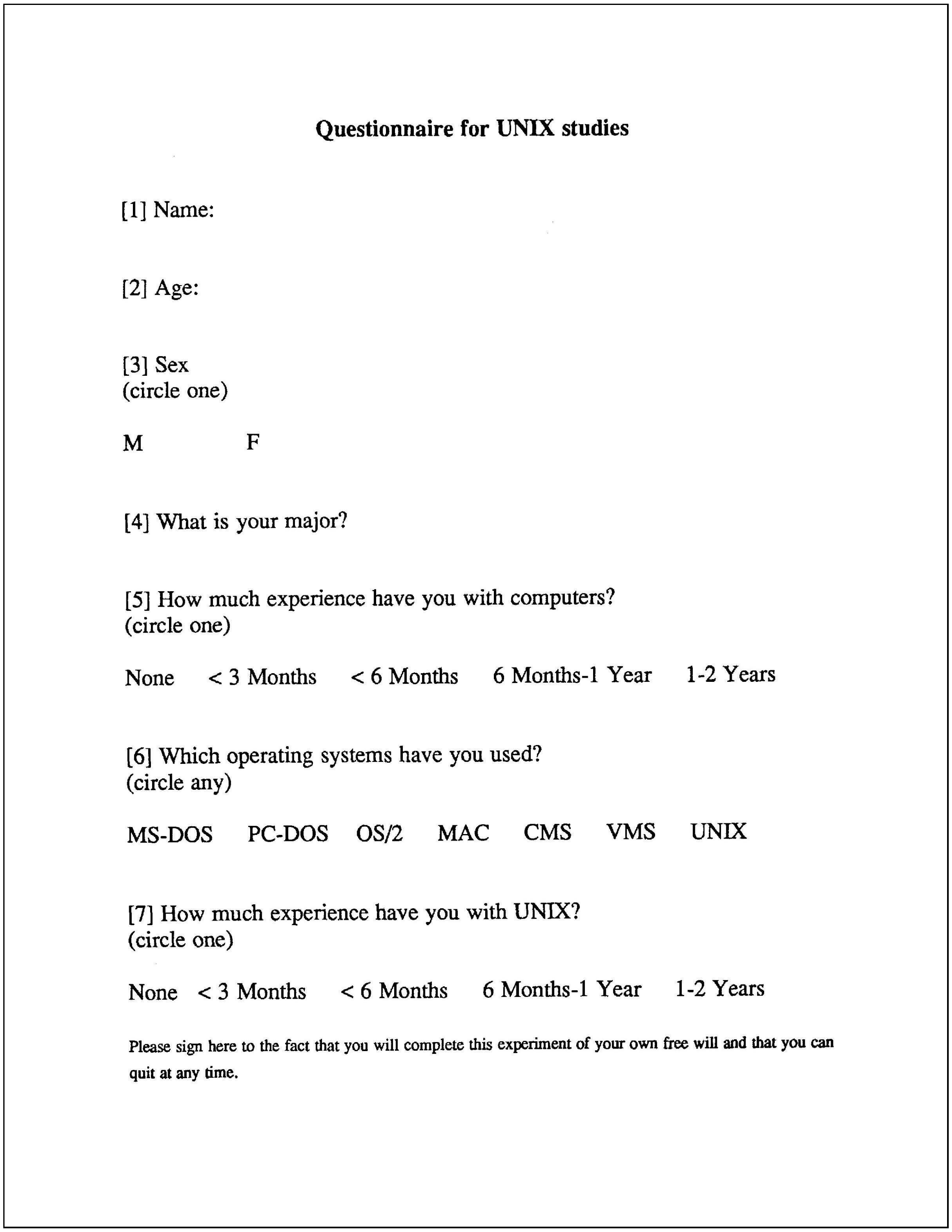
```
0. Print the file oscon

1. There is a file called "oscon" is in your directory.
Take a look inside this file

2. There is a subdirectory named "help" is in your current directory.
Find out what files are in "help"

3. Ask the computer for today's date

4. Find out who else is using the computer today

5. Produce a printed copy of the file called "oscon"

6. (a) Create a directory called "uswest" in your directory
(b) Check to see "uswest" now exists in your directory

7. (a) Make a copy of the file called "oscon" and call the new copy "oscon1"
(b) Look at your directory

8. (a) Make a copy of the subdirectory called "help" and
call the new copy "help1"
(b) Check to see that it now exists in your directory

9. (a) Change the name of the file called "file1" to "file2"
(b) Check that the name of the file has changed

10. (a) Change the name of the directory called "dir" to "dir1"
(b) Check that the directory name has changed

11. (a) Move from your current directory to the subdirectory named "help"
(b) Look at the files in your new working directory
(c) Go back to your original directory
(d) Check to see that you are in the directory called "oscon"

12. (a) Move the file called "moveme" in your current directory
to the directory named "here"
(b) Check to see that the file "moveme" has gone
(c) Check that "moveme" is now in "here"

13. (a) Put the file called "movemeto" in the subdirectory called "tohere"
(b) Check that "movemeto" has been moved
(c) Check that "movemeto" is in "tohere"

14. (a) Get rid of the file called "dumpme"
(b) Check to see that "dumpme" has gone

15. (a) Get rid of the directory named "rubbish"
(b) Check that "rubbish" is gone

16. Stop, and tell the computer you are finished for today
```

Figure C.4: *Subject* tasks

```
D: Thu Oct 12 12:57:37 MDT 1989

T: 05:47 Print the file oscon

U: 06:36 How do I print the file

W: 06:37 To print a file, use 'lpr <filename>'

C: 07:02 lpr oscon

C: 07:09

T: 07:50 (a) Move from your current directory to the subdirectory named "help" (b)
Look at the files in your new working directory (c) Go back to your original
directory (d) Check to see that you are in the directory called "oscon"

U: 08:32 How do I change the directory?

W: 09:11 to change the directory use 'cd <directoryname>'

C: 09:23 cd help

U: 09:53 How do I list files?

W: 10:11 to list the files use 'ls'

C: 10:17 ls

N: 10:52 * i took over from bill here.

U: 11:09 How do I find out what the last directory used was?
W: 11:51 to find out the last directory use 'cd ..' and then use 'ls'

C: 11:58 cd ..

C: 12:02 ls

T: 12:25 Find out who else is using the computer today
```

Figure C.5: Sample log data

operating systems, and UNIX experience, of subjects was recorded. A sample questionnaire is shown in Figure C.6.



Questionnaire for UNIX studies

[1] Name:

[2] Age:

[3] Sex
(circle one)

M F

[4] What is your major?

[5] How much experience have you with computers?
(circle one)

None < 3 Months < 6 Months 6 Months-1 Year 1-2 Years

[6] Which operating systems have you used?
(circle any)

MS-DOS PC-DOS OS/2 MAC CMS VMS UNIX

[7] How much experience have you with UNIX?
(circle one)

None < 3 Months < 6 Months 6 Months-1 Year 1-2 Years

Please sign here to the fact that you will complete this experiment of your own free will and that you can quit at any time.

Figure C.6: Questionnaire

Appendix D

Experiment III

This appendix describes a second Wizard-of-Oz experiment conducted in the form of a written/interactive natural-language dialogue for the UNIX help domain. Results, and implications of this experiment are described in Chapter 8. This appendix gives details of how the experiment was conducted. This experiment differs from the first in that it incorporates (1) a menu where the wizard can select pre-stored responses, (2) tasks are specified in picture form rather than in English, and (3) two subject groups with different levels of expertise are compared. The experiment involved comparing an *experimental*, a group of subjects who had taken a class on UNIX, to a *control*, a group who had little UNIX experience.

D.1 Environment

The second experiment served as a comparative study to determine if there was a significant difference in the questions asked by subject groups with different levels of expertise. The experiment was conducted in the domain of computer operating systems.

There were two groups of subjects: (1) those who had taken a class on UNIX, learning simple UNIX commands, and (2) those who had *not* taken the class and had little UNIX experience. The subjects who had taken the UNIX class were Psychology graduate students at New Mexico State University (NMSU), in the state of New Mexico, USA. The subjects who had not taken the class were, on the whole, graduates students from NMSU. There were 16 subjects in all, eight in each group.

Instructions were read to each subject before the experiment commenced. These are shown in Figure D.1 below. The environment for this experiment was similar to that used in Experiment II. The subjects were told that a person would be monitoring their input and they could call for help at any time. Again, subjects were *not* told that it was a person who was answering their questions. Each subject took about one hour to complete the tasks.

USER INSTRUCTIONS

The function of this test is to familiarize you with some of the basic commands and their options in the UNIX operating system. However, rather than learning the UNIX commands in a more traditional way, we want you to obtain your knowledge about UNIX from the computer itself. We are testing an approach that will allow you to do this. You will be able to type questions about UNIX commands into a special window on the computer screen. The system will attempt to understand your question and will return an answer in another window on the screen.

First let me describe some basics about UNIX which you will need to know in order to ask questions. UNIX stores information in units called *files*. Files contain words or numbers and there are UNIX commands for displaying the contents of a file on the screen, printing them on a printer, and for deleting, creating, and renaming files. Part of what you need to do is to discover what these command are called in UNIX. In addition, files are contained within units called directories. We have them drawn here to look like folders. Directories or folders can contain files, or they can contain other directories. Here we have them drawn with the name of the directory shown in the tab section of the folder (for example, *home* is the name of this directory [POINT]; *letters* is the name of this directory). Within each directory is a list of the names of the files contained in that directory (for example *edit* and *spreadsheet* are names of files in the *programs* directory). If a directory has a line coming out the bottom of the picture of its folder, it also contains a sub-directory (for example, *programs* contains the sub-directory *data*). Now, there are UNIX commands for displaying the contents of directories, moving files from one directory to another, changing the names of directories, and again these are the commands that you will need to ask about. So, now you know that UNIX stores information in files, files are stored in directories, and directories can contain other directories, and that UNIX provides commands for creating and modifying all of these units. One more thing that you will need to know is that UNIX has a concept of a current directory. When you type a UNIX command, you usually type a command name followed by either a file name or a directory name. The file or directory name you use, MUST be in your current directory (... unless you are creating a new one, in which case it will be in your current directory with the command is executed). Therefore, you must be able to change your current directory, so that you will be able to refer to the right items. Fortunately, there are also UNIX commands for changing the current directory, either by going down, or by going up, in the hierarchy of directories. You will be starting out with your current directory at the top of the picture you see. That is, the name or you current directory will be *home*.

Here is what we want you to do. What you have in front of you is a picture of an arrangement of files and directories that we want you to create. But, you do not have to do it from scratch. In fact, every file you need is already in the system, you just have to rename and move things around a bit. But in order to do that, you first have to discover what is there. If you find a file with the same name as what is on your picture, you can assume it is the same file, but you may have to move it to a new directory, or change the name of the directory. You may also have to create some new directories to move the files into. Some files in your picture do not already exist but the contents of those files exist in other files together to create a third. The contents of the files you need to create are listed at the bottom of the picture. Remember, the contents already exist in other files. If you find a file that is not on your picture, delete it, but making sure it does not have the contents you need for another file. Your job is to ask our experimental help system, about the commands that will allow you to accomplish all of this work.

⋮

⋮

Here is an example> In your home directory there is a directory named *fun* and in it contains the files *Jolt* and *CrystalWonder*. To discover this you first have to get a list of the directories you have in *home* (in the process, discovering *fun*) and then listing the files contained in *fun*. So you first have to ask what command in UNIX allows you to look at the contents of directories. Then you notice that in your source picture you have a directory called *Games* that has the same two files as *fun* has. So you need to rename the *fun* directory to be the *Games* directory. So you have to ask what command changes the names of directories. Any questions?

One final thing, there is a file in the home directory named *ToDo*. We want you to also follow the instructions that are contained in that file.

We are primarily concerned with how you obtain information from the computer, so if you have any questions, please ask the computer. In other words, we are mostly interested in how the question-answer system works. We are NOT testing YOUR ability to learn UNIX and we are not testing YOU. We are testing the system, so please help us out and ask lots of questions.

The setup of the experiment is as follows.

You have three windows on the screen:

The first window is a question window in which you can type any questions you have about how to do things with UNIX. To ask a question, move your mouse until you see the cursor in the *Question* window. Then type your question. Then move the mouse to the *ASK* button and press the left button on the mouse. The system may take a while to respond but eventually you will see a response to your question in the *Answer* window. If the response is what you need, move your mouse to the UNIX window and use UNIX. If the response does not quite answer your question, or if you have another question, you can move your mouse to the CLEAR button, press the left mouse button, move your mouse back to the question window and type in a new question.

The answer will give you information about a command and will show you how to enter the command after a colon. Type everything you see after the colon EXCEPT substitute the name of the particular file or directory name in place of the words *filename oldname*, *newname*, *directoryname*, etc.

Try it now, type in a question about how you could get a file printed on the printer.

To create the arrangement of files in the picture you need to enter the UNIX commands into the UNIX window. You have to remember to move your mouse pointer into the window before you type a command. Also, remember to type the `<RETURN>` key at the end of each UNIX command.

ANY QUESTIONS??

Figure D.1: Instructions read to subjects

D.2 Subject's screen layout

There were two windows displayed on the subject's monitor as shown in Figure D.2 below. The setup was virtually the same as in experiment II. However, the window to the right contained only two frames: (1) subject *question* frame, and (2) wizard *answer* frame. The subject question frame was on top, and the wizard answer frame below that. The question frame allowed the subject to type queries and then send them to the computer by clicking a button marked *QUESTION*. There was also a button marked *CLEAR* which the subject used to clear the question buffer before asking a new question. The answer frame marked *ANSWER* displayed the wizard's answer. This frame was not cleared, but scrolled up, as the answers filled up the frame.

Below the answer frame there were three buttons. One button marked *RESET* was used for resetting the wizard program for a new subject. Another button marked *QUIT* was used for quitting the program. The third button marked *Printscreen* allowed one to print the screen on the laserwriter at any given moment during the experiment.

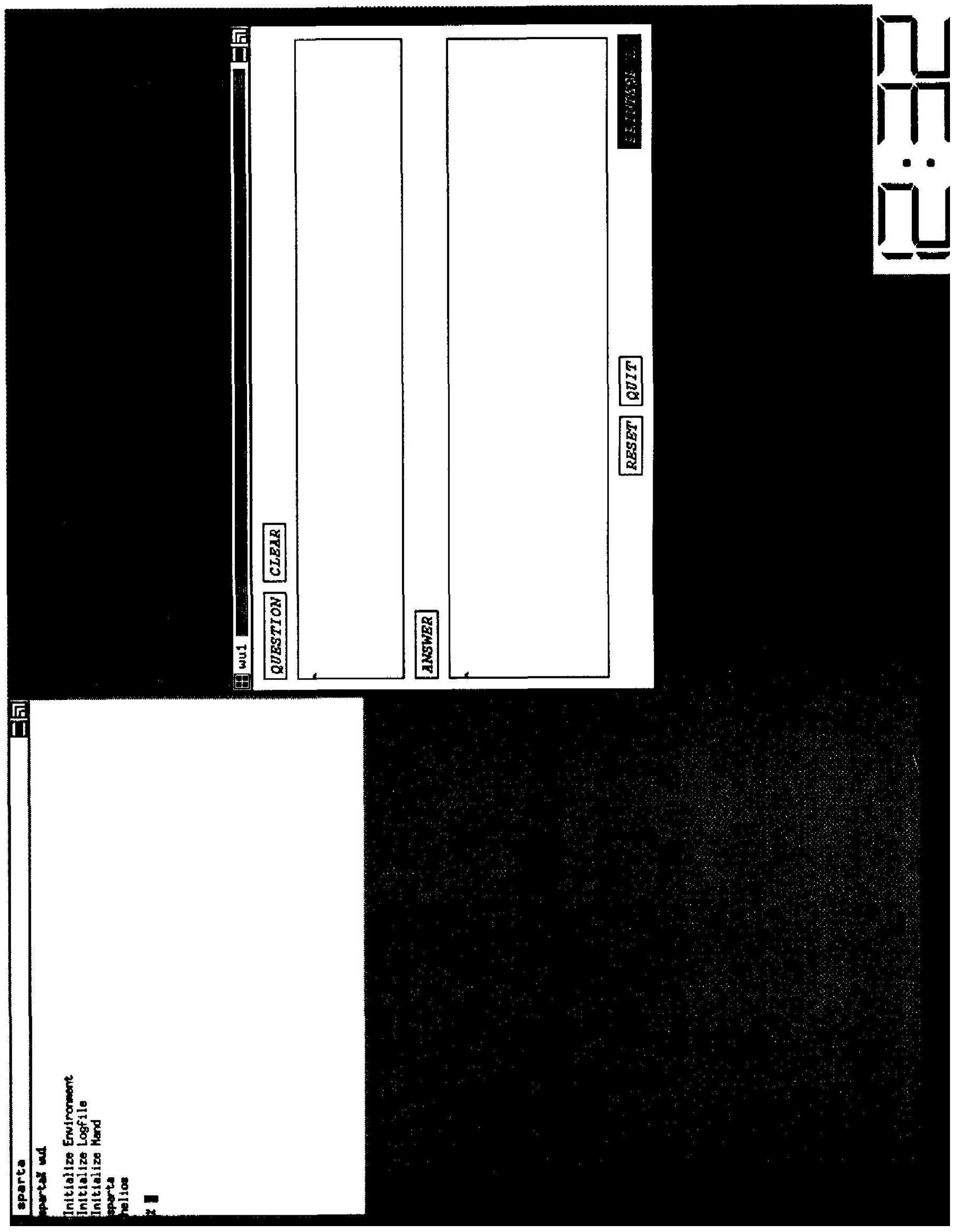
D.3 Wizard's screen layout

The wizard's screen layout was virtually the same as that in Experiment II. In this case however, there was a frame where the Wizard could type a note to the log file. The wizard's screen layout is shown in Figure D.3 below. Also, the wizard had a POPUP window which enabled the selection of canned responses to the subject's questions. The wizard could move the cursor to the *POPUP* button, click on it, and call up the POPUP window. The POPUP window is shown in the bottom left of the screen in Figure D.3 below. The canned responses displayed in the popup menu are shown in Figure D.4.

D.4 Conducting the experiment

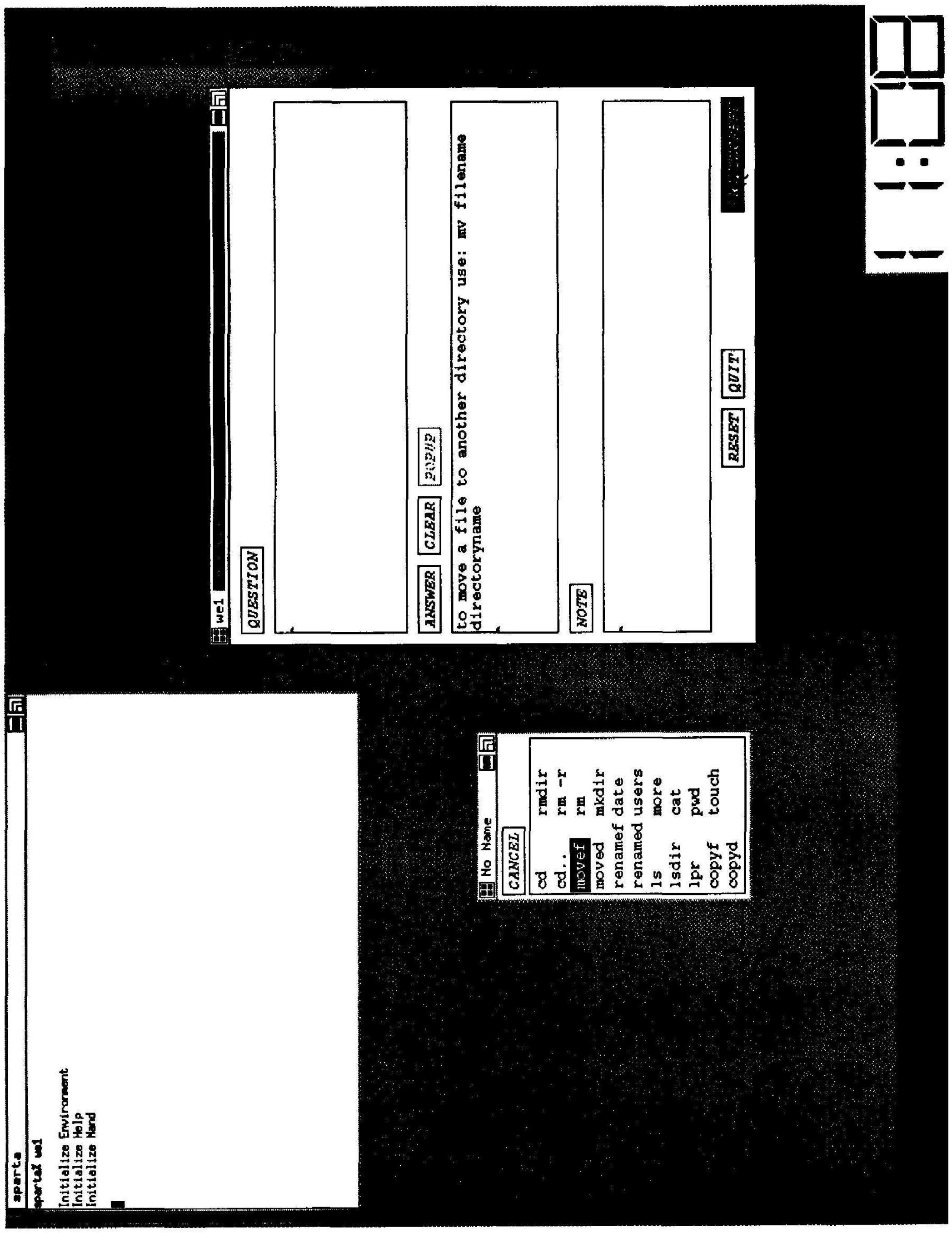
During the experiment subjects had to know about moving the mouse to place the cursor in the correct windows and on the correct buttons. To ask a question the subject had to move the cursor into the QUESTION frame, and to use UNIX, the cursor had to be in the UNIX window. In cases where the subject cleared the QUESTION frame the cursor would automatically reset to the correct position in the QUESTION frame where the subject could type a question.

Each time a subject or wizard sent a message there was a bell beep from the wizard's or subject's console respectively. Also, when the subject typed a command to the UNIX shell there was a bell beep on the wizard's console. Subjects were not given a pen and paper, but it was noted that some of them jotted down information they were given through responses from the computer.



```
sparta% wu1
Initialize Environment
Initialize Logfile
Initialize Hand
sparta
helios
%
```

Figure D.2: *Subject* screen layout



to move a file to another directory use: mv filename directoryname

```
sparta
spartaX we1
Initialize Environment
Initialize Help
Initialize Hand
```

Figure D.3: *Wizard* screen layout

```
cd
to move to a directory use: cd directoryname
cd..
to move to a parent directory use: cd ..
movef
to move a file to another directory use: mv filename directoryname
moved
to move a directory to another directory use: mv directoryname directoryname
renamef
to rename a file use: mv oldfilename newfilename
renamed
to rename a directory use: mv olddirectoryname newdirectoryname
ls
to display a list of files and directories in the current directory use: ls
lsdir
to display a list of files in a directory use: ls directoryname
lpr
to print a file on the printer use: lpr filename
copyf
to make a copy of a file use: cp oldfilename newfilename
copyd
to make a copy of a directory use: cp -r olddirectoryname newdirectoryname
rmdir
to remove a directory use: rmdir directoryname
rm -r
to remove a directory and all of its files and subdirectories use: rm -r
directoryname
rm
to remove a file use: rm filename
mkdir
to create a new directory use: mkdir directoryname
date
to display today's date use: date
users
to display a list of the users logged in use: users
more
to display a file use: more filename
cat
to merge two files and place the result in a third file use: cat file1 file2 > file3
pwd
to display the full path name of the current directory use: pwd
touch
to create a new empty file use: touch filename
```

Figure D.4: *Prestored* responses for popup menu

D.4.1 Tasks

Subjects had to map a source directory structure into a target directory structure. Also, there was a file called *ToDo* containing a list of three other tasks to be completed.

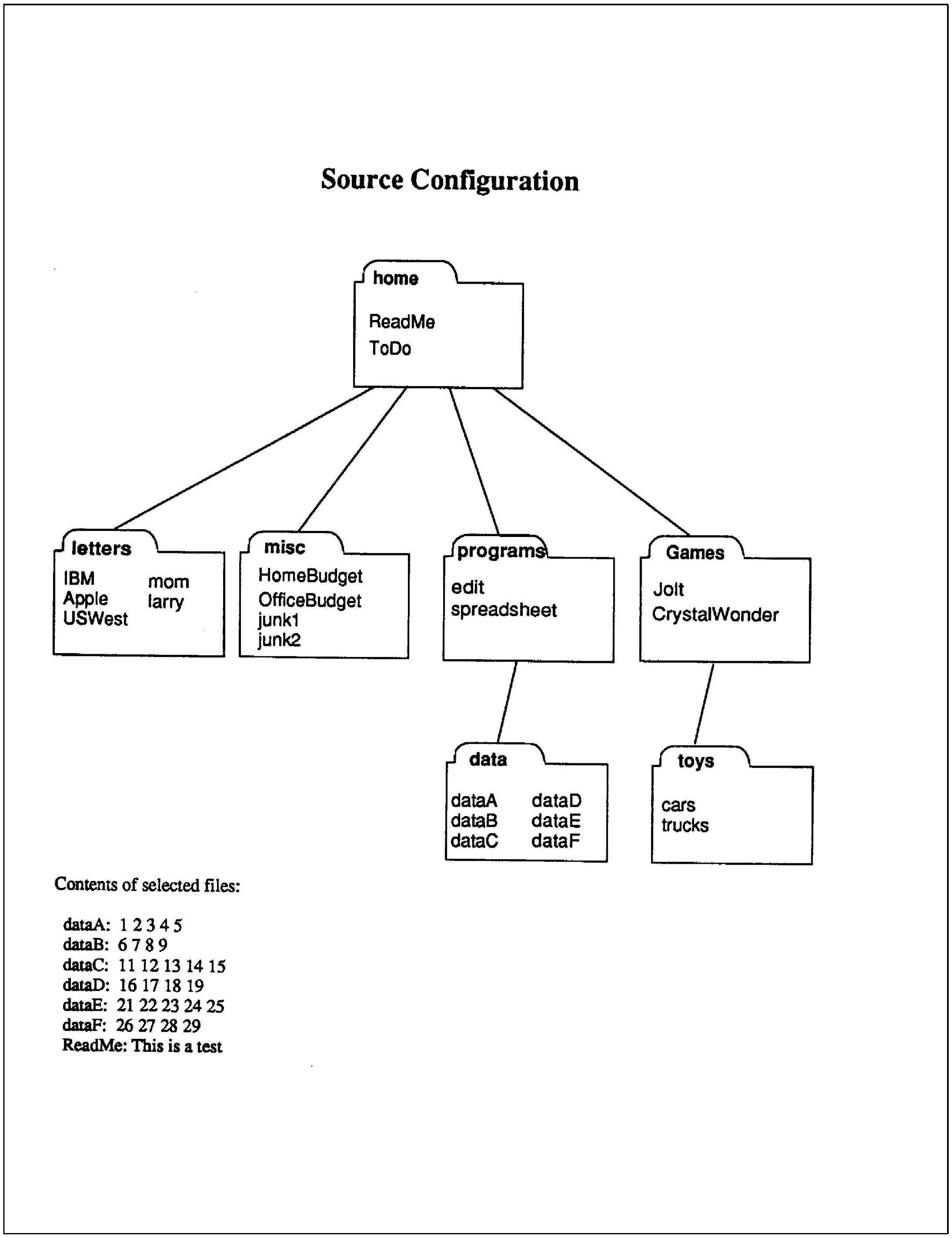
The tasks that subjects had to execute were simple operating system tasks. Some of the tasks were specific to UNIX, others possible in any operating system. The tasks in Experiment III, unlike Experiment II, were presented in picture form rather than on the screen. Subjects were given copies of the source and target directory structure for the tasks. The tasks involved simple operating systems basics such as displaying, printing, moving, and removing files and directories. Some tasks involved displaying information about the system too. The tasks involved mapping a source directory structure shown in Figure D.5 into a target directory structure shown in Figure D.6. Each subject was given a copy of Figure D.5 and Figure D.6.

D.4.2 Logging the interaction

During each subject session data was collected and placed in a log file. The format was similar as for experiment II, except that *E:* was also used to mark user errors in the UNIX shell. Sample log data is shown in Figure D.7.

D.4.3 Questionnaire

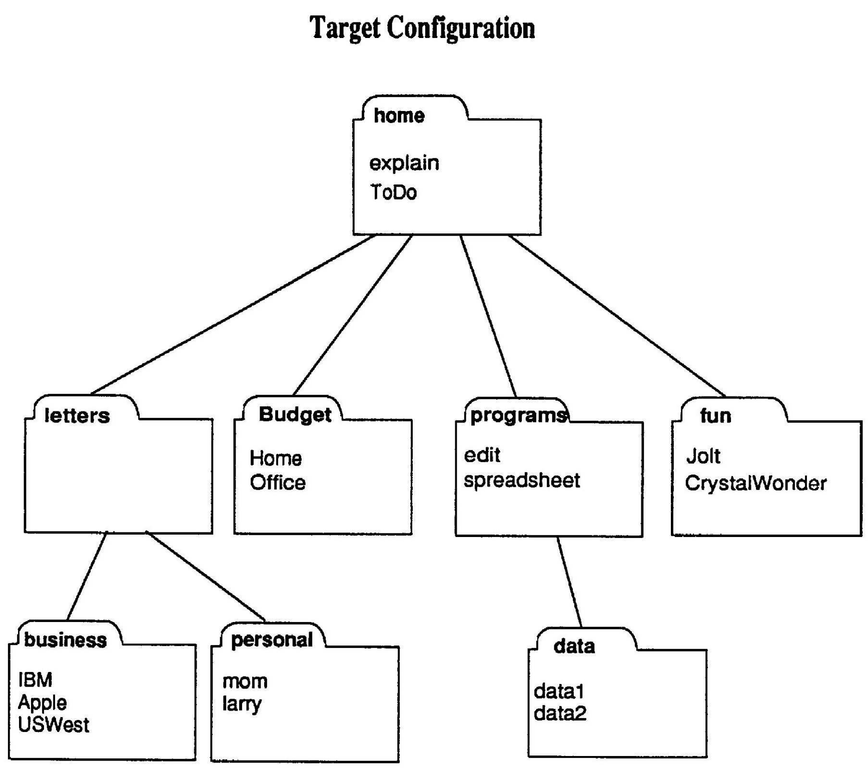
Each subject was asked to fill out a questionnaire which recorded data about each subject. The questionnaire was the same as that used in experiment II.



Contents of selected files:

dataA: 1 2 3 4 5
dataB: 6 7 8 9
dataC: 11 12 13 14 15
dataD: 16 17 18 19
dataE: 21 22 23 24 25
dataF: 26 27 28 29
ReadMe: This is a test

Figure D.5: *Source* directory structure



Contents of selected files:

data1:
```
1 2 3 4 5
6 7 8 9
```

data2:
```
11 12 13 14 15
 16 17 18 19
```

explain:
```
This is a test
```

Figure D.6: *Target* directory structure

```
D: Thu Mar 15 18:50:01 MST 1990

C: 55:45 dir

U: 56:32 how do i copy files

W: 56:42 to make a copy of a file use: cp oldfilename newfilename


U: 57:21 how do i move files

W: 57:31 to move a file to another directory use: mv filename directoryname


U: 58:41 how do i make a new directory

W: 58:51 to create a new directory use: mkdir directoryname


C: 59:14 cd letters

C: 59:19 pwd

C: 59:21

C: 59:38 mkdir business

C: 59:55 mkdir personel

C: 00:01 dir

U: 00:49 how do i remove a directory

W: 01:03 to remove a directory use: rmdir directoryname


C: 02:06 mv ibm business

E: 02:07 mv: ibm: Cannot access: No such file or directory

U: 03:37 in moving a file how is the destination directory specified

W: 03:50 to move a file to another directory use: mv filename directoryname


N: 04:08 * he has the upper case problem.

C: 04:18

C: 04:22 pwd

N: 04:49 * he knows the command pwd.

C: 05:28 mv /sparta/wizard/home/letters/ibm business
```

Figure D.7: Sample log data

References

Allen, James F. (1979) *A plan-based approach to speech act recognition.* Doctoral Dissertation. Also as, Technical Report No. 131/79, University of Toronto, Toronto, Canada, February.

Allen, James F. (1981a) *Maintaining knowledge about temporal intervals.* TR-86, Computer Science Dept., University of Rochester, January. Also in *Communications of the ACM*, 26, 832-843, 1983.

Allen, James F. (1981b) An interval-based representation of temporal knowledge. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, 221-226, Vancouver, British Columbia, Canada.

Allen, James F. (1983) Recognising intentions from natural language utterances. In *Computational Models of Discourse*, M. Brady and R.C. Berwick (Eds.), 107-166. Cambridge, MA: MIT Press.

Allen, James F. (1984) Towards a general theory of action and time. In *Artificial Intelligence*, 23, 123-154.

Allen, James F. (1987) *Natural language understanding.* Benjamin/Cummings Series in Computer Science. Menlo Park, California: Benjamin/Cummings.

Allen, James F. and C. Raymond Perrault (1980) Analyzing intentions in utterances. In *Artificial Intelligence*, 15, 143-178.

Alshawi, Hiyan (1987) *Memory and context for language interpretation.* Cambridge, United Kingdom: Cambridge University Press.

Alterman, R. (1985) A dictionary based on concept coherence. In *Artificial Intelligence*, 25, 153-186.

Andersen, S. and B.M. Slator (1990) Requiem for a theory: the 'story grammar' story. In *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 2,3, 253-275, July-September.

Anderson, S. (1971) *On the linguistic status of the performative/constative distinction.* Indiana University Linguistics Club, Bloomington, IN, USA.

Antaki, Charles (1988) *Analysing everyday explanation: A case book of methods.* London: Sage Publications.

Appelt, Douglas E. (1981) *Planning natural-language utterances to satisfy multiple goals.* Technical Note 259, SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025, USA.

Appelt, Douglas E. (1985) *Planning English sentences.* Cambridge, United Kingdom: Cambridge University Press.

Appelt, Douglas and Amichai Kronfeld (1987) A computational model of referring. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, Vol.2, 640-647, Milan, Italy, European Community (EC).

Arens, Yigal (1986) *CLUSTER: An approach to contextual language understanding.* Doctoral Dissertation, Department of Computer Science, University of California at Berkeley. Also as, Report No. UCB/CSD 86/293, Computer Science Division (EECS), University of California, Berkeley, California, USA, April.

Austin, J.L. (1962) *How to do things with words (Second Edition).* J.O. Urmson and Marina Sbisà (Eds.). Cambridge, MA: Harvard University Press.

Bach, Kent and Robert M. Harnish (1979) *Linguistic communication and speech acts.* Cambridge, MA: The MIT Press.

Bahl, Lalit R., Frederick Jelinek, and Robert L. Mercer (1983) A maximum likelihood approach to continuous speech recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAM1-5, No. 2, March.

Bahl, Lalit R., Peter F. Brown, Peter V. De Souza, and Robert L. Mercer (1989) A tree-based statistical language model for natural language speech recognition. In *IEEE Transactions on*

Acoustics, Speech, and Signal Processing, Vol. 37, No. 7, July.

Baker, Janet M. and James K. Baker (1989) Dragon Systems Inc. speech system. In *DARPA Speech and Natural Language Workshop*, Pittsburgh, Pennsylvania, USA, February.

Ballim, Afzal and Yorick Wilks (1990) Stereotypical belief and dynamic agent modeling. In *Proceedings of the Second International Conference on User Modelling*, University of Hawaii at Manoa, Honolulu, Hawaii, USA.

Ballim, Afzal and Yorick Wilks (1991) *Artificial Believers.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Bar-Hillel, Y. (1954) Indexical expressions. In *Mind*, 63, 359-379.

Barnden, John A. (1984) On short-term information processing in connectionist theories. In *Cognition and Brain Theory*, 7 (1), 25-59.

Barnden, John A. (1990) *Naive metaphysics: a metaphor-based approach to propositional attitude representation (unabridged version).* Memorandum in Computer and Cognitive Science, MCCS-90-174, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA.

Barnden, John A. and Jordan B. Pollack (1991) (Eds.) *Advances in connectionist and neural computation theory, Vol. I: high level connectionist models.* Norwood, N.J.: Ablex Publishing Corporation.

Barnden, John A., Sylvia Candelaria de Ram, David Farwell, Louise Guthrie, Stephen Helmreich, Paul Mc Kevitt, and Yorick Wilks (1991) The relevance of beliefs to natural language communication. In *Intercultural Communication Studies*, 1(1), 237-272. Also as, "The need for belief modelling in natural language processing," In *Proceedings of the International Conference on Cross-Cultural Communication (ICC-CC-89)*, Trinity University, San Antonio, Texas, USA, March, 1989.

Barwise, J. and J. Perry (1983) *Situations and attitudes.* Cambridge, M.A.: Bradford Books, MIT Press.

Bates, M. and R.J. Bobrow (1983) A transportable natural language interface for information retrieval. In *Proceedings of the Sixth International ACM SIGIR Conference*, .

Bates, Madeline, M. G. Moser and David Stallard (1986) The IRUS transportable natural language database interface. In *Expert Database Systems: Proceedings from the First International Workshop*, 617-630, Kiawah Island, South Carolina, USA, October. Also in, *Expert Database Systems*, L. Kerschberg (Ed.). Menlo Park, CA: Benjamin/Cummings.

Benyon, David (1984) MONITOR – a self-adaptive user interface. In *Proceedings of Interact '84, First IFIP Conference on Human-Computer Interaction, B. Shackel (Ed.)*, Amsterdam: Elsevier/North-Holland.

Benyon, David (1987) User models –what's the purpose?. In *Proceedings of the Second Intelligent Interface Meeting*, M. Cooper and D. Dodson (Eds.). Alvey Knowledge-Based Systems Club, Intelligent Interfaces Special Interest Group, 28-29 May, London. London: The Alvey Directorate.

Benyon, David and Dianne Murray (1988) Experience with adaptive interfaces. In *The Computer Journal*, Vol. 31, No. 5, 465-474.

Benyon, David, Steve Milan, and Dianne Murray (1988) *Modelling users' cognitive abilities in an adaptive system.* NPL Report DITC 115/88, March.

Billmers, Meyer A. and Michael G. Carifio (1985) Building knowledge-based operating system consultants. In *Proceedings of the Second Conference on Artificial Intelligence Applications*, Miami Beach, USA, December, 449-454.

Bobrow, D., R. Kaplan, M. Kay, D. Norman, H. Thompson, and T. Winograd (1977) Gus, a frame driven dialog system. In *Artificial Intelligence*, 8, 155-173, April.

Bobrow, D. and Winograd, T. (1977) An overview of KRL, a knowledge and representation language. In *Cognitive Science*, 1, 3-46.

Brady, M. and R.D. Berwick (Eds.) (1983) *Computational models of discourse.* Cambridge, MA: MIT Press.

Bransford, J. and M.K. Johnson (1973) Consideration of some problems in comprehension. In *Visual Information Processing*, W.G. Chese (Ed.). New York: Academic Press.

Brennan, S.E., M.W. Friedman, and C. Pollard (1987) A centering approach to pronouns. In *Proceedings of the 25th Annual Meeting of the Association of Computational Linguistics*, .

Brown, John Seely, Richard R. Burton and Alan G. Bell (1975) SOPHIE: a step towards creating a reactive learning environment. In *International Journal of Man-Machine Studies*, 7, 675-696.

Bruce, B.C. (1972) A model for temporal references and its application in a question-answering program. In *Artificial Intelligence*, 3, 1-25.

Bruning, James L. and Kintz, B.L. (1977) *Computational handbook of statistics (Second Edition).* Glenview, Illinois: Scott, Foresman and Company.

Brunner, Hans, Kathleen Ferrara, and Greg Whittemore (1989a) *On the implications of frequency and distribution for anaphora resolution in written/interactive dialogue.* MCC Technical Report Number ACT-HI-243-89, (MCC Nonconfidential), Microelectronics and Computer Technology Corporation (MCC), Advanced Computing Technology, Human Interface Laboratory, 3500 West Balcones Center Drive, Austin, Texas, USA.

Brunner, Hans, Kathleen Ferrara, and Greg Whittemore (1989b) *An assessment of written/interactive dialogue for information retrieval applications.* MCC Technical Report Number ACT-HI-245-89, (MCC Nonconfidential), Microelectronics and Computer Technology Corporation (MCC), Advanced Computing Technology, Human Interface Laboratory, 3500 West Balcones Center Drive, Austin, Texas, USA.

Bundy, Alan (1990) What kind of field is AI?. In *The foundations of Artificial Intelligence: a sourcebook*, Derek Partridge and Yorick Wilks (Eds.), 215-222. Cambridge, United Kingdom: Cambridge University Press.

Bundy, Alan and Stellan Ohlsson (1990) The nature of AI principles: a debate in the AISB Quarterly. In *The foundations of Artificial Intelligence: a sourcebook*, Derek Partridge and Yorick Wilks (Eds.), 135-154. Cambridge, United Kingdom: Cambridge University Press.

Burton, R. (1976) *Semantic grammar: An engineering technique for constructing natural-language understanding systems.* BBN Report No. 3453, Bolt, Beranek and Newman, Cambridge, MA, USA.

Carberry, Sandra (1985) A pragmatics based approach to understanding intersentential ellipsis. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, 188-197, Chicago, Illinois, USA.

Carberry, Sandra (1989) A pragmatics-based approach to ellipsis resolution. In *Computational Linguistics*, Vol. 15(2), June, 75-96.

Cashman, Paul M. and Anatol W. Holt (1980) A communication oriented approach to structuring of software maintenance environment. In *ACM SIGSOFT Software Engineering*, Notes 5:1, 4-17, January.

Charniak, Eugene (1977) A framed painting: the representation of a commonsense knowledge fragment. In *Cognitive Science*, 1(4).

Charniak, Eugene (1981) *Passing markers: a theory of contextual influence in language comprehension.* Technical Report CS-80, Department of Computer Science, Brown University, Providence, RI, USA, 1981. Also, in *Cognitive Science*, 7(3), 171-190, July-September, 1983.

Carbonell, Jaime G. (1983) Discourse pragmatics and ellipsis resolution in task-oriented natural language interfaces. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, Boston, MA, USA.

Chin, David (1984) An analysis of scripts generated in writing between users and computer consultants. In *AFIPS Proceedings of the National Computer Conference 53*, Las Vagas, NV, USA, July.

Chin, David (1988) Exploiting user expertise in answer expression. In *Proceedings of the Seventh National American Conference on Artificial Intelligence (AAAI-88)*, Vol. 2, 756-760, St. Paul, Minnesota, USA, August.

Chomsky, Noam (1965) *Aspects of the theory of syntax.* Cambridge, MA: MIT Press.

Churchland, Paul M. and Patricia Smith Churchland (1990) Could a machine think?. In *Scientific American*, 32-37, January.

Clark, H.H. (1979) Responding to indirect speech acts. In *Cognitive Psychology*, 11, 430-477.

Clark, Herbert and Eve Clark (1977) *Psychology and language acquisition: an introduction to psycholinguistics.* New York: Harcourt Brace Jovanovich.

Cohen, Phil R. and C. Raymond Perrault (1979) Elements of a plan-based theory of speech acts. In *Cognitive Science*, 3: 177-212.

Cohen, Phil R., C. Raymond Perrault and James F. Allen (1982) Beyond question answering. In *Strategies for natural language processing*, Wendy G. Lehnert and Martin H. Ringle (Eds.), 245-274. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Cohen, Phil R. and Hector Levesque (1985) Speech acts and rationality. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, Also in, *Intentions in Communication*, Philip R. Cohen, J.L. Morgan and M.E. Pollack (Eds.), Cambridge, MA: MIT Press, 1990.

Cohen, Phil R. and Hector Levesque (1987) Rational interaction as the basis for communication. In *Symposium on Intentions and Plans in Communication and Discourse*, .

Cohen, Phil R., J.L. Morgan, and M.E. Pollack (Eds.) (1990) *Intentions in Communication.* Cambridge, MA: MIT Press.

Cohen, R. (1984) A computational theory of the function of clue words in argument understanding. In *Proceedings of COLING-84*, Stanford, CA, USA.

Cohen, R. (1987) Analysing the structure of argumentative discourse. In *Computational Linguistics*, 13 (1-2).

Collins, A.M. and Quillian, M.R. (1972) How to make a language user. In *Organization of Memory*, E. Tulving and W. Donaldson (Eds.). New York: Academic Press.

Coombs, Mike J. and Jim L. Alty (1981) Communicating with university computer users: a case study. In *Computing Skills and the User Interface*, M.J. Coombs and J.L. Alty (Eds.), 3-71, Computers and People Series, B. R. Gaines (Ed.). London: Academic Press.

Cullingford, R.E. (1977) *Script application: computer understanding of newspaper stories.* Doctoral Dissertation. Also as, Research Report 116, Department of Computer Science, Yale University, New Haven, CT, USA, January.

Cullingford, R.E. (1981) Sam. In *Inside Computer Understanding: Five Programs Plus Miniatures*, R. Schank and C. Riesbeck (Eds.), 75-119. Hillsdale, New Jersey: Lawrence Erlbaum.

Dale, Robert (1988) The generation of subsequent referring expressions in structured discourses. In *Advances in Natural Language Generation, An interdisciplinary perspective*, Communication in Artificial Intelligence Series, Michale Zock and Sabah, Gérard (Eds.), 58-75. London: Pinter Publishers.

Dale, Robert (1989) Cooking up reference expressions. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics, 68-75*, University of British Columbia, Vancouver, British Columbia, Canada, June.

Damerau, Fred J. (1981) Operating statistics for the transformational question answering system. In *American Journal of Computational Linguistics*, 7: 30-42.

Dennett, Daniel (1981) *Brainstorms.* Harvester Press.

Dietrich, Eric (1990) Programs in the search for intelligent machines: the mistaken foundations of AI. In *The foundations of Artificial Intelligence: a sourcebook*, Derek Partridge and Yorick Wilks (Eds.), 223-233. Cambridge, United Kingdom: Cambridge University Press.

Donnellan, K.S. (1966) Reference and definite descriptions. In *Philosophical Review*, Vol. 75, 281-304. Also in, *Semantics*, D. Steinberg and L. Jacobovits (Eds.), Cambridge, United Kingdom: Cambridge University Press, 1971.

Douglass, Robert J. and Stephen J. Hegner (1982) An expert consultant for the UNIX operating system: Bridging the gap between the user and command language semantics. In *Proceedings of the Fourth National Conference of the Canadian Society for Computational Studies of Intelligence (CSCSI)/SCIEO Conference*, 119-127, Saskatoon, Saskatchewan, Canada, May.

Ernst, G. and A. Newell (1969) *GPS: A case study in generality and problem solving.* New York: Academic Press.

Farwell, David and Yorick Wilks (1990a) The pragmatics of ellipsis. In *Abstracts of The 1990 International Pragmatics Conference*, E- Barcelona, Spain, European Community (EC), July.

Farwell, David and Yorick Wilks (1990b) *A white paper on research in pragmatics-based machine translation.* Memorandum in Computer and Cognitive Science, MCCS-90-188, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA.

Fass, Dan (1988) *Collative semantics: a semantics for natural-language processing.* Ph.D. Dissertation, Department of Computer Science, University of Essex. Also, as Memorandum in Computer and Cognitive Science, MCCS-88-118, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA.

Fass, Dan, James Martin and Elizabeth Hinkelman (1991) (Eds.) Proceedings of the IJCAI-91 workshop on computational approaches to non-literal language: metaphor, metonymy, idiom, speech acts, and implicature. In *Workshop at The Twelvth International Joint Conference on Artificial Intelligence*, Sydney, Australia, August. Also in, *Computational Intelligence*, Dan Fass, James Martin and Elizabeth Hinkelman (Eds.), 1992, (Forthcoming).

Fauconnier, Giles (1985) *Mental Spaces.* Cambridge, MA: MIT Press.

Ferguson, C.A. (1982) Simplified registers and linguistic theory. In *Exceptional Language and Linguistics*, Obler, L.K. and L. Menn (Eds.), 49-66. New York: Academic Press.

Fikes, Richard and Nils Nilsson (1971) STRIPS: A new approach to the application of theorem proving to problem solving. In *Artificial Intelligence*, 2:89-105.

Fillmore, C.J. (1975) *Santa Cruz lectures on deixis, 1971.* Technical Report/Mimeo, Indiana University Linguistics Club, Bloomington, Indiana, USA.

Findlay, J.N. (1941) Time: a treatment of some puzzles. In *Australasian Journal of Philosophy*, 19, 216-235.

Finin, Timothy W. (1983) Providing help and advice in task oriented systems. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, 176-178, Karlsruhe, Germany, European Community (EC).

Flores, Fernando (1981) Doing and speaking in the office. In *DSS: Issues and Challenges*, G. Fick and R. Sprague (Eds.). London: Pergamon Press.

Flores, Fernando (1982) *Management and communication in the office of the future.* Report printed by Hermenet Inc., San Francisco.

Fodor, J.A. and Z.W. Pylyshyn (1988) Connectionism and cognitive architecture: a critical analysis. In *Connections and symbols*, S. Pinker and J. Mehler (Eds.). Cambridge, MA: MIT Press, and Amsterdam: Elsevier/North-Holland. Also in, *Cognition*, 28, 3-71, 1988.

Fraser, B. (1971) *An examination of the performative analysis.* Indiana University Linguitstics Club, Bloomington, Indiana, USA. Also, expanded in, *Papers in Linguistics*, 7, 1-40, 1974.

Galton, Antony (1984) *The logic of aspect: an axiomatic approach.* Oxford: Claredon Press.

Garside, R.G., G.N. Leech, and G.R. Sampson (Eds.) (1987) *The computational analysis of English.* London: Longman.

Gazdar, Gerald (1979) *Pragmatics, implicature, presupposition, and logical form.* New York: Academic Press.

Gazdar, Gerald and Chris Mellish (1989) *Natural language processing in PROLOG: an introduction to computational linguistics.* Wokingham, United Kingdom: Addison-Wesley.

Godden, Kurt (1989) Computing pronoun antecedents in an English query system. In *Proceedings of the Eleventh International Conference on Artificial Intelligence (IJCAI-89)*, Vol. 2, 1498-1503, Detroit, Michigan, USA, August.

Green, Georgia (1989) *Pragmatics and natural language understanding.* Hillsdale, N.J.: Lawrence Erlbaum Associates.

Green, Bert. F., Alice K. Wolf, Carol Chomsky, and Kenneth Laughery (1963) BASEBALL: an automatic question answerer. In *Computers and Thought*, Edward Feigenbaum and J. Feldman (Eds.), 207-216. McGraw-Hill Books Co. Inc..

Grice, H.P. (1957) Meaning. In *Philosophical Review*, LXVI, No. 3, 377-388, October. Also in, *Semantics*, D. Steinberg and L. Jakobvits (Eds.), New York: Cambridge University Press, 1971.

Grice, H.P. (1969) Utterer's meaning and intentions. In *Philosophical Review*, LXXVIII, No. 2, 147-177, April.

Grice, H.P. (1971) Utterer's meaning, sentence-meaning, and word-meaning. In *Oxford readings in philosophy*, J.R. Searle (Ed.), 54-70. Oxford, United Kingdom: Oxford University Press.

Grice, H.P. (1975) Logic and conversation. In *Syntax and Semantics, Vol. 3, Speech Acts*, P. Cole and J.L. Morgan (Eds.), 41-58. New York: Academic Press.

Grishman, Ralph (1986) *Computational linguistics: an introduction.* Studies in Natural Language Processing. Cambridge University Press: Cambridge.

Grosz, B.J. (1977a) *The representation and use of focus in dialogue understanding.* Doctoral Dissertation, University of California at Berkeley, Berkeley, CA, June. Also as, Technical Note No. 151, Artificial Intelligence Center, SRI International, Menlo Park, California, USA, July.

Grosz, B.J. (1977b) The representation and use of focus in a system for understanding dialogs. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, 67-76, Cambridge, MA, August. Also as, Technical Note 150, Artificial Intelligence Center, SRI International, June 1977, and in *Readings in Natural Language Processing*, Barbara Grosz, Karen Sparck Jones, and Bonnie Webber (Eds.), 353-362, 1986.

Grosz, Barbara J. (1978a) Focusing in dialog. In *Proceedings of the Second Workshop on Theoretical Issues in Natural Language Processing (TINLAP-2)*, 96-103, University of Illinois, Urbana-Champaign, July. Also in, *Proceedings of the 1978 Meeting of the Association for Computational Linguistics, American Journal of Computational Linguistics*, 3, 1978.

Grosz, Barbara J. (1978b) Understanding spoken language. In *Discourse Analysis*, Walker, D. (Ed.), Chapter IX, 235-268. New York: Elsevier/North-Holland.

Grosz, Barbara J. (1981) Focusing and description in natural language dialogues. In *Elements of discourse understanding*, A. Joshi, B. Webber and I. Sag (Eds.), 84-105. Cambridge: Cambridge University Press. Also as, Technical Note 185, Artificial Intelligence Center, SRI International, Menlo Park, California, USA, July 1978.

Grosz, Barbara J. (1983) TEAM, a transportable natural language interface system. In *Proceedings of the 1983 Conference on Applied Natural Language Processing*, 39-45, Santa Monica, California, USA, February.

Grosz, Barbara, Aravind Joshi and S. Weinstein (1983) Providing a unified account of definite noun phrases in discourse. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, 44-50, Cambridge, Massachusetts, USA. Also in, *Proceedings of the International Joint Conference on Artificial Intelligence*, Cambridge, MA.

Grosz, B.J. and C.L. Sidner (1986) Attention, intentions and the structure of discourse. In *Computational Linguistics*, Vol. 12, 3, 175-204.

Grosz, Barbara J., K.S. Jones and Bonnie Lynn Webber (1986) *Readings in natural language processing.* Los Altos, California: Morgan Kaufmann.

Grosz, Barbara and Candy Sidner (1990) Plans for discourse. In *Intentions in Communication*, Cohen, P.R., J.L. Morgan, and M.E. Pollack (Eds.). Cambridge, MA: Bradford Books, MIT Press.

Grosz, Barbara and Candy Sidner (1990) Plans to support communication and collaboration among agents. In *Proceedings of the 1990 AAAI Spring Symposia*, March.

Grosz, Barbara, Martha E. Pollack and Candace Sidner (1989) Discourse. In *Foundations of Cognitive Science*, Michael Posner (Ed.), Domains, Chapter 11, 435-468. Cambridge, MA: Bradford Books, MIT Press.

Guenthner, F., H. Lehmann, and W. Schoenfield (1986) A theory for the representation of knowledge. In *IBM Journal of Research and Development*, 30(1): 39-56.

Guindon, Raymonde (1988) A multidisciplinary perspective in dialogue structure in user-advisor dialogues. In *Cognitive Science and its Applications for Human-Computer Interaction*, R. Guindon (Ed.). Hillsdale, N.J.: Lawrence Erlbaum and Associates.

Guindon, Raymonde, Paul Sladky, Hans Brunner and Joyce Conner (1986) The structure of user-advisor dialogues: is there method in their madness?. In *Proceedings of the 24th Annual Conference of the Association for Computational Linguistics*, 224-230, Columbia University, New York, USA.

Guindon, Raymonde, Joyce Connor, Kelly Schulberg (1989) *Why a restricted natural language is sufficient for advisory systems: effects of real-time constraints and limited shared context.* MCC Technical Report Number STP/ACT-HI-195-89, (MCC Nonconfidential), Microelectronics and Computer Technology Corporation (MCC), Advanced Computing Technology, Human Interface Laboratory, 3500 West Balcones Center Drive, Austin, Texas, USA.

Guthrie, Louise, Paul Mc Kevitt and Yorick Wilks (1989) OSCON: An operating system consultant. In *Proceedings of the Fourth Annual Rocky Mountain Conference on Artificial Intelligence (RMCAI-89), Subtitled, 'Augmenting Human Intellect By Computer'*, 103-113, Registry Hotel, Denver, Colorado, USA, June.

Guthrie, Louise, Paul Mc Kevitt, and Yorick Wilks (1990) Using words. In *Proceedings of the Fifth Annual Rocky Mountain Conference on Artificial Intelligence (RMCAI-90), Subtitled 'Pragmatics in Artificial Intelligence'*, Yorick Wilks and Paul Mc Kevitt (Eds.), 263-271. Computing Research Laboratory, New Mexico State University, Las Cruces, New Mexico, USA, June.

Halliday, M.A.K. and R. Hasan (1976) *Cohesion in English.* London: Longman.

Harary, Frank (1969) *Graph theory.* Reading, MA: Addison-Wesley.

Hauptmann, A. G., and B. F. Green (1983) A comparison of command, menu-selection, and natural language programs. In *Behavior and Information Technology*, 2(2), 163-178.

Hayes, Philip J. (1982a) Uniform help facilities for a cooperative user interface. In *Proceedings of the National Computer Conference*, 469-474, Houston, Texas, USA.

Hayes, Philip J. (1982b) Cooperative command interaction through the Cousin system. In *Proceedings of the International conference on Man/Machine Systems*, Institute of Science and Technology, University of Manchester, Manchester, United Kingdom, European Community (EC), July.

Hayes, Philip J. and Pedro A. Szekely (1983) *Graceful interaction through the Cousin command interface.* CMU-CS-83-102, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, January.

Hayes, Philip J., Peggy M. Anderson, Scott Safier (1985) Semantic caseframe parsing and syntactic generality. In *Proceedings of the 23rd Conference of the Association for Computational Linguistics*, 153-160, Chicago, Illinois, USA.

Hecking, M. C. Kemke, E. Nessen, D. Dengler, M. Gutmann and G. Hector (1988) *The SINIX consultant – A progress report.* Memo No. 28, Department of Computer Science, University of Saarbrücken, D– 6600 Saarbrücken 11, Germany, European Community (EC), August.

Hegner, Stephen J. (1987) *Representation of command language behavior for an operating system consultation facility.* Technical Report CS/TR87-02, CS/EE Department, University of Vermont, Burlington, Vermont, USA.

Hegner, Stephen J. and Robert J. Douglass (1984) Knowledge base design for an operating system expert consultant. In *Proceedings of the Fifth National Conference of the Canadian Society for Computational Studies of Intelligence (CSCSI)*, 159-161, London, Ontario, Canada, December.

Heim, I. (1982) *The semantics of definite and indefinite noun phrases.* Unpublished Ph.D. dissertation, University of Massachusetts, Amherst, Amherst, Massachusetts, USA.

Hendrix, Gary G. (1980) Future prospects for computational linguistics. In *Proceedings of the 18th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA, June.

Hendrix, G.G., E.D. Sacerdoti, D. Sagalowicz and J. Slocum (1978) Developing a natural language interface to complex data. In *ACM Transactions on Database Systems (TODS)*, Vol. 3, No. 2, 105-147, June.

Heritage, John (1986) *Garfinkel and ethnomethodology.* Cambridge: Polity Press.

Heritage, John (1988) Explanations as accounts: a conversation analytic perspective. In *Analysing everyday explanation: a casebook of methods*, Charles Antaki (Ed.), 127-144. London: Sage Publications.

Hill, William C. (1987) *Two hidden operator studies of advice-giving for a direct representation interface.* MCC Technical Report Number HI-169-87-P, (ACA Confidential & Proprietary), Microelectronics and Computer Technology Corporation (MCC), Advanced Computing Technology, Human Interface Laboratory, 3500 West Balcones Center Drive, Austin, Texas, USA, May.

Hill, William C. and James R. Miller (1987) *Justified advice: a semi-naturalistic study of advisory strategies.* MCC Technical Report Number ACA-HI-297-87, (MCC Non-Confidential), Microelectronics and Computer Technology Corporation (MCC), Advanced Computing Technology, Human Interface Laboratory, 3500 West Balcones Center Drive, Austin, Texas, USA, September.

Hinkelman, Elizabeth and James Allen (1989) Two constraints on speech act ambiguity. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, 212-219, University of British Columbia, Vancouver, British Columbia, Canada, June.

Hintikka, J. (1963) *Knowledge and belief.* Ithaca, NY: Cornell University Press.

Hirst, Graeme (1987) *Semantic interpretation and the resolution of ambiguity.* Studies in Natural Language Processing. Cambridge, United Kingdom: Cambridge University Press.

Hirscheberg, J. and Pierrehumbert, J. (1986) The intonational structuring of discourse. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, 136-144, New York, New York, USA.

Hirscheberg, J., D.J. Litman, and G.L. Ward (1987) Intonation and the intentional structure of discourse. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 636-639, Milan, Italy, European Community (EC).

Hirschman, Lynette (1988) *A meta-rule treatment of English wh-constructions.* Technical Report PRC-LBS-8809, Unisys Corporation, Paoli Research Center, Unisys Defense Systems, P.O. Box 517, Paoli, PA 19301, USA, June.

Hobbs, Jerry (1979) Coherence and coreference. In *Cognitive Science*, 3, 1: 67-90.

Hobbs, Jerry (1982) Towards an understanding of coherence in discourse. In *Strategies for Natural Language Processing*, Wendy G. Lehnert and Martin H. Ringle (Eds.), 223-243. Hillsdale, New Jersey: Lawrence Erlbaum.

Hobbs, Jerry and P. Martin (1987) Local pragmatics. In *Proceedings of the 10th International Conference on Artificial Intelligence*, 520-523, Milan, Italy, European Community (EC), August.

Hofstadter, Douglas (1979) *Gödel, Escher, Bach: an eternal golden braid.* New York: Basic Books.

Holt, Anatol W., H.R. Ramsey, and J.D. Grimes (1983) Coordination system technology as the basis for a programming environment. In *Electrical Communications*, 57:4, 307-314.

Hovy, Eduard (1988) *Generating natural language under pragmatic constraints.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Iida, Masayo; Stephen Wechsler, and Draga Zec (Eds.) (1987) *Working papers in grammatical theory and discourse structure: interactions of morphology syntax, and discourse.* Centre for the Study of Language and Information (CSLI) Lecture Notes, No. 11. Chicago, Illinois: University of Chicago Press.

Jack, M.A. and J. Laver (1988) *Aspects of speech technology.* EDITS, Edinburgh Information Technology Series, 4, Sidney Michaelson and Yorick Wilks (Series Eds.). Edinburgh: Edinburgh University Press

Jarke, Matthias, Jon A. Turner, Edward A. Stohr, Yannis Vassiliou, Norman H. White and Ken Michielsen (1985) A field evaluation of natural language for data retrieval. In *IEEE Transactions on Software Engineering*, SE-11, No. 1, 97-103.

Joyce, James (1939) *Finnegans wake.* London: Faber and Faber.

Jung, John (1968) *Verbal learning.* New York: Holt, Rinehart and Winston.

Kahn, K. and G.A. Gorry (1977) Mechanizing temporal knowledge. In *Artificial Intelligence*, 9, 87-108.

Kamp, H. (1981) A theory of truth and semantic representation. In *Formal methods in the study of language, Part I*, J. Groenendijik, Th. Janssen, and M. Stokoff (Eds.), 277-322. Amsterdam, The Netherlands: MC TRACT 135.

Kantor, R.N. (1977) *The management and comprehension of discourse connection by pronouns in English.* Unpublished Doctoral Dissertation, Ohio State University, Ohio, USA.

Kautz, H.A. (1989) A circumscriptive theory of plan recognition. In *Intentions in communication*, P.R. Cohen, J.L. Morgan and M.E. Pollack (Eds.). Cambridge, MA: Bradford Books, MIT Press, 1990.

Kemke, Christel (1986) *The SINIX Consultant – requirements, design, and implementation of an intelligent help system for a UNIX derivative.* Report No. 11, SC-Project, Department of Computer Science, University of Saarbrücken, Saarbrücken, Germany, European Community (EC), October.

Kemke, Christel (1987) Representation of domain knowledge in an intelligent help system. In *Human-Computer Interaction – INTERACT '87*, H.J. Bullinger and B. Shakel (Eds.), 215-220. Amsterdam: Elsevier Science Publications B.V./North-Holland.

Kobsa, Alfred and Wolfgang Wahlster (1988) *User models in dialog systems.* Berlin: Springer-Verlag.

Kowalski, R.A. (1979) *Logic for problem solving.* New York: Elsevier/North-Holland.

Kowalski, R.A. and Marek Sergot (1986) A logic-based calculus of events. In *New Generation Computing*, 4, 67-95.

Krause, J. (1980) Natural language access to information systems: an evaluation study of its acceptance by end users. In *Information Systems*, 5(4), 297-319.

Kronfeld, Amichai (1986) Donnellan's distinction and a computational model of reference. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, New York, New York, USA.

Kuno, S. (1972) Functional sentence perspective: a case study from Japanese and English. In *Linguistic Enquiry*, 3, 269-320.

Kuno, S. (1975) Three perspectives in the functional approach to syntax. In *Papers from the parasession on functionalism*, R. Grossman, L.J. San, and T.J. Vance (Eds.), 276-336. Chicago: Chicago Linguistic Society.

Lakoff, G. (1968) *Counterparts; or, the problem of reference in transformational grammar.* Paper presented at the July 1968 Linguistic Society of America meeting, Champaign-Urbana, Illinois, USA.

Lakoff, R. (1968) *Abstract syntax and Latin complementation.* Cambridge, MA: MIT Press.

Lakoff, G. and M. Johnson (1980) *Metaphors we live by.* Chicago, Illinois: University of Chicago Press.

Lee, R.M., H. Coelho, and J.C. Cotta (1985) Temporal inferencing on administrative databases. In *Information Systems*, 10, 197-206.

Lehmann, H. (1978) Interpretation of natural language in an information system. In *IBM Journal of Research and Development*, 22(5), 560-572.

Lehnert, Wendy (1978) *The process of question answering.* Hillsdale, N.J.: Lawrence Erlbaum Associates.

Lehnert, Wendy (1982) Plot Units: a narrative summarization strategy. In *Strategies for natural language processing*, W.G. Lehnert and M.H. Ringle (Eds.). Hillsdale, N.J.: Lawrence Erlbaum Associates.

Levinson, Stephen C. (1983) *Pragmatics.* Cambridge textbooks in Linguistics. Cambridge, United Kingdom: Cambridge University Press.

Linde, C. (1979) Focus of attention and the choice of pronouns in discourse. In *Syntax and Semantics, Vol. 12, Discourse and Syntax*, Givon, T., (Ed.). New York, New York: Academic Press.

Linebarger, Marcia C., Deborah A. Dahl, Lynette Hirschman, Rebecca J. Passonneau (1988) *Sentence fragments regular structures.* Technical Report PRC-LBS-8807, Unisys Corporation, Paoli Research Center, P.O. Box 517, Paoli, PA 19301, USA, June.

Linsky, Leonard (1967) Referring. In *International Library of Philosophy and Scientific Method*, Ted Honderich (Ed.), Bernard Williams (Advisory Ed.). London: Routledge and Kegan Paul.

Linsky, Leonard (1971) Reference and modality. In *Oxford Readings in Philosophy*, Susan Warwick (Series Ed.). London: Oxford University Press.

Litman, Diane (1985) *Plan recognition and discourse analysis: an integrated approach for understanding dialogues.* Ph.D. Dissertation and Technical Report TR-170, University of Rochester, Rochester, New York, USA.

Litman, Diane (1986) Understanding plan ellipsis. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, 619-624, Philadelphia, PA, USA.

Litman, Diane and James Allen (1984) A plan recognition model for clarification subdialogues. In *Proceedings of the COLING-84*, 302-311.

Litman, Diane and James Allen (1987) A plan recognition model for subdialogues in conversation. In *Cognitive Science*, 11, 2: 163-200.

Mann, William C. (1975) *Why things are so bad for the computer naive user.* Technical Report RR-75-32, Information Sciences Institute, University of Southern California, California, USA, March.

Mann, William C., James Moore, and James Levin (1977) A comprehension model for human dialog. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 77-87, Cambridge, MA, USA.

Martin, Paul, Douglas Appelt and Fernando Pereira (1983) Transportability and generality in a natural-language interface system. In *Proceedings Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, Alan Bundy (Ed.), 573-581, Karlsruhe, Germany, European Community (EC), August.

Marr, P. (1982) *Vision: A computational investigation into the human representation and processing of visual information.* San Francisco, CA: Freeman. Also as, *Vision*, Cambridge, MA: MIT Press.

Marr, David (1990) *AI: a personal view.* The foundations of Artificial Intelligence: a sourcebook. Derek Partridge and Yorick Wilks (Eds.), 99-107 Cambridge, United Kingdom: Cambridge University Press

Masson, Michael E.J., William C. Hill, Joyce Connor and Raymonde Guindon (1987) *Misconceived misconceptions?.* MCC Technical Report Number ACA-HI-298-87, (MCC Nonconfidential), Microelectronics and Computer Technology Corporation (MCC), Advanced Computing Technology, Human Interface Laboratory, 3500 West Balcones Center Drive, Austin, Texas, USA, September.

Matthews, Manton and Walter Pharr (1987) Knowledge acquisition for active assistance. In *Preprints of the First International Workshop on Knowledge Representation in the UNIX Help Domain*, University of California at Berkeley, Berkeley, California, December..

McCarthy, John and P.J. Hayes (1969) Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, 4, B. Meltzer and D. Michie (Eds.). Edinburgh, United Kingdom: Edinburgh University Press.

McDermott, D. (1982) A temporal logic for reasoning about processes and plans. In *Cognitive Science*, 6, 101-155.

McDonald, David D., J. Pustejovsky, and M.M. Vaughan (1987) Factors contributing to efficiency in natural language generation. In *Natural Language Generation: New Results in Artificial Intelligence, Psychology, and Linguistics*, G. Kempen (Ed.). Dordrecht: Kluer Academic.

McDonald, James E. and Roger W. Schvaneveldt (1987) The application of user knowledge to interface design. In *Cognitive Science and its Applications for Human-Computer Interaction*, R. Guindon (Ed.), 289-338. Hillsdale, N.J.: Lawrence Erlbaum and Associates. Also, as Memorandum in Computer and Cognitive Science, MCCS-87-93, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA.

McKeown, Kathleen (1985) *Text generation: using discourse strategies and focus constraints to generate natural language text.* Studies in Natural Language Processing. Cambridge, United Kingdom: Cambridge University Press.

Mc Kevitt, Paul (1985) *AD-HAC Inference Diagrams.* Bachelor's Thesis, Computer Science Department, University College Dublin, Belfield, IRL- Dublin 4, Dublin, Ireland, European Community (EC).

Mc Kevitt, Paul (1986a) *Object and action frames for Transfer Semantics.* Memorandum in Computer and Cognitive Science, MCCS-86-69, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA.

Mc Kevitt, Paul (1986b) *Selecting and instantiating formal concept frames.* Memorandum in Computer and Cognitive Science, MCCS-86-71, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA.

Mc Kevitt, Paul (1986c) *Building embedded representations of queries about UNIX.* Memorandum in Computer and Cognitive Science, MCCS-86-72, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA.

Mc Kevitt, Paul (1986d) *Formalization in an English interface to a UNIX database.* Memorandum in Computer and Cognitive Science, MCCS-86-73, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA.

Mc Kevitt, Paul (1987) Natural language interfaces in computer aided instruction – What happened before and after the 80's AICAI coup. In *Proceedings of the Fourth International Symposium on Modeling and Simulation Methodology*, University of Arizona, Tucson, Arizona, USA, January.

Mc Kevitt, Paul (1988a) *Representing and extending the Rule of Composition.* Memorandum in Computer and Cognitive Science, MCCS-88-125, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA.

Mc Kevitt, Paul (1988b) *Artificial Communicators: An operating system consultant.* Master's Thesis, Computer Science Department, Dept. 3CU, Box 30001, New Mexico State University, Las Cruces, New Mexico 88003-0001, USA, March.

Mc Kevitt, Paul (1988c) Natural language meaning representations of queries about operating systems. In *Proceedings of the Third Rocky Mountain Conference on Artificial Intelligence (RMCAI-88)*, 233-247, Sheraton Denver Technology Center, Denver, Colorado, USA, June.

Mc Kevitt, Paul (1988d) Rules of inference in an operating system consultant. In *Proceedings of the First Irish National Conference on Artificial Intelligence and Cognitive Science (AI/CS-88)*, Vol. 1, University Industry Center, University College Dublin, IRL- Dublin 4 Dublin, Ireland, European Community (EC), September.

Mc Kevitt, Paul (1990a) Acquiring user models for natural language dialogue systems through Wizard-of-Oz techniques. In *Proceedings of the Second International Workshop on User Modeling, David Chin (Ed.)*, University of Hawaii at Manoa, Honolulu, Hawaii, USA, March.

Mc Kevitt, Paul (1990b) Using speech acts for segmentation in user-advisor dialogue. In *Abstracts of The 1990 International Pragmatics Conference*, University of Barcelona, E- Barcelona, Spain, European Community (EC), July.

Mc Kevitt, Paul (1990c) The role of the user in designing natural language dialogue interfaces. In *Preprints of the Workshop on The Role of the User in Participatory Systems Design at the International Workshop on Human Centered Systems*, Bristol Polytechnic, Brighton, United Kingdom, European Community (EC), September.

Mc Kevitt, Paul (1990d) *Data acquisition for natural language interfaces.* Memorandum in Computer and Cognitive Science, MCCS-90-178, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA.

Mc Kevitt, Paul (1991a) Principles and practice in an operating system consultant. In *Artificial Intelligence and Software Engineering, Vol. 1*, Chapter on 'AI Mechanisms and techniques in practical software', Derek Partridge (Ed.). New York: Ablex Publishing Corporation.

Mc Kevitt, Paul (1991b) The OSCON operating system consultant. In *Intelligent Help Systems for UNIX – Case Studies in Artificial Intelligence*, Springer-Verlag Symbolic Computation Series, Peter Norvig, Wolfgang Wahlster and Robert Wilensky (Eds.). Berlin: Springer-Verlag.

Mc Kevitt, Paul (1992) Legalising reliable computer programs. In *International Journal of Law and Artificial Intelligence*, (Forthcoming). Also in, *Proceedings of the Second National Conference on Law, Computers and Artificial Intelligence, Theme: The Legal Implications of Computer Misuse and Abuse*, University of Exeter, United Kingdom, European Community (EC), November.

Mc Kevitt, Paul and Yorick Wilks (1987) Transfer Semantics in an Operating System Consultant: the formalization of actions involving object transfer. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, Vol. 1, 569-575, Milan, Italy, European Community (EC), August.

Mc Kevitt, Paul and William C. Ogden (1989a) *Wizard-of-Oz dialogues in the computer operating systems domain.* Memorandum in Computer and Cognitive Science, MCCS-89-167, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA.

Mc Kevitt, Paul and William C. Ogden (1989b) *OSWIZ II: Wizard-of-Oz dialogues in the computer operating systems domain.* Memorandum in Computer and Cognitive Science, MCCS-90-181, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA.

Mc Kevitt, Paul and Zhaoxin Pan (1990a) A general effect representation for Operating System Commands. In *AI and Cognitive Science '89*, Springer-Verlag British Computer Society Workshop Series, Alan Smeaton and Gabriel McDermott (Eds.), 68-85. Berlin: Springer-Verlag. Also in, *Proceedings of the Second Irish National Conference on Artificial Intelligence and Cognitive Science (AI/CS-89)*, 50-65, School of Computer Applications, Dublin City University (DCU), IRL- Dublin 4, Dublin, Ireland, European Community (EC), September, 1989.

Mc Kevitt, Paul and Zhaoxin Pan (1990b) *OUI – A user interface for OSCON.* Memorandum in Computer and Cognitive Science, MCCS-90-179, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA.

Mc Kevitt, Paul and D. Partridge (1991) Problem description and hypothesis testing in Artificial Intelligence. In *Artificial Intelligence and Cognitive Science '90*, Springer-Verlag British Computer Society Workshop Series, Michael McTear and Norman Creaney (Eds.). Berlin: Springer-Verlag. (In Press). Also as abstract in, *Abstracts of the Third Irish Conference on Artificial Intelligence and Cognitive Science (AI/CS-90)*, University of Ulster at Jordanstown, GB- Jordanstown, Northern Ireland, European Community (EC), September, 1990.

Mellish, Chris S. (1980) Some problems in early noun-phrase interpretation. In *Proceedings of the AISB Conference. Amsterdam: Elsevier/North-Holland*, .

Mellish, Chris S. (1985) *Computer interpretation of natural-language descriptions.* Ellis Horwood Series in Artificial Intelligence. Chicester: Ellis Horwood/John-Wiley.

McKeown, K.R. (1985) *Text generation: using discourse strategies and focus constraints to generate natural language text.* Cambridge, United Kingdom: Cambridge University Press.

McTear, Mike (1987) *The articulate computer.* Oxford: Basil Blackwell Ltd..

Minsky, Marvin (1975) A framework for representing knowledge. In *The Psychology of Computer Vision*, P.H. Winston (Ed.), 211-217. New York: McGraw-Hill.

Moore, R.C. (1980) *Reasoning about knowledge and action.* Technical Report No. 191, Artificial Intelligence Center, SRI International, Menlo Park, CA, USA.

Morgan, Jerry L. (1978) Two types of convention in indirect speech acts. In *Syntax and Semantics*, Vol. 9, Pragmatics, P. Cole (Ed.), 9, 261-280. New York: Academic Press.

Napier, Albert, David M. Lane, Richard R. Batsell, Norman S. Guadango (1989) Impact of a restricted natural language interface on ease of learning and productivity. In *Communications of the Association of Computing Machinery*, 32, 10, 1190-1198, October.

Narayanan, Ajit (1986) Why AI cannot be wrong. In *Artificial Intelligence for Society*, K.S. Gill (Ed.), 43-53. Chichester, UK: John Wiley and Sons.

Narayanan, Ajit (1988) *On being a machine.* Volume 1, Formal Concepts of Artificial Intelligence, Ellis Horwood Series in Artificial Intelligence Foundations and Concepts. Sussex, United Kingdom: Ellis Horwood Limited.

Narayanan, Ajit (1990a) The intentional stance and the imitation game. In *Proceedings of the Turing '90 Colloquium*, . London: Oxford University Press.

Narayanan, Ajit (1990b) *On being a machine.* Volume 2, Philosophy of Artificial Intelligence, Ellis Horwood Series in Artificial Intelligence Foundations and Concepts. Sussex, United Kingdom: Ellis Horwood Limited.

Neisser, Ulric (1987) *Conepts and conceptual development: ecological and intellectual factors in categorisation.* Expanded versions of talks given at the *First Emory Cognition Project Conference*, October 11-12, 1984. Cambridge, United Kingdom: Cambridge University Press..

Nirenberg, Sergei (Ed.) (1987) *Machine Translation: Theoretical and methodological issues.* Theroetical and Methodological Issues in Natural Language Processing. Cambridge: United Kingdom: Cambridge University Press.

Norvig, Peter, Wolfgang Wahlster and Robert Wilensky (1991) *Intelligent Help Systems for UNIX – Case Studies in Artificial Intelligence.* Springer-Verlag Symbolic Computation Series. Berlin: Springer-Verlag. (In Press).

Partridge, Derek (1986) *Artificial Intelligence: applications in the future of software engineering.* Halsted Press, Chichester: Ellis Horwood Limited.

Partridge, Derek (1991a) *A new guide to Artificial Intelligence.* Norwood, New Jersey: Ablex Publishing Corporation.

Partridge, Derek (1991b) What the computer scientist should know about AI – and vice versa. In *Artificial Intelligence and Cognitive Science '90*, Springer-Verlag British Computer Society Workshop Series, Michael McTear and Norman Creaney (Eds.). Berlin: Springer-Verlag. Also as an abstract in, *Abstracts of the Third Irish Conference on Artificial Intelligence and Cognitive Science*, University of Ulster, Jordanstown, Northern Ireland, September, 1990.

Partridge, Derek and Yorick Wilks (1990a) Does AI have a methodology different from software engineering?. In *The Foundations of Artificial Intelligence: a sourcebook*, Partridge, Derek and Yorick Wilks (Eds.), 363-372. Cambridge, United Kingdom: Cambridge University Press. Also as, "Does AI have a methodology which is different from software engineering?," in *Artificial Intelligence Review*, 1, 111-120, 1990.

Partridge, Derek and Yorick Wilks (1990b) *The foundations of Artificial Intelligence: a sourcebook.* Cambridge, United Kingdom: Cambridge University Press.

Pereira, Fernando and David Warren (1980) Definite clause grammars for language analysis – a survey of the formalism and a comparison with augmented transition networks. In *Artificial Intelligence*, 13, 231-278.

Perrault, Raymond (1990) An application of default logic to speech act theory. In *Intentions in Communication*, P.R. Cohen, J.L. Morgan, and M.E. Pollack (Eds.). Cambridge, MA: MIT Press.

Perrault, Raymond (1978) Speech acts as a basis for understanding dialogue coherence. In *Proceedings of Second Conference on Theoretical Issues in Natural Language Processing (TINLAP-2)*, .

Perrault, Raymond C. and James F. Allen (1980) A plan-based analysis of indirect speech acts. In *American Journal of Computational Linguistics*, Vol. 6, No. 3-4, 167-182, July-December.

Pollack, Martha E. (1986) *Inferring domain plans in question-answering.* Unpublished Doctoral Dissertation, University of Pennsylvania, Philadelphia, Pennsylvania, USA.

Pollack, Martha E. (1990) Plans as complex mental attitudes. In *Intentions in Communication*, P.R. Cohen, J.L. Morgan, and M.E. Pollack (Eds.). Cambridge, MA: MIT Press.

Pollack, Martha E. and J. Hirschberg and Bonnie Webber (1982) User participation in the reasoning processes of expert systems. In *Proceedings of the Second American National Conference on Artificial Intelligence (AAAI-82)*, 358-361, Pittsburgh, Pennsylvania, USA.

Pollack, Martha E. and Fernando Pereira (1988) An integrated framework for semantic and pragmatic interpretation. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguitstics*, Buffalo, NY, USA.

Postman, L. and Keppel, G. (1970) (Eds) *Norms of word association.* New York: Academic Press.

Prior, A.N. (1955) Diodran modalities. In *Philosophical Quarterly*, 5,205-213.

Pustejovsky, James (1987) An integrated theory of discourse analysis. In *Machine Translation: Theoretical and methodological issues*, Studies in Natural Language Processing, Sergei Nirenberg (Ed.), 168-191. Cambridge, United Kingdom: Cambridge University Press.

Quillian, M. R. (1969) The teachable language comprehender: a simulation program and theory of language. In *Communications of the Association for Computing Machinery*, 459-76, 12, 8, August.

Quirk, R., S. Greenbaum, G. Leech, and J. Svartvik (1985) *A comprehensive grammar of the English language.* Harlow, UK: Longman Group Ltd..

Reichgelt, H. (1989) A comparison of first order and modal logics of time. In *Logic-based Knowledge Representation*, P. Jackson, F. Reichgelt, and F. van Harmelen (Eds.), MIT Press Series in Logic Programming, 7, 143-176. Cambridge, Mass: MIT Press.

Reichman, R. (1978) Conversational coherency. In *Cognitive Science*, 2:283-327.

Reichman-Adar, R. (1984) Extended person-machine interface. In *Artificial Intelligence*, 22(2), 157-218.

Reichman, R. (1985) *Getting computers to talk like you and me.* Cambridge, MA: MIT Press.

Reid, T.B.W. (1956) Linguistics, structuralism and philosophy. In *Archivum Linguisticum*, 8, 28-37.

Reilly, Ronan (Ed.) (1987) *Communication failure in dialogue and discourse: detection and repair processes.* Amsterdam: Elsevier Science Publishers/North-Holland.

Reilly, Ronan; Ferrari, Giacomo and Prodanof Inria (1988) Framework for a model of dialogue. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING-88)*, 540-543, Vol. 2, Budapest, Hungary, August.

Ross, J. R. (1970) On declarative sentences. In *Readings in English transformational grammar*, R. Jacobs and P.S. Rosenbaum (Eds.), 222-272. Waltham, MA: Ginn.

Rowe, Jon and Paul Mc Kevitt (1991) An emergent computation approach to natural language processing. In *Proceedings of the Fourth Irish Conference on Artificial Intelligence and Cognitive Science*, University College Cork, IRL Cork, Ireland, European Community (EC), September.

Rossborough, Michael J. and Jerry T. Ball (1989) *Phase I development of an ULTRA interface.* Memorandum in Computer and Cognitive Science, MCCS-90-176, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA.

Rumelhart, David (1975) Notes on a schema for stories. In *Representation and Understanding*, D. Bobrow and A. Collins (Eds.), 211-236. New York: Academic Press.

Sacerdoti, E. (1977) *A structure for plans and behaviour.* New York: Elsevier/North-Holland.

Sampson, Geoffrey, Robin Haigh and Eric Atwell (1989) Natural language analysis by stochastic optimization: a progress report on project APRIL. In *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, Vol. 1, No. 4, 271-287.

Sarantinos, E. and Peter Johnson (1990) Explanation dialogues: a theory of how experts provide explanations to novices and partial experts. In *Artificial Intelligence*, Also in, *Proceedings of the Fifth Rocky Mountain Conference on Artificial Intelligence (RMCAI-90), Subtitled, 'Pragmatics in Artificial Intelligence,'* Yorick Wilks and Paul Mc Kevitt (Eds.), 307-312, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA, June, 1990.

Schank, Roger C. (1972) Conceptual dependency: a theory of natural language understanding. In *Cognitive Psychology*, 3(4): 552-631.

Schank, Roger C. (1973) Identification and conceptualizations underlying natural language. In *Computer Models of Thought and Language*, R. Schank and K. Kolby (Eds.). San Francisco, CA: Wh Freeman and Co..

Schank, Roger C. (1975) *Conceptual information processing.* Fundamental Studies in Computer Science, 3. Amsterdam: North-Holland.

Schank, Roger C. (1990) What is AI anyway?. In *The foundations of Artificial Intelligence: a sourcebook*, Derek Partridge and Yorick Wilks (Eds.), 1-13. Cambridge, United Kingdom: Cambridge University Press.

Schank, Roger C. and Robert P. Abelson (1977) *Scripts, plans, goals and understanding: an inquiry into human knowledge structures.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, Roger and Chris Riesbeck (Eds.) (1981) *Inside Computer Understanding: Five Programs Plus Miniatures.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Schuster, Ethel; David Chin, Robin Cohen, Alfred Kobsa, Kathrina Morik, Karen Sparck Jones, Wolfgang Wahlster (1988) Discussion section on the relationship between user models and discourse models. In *Computational Linguistics, Special Issue on User Modeling, Alfred Kobsa and Wolfgang Wahlster (Guest Eds.)*, Vol. 14, No. 3, 79-103, September.

Searle, John R. (1969) *Speech acts: An essay in the philosophy of language.* London: Cambridge University Press.

Searle, John R. (1978) *Literal meaning.* Erkenntnis. 13:1, 207-224, July

Searle, John R. (1980) Minds, brains and programs. In *The Behavioral and Brain Sciences*, 3, 417-424. Also in, *Mind Design*, J. Haugeland (Ed.), Cambridge, MA:MIT Press, and in, *The Mind's I: Fantasies and Reflections on Self and Soul*, D.R. Hofstadter and D.C. Dennett (Eds.), Penguin.

Searle, John R. (1987) Minds and brains without programs. In *Mindwaves*, C. Blakemore and S. Greenfield (Eds.). Oxford: Blackwell.

Searle, John R. (1990) Is the brain's mind a computer program?. In *Scientific American*, 26-31, January.

Sharkey, Noel E. and G.D.A. Brown (1986) Why AI needs an empirical foundation. In *AI: Principles and applications*, M. Yazdani (Ed.), 267-293. London, United Kingdom: Chapman-Hall.

Sharkey, Noel E., R.F.E. Sutcliffe, and W.R. Wobcke (1986) Mixing binary and continuous connection scheme: for knowledge access. In *Proceedings of the Fifth American National Conference on Artificial Intelligence (AAAI-86)*, Philadelphia, PA, USA, August.

Sharkey, Noel E. and Amanda J.C. Sharkey (1987) KAN: A knowledge access network model. In *Communication failure in dialogue and discourse: detection and repair processes*, Ronan Reilly (Ed.), 287-307. Amsterdam: Elsevier Science Publishers/North-Holland.

Shneiderman, Ben (1987) *Designing the user interface: Strategies for effective human-computer interaction.* Reading, Massachusetts: Addison Wesley.

Shrager, and Tim Finin (1982) An expert system that volunteers advice. In *Proceedings of the American National Conference on Artificial Intelligence*, 339-340, Pittsburgh, Pennsylvania.

Sidner, Candice L. (1979) *Towards a computational theory of definite anaphora comprehension in English discourse.* Doctoral Dissertation. Also as, Technical Report 537, Artificial Intelligence

Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA.

Sidner, Candace L. (1981) Focusing for interpretation of pronouns. In *Computational Linguistics*, 7(4): 217-231.

Sidner, Candace L. (1983) What the speaker means: The recognition of speakers' plans in discourse. In *International Journal of Computers and Mathematics, Special Issue in Computational Linguistics*, 9(1): 71-82.

Sidner, Candice L. (1985) Plan parsing for intended response recognition in discourse. In *Computational Intelligence*, 1(1): 1-10, February.

Simon, Thomas W. (1990) Artificial methodology meets philosophy. In *The foundations of Artificial Intelligence: a sourcebook*, Derek Partridge and Yorick Wilks (Eds.), 155-164. Cambridge, United Kingdom: Cambridge University Press.

Skinner, B. F. (1957) *Verbal behavior.* Englewood Cliffs, New Jersey: Prentice-Hall.

Slator, Brian M., Matthew P. Anderson and Walt Conley (1986) Pygmalion at the Interface. In *Communications of the ACM, Special Section on the Human Aspects of Computing, Henry Ledgard (Ed.)*, Vol. 29, No. 7, July, 599-605. Also, as *Pygmalion at the interface: impatient users and foreign-speak*, Memorandum in Computer and Cognitive Science, MCCS-85-26, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA.

Small, Duane W. and Linda J. Weldon (1983) An experimental comparison of natural and structural query languages. In *Human Factors*, 25(3), 253-263.

Smolensky, P. (1988) The constituent structure of connectionist mental states: a reply to Fodor and Pylyshyn. In *Connectionism and the Philosophy of Mind, Supplement to Vol. XXVI of The Southern Journal of Philosophy*, T. Horgan and J. Tienson (Eds.). .

Soderlund, Cari (1988) *A TCP remote message passing service.* Memorandum in Computer and Cognitive Science, MCCS-88-130, Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA.

Sparck Jones, Karen (1990) What sort of thing is an AI experiment. In *The foundations of Artificial Intelligence: a sourcebook*, Derek Partridge and Yorick Wilks (Eds.), 274-285. Cambridge, United Kingdom: Cambridge University Press.

Sperber, Dan and Deirdre Wilson (1986) *Relevance – Communication and Cognition.* Oxford: Basil Blackwell Ltd..

Stampe, Dennis W. (1975) Meaning and truth in the theory of speech acts. In *Syntax and Semantics, Speech Acts, Vol. 3*, Peter Cole and Jerry L. Morgan (Eds.), 1-39. New York: Academic Press.

Stampe, D. (1975) Meaning and truth in the theory of speech acts. In *Syntax and semantics, Vol. 9, Pragmatics*, 315-332. New York: Academic Press

Suchman, Lucy (1990) *Plans and situated actions: the problem of human-machine communication.* Cambridge, United Kingdom: Cambridge University Press.

Sutcliffe, Richard (1991) Representing meaning using microfeatures. In *Connectionist approaches to natural language processing, Vol. 1*, R. Reilly and N.E. Sharkey (Eds.). Hillsdale, NJ: Lawrence-Erlbaum.

Tennant, Harry R. (1979) *Evaluation of natural language processors.* Ph.D. Dissertation, University of Illinois, Urbana-Champaign, Illinois.

Tennant, Harry R., Ross, K.M. and Thompson, C.W. (1983) Usable natural language interfaces through menu-based natural language understanding. In *Proceedings of Conference on Computer and Human Interaction (CHI) '83*, 154-160.

Thompson, Bozena H. (1980) Linguistic analysis of natural language communication with computers. In *Proceedings of the Eighth International Conference on Computational Linguistics*, 190-201, Tokyo, Japan.

Tulving, E. (1972) Episodic and semantic memory. In *Organization of memory*, E. Tulving and W. Donaldson (Eds.). New York: Academic Press.

Turing, Alan M. (1950) Computing machinery and intelligence. In *Mind*, 59, 433-460.

van Dijk, T.A. (1972) *Some aspects of text grammars.* The Hague: Mouton.

van Dijk, T.A. (1977) *Text and context: Explorations in the semantics and pragmatics of discourse.* London: Longmans.

Walker, Donald (1978) *Understanding spoken language.* New York: Elsevier/North-Holland.

Walker, Marilyn (1989) Evaluating discourse processing algorithms. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, 251-261, University of British Columbia, Vancouver, British Columbia, Canada, June.

Walker, Marilyn and Steve Whittaker (1989) *When natural language is better than menus: a field study.* Technical Report, Hewlett Packard Laboratories, Advanced Information Management Dept., Filton Road, Stoke Gifford, Bristol, United Kingdom, European Community (EC).

Waltz, David (1975) Natural language access to a large database: an engineering approach. In *Advance papers for the Fourth International Joint Conference on Artificial Intelligence (IJCAI-75)*, 868-872, Tbilisi, Georgia, USSR.

Waltz, David (1978) An English language question answering system for a large relational database. In *Communications of the ACM*, 526-539, Vol. 21, No. 7, July.

Waltz, David L. and Pollack, Jordan P. (1985) Massively parallel parsing: A strongly interactive model of natural language interpretation. In *Cognitive Science*, 9(1), 51-74, January-March.

Webber, Bonnie (1978) *A formal approach to discourse anaphora.* Doctoral Dissertation. Also in, BBN Report No. 3761, Bolt, Beranek and Newman, Cambridge, MA, USA.

Webber, Bonnie (1980) *A computational approach to discourse anaphora.* New York: Garland.

Webber, Bonnie (1983) So what can we talk about now. In *Computational Models of discourse*, M. Brady and Robert C. Berwick (Ed.). Cambridge, MA: MIT Press.

Webber, Bonnie, A. Joshi, and I. Sag (1981) *Elements of discourse understanding.* Cambridge, United Kingdom: Cambridge University Press.

Weischedel, David and Norman K. Sondheimer (1982) An improved heuristic for ellipsis processing. In *Proceedings of the 20th Annual Meeting of the Association for Computational Linguistics*, 85-88, Toronto, Canada.

Weischedel, R.M. and W.K. Sondheimer (1982) An improved heuristic for ellipsis processing. In *Proceedings of the 20th Annual Meeting of the Association for Computational Linguistics*, 85-88, Toronto, Canada.

Weizenbaum, J. (1965) ELIZA – A computer program for the study of natural language communication between man and machine. In *Communications of the ACM*, 9(1): 36-45.

Whittaker, Steve and Phil Stenton (1988) Cues and control in expert-client dialogues. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, 123-130, State University of New York at Buffalo, Buffalo, New York, USA.

Whittaker, Steve and Phil Stenton (1989) User studies and the design of natural language systems. In *Proceedings of the Fourth Annual Meeting of the European Association of Computational Linguistics*, 116-122, Manchester, United Kingdom.

Whittaker, Steve and Marilyn Walker (1989) *Comparing two user-oriented database query languages: a field study.* Technical Report, Hewlett Packard Laboratories, Advanced Information Management Dept., Filton Road, Stoke Gifford, Bristol, United Kingdom, European Community (EC).

Whittemore, Greg, Kathleen Ferrara, Hans Brunner (1990) *Post-modifier prepositional phrase ambiguity in written/interactive dialogue.* U S WEST Advanced Technologies, Science and Technology Technical Report, ST 04-01, U S WEST Advanced Technologies, Engelwood, Colorado, USA.

Wilensky, Robert (1983) *Planning and understanding.* Reading, MA: Addison-Wesley.

Wilensky, Robert (1987) Some complexities of goal analysis. In *Preprints of the Third Conference on Theoretical Issues in Natural Language Processing (TINLAP-3)*, 97-99, Computing Research Laboratory, New Mexico State University, Las cruces, New Mexico, USA, January. Also in, *Theoretical issues in natural language processing*, Yorick Wilks (Ed.). Hillsdale: N.J.: Lawrence Erlbaum Associates.

Wilensky, Robert, Yigal Arens and David Chin (1984) Talking to UNIX in English: An overview of UC. In *Communications of the ACM*, 574-593, Vol. 27, No. 6, June.

Wilensky, Robert, Jim Mayfield, Anthony Albert, David Chin, Charles Cox, Marc Luria, James Martin, and Dekai Wu (1986) *UC – a progress report.* Report No. UCB/CSD 87/303, Computer Science Division (EECS), University of California, Berkeley, California 94720, USA, July.

Wilensky, Robert, David N. Chin, Marc Luria, James Martin, James Mayfield and Dekai Wu (1988) The Berkeley UNIX Consultant project. In *Computational Linguistics*, 35-84, Vol. 14, No. 4, December.

Wilks, Yorick (1973) An artificial intelligence approach to machine translation. In *Computer Models of Thought and Language*, R. Schank and K. Kolby (Eds.). San Francisco, CA: Wh Freeman and Co..

Wilks, Yorick (1975a) Preference semantics. In *Formal semantics of natural language*, Keenan, Edward (Ed.). Cambridge, United Kingdom: Cambridge University Press. Also as, Memo AIM-206, Artificial Intelligence Laboratory, Stanford University, Stanford, California, USA, July 1973.

Wilks, Yorick (1975b) An intelligent analyzer and understander of English. In *Communications of the ACM*, 18(5), 264-274, May. Also in, *Readings in Natural Language Processing*, Barbara Grosz, Karen Sparck Jones and Bonnie Webber (Eds.), 193-204, Los Altos, California: Morgan Kaufmann, 1986.

Wilks, Yorick (1975c) A preferential, pattern-seeking semantics for natural language inference. In *Artificial Intelligence*, 6, 53-74.

Wilks, Yorick (Ed.) (1987) *Theoretical issues in natural language processing.* Hillsdale: N.J.: Lawrence Erlbaum Associates.

Wilks, Yorick (1990) One small head: models and theories. In *The foundations of Artificial Intelligence: a sourcebook*, Derek Partridge and Yorick Wilks (Eds.), 121-134. Cambridge, United Kingdom: Cambridge University Press.

Wilks, Yorick and J.S. Bien (1983) Beliefs, points of view and multiple environments. In *Cognitive Science*, 795-119. Also in, *Sixth International Joint Conference on Artificial Intelligence (IJCAI-)*, 431-453, USA.

Wilks, Yorick and Afzal Ballim (1987) Multiple agents and the heuristic ascription of belief. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI-87)*, 119-124, Milan, Italy, August.

Wilks, Yorick and Paul Mc Kevitt (Eds.) (1990) *Proceedings of the Fifth Rocky Mountain Conference on Artificial Intelligence (RMCAI-90), Subtitled, 'Pragmatics in Artificial Intelligence'.* Computing Research Laboratory, Dept. 3CRL, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA, June.

Winograd, Terry (1971) *Procedures as a representation for data in a computer program for understanding natural language.* Doctoral Dissertation, Department of Mathematics, Massachusetts Institute of Technology, August. Also as, Report No. TR-17, AI Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA, February.

Winograd, Terry (1972) *Understanding natural language.* New York: Academic Press.

Winograd, Terry and Fernando Flores (1986) *Understanding computers and cognition: a new foundation for design.* Norwood, New Jersey: Ablex Publishing Corporation.

Wittgenstein, Ludwig (1963) *Philosophical Investigations (translated by G.E. Anscombe).* Oxford: Blackwell.

Woods, W.A. (1970) Transition network grammars for natural-language analysis. In *Communications of the ACM*, 13(10), 591-606, October.

Woods, W.A. (1978) Semantics and quantification in natural language question answering. In *Advances in Computers*, M. Yovits, (Ed.), Vol. 17, 1-87. New York: Academic Press.

Woods, W.A., R.M. Kaplan, and B.L. Nash-Webber (1972) *The lunar sciences natural language information system: final report.* BBN Report 2378, Bolt, Beranek and Newman Incorporated, Cambridge, MA, USA, June.

Young, Sheryl R. (1989) The MINDS system: using context and dialog to enhance speech recognition. In *DARPA Speech and Natural Language Workshop*, Pittsburgh, Pennsylvania, USA, February.

Yun, David Y. and David Loeb (1984) The CMS-HELP expert system. In *Proceedings of the International Conference on Data Engineering*, IEEE Computer Society, 459-466, Los Angeles..

Zock, Michael and Gérard Sabah (1988) (Eds.) *Advances in natural language generation: an interdisciplinary perspective.* Communication in Artificial Intelligence Series, Vol. 2, 58-75. London: Pinter Publishers.