Int. J. Human-Computer Studies (1999) **51,** 947–989
Article No. ijhc.1999.0271
Available online at http://www.idealibrary.com on IDEAL®

Why machines should analyse intention in natural language dialogue†

PAUL MC KEVITT
*Center for PersonKommunikation (CPK), Fredrik Bajers Vej 7-A5,
Institute of Electronic Systems (IES), Aalborg University, DK-9220, Aalborg,
Denmark. email: pmck@cpk.auc.dk*

DEREK PARTRIDGE
*Department of Computer Science, University of Exeter, GB-EX4 4PT, Exeter, England.
email: derek@dcs.exeter.ac.uk*

YORICK WILKS
*Department of Computer Science, Regent Court, University of Sheffield,
211 Portobello Street, GB-S1 4DP, Sheffield, England. e-mail: yorick@dcs.shef.ac.uk*

(*Received 16 February 1998 and accepted in revised form 13 May 1999*)

One of the most difficult problems in Artificial Intelligence (AI) is to construct a natural language processing system which can interact with users through a natural language dialogue. The problem is difficult because there are so many ways by which a user can phrase his/her utterances to such a system. An added problem is that different types of users have different types of intentions and will conduct different exchanges with the system. While many have proposed theories and models of the processing of intentions in dialogue, few of these have been incorporated within working systems and tested empirically. Here, an experiment is conducted to test 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. The experiment provides evidence to support the hypothesis. In turn, the hypothesis provides evidence for a theory of intention analysis for 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. A computational model, called Operating System CONsultant (OSCON), implemented in Quintus Prolog, makes use of the theory and hypothesis to understand, and answer in English, English questions about computer operating system.

1. Introduction

One of the most difficult problems within the field of Artificial Intelligence (AI) is that of processing language by computer, or natural-language processing (see Allen, 1987;

†This work has been funded, in part, by a Research Fellowship from the Computing Research Laboratory (CRL), New Mexico State University, Las Cruces, New Mexico, US, a grant from US WEST Advanced Technologies, Denver, Colorado, US, and and a Visiting Fellowship from The School of Electronic Engineering, Dublin City University, Dublin, Ireland.

1071-5819/99/110947 + 43 $30.00/0

Partridge, 1991; Mc Kevitt, Partridge & Wilks, 1992). Much work on natural-language processing has concentrated on modelling the structure, meaning and usage of individual utterances.† However, it is not often that natural-language utterances occur on their own, independent of some context or other. Hence, one problem 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. Many of the approaches stress one theme over all the others.

Our aim here is to test 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. While many have proposed theories and models of the processing of intentions in dialogue, few of these have been incorporated within working systems and tested empirically. We have conducted empirical evidence in experiments for the computer operating systems domain (see Mc Kevitt & Ogden, 1989; Mc Kevitt, 1991*b*) shows that there is a correlation between level of user expertise and frequency of intention types used. Our focus of attention will be on natural-language dialogue, or the form of natural language that exists between communicating agents, whether they be people or computers. The focus will be on written dialogues rather than spoken ones, and other forms of natural-language discourse such as texts are not considered.

To test the Intention-Computer Hypothesis, an experiment was conducted in the domain of natural-language consultancy for computer operating systems where we chose the UNIX‡ operating system. A sample dialogue is selected and processed by a computer program called Operating System CONsultant (OSCON) which understands, and answers in English, English questions about computer operating systems. We claim that the sample dialogue is representative of a typical natural-language dialogue. The experiment consisted of an empirical comparison on intention sequence analysis over the sample dialogue.

Our motivation for testing the Intention-Computer Hypothesis is that we have provided a theory of intention analysis in Mc Kevitt (1991*b*). The theory argues that natural-language dialogue can be modelled, in part, by the analysis of intentions. The Intention-Computer Hypothesis exists as part of the theory which has been incorporated within OSCON. Let us now move on the discuss briefly some existing theories of natural-language discourse processing.

2. Background

There has been much research conducted on developing theories, and designing computational models for understanding natural-language discourse. Much of the work has

†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.

‡UNIX is a trademark of X/Open, Inc.

been on developing models for text and speech processing. Some work has concentrated more specifically on processing natural-language dialogue. Theories concentrate on the themes of semantics, structure and intention. Many of the approaches have common themes, while there are differing points of view between them. The one common theme of all the models is that of coherence of discourse. The motivation for a concentration on discourse coherence is the fact that rational speakers usually use coherent discourse.

Semantic theories 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 dicourse such as those proposed by Fass (1988), Schank (1972, 1973, 1975), Schank and Abelson (1977), and Wilks (1973, 1975*a*–*c*). These approaches concentrate on recognizing 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 maximize the coherence of the discourse. Semantics-based approaches argue that by modelling the semantics of a discourse, other information will fall out of that.

Structure-based theories argue that a discourse can be modelled in terms of structural units which can be recognized and marked out in the discourse. Examples of such theories are proposed in Alshawi (1987), Dale (1992), Grosz (1983), Grosz and Sidner (1986), Grosz, Joshi and Weinstein (1995), Hobbs (1979, 1983, 1985), Reichman (1985), Sidner (1983, 1985) and Webber (1978). Many of the theories model discourse coherence by defining explicit spaces which represent implicit spaces in the discourse. Much of the work involves recognizing and representing these spaces and the relationships between them. A number of different names are given to spaces which represent discourse structure. Grosz and Sidner use the term *focus space* (as does Alshawi) while Reichman uses the term *context factor*. All the approaches argue that discourse structure can be recognized 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, Reichman considers intention as being sub-ordinate to topic.

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. Examples of such approaches are given in Allen (1983), Allen *et al.* (1995), Appelt (1981, 1985), Carberry (1989), Cohen, Perrault and Allen (1982), Grosz and Sidner (1986), Grosz and Kraus (1996), Hinkelman and Allen (1989), Hobbs (1979), Litman and Allen (1987), Moore and Paris (1993), Schank and Abelson (1977), Sidner (1994) and Wilensky (1983). Formal models of intention are found in Cohen and Levesque (1985) and Moore (1980), and on speech acts in Perrault (1990). A comprehensive collection of work on intentions in dialogue is found in Cohen, Morgan and Pollack (1990). Ideas here emerge from claims made by philosophers of language such as Austin (1962), Searle (1969, 1983) and Bratman (1987). Such philosophers argue that the motivation for people to use language is to achieve their intentions.

The coherence of a discourse is determined by the coherence of the intentions of the participants. Much of the work here involves recognizing 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. The approaches argue that people's intentions underlie their use of language, and that by modelling these intentions one can model language. It is not argued that intentions in people's brains can be seen but that people's intentions can be recognized and inferred from the utterances they use.

An area where intention-based models of discourse have been applied is that of integration of natural language and vision processing (see Mc Kevitt 1995/1996) and intelligent multimedia interfaces (see Maybury, 1993; Maybury & Wahlster, 1998). A number of natural language systems for the description of image sequences have been developed (see Neumann & Novak, 1986; Herzog & Retz-Schmidt, 1990). These systems can verbalize the behaviour of human agents in image sequences about football and describe the spatio-temporal properties of the behaviour observed. Retz-Schmidt (1991) and Retz-Schmidt and Tetzlaff (1991) describe an approach which yields plan hypotheses about intentional entities from spatio-temporal information about agents. The results can be verbalized in natural language.

Maybury (1993) considers the use of communicative acts for generating multimedia explanations to compose route plans for a cartographic information system. Three types of communicative act are described: linguistic (e.g. illocutionary and locutionary), visual (e.g. deictic) and media-independent/rhetorical (e.g. identify, describe). Wahlster, André, Finkler, Profitlich and Rist (1991) and André and Rist (1996) discuss how communicative act sequences can be used for the generation of multimodal documents. They describe the WIP system which can provide information on assembling, using and maintaining physical devices like an expresso machine or a lawnmower. In the latter multimedia systems, natural-language processing is used more for annotation through text generation whereas we are interested in analysis.

While the approaches above stress meaning, structure or intention more or less in their view of discourse, many attempt to provide an integrated account. Examples are where Grosz and Sidner (1986) model attention, intentions and the structure of discourse, Grosz and Kraus (1996) model collaboration for complex group action and Sidner (1994) provides a language for collaborative negotiation. Rhetorical structure theory (RST) (see Mann & Thompson, 1987) is a major general approach to discourse analysis/synthesis focussing more on texts than dialogue. Pustejovsky (1987) also 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. While many of the approaches discussed above propose theories and models for processing discourse, few of these have been incorporated within working systems and tested empirically which is what we focus on here.

3. Intention sequences and satisfaction

A theory of intention sequencing is given in Mc Kevitt (1991*b*), where we proposed a set of basic intentions which a speaker would be expected to use in the domain of natural-language consultancy (see Table 1). We have conducted Wizard-Of-Oz experiments (see Mc Kevitt & Ogden, 1989; Mc Kevitt, 1991*b*) where there were the only nine intention types derived from empirical data of natural-language dialogues on operating systems. It is difficult to compare this set of intentions to the set of *rhetorical relations* in RST (see Mann & Thompson, 1987) which is intended primarily for general written texts.

TABLE 1
Definitions of principal intentions for the consultancy domain

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?	An intention requesting a UNIX command to achieve a UNIX 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 UNIX concepts, objects, or commands. E.g. "what is UNIX?"
Instruction	An intention acting as an instruction to achieve a GOAL rather than 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 UN resolution 611?"	An intention requesting explanation of a response from the UNIX shell. 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."
Repetition	An intention which is a repeated request. E.g. "How many people live in the Gulf ?" followed by "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 system. E.g. How do I eat a file?

[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.

Similarly, Hobbs' (1985) set of *coherence relations* apply more to written texts than task-specific dialogues. However, *elaboration* occurs in both RST and in Hobbs' set. Grosz and Sidner's (1986) *Discourse Segment Purposes* (*DSPs*) are meant for larger chunks of discourse where a segment of text contributes some general intention.

Our Wizard-Of-Oz empirical data showed that expert subjects have mainly *information* and *description* intentions, whereas less expert subjects have much more of the other intentions such as *elaboration*, *explanation* and *repetition*. We proposed that the set of intentions could be ordered into two groups based on the amount of satisfaction they represent (see Figure 1). *Nointentions* are not placed in the ordering because as they are not relevant to the domain they are not an obvious measure of the level of speaker satisfaction.

Based on this grouping of intentions, and the fact that some sequences indicate less satisfaction than others, we proposed a weighted formula for the measure of *satisfaction* as follows (where $[X \rightarrow Y]$ represents the frequency of intention pairs from intention X to intention Y):

$$\begin{aligned} 3([I \rightarrow I]) &+ [I \rightarrow De] + [I \rightarrow El] + [I \rightarrow Ex] + [I \rightarrow Re] \\ &+ [De \rightarrow I] \rightarrow [El \rightarrow I] + [Ex \rightarrow I] + [Re \rightarrow I] \\ &+ 3([De \rightarrow De]) + [De \rightarrow El] + [De \rightarrow Ex] + [De \rightarrow Re] \\ &+ [El \rightarrow De] + [Ex \rightarrow De] + [Re \rightarrow De] \end{aligned}$$

Relative dissatisfaction can be calculated by summing dissatisfaction intention pairs. The formula for a measure of *dissatisfaction* is

$$\begin{aligned} 3([El \rightarrow El] &+ [Ex \rightarrow Ex] + [Re \rightarrow Re]) \\ &+ 2([Ex \rightarrow El] + [Re \rightarrow El] + [Re \rightarrow Ex] \\ &+ [El \rightarrow Ex] + [El \rightarrow Re] + [Ex \rightarrow Re]) \end{aligned}$$

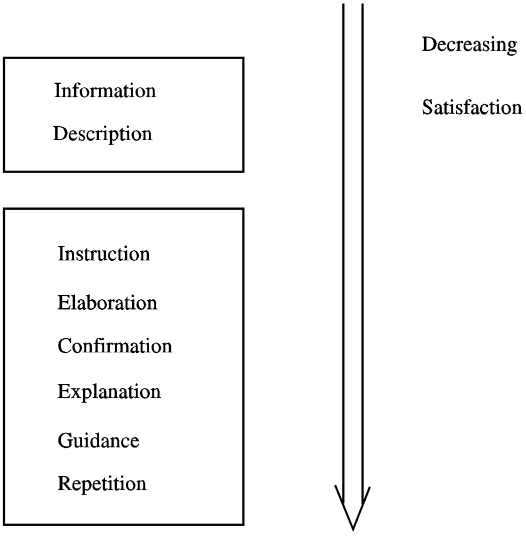

FIGURE 1. *Satisfaction* ordering of intention.

The theory of intentions analysis and measure of user satisfaction have been integrated into OSCON which we shall now briefly describe.

4. 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 & Wilks, 1987; Mc Kevitt, 1988, 1991*a*). OSCON enables a user to enter written English queries and then answers them in English. The program is written in Quintus Prolog and runs on a Sun Sparcstation computer in real time. It can answer queries for over 30 commands from each of the UNIX and MS-DOS† operating systems. OSCON handles four basic query types which are specializations of five types of intention. 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. Examples of sample input and output for OSCON are shown in Appendix A.

The main differences with the Unix with the Unix Consultant (UC) (see Wilensky *et al.*, 1984, 1988; Chin, 1988) are (1) that OSCON is an Operating System consultant representing data on both UNIX and MS-DOS, (2) that OSCON is a specialized system with detailed representations for Operating System concept whereas UC, with its general knowledge representation scheme, has to perform more elaborate procedures to resolve queries and (3) OSCON can handle a larger range of query and intention types.

The architecture of the OSCON system consists of six basic modules and two extension modules. There are at least two arguments for modularizing 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, Burton & Bell, 1975; 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 both context and user-sensitive natural-language generation. A diagram of OSCON's architecture is shown in Figure 2.

ParseCon consists of a set of Prolog predicates which read natural-language input and determine the type of query being asked (a specialization of intention type) by the user.

†MS-DOS is a trademark of Microsoft Corporation.

‡ParseCon uses a grammar in the definite clause grammar (DCG) formalism of Prolog. DCGs were first developed by Pereira and Warren (1980) as a tool to be used in Prolog for natural-language processing.

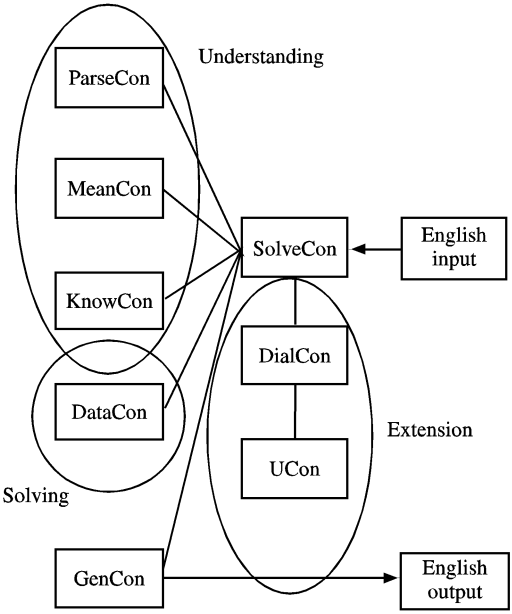

FIGURE 2. Architecture of the Operating System CONsultant (OSCON) system.

There are four basic types of query, which are specializations of five intention types, recognized. 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 a matrix of intention pairs in the dialogue, called an *intention matrix*, by locating the relevant cell in the matrix which needs to be updated and increasing its count. 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. We will not discuss details of processing within components of the OSCON system. Such details can be found in Mc Kevitt (1991*a*,*b*).

Now we can move on to discuss the experiment which tests the Intention-Computer Hypothesis. In order to commence any experimental analysis one must ask oneself what sort of data one will collect. The answer to this question is provided by answering the question: what do we wish to do with that data? It will 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. 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.

5. The experiment

The goals of the experiment were two-fold. First, to test whether the OSCON system can conduct intention analysis over a sample natural-language dialogue by the processes of intention and intention-sequence recognition and representation which can then be used by OSCON to facilitate user modelling. In turn, the user-model is used to generate natural-language responses which are sensitive to the level of user expertise. The latter problem of user-modelling has received much attention in the field (see André, Cohen, Graf, Kass, Paris & Wahlster, 1992; Chin, 1988; Kobsa & Wahlster, 1988). Second, three versions of OSCON were tested over the same natural-language input dialogue. The sample natural-language dialogue was representative of a typical natural-language dialogue for UNIX natural-language consultancy.

First, the OSCON system was tested for its capability of intention-sequence analysis over a sample natural-language dialogue on UNIX help. (*full analysis*, experimental treatment). This version of OSCON was programmed to discriminate five types of intention: *information*, *description*, *elaboration*, *explanation* and *repetition* as well as *nointentions*. This intention set is sufficient for demonstrating the Intention-Computer Hypothesis.

Then, a test was conducted to show the performance of OSCON over the *same* input natural-language dialogue, when "handicapped", i.e. with the capability of intention recognition but without any capability of intention-sequence analysis (*no analysis*, control treatment). This version of OSCON did not have 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 third test was conducted to show the performance of OSCON over the *same* input natural-language dialogue, when "handicapped" differently, i.e. with complete capability of intention recognition but without the capability of intention-pair analysis (*frequency-only analysis*, second control treatment). Hence, this "handicapped" version of OSCON would only be able to determine intention frequencies. This condition serves the purpose of showing that determining absolute intention frequencies alone, rather than sequences of intention, will handicap OSCON's ability to model the subtleties of intention change over time and in turn ability to build an accurate model of the user. Sample traces describing detailed results for the performance of OSCON on *full analysis*, *no analysis* and *frequency-only* analysis are shown in Appendix A.

It is argued that the natural-language consultancy dialogue on UNIX, for which the empirical comparison was conducted, is representative of a typical natural-language consultancy dialogue. It contains examples of many of the intention pairs discovered in the data collection experiments we have conducted (see Mc Kevitt & Ogden, 1989; Mc Kevitt, 1991*b*). All intention pairs appearing in the sample dialogue are marked by "X" in Figure 3. Looking at Figure 3 it is noted that the intention pairs existing in the sample dialogue act as a reasonable covering of the complete set of possible intention pairs. The data from our data collection had a large proportion of pairs with *information* intentions as sources and sinks. Many of the intention sequences in the sample dialogue exhibit 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.

It is possible to compare the behaviour of *full analysis* and *no analysis* to determine respective performances. One criterion for comparison is the correctness of intention-type discrimination. *Full analysis* discriminates all intention types for all 23 queries with

t_n

t_{n+1}

Intension	In	De	El	Ex	Re	No
Information	X		X	X	X	X
Description	X		X			
Elaboration	X	X	X	X		
Explanation	X			X		
Repetition	X				X	
No intention		X			X	

FIGURE 3. Intention-pair sequences in sample natural-language dialogue.

no errors. *No analysis* has nine intention discrimination errors in the 23 queries (error-rate = 39%). The gap between *full analysis* and *no analysis* is reflected even more when one looks at the differences in intention-sequence determination. *Full analysis* can recognize the sequences shown in Figure 3 while *no analysis* can only recognize the sequences in Figure 4. *Full analysis* determines all 22 intention sequence for all 23 queries with no errors. *No analysis* has 18 intention-sequence discrimination errors out of a possible 22 (error-rate = 82%). This performance rate is obtained by finding the ratio of the differences in intention sequence totals between *full analysis* (7,2,4,2,5,2) and *no analysis* (14,3,0,3,0,2) over the total number of possible sequences:

$$= \frac{(14-7)+(3-2)+(4-0)+(3-2)+(5-0)+(2-2)}{22} \times 100$$

$$= \frac{18}{22} \times 100$$

$$= 81.8\%$$

Note that *no analysis* is penalized for attributing intention pairs where they should not be attributed (nine times) and also for not attributing them where they should be attributed (nine times).

It is noted that the levels of *satisfaction* and *dissatisfaction*, which act as a measure of user expertise, are dynamically changing for *full analysis* but hardly change at all for *no analysis*. During most of the dialogue, *no analysis* responds with too little information because of its eagerness to ascribe satisfaction to the user and in fact the user is *never* recorded as dissatisfied. *Full analysis* continuously modifies the character of its natural-language responses to the user, whereas *no analysis never* changes the character of its natural-language responses. Out of 23 queries, the *no analysis* version of OSCON increases the wrong member of the set {satisfaction, dissatisfaction} five times out of a possible 23 (error-rate = 22%).

t_n

t_{n+1}

Intension	In	De	El	Ex	Re	No
Information	✕			✕		✕
Description	✕					
Elaboration						
Explanation	✕			✕		
Repetition						
No intention		✕				

FIGURE 4. Intention pair sequences recognized by *no analysis.*

TABLE 2
Comparison of performance data on variations of sequence analysis

Sequence analysis	Intention-discrimination errors (max. = 23)	Intention-sequence errors (max. = 22)	Satisfaction-rating errors (max. = 23)
Full	(0%) 0	(0%) 0	(0%) 0
None	(39%) 9	(81%) 18	(22%) 5
Frequency-only	(0%) 0	—	(35%) 8

While comparing *full analysis* against *frequency-only analysis*, the latter performs just as well as *full analysis* in terms of intention discrimination. With respect to user-modelling, *frequency-only analysis* does not perform similar to *full analysis*. Out of 23 queries, the *frequency-only analysis* version of OSCON increases the wrong member of the set {satisfaction, dissatisfaction} eight times out of a possible 23 (error-rate = 35%).

It is noted that the levels of satisfaction and dissatisfaction, which act as a measure of user expertise, are dynamically changing for *full analysis* but hardly change at all for *frequency-only analysis*. During most of the dialogue, *frequency-only analysis* responds with too much information because of its eagerness to ascribe dissatisfaction to the user. *Sequence-analysis* continuously modifies the character of its natural-language responses to the user, whereas *frequency-only analysis* only changes once, at the start, the character of its natural-language responses. This phenomenon is caused by the fact that *frequency-only analysis*, 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. These results are summarized in Table 2.

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. The results show that *full analysis* performs much better than *no analysis* with respect to intention discrimination, intention-sequence analysis and user-modelling. Also, *full analysis* performs much better than *frequency-only analysis* with respect to user-modelling. Hence, the experiment gives positive evidence for the Intention-Computer Hypothesis. Note that these results are mean 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.

6. Conclusion and future work

While many have proposed theories and models of the processing of intentions in dialogue, few of these have been incorporated within working systems and tested empirically. An experiment has been conducted to test what we call the *Intention-Computer Hypothesis* where the goals of the experiment were two-fold. First, we wanted to demonstrate that the OSCON system can conduct intention analysis over a natural-language dialogue by the processes of intention and intention-sequence recognition and representation, and, in turn, these processes can be used by OSCON to facilitate user-modelling. Second, we compared three versions of OSCON, over the same input natural-language dialogue, in order to demonstrate that OSCON could conduct intention analysis over a natural-language dialogue and that while not having that capability OSCON's performance decreases. The results of the experiments show that *full analysis* of intentions performs much better than *no analysis* with respect to intention discrimination, intention-sequence analysis and user-modelling. Also, *full analysis* performs much better than *no analysis* with respect to user-modelling. The experiment provides positive evidence for the hypothesis.

The experiment does not only provide a means of testing the Intention-Computer Hypothesis but also provides evidence for a theory of intention analysis proposed in Mc Kevitt (1991*b*). The theory claims that natural-language dialogue can be modelled in part by the analysis of intention. A central principle of the theory is that the coherence of natural-language dialogue can be modelled in part by the analysis of intentions. The experiment conducted here shows that the theory is empirically valid with respect to real-world data.

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 experiment conducted here, it could be argued that it is too constrained in the sense that the environment is not like a real environment of human-computer natural-language consultancy and in the sense that the interaction was conducted in a controlled way with specific tasks and a limited set of UNIX commands.† 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

†This is considered a problem with scientific experimentation in general. Neisser (1987) gives a discussion of the problem in Psychology.

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.

A number of other empirical studies could be conducted with computational models, like OSCON. 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 been shown that the OSCON system, as a result of user-modelling, can modify its response 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 representing 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 recognizing 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 types in natural-language utterances?, 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 do not have the capability of detecting the rationality of the users interacting with them. Computer programs, especially AI ones, should be able to detect irrational behaviour and 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.

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. More computer programs and empirical studies are needed so that 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, computers will be used more effectively by people.

We would like to thank David Benyon, Antony Galton, Thomas Green and two anonymous referees for comments on this research.

References

ALLEN, J. F. (1983). Recognising intentions from natural language utterances. In M. BRADY and R. C. BERWICK, Eds. *Computational Models of Discourse*, pp. 107–166. Cambridge, MA: MIT Press.

ALLEN, J. F. (1987). *Natural Language Understanding*. Benjamin/Cummings Series in Computer Science. Menlo Park, CA: Benjamin/Cummings.

ALLEN, J. F., SCHUBERT, L. K., FERGUSON, G., HEEMAN, P., HWANG, C. H., KATO, T., LIGHT, M., MARTIN, N. G., MILLER, B. W., POESIO, M. & TRAUM, D. R. (1995). The TRAINS project: a case study in building a conversational planning agent. *Journal of Experimental and Theoretical*, Artificial Intelligence, **7**.

ALSHAWI, H. (1987). *Memory and Context for Language Interpretation*. Cambridge, UK: Cambridge University Press.

ANDRÉ, E. & RIST, T. (1996). Coping with temporal constraints in multimedia presentation planning. *Proceedings of the* 13*th National Conference on Artificial Intelligence*, pp. 142–147. Portland, Oregon.

ANDRÉ, E., COHEN, R., GRAF, W., KASS, B., PARIS, C., & WAHLSTER, W. Eds. (1992). *UM*92: 3*rd International Workshop on User Modeling Proceedings*. DFKI Document D-29-17, German National Research Centre for Artificial Intelligence (DFKI), Postfach 20 80, D-6750 Kaiserslautern, Germany.

APPELT, D. E. (1981). *Planning natural-language utterances to satisfy multiple goals*. Technical Note 259, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, USA.

APPELT, D. E. (1985). *Planning English Sentences*. Cambridge, UK: Cambridge University Press.

AUSTIN, J. L. (1962). In J. O. URMSON & M. SBISÀ, Eds. *How to do things with words* (2 edn). Cambridge, MA: Harvard University Press.

BRATMAN, M. E. (1987). *Intention, Plans and Practical Reason*. Cambridge, MA: Harvard University Press.

BROWN, J. S., BURTON, R. R. & BELL, A. G. (1975). SOPHIE: a step towards creating a reactive learning environment. *International Journal of Man-Machine Studies*, **7,** 675–696.

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, S. (1989). A pragmatics-based approach to ellipsis resolution. *Computational Linguistics*, **15,** 75–96.

CHIN, D. (1988). Exploiting user expertise in answer expression. *Proceedings of the* 7*th National American Conference on Artificial Intelligence* (*AAAI*-88), Vol. 2, pp. 756–760. St. Paul, Minnesota, USA.

COHEN, P. R. & LEVESQUE, H. (1985). Speech acts and rationality. *Proceedings of the* 23*rd Annual Meeting of the Association for Computational Linguistics*.

COHEN, P. R. & LEVESQUE, H. (1990). In P. R. COHEN, J. L. MORGAN & M. E. POLLACK, Eds. *Intentions in Communication*, Cambridge, MA: MIT Press.

COHEN, P. R., MORGAN, J. & POLLACK, M. E. Eds. (1990). *Intentions in Communication*. Cambridge, MA: MIT Press.

CHOEN, P. R., PERRAULT, C. R. & ALLEN, J. F. (1982). Beyond question answering. In W. G. LEHNERT & M. H. RINGLE, Eds. *Strategies for Natural Language Processing*, pp. 245–274. Hillsdale, NJ: Lawrence Erlbaum Associates.

DALE, R. (1992). *Generating Referring Expressions*. Cambridge, MA: MIT Press.

FASS, D. (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.

GROSZ, B. J. (1983). TEAM, a transportable natural language interface system. *Proceedings of the Conference on Applied Natural Language Processing*, pp. 39–45. Santa Monica, CA, USA, February.

GROSZ, B. J. & KARUS, S. (1996). Collaborative plans for complex group action. *Artificial Intelligence*, **86**.

GROSZ, B. J., JOSHI, A. K. & WEINSTEIN, S. (1995). Centering: a framework for modelling the local coherence of discourse. *Computational Linguistics*, **21,** 203–225.

GROSZ, B. J. & SIDNER, C. L. (1986). Attention, intentions and the structure of discourse. *Computational Linguistics*, **12,** 175–204.

HERZOG, G. & RETZ-SCHMIDT, G. (1990). Das System SOCCER: Simultane Interpretation und naturalichsprachliche Beschreibung zeitveranderlicher Szenen. In J. Perl, Ed. *Sport und Informatik*, pp. 95–119. Schorndorf: Hofmann.

HINKELMAN, E. & ALLEN, J. (1989). Two constraints on speech act ambiguity. *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pp. 212–219, University of British Columbia, Vancouver, British Columbia, Canada, June.

HOBBS, J. (1979). Coherence and coreference. *Cognitive Science*, **3,** 67–90.

HOBBS, J. (1983). Why is discourse coherent?. In F. NEUBAUER, Ed. *Coherence in Natural Language Texts*. Hamburg: H. Buske Verlag.

HOBBS, J. (1985). *On the coherence and structure of discourse*. Technical Report CSLI-85-37, Center for the Study of Language and Information, Stanford, CA, USA

KOBSA, A. & WAHLSTER, W. (1988). *User Models in Dialog Systems*. Berlin: Springer-Verlag.

LITMAN, D. & ALLEN, J. (1987). A plan recognition model for subdialogues in conversation. *Cognitive Science*, **11,** 163–200.

MANN, W. C. & THOMPSON, S. A. (1987). *Rhetorical structure theory: a theory of text organization*. Technical Report ISI/RS-87-190, Information Sciences Institute, Los Angeles, CA, USA.

MAYBURY, M., Ed. (1993). *Intelligent Multimedia Interfaces*. Menlo Park, CA: AAAI Press.

MAYBURY, M. & WAHLSTER, W. Eds. (1988). *Readings in Intelligent User Interfaces*. Los Altos, CA: Morgan Kaufmann Publishers.

MC KEVITT, P. (1988). *Artificial Communicators: an operating system consultant*. Master's Thesis, Computer Science Department, Dept. 3CU, Box 30001, New Mexico State University, Las Cruces, NM 88003-0001, USA, March.

MC KEVITT, P. (1991*a*). Principles and practice in an operating system consultant. In D. PARTRIDGE, Ed. *Artificial Intelligence and Software Engineering, Vol.* 1, *Chapter on 'AI Mechanisms and Techniques in Practical Software'*, pp. 439–462. New York: Ablex Publishing Corporation.

MC KEVITT, P. (1991*b*). *Analysing coherence of intention in natural language dialogue*. Ph.D. Thesis, Department of Computer Science, University of Exeter, GBEX4 4PT, England.

MC KEVITT, P. (1995/1996). *Integration of Natural Language and Vision Processing, Vols. I–IV*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

MC KEVITT, P. & OGDEN, W. C. (1989). *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, P., PARTRIDGE, D. & WILKS, Y. (1992). Approaches to natural language discourse processing. *Artificial Intelligence Review*, **6,** 333–364.

MC KEVITT, P. & WILKS, Y. (1987). Transfer semantics in an operating system consultant: the formalization of actions involving object transfer. *Proceedings of the 10th International Joint Conference on Artificial Intelligence (IJCAI-*87), Vol. 1, pp. 569–575, Milan, Italy, August.

MOORE, R. C. (1980). *Reasoning about knowledge and action*. Technical Report No. 191, Artificial Intelligence Center, SRI International, Menlo Park, CA, USA.

MOORE, J. D. & PARIS, C. L. (1993). Planning text for advisory dialogues: capturing intentional and rhetorical information. *Computational Linguistics*, **19**, 651–694.

NEISSER, U. (1987). *Concepts and conceptual development: ecological and intellectual factors in categorisation*. Expanded versions of talks given at the *First Emory Cognition Project Conference*, 11–12 October 1984. Cambridge, England: Cambridge University Press.

NEUMANN, B. & NOVAK, H.-J. (1986). NAOS: Ein System zur naturalichsprachlichen Beschreibung zeitveranderlicher Szenen. *Informatik. Forschung und Entwicklung*, **1,** 83–92.

PARTRIDGE, D. (1991). *A New Guide to Artificial Intelligence*. Norwood, NJ: Ablex Publishing Corporation.

PEREIRA, F. & WARREN, D. (1980). Definition clause grammars for language analysis — a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, **13,** 231–278.

PERRAULT, R. (1990). An application of default logic to speech act theory. In P. R. COHEN, J. L. MORGAN & M. E. POLLACK, Eds. *Intentions in Communication*. Cambridge, MA: MIT Press.

PUSTEJOVSKY, J. (1987). An integrated theory of discourse analysis. In S. Nirenberg, Ed. *Machine Translation: Theoretical and Methodological Issues, Studies in Natural Language Processing*, pp. 168–191. Cambridge, UK: Cambridge University Press.

REICHMAN, R. (1985). *Getting Computers to Talk like You and Me*. Cambridge, MA: MIT Press.

RETZ-SCHMIDT, G. (1991). Recognizing intentions, interactions, and causes of plan failures. *User Modeling and User-Adapted Interaction*, **1,** 173–202.

RETZ-SCHMIDT, G. & TETZLAFF (1991). *Methods for the intentional description of image sequences*. Bereich No. 80, Universitat des Saarlandes, FB 14 Informatik IV, Im Stadtwald 15, D-6600, Saarbrucken 11, Germany, August.

SCHANK, R. C. (1972). Conceptual dependency: a theory of natural language understanding. *Cognitive Psychology*, **3,** 552–631.

SCHANK, R. C. (1973). Identification and conceptualizations underlying natural language. In R. SCHANK & K. KOLBY, Eds. *Computer Models of Thought and Language*. San Francisco, CA: Freeman and Co.

SCHANK, R. C. (1975). *Conceptual Information Processing*. Fundamental Studies in Computer Science, Vol. 3. Amsterdam: North-Holland.

SCHANK, R. C. & ABELSON, R. P. (1977). *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures*. Hillsdale, NJ: Lawrence Erlbaum Associates.

SEARLE, J. R. (1969). *Speech Acts: An Essay in the Philosophy of Language*. Cambridge, UK: Cambridge University Press.

SEARLE, J. R. (1983). *Intentionality*. Cambridge, UK: Cambridge University Press.

SIDNER, C. L. (1983). What the speaker means: the recognition of speakers' plans in discourse. *International Journal of Computers and Mathematics (Special Issue in Computational Linguistics)*, **9,** 71–82.

SIDNER, C. L. (1985). Plan parsing for intended response recognition in discourse. *Computational Intelligence*, **1,** 1–10.

SIDNER, C. L. (1994). An artificial discourse language for collaborative negotiation. *Proceedings of the* 12*th National Conference on Artificial Intelligence* (*AAAI*-94), Vol. 1, pp. 814–819, Seattle, Washington, August.

WAHLSTER, W. ANDRÈ, E., FINKLER, W., PROFITLICH, H. J. & RIST, T. (1991). Plan-based integration of natural language and graphics generation. *Artificial Intelligence* (*Special Issue on Natural Language Generation*), **63,** 387–427.

WEBBER, B. (1978). *A formal approach to discourse anaphora*. Doctoral Dissertation. Also in, BBN Report No. 3761, Bolt, Beranek and Newman, Cambridge, MA, USA.

WILENSKY, R. (1983). *Planning and Understanding*. Reading, MA: Addison-Wesley.

WILENSKY, R., CHIN, D. N., LURIA, M., MARTIN, J., MAYFIELD, J. & WU, D. (1988). The Berkeley UNIX Consultant project. *Computational Linguistics*, **14,** 35–84.

WILENSKY, R., MAYFIELD, J., ALBERT, A., CHIN, D., COX, C., LURIA, M., MARTIN, J. & WU, D. (1986). *UC—a progress report*. Report No. UCB/CSD 87/303, Computer Science Division (EECS), University of California, Berkeley, CA 94720, USA, July.

WILKS, Y. (1973). An artificial intelligence approach to machine translation. In R. SCHANK & K. KOLBY, Eds. *Computer Models of Thought and Language*. San Francisco, CA: Freeman and Co.

WILKS, Y. (1975*a*). Preference semantics. In E. Keenan, Ed. *Formal Semantics of Natural Language*. Cambridge, UK. Cambridge University Press. Also as, Memo AIM-206, Artificial Intelligence Laboratory, Stanford University, Stanford, CA, USA, July 1973.

WILKS, Y. (1975*b*). An intelligent analyzer and understander of English. *Communications of the ACM*, **18,** 264–274. Also in B. GROSZ, K. S. JONES & B. WEBBER, Eds. *Readings in Natural Language Processing*, pp. 193–204, Los Altos, CA: Morgan Kaufmann.

WILKS, Y. (1975*c*). A preferential, pattern-seeking semantics for natural language inference. *Artificial Intelligence*, **6,** 53–74.

Paper accepted for Publication by Associate Editor, Dr R. Agarwal

Appendix A: Experimental details

The goals of this experiment were two-fold. First, to demonstrate that the OSCON system can conduct intention analysis over a sample natural-language dialogue by the processes of intention and intention-sequence recognition and 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 has the full capability of intention sequence analysis (*full analysis*, experimental treatment). The second version has no capability of sequence analysis (*no analysis*, control treatment). The third version has the capability of recognizing sequences with only single intentions (*frequency-only*, second control).

Sample runs of the OSCON system demonstrating *full analysis*, *no analysis* and *frequency-only* analysis 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 five “real” types of intention, and nointentions. The “intention matrix” is a two-dimensional, 6 × 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 guery 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.*

```
→ 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 caused "dissatisfaction" to increase, whereas a "description" would have caused "satisfaction" to increase.

```
→ what is ada?

Ada is developed on behalf of the US 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 treatment (*no analysis*) 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.

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.

US 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 US 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 another control treatment (*frequency-only*) 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.

The user-modelling function is changed so that it compares intention frequency counts, or single-intention sequences, rather than intention-pair sequences.

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* sequence analysis *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 on 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 recognizes on elaboration intention. 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 US 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 responses for the complete dialogue except for the second 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.