Report Number: WUCS-92-21

1992

The DIM system: An Empirically Derived NLP System

Jennifer Balentine, Anne Johnstone, and David Mathias

Current natural language processing systems have a wide coverage of English, but are unforgiving of errors and generate numerous alternative but irrelevant analyses. These problems are exacerbated when speech is the input medium rather than text, because speech recognizers generate their own errors and ambiguities. These errors, together with the user's informality of speech, can lead to extremely large search spaces of possible interpretations. To add to the complexity, only very small subsets of English can be analyzed semantically. For our domain, which is very limited, our approach is to avoid detailed syntactic and semantic processing wherever possible. This... **Read complete abstract on page 2.**

The DIM system: An Empirically Derived NLP System

Jennifer Balentine, Anne Johnstone, and David Mathias

Complete Abstract:

Current natural language processing systems have a wide coverage of English, but are unforgiving of errors and generate numerous alternative but irrelevant analyses. These problems are exacerbated when speech is the input medium rather than text, because speech recognizers generate their own errors and ambiguities. These errors, together with the user's informality of speech, can lead to extremely large search spaces of possible interpretations. To add to the complexity, only very small subsets of English can be analyzed semantically. For our domain, which is very limited, our approach is to avoid detailed syntactic and semantic processing wherever possible. This is less error prone and increases the level of robustness. Rather than perform complete compositional analysis of utterances, we are using patternmatching techniques to get an interpretation of utterances by generating information from significant seeming pieces.

The DIM system :

An Empirically Derived NLP System

Jennifer Balentine, Anne Johnstone and David Mathias

WUCS-92-21

July 1, 1992

Department of Computer Science
Campus Box 1045
Washington University
One Brookings Drive
St. Louis, MO 63130.

Abstract

Current natural language processing systems have a wide coverage of English, but are unforgiving of errors and generate numerous alternative but irrelevant analyses. These problems are exacerbated when speech is the input medium rather than text, because speech recognizers generate their own errors and ambiguities. These errors, together with the user's informality of speech, can lead to extremely large search spaces of possible interpretations. To add to the complexity, only very small subsets of English can be analyzed semantically. For our domain, which is very limited, our approach is to avoid detailed syntactic and semantic processing wherever possible. This is less error prone and increases the level of robustness. Rather than perform complete compositional analysis of utterances, we are using patternmatching techniques to get an interpretation of utterances by generating information from significant seeming pieces.

Contents

1. Introduction

The dialogue project is a 2-3 year research effort with the goal of building a habitable speech-driven system that responds to a set of commands and questions in a particular domain. The hope is that the domain itself should lead to natural restrictions on the vocabulary and speech of the users. This rather than constraints imposed by the system designers would create the impression of habitability and an inherent robustness.

The overall aim of the DIalogue Manager (DIM) system is to interpret a user's spoken utterance with respect to various goals the user might be trying to achieve. The goals are limited to a certain domain, in the current case, the custom-calling features provided by Southwestern Bell Telephone. Once the goal has been identified the system must then formulate an appropriate response. This report discusses a first attempt to provide natural language processing (NLP) in the DIM system.

The role of the NLP system is:

- to identify the type of utterance, i.e. whether it is a question, a command, a statement, a request for help, etc,
- to extract as much of the propositional content of the utterance as is useful for making a response.

This information is then integrated into the context of the dialogue by the plan recognizer (PR). The PR component relates the current utterance to the user's overall goal as established during previous exchanges.

Input to the NLP component would normally be the word hypotheses output by a speech recognizer. When this work was begun a speech recognizer was not yet available. We decided, therefore, to use the text transcripts taken from the first Wizard of Oz experiment (Balentine, Berry, Johnstone 1992) as a starting point.

We will first discuss some of the theoretical issues in NLP with particular reference to processing spoken dialogues with "intelligent agents". We then report the results of analyzing the WOZ dialogues. Finally, we assess the prototype system which was built to test the rules developed through dialogue analysis.

2. Theoretical Issues in NLP

2.1. The TRAINS project

Our project bears many similarities to the TRAINS project (Allen & Schubert 1991), but is less ambitious in scope. We will briefly discuss the issues raised by researchers on the TRAINS project and then discuss our approach to those problems, comparing and contrasting, where appropriate the approach taken in TRAINS.
Allen & Schubert make the following observations on the state of the art in NLP:

2.1.1. Syntax

- Current systems have a wide coverage of English, but
- are unforgiving of errors, and
- generate numerous alternative but irrelevant analyses.

These problems are exacerbated when speech is the input medium rather than text, because speech recognizers generate their own errors and ambiguities. These errors, together with the user's informality of speech, can lead to extremely large search spaces of possible interpretations. The potential size of this search space can be reduced by restricting the coverage of English, i.e. using a limited vocabulary and a grammar that restricts the number of possible word choices at any particular point in the utterance.

2.1.2. Semantics

Allen and Schubert note that the situation in semantics is "much worse. Only *very* small subsets of English can be analyzed semantically". The reasons they cite are:

- expressive inadequacy. There are many common phenomena such as generalized quantifiers, tense, aspect, causal relations and so on, which current meaning representations cannot handle.
- the difficulty of mapping from syntax to semantics.
- poor understanding of the role of context.
- poor understanding of the role of inference and plan reasoning.

2.2. Comparison of TRAINS and DIM

The TRAIN's system requires detailed and accurate information about the propositional content and illocutionary force of an utterance. The researchers are therefore working to implement very sophisticated parsers and knowledge representation structures.

They plan to get round the problems of ungrammaticality mentioned above by employing an error-tolerant parser that uses elastic unification and preference-seeking techniques.

Given that our domain is so much less complex, at least at the level of NLP, our approach is to *avoid* detailed syntactic and semantic processing wherever possible. This leaves us less open to errors in analysis, and increases the level of robustness in the face of poor quality speech input. Rather than perform a complete compositional analysis of an utterance, we will try to cobble together an interpretation out of significant seeming pieces. The question is how little we can get away with and still capture the user's intention.

We also, for the first system, plan to collapse the distinction between syntax and semantics and go directly to a logical form based on *utterance act* types, such as requests for help, commands for a certain action, acknowledgements, etc. This may be possible because of the limited nature of the domain. Utterances are viewed as acts or parts of actions whose intended effect is to change the state of the hearer's beliefs such that some action (spoken and/or physical) is taken by the hearer. The number and type of actions in our domain is very limited, even though the number of ways of expressing that intention may be quite large.

3. The Data

3.1. Collection

The initial domain for our problem is custom-calling features, a service currently offered by SWBT using a touch-tone command interface. The features are call-forwarding, remote call-forwarding, speed-dialling and call-waiting. The purpose of the DIM system is to accept natural spoken commands to manage these calling features.

Since the existing system does not use natural language, simulations were required to provide dialogues for the DIM system development. The first experiment using such a simulation was designed to collect the full range of vocabulary and syntax from the users. Details of the experiment are given in (Balentine, Berry &

Johnstone 1992). The Wizard of Oz methodology (Kelley 1983) was employed. Subjects talked with a "wizard", via a phone line, who performed the speech recognition and NLP task and then selected a pre-recorded response. The goal of the experiment was to capture realistic conversations similar to those the user might have with an actual system. The experiment was carefully designed and run to maintain the illusion that the user was conversing with a machine. The dialogues that were collected were annotated and are given in Appendix A.1.

3.2. Examples

The dialogues range from the simple (Figure 1) in which the user accomplishes their task in one utterance, to the complex (Figure 2). The first dialogue starts with the system greeting. The user accomplishes their task using a simple command with common domain words. There is little superfluous information to be processed. This is an example of the simplest type of utterance and one that would pose few problems to a speech recognizer and NLP component. All necessary information is present in one command. The system confirms the user's request, and the conversation ends. The complex dialogue in Figure 2 states far more indirectly what features are needed. Much more sophisticated NLP would be needed to work out the relevance of "I call a number quite frequently", and of other utterances. The user also appears to be confused by the experimental scenario, and is really asking for clarification on that. These conversations represent the range of input to the NL processor.

```
S: Hello, this is South Western Bell's phone
service.
U: Cancel call-forwarding.
S: Call-forwarding is off.
```

Figure 1.

S: Hello, this is South Western Bell's phone service.
U: Yes, um I call a number quite frequently and need to know if um, speed-calling and or dialling by name or number is the same type of service?
S: You can dial by speaking a phone number, a speed dial code, or a speed dial name.
U: OK I would like to have this speed-call number set up by dialling by name. How can I do that?
S: With speed calling, you can reach important numbers more quickly and easily. You store a person's name and phone number under a one- or two-digit code. Then you can call using either the person's name or their code instead of their phone number.
U: OK so I would need to put that person's phone number in and then I can just list it by name, and that works the same way, correct? Or is it the same thing by (pause) mmh I'm confused

(system talks over user)

U: Can you
S: Please say what service you want.
U: I want dialling by name.
S: Whom shall I dial?
U: Mmm I want to dial Jane.
S: You want to dial Jane at 225-0099?
U: Correct
S: Now dialling Jane.
U: OK Mmm now does that mean my service is set up?
S: Yes
U: OK Thank you very much.

Figure 2

4. Data Analysis

4.1. Issues and aims

As mentioned above, the aim of the NLP is to provide a sufficiently detailed description of the input utterance to allow the plan recognizer to recognize the user's goal and formulate an appropriate response. We examined the dialogue transcripts in order to identify

- what the wizard had taken the user's intention to be, and
- what linguistic cues were used to signal the intention.

The goal was to come up with a set of rules, the left-hand side of which would be certain linguistic constructs, and the right-hand side would be the surface-form representing the structure of the utterance.

4.2. The left-hand side

4.2.1. keywords

We were concerned not to place too great a burden on the speech recognizer, so when going through the transcripts of the dialogues the analyst tried to pick out just the minimum information necessary to signal the user's intentions. We found that in the majority of cases simple key words and their synonyms such as *help, information, directions* or *dial, call, phone* were sufficient to build the appropriate surface description of the utterance act. It was still important to relate words in different parts of the utterance, however. For example, it is important to recognize that the user is asking for help on call-forwarding rather than to recognize call-forwarding by itself. These constraints were captured by the ordering of the process rules.

4.2.2. phrases

There is other important information in phrases that we would like to capture. Positional information provides clues to the high level structure of an utterance, i.e. declarative versus interrogative. For example: *I can* .. appears the same to our keyword pattern matcher as does *Can I* ... yet in the first utterance the speaker is asserting a fact while in the second the speaker is possibly asking a yes-no question. If we could match entire phrases rather than single words then we could capture the relevant positional information. That is, inherent in the phrase is some word order which, by recognizing the entire phrase, we can exploit to determine the relationship of certain words.

A second way in which we can use phrases is to identify standard meaningful locutions, such as *I need to know what..* without performing extensive and sophisticated syntactic and semantic analysis of its component parts. Such an analysis, as we discussed above, can be expensive and error-prone. A more useful approach, given our limited domain, might be a best-fit matching of inputs to stock phrases.

Linguistics has traditionally ignored phrases in discussions of the role of a lexicon in language. In "The Phrasal Lexicon", (Becker, 1975) proposes that ignoring the important role of phrases has seriously impaired the ability of linguists to account for the way in which humans deal with language. Becker outlines six major categories of phrases which will be briefly discussed here. The following definitions are taken from "The Phrasal Lexicon".

Polywords: Multi-word phrases admitting no variability; interchangeable with single words or phrases. Example
to blow up

Phrasal Constraints: Units consisting of a small number of words, some of which constrain the variability of others. Example
by pure coincidence

Deictic Locutions: Phrases with low variability; short to medium length. Example:
for that matter

Sentence Builders: Phrases up to sentence length, often containing slots for parameters or arguments. Example:
(person A) gave (person B) a (long) song and dance ...

Situational Utterance: Usually complete sentences with little variability. Example:
It only hurts when I laugh.

Verbatim Texts: Texts of any length memorized verbatim, or approximately so. Example:
Better late than never.

By studying the data collected during the experiments we found that the two classes of most concern to us were Deictic Locutions and Sentence Builders.

Deictic Locutions appeared quite frequently in the sample dialogues. Typical occurrences were:

Can you tell me if ...
I need to know what ...

Notice that each of these directs the course of the conversation and is short in length with surprisingly low variability. Occurring less

frequently, but frequently enough to merit attention, were Sentence Builders. These typically took a form similar to the following:

I want to call my (Mom). What is (her) (number).

In this initial study we concentrated on Deictic Locutions.

4.2.2.1. Types of phrases

Initially the intent of this work was to implement a phrase grammar that would allow greater recognition of utterances in the interactive phone services domain. No restrictions were to be placed on the types of utterances with which the grammar would aid the system. Theoretically this is sound because Becker makes no mention of applicability of phrases to only some utterance types. Empirically, however, we found that due to the limited domain, phrases appeared in a limited number of utterances and these were in similar utterance types. Specifically, the types of utterances in which relevant phrases appeared are:

• Yes-no questions: These require answers of yes or no. Examples:
Is call waiting on?
Is that the only name in my speed calling directory?
• Wh-questions: These require the referent of some pronoun or set. We define a non-specific noun as a noun that is not instantiated. Examples:
What are the services I can obtain?
Who is in my speed calling directory?

Since both of the utterance types in which phrases appear in the sample dialogs are questions, we implement a question-phrase grammar rather than a general phrase grammar.

An important policy issue presents itself at this point. If we make the grammar too specific then some questions may not be recognized as such and will be treated as requests. Conversely, if the grammar is not specific enough then it will likely recognize all questions as well as some utterances intended as requests. Due to the hierarchical construction of our system if a question is missed by the grammar it will not be ignored but will be processed by the pattern matcher. Although the user's question may not be answered in this case no harm will be done since any unintended command can be undone. We determined this to be preferable to erring in the other direction.

That is, we do not want to overly burden, and frustrate, the user by treating requests as questions, thus delaying them in accomplishing their task. Therefore, we have made the grammar as specific as possible. The entire grammar appears in the Appendix A.2.

This is an example of the different types of inferencing "buried" in the system. The assumption is made that most of the time the user want's a request to be carried out.

4.3. The right-hand side

4.3.1. Types of Logical Forms

The main intention behind an utterance can be classified according to the speech act that the utterance is said to perform. In most ordinary dialogues between people there is no one-to-one correspondence between utterances and speech acts. Often a speech act takes several utterances to be completely expressed.

An example of a warning over two utterances is:

> "The United Heat and Light Company is going out of business."
> "You have shares in that company."

Frequently the same utterance can be used to express different speech acts depending on the context. For example, the utterance *Do you know the time?* could be used as a request for information, an offer to tell the time, a yes-no question, or the beginning of an argument about punctuality.

In our domain we have come across the problem of ambiguity. For example, the phrase *call forwarding* could be a request for the call forwarding service, or a parameter to a previous request for help. Similarly, a number could be a command to dial that number, or a parameter to a previous request for call-forwarding. We have addressed the problem of ambiguity by outputting the surface-form for the utterance and letting the plan recognizer decide on the basis of the discourse context what was the intended meaning behind the utterance.

We are following Allen and Schubert (1991) in referring to utterances as utterance acts, which can initiate, continue or correct what has been said so far. A speech act is made up of one or more utterance acts.

In the prototype system, utterances were mapped into one of seven forms. These are an initial attempt at defining surface structures of utterances. We may modify these in the near future.

- **surface-request** -- the speaker intends the system to perform some action. This is signalled by use of key words such as *please.*, and absence of anything that suggests a question.
- **surface-inform** -- the speaker intends to make a claim about the world. A frequent surface-inform which would be reinterpreted as a request in context is a phrase of the form: *I want call-forwarding cancelled.*
- **surface-yn-question** -- the speaker asks a yes-no question to the hearer. For example *Is call waiting on?* or *Can you forward my calls?*
- **surface-wh-question** -- the speaker intends the hearer to respond with some information. These include questions like *How can I forward my calls?* or *What are the entries in my speed call directory?*
- **surface-informif** -- the speaker informs the hearer about the truth or falsity of some proposition. Utterances like *Yes* and *No, I don't want help* would fall into this category.
- **surface-informref** -- the speaker supplies some information to the hearer. An example would be *The number is 534-6749.*
- **surface-acknowledge** -- the speaker acknowledges his or her agreement that the system has properly carried out a major goal. Key words such as *thank you*, or *goodbye*, signalling the end of the conversation are translated to surface-acknowledge.

The form of each speech act is similar:

surface-speech-act-type(u,s, action-type)

For example surface-request(u,s,repeat) is a request from the user to the system for the action *repeat.*

5. The Natural Language Processing Architecture

The natural language processor consists of four components: the question grammar, the phrase grammar, the patternmatcher, and the lexicon. Input to the system is currently the list of written words

taken from the experimental transcripts. The output is one of the surface-forms given above.

The patternmatcher does the primary processing for the system. It consists of a hierarchy of rules. These rules search for specific words in order of their precedence. Words of the highest precedence are searched for initially. This establishes the major propositional content of the utterance. Then other words, which supply information are looked for. If something expected is missing, it is recorded in the surface-form as an uninstantiated variable.

The lexicon serves as the domain dependent vocabulary of the processor. It contains a dictionary of synonyms, as well as application-dependent information. The lexicon currently contains 57 words and synonyms.

The question grammar is a preprocessor for the patternmatcher. It parses each utterance, determining if it is a question. If it is a question, the question grammar determines what type of question it is. Utterances which are not recognized as questions are sent to the patternmatcher for processing. Otherwise, the question grammar outputs a surface-form, and the patternmatcher is bypassed.

The phrase grammar is similar to the question grammar. It serves as a preprocessor for the patternmatcher. Unlike the question grammar it rarely bypasses the patternmatcher, even when it successfully finds a phrase. Instead, the information about the phrase is passed to the patternmatcher to aid its processing.

6. Implementation

6.1. Question Grammar

When one uses a grammar it is typically to recognize an entire string (or, in our domain, utterance). If the input is exhausted at the same time that the grammar has completed a derivation (expanding all non-terminal characters) then the string is accepted. In the case of a grammar to recognize phrases, however, this will not work because it is not our intention to recognize complete utterances. Therefore, when our grammar has entered an accepting state (to use the automaton term) some of the utterance is likely to remain. To deal with this, on a theoretical level, we introduce a non-terminal called "Anything" which leads to any word that might possible appear in an utterance, and is, in all cases, right-recursive. Thus, if we wish to recognize the phrase "Can you tell me if" which indicates that the utterance is a yes-question , we invoke a production of the form:

- YN-question --> Modal Secondprsn Inform Firstprsn if Anything

where non-terminals have initial capitals and terminals are all lower case. The first five elements of the right side properly parse our phrase and then "anything" will consume the remainder of the utterance.

The method described above is fine in theory. In practice, however, there is a problem: it is very difficult to create productions that will allow for every word that might possibly appear in an utterance. This is true even though, in this case, we are not concerned with word order (by the time we get to "Anything" we just want to consume the remaining input). Obviously, the problem comes in enumerating all of the possible words. Some measure of help comes from the fact that the current speech recognizers have vocabularies of only about one thousand words. While it would not be pleasant, certainly we could enumerate the entire vocabulary of such a system. However, the vocabularies of these systems are sure to grow dooming us to forever increase the number of productions in the grammar without increasing its capabilities. Another, more subtle but perhaps more troubling, problem is that of recognizing proper nouns -- specifically people's names. There is simply too much variability.

There is a solution to this problem. Prolog allows its grammars to operate on a structure called a difference list. A difference list is of the form ([list1], [list2]) where [list1] is assumed to consist of [list2] appended to some [list1a]. It is [list1a] that the grammar is to recognize. An example from the domain is:

([i,need,to,know,if,call,waiting,is,on],[call,waiting,is,on])

Here [list1a] is [i,need,to,know,if] which is the relevant phrase.

This raises another implementation issue: How do we properly divide the utterance into the appropriate sublists? Clearly, we do not know *a priori* where to make the division. Once again, Prolog provides a solution. Prolog allows us to iteratively divide the list until we obtain a successful parse or have tried all possible divisions of the list. There is a problem with this approach, however; Prolog begins dividing the list at the beginning and moves toward the end one word at a time. Thus, the first meaningful attempt at a parse of our example (after the null list[]) would be:

([i,need,to,know,if,call,waiting,is,on],[need,to,know,if,call,waiting,is,on])

It is not difficult to imagine an instance where this might allow a "sub-phrase" to be recognize, rather than the full phrase present in

the utterance, and thus, cause mis-identification of the phrase type. To prevent this we want to attempt to match the longest part of the utterance possible, In other words, we want to first try to parse the entire utterance and then if unsuccessful, iteratively remove one word from the end of the utterance and attempt to parse again. This method guarantees that the first successful parse is also the longest phrase we can recognize in the utterance. We are able to do this by reversing the utterance, then dividing it and re-reversing the sublists.

The final implementation issue we will discuss is that of "um words". "Um words" are words such as: um, uhh, mmm, ok, well, etc. These are quite common in the collected dialogs and in spoken conversation in general. To deal with these we create a non-terminal in the grammar, similar to Anything, called Um. This does not involve the problems encountered with Anything because there are a small number of "um words". We then place this non-terminal between adjacent symbols in the right sides of productions.

The grammar determines if the utterance is a yes-no-question or a wh-question (or neither) but we must still examine the utterance to determine the information requested. Recall, however that this is precisely the kind of task at which pattern matchers are quite good. Therefore after determining the utterance type we pass it to a pattern matcher. If it is neither a yes-no-question or a wh-question the utterance "falls through" to the part of the pattern matching system discussed earlier. If, however, the utterance is determined to be a question, it is passed to a part of the pattern matcher designed specifically to handle questions. This is necessary because the logical form output by the system is significantly different for questions than it is for other utterance types. Since these logical forms are "hard-coded" we must create a separate pattern matching sub-system. The operation of this sub-system is identical, conceptually, to that discussed in the first sections of this paper.

Notice that the grammar sits atop the pattern matcher. If the input is successfully parsed the it is sent to the pattern matcher flagged as a question. Otherwise, the input is sent to the pattern matcher and will be treated just as it would have been before creation of the grammar.

6.2. Pattern Matcher

The patternmatcher was implemented using a hierarchy of rules. The patternmatcher receives an utterance as input if it has

unsuccessfully passed through the question grammar. Once in the patternmatcher, the utterance is scanned for the presence of specific words based on a hierarchy. The rules of process implement the hierarchy. (see Appendix A.2)

Each rule of process searches for a word or the synonyms of that word. This search is implemented by the rules of present. In this domain, the words of highest precedence are those associated with *help*. If one of these words is found in the utterance, the next step is to look for the presence of a feature word (i.e. *forwarding, waiting, dialing,* etc). If a feature word can be found, then a logical form for the action help is outputted, including the feature for which the user wants help. If a feature word cannot be found, a logical form for the action help is outputted without a feature word included. All the rules of process are similar to the first one. Each rule searches for a word, and then looks for the parameters of that action in the utterance. Process instantiates a logical form based on the words found in the utterance. This surface form is the output of the natural language processor.

6.3. Example Output

Let us take the input "call forwarding help" as an example. This input is processed using the algorithm described above. The first step is to look for a word of highest precedence, i.e. *help*. This word is present, so the utterance is then searched for a feature word. Forwarding is a synonym for a feature word. A complete logical form can be outputted:

Output = surface-request(u, s, help(s, u, forwarding))

This indicates a request, from the user to the system, that the system provide help on call-forwarding to the user.

An example of a more complex utterance can also be translated. For example:

yes I would like my home phone call forwarded to a neighbour and I need some directions on how to do that:

is translated to:

Output = surface-inform(u, s, want(u, help(s, u, forwarding)))

The presence of the word *need* indicates that the logical form is a surface-inform. *Directions* is a synonym for *help*, and finally *forwarded* provides the custom-calling feature word.

7. Testing

7.1. Method

For initial testing, we used a large and representative sample of utterances from the WOZ dialogues. These were run through the NLP system, and checked to see if the representation of the intention was appropriate. An example output file is given in Appendix A.3. In that batch, out of 33 input utterances all but 2 received an appropriate interpretation. The two which failed were:

Ok, does that mean my service is set up?

which was interpreted as a surface-informif because of the keyword ok, and

Hi, I forgot my mothers number and I would like to call her.

which was interpreted as :

surface-inform(u,s,want(u,dial(s,u,?number))).

that is, the patternmatcher could not resolve the referrent of the pronoun *her* in the utterance.

The question of how to evaluate a dialogue system is not a simple one. We are currently designing evaluation methods. These will be applied to the system as a whole, NLP and plan recognition.

7.2. Problems Encountered

During analysis of the dialogues some especially difficult problems arose. We will describe our attempts at solutions, and those that so far remain without a solution.

First, the same word may be used in different senses depending on the context. For example, the word *remove* was used in two different senses in association with two features: call-forwarding and speed-dialing. Users might request the removal of call-forwarding indicating they wanted it canceled. Alternatively, users might request that a user's name be *removed* from their speed dialing directory.

We solved this problem by looking for feature words in the utterance that would be consistent with canceling a service. Both call forwarding and call waiting can be cancelled. If one of these feature words is found in the utterance, we assume the user is using *remove* as a synonym for *cancel*. If neither of these feature words can be found, we assume *remove* is used as a synonym for *delete*. This has the effect of adding selectional restrictions on the use of words. A more principled approach would link the two meanings of the words *remove* and allow the appropriate one to be used in the right context.

A second potential difficulty is that our current approach does not allow users to request multiple actions in one utterance. Only one of the requests will be processed. This is not a problem with the current dialogues as the experimental scenarios focused on one goal at a time. We do, however, allow users to request help with multiple features in one utterance. For example:

I need directions on how to delete a name from speed dial and dialling by name.

is translated into the logical form:

surface-request(u, s, help(s, u, delete, dial))

This logical form indicates that the user wants help with two calling features.

Another ambiguity in the dialogue was mentioned earlier. The utterance, 224-0099 could be a request to dial the phone number or could be a response to a previous question. For instance consider the following dialogue:

System: Please say what service you want.
User: Call forwarding
System: What number do you want your calls forwarded to?
User: 224-0099

The natural language processor has no contextual information and it is therefore impossible to resolve this ambiguity, and correctly map it to a single logical form. This raises issues of how "bottom-up" and top-down" information should be integrated. Suppose the system has just asked a question that expects a number as a reply, at what level should this expectation be used to filter hypotheses about the input?

Our current solution is to map the utterance into a request to dial form. The plan recognizer can then use contextual information to decide whether the phone number is in fact the instantiation of a parameter rather than a request for dialling. We are currently revising our definitions of the utterance act and speech act types.

The system is very robust, and generally selects an appropriate response. However, it is not very sensitive either to similarities or differences in the input. For example:

I need to know if speed-calling and dialling ... and
I would like to have this speed-call number set up. How ...?

are interpreted differently, although they have a similar form. One of the ways in which we need to test the system is to check exactly which keywords and phrases are responsible for the interpretation, so that we can see if they are being used appropriately.

A related problem is that a great deal of default reasoning and hidden inferencing is buried in the system and needs to be made explicit even if it used in a "compiled" form in the actual implementation.

7.3. Unresolved Problems

Several potential problems occurred during dialogue analysis that we couldn't see an immediate solution to. One of the primary problems was the use of *from* and *to*. These words frequently precede phone numbers. The use of *from* and *to* is critical in remote call forwarding. Users must specify what number or speed call name they are forwarding their calls from as well as to. Unfortunately, it is probable that current speech recognition technology will be unable to reliably recognize *from* and *to* in an utterance. We will see how bad this problem really is when we test with the speech recognizer.

Ideally we need other means of recognizing these important pieces of information. We considered using positional information, but users did not exhibit a predictable pattern of presenting the *to* and *from* locations. It is possible that the system will have to ask for clarification on all remote call forwarding requests.

A similar problem is locating a name in an utterance, though this problem resides more properly in the speech recognizer. When deleting form the speed call directory or dialing a speed call entry, words in the utterance can be compared to speed call directory until a match is found. This is not the case if a user wants to add a name to

the directory or change the entries in the directory. The only solution to this problem may be to recognize that the user is trying to do some editing of the directory and then lead the user through a dialogue designed to capture the name.

One further problem is a user's faulty mental model. When users do not clearly understand the task they are trying to accomplish, expressing a request is difficult. For example:

> *trying to use a modem to place a call. What service would you recommend?*

The user needs to cancel call waiting. His utterance seems to indicate that he does not completely understand the service. If users don't have a correct model of a service, it is unlikely that they will use the vocabulary and structure that the system is set up to expect. One possible solution to this problem is user training. We would also like to be able to detect situations when an operator should be brought on-line.

8. Conclusions

The natural language processor is able to correctly map most of the utterances into logical forms. A rough estimate of the percentage of utterances it can handle is 75-95 per cent.

An advantage of the pattern matcher is the simple techniques it employs. They are easy to understand, and modify. In addition, vocabulary is added and deleted easily from the lexicon.

A major disadvantage of the system is its dependence on specific words. Current speech recognition technology is not capable of capturing each word in an utterance accurately. If a word of high precedence is missed, the whole utterance may be inaccurately or incompletely interpreted. Foregoing the use of fuller syntactic and semantic constraints makes it much harder to notice when the speech recognizer has got it wrong as in the following utterance:

I want twelve on call-forwarding.

Our current methods would interpret this as a command to forward calls, rather than realizing *twelve* must have meant something more like *help*. As with all realistic systems, we need to find a workable balance between robustness and sensitivity.

One great advantage of the system overall, is that it is possible to have clarification dialogues with the user. Information which is ambiguous or incomplete can be clarified.

9. Future Work

There are several ways in which we could extend our current work. Some minor changes such as using a pre-processor to eliminate "um-words" would greatly speed up the parsing process. We also want to explore the pros and cons of using traditional grammars versus "stretchy" templates for phrases. The phrase recognizer, question grammar and pattern matcher could be much better integrated than at present.

The most important and pressing task is to do a "rational reconstruction" of the current prototype and tease out the different types of knowledge and reasoning being used. In particular we need a more principled account of how the various utterance acts are identified from the surface characteristics of the input utterance.

We plan to start working with the actual output of a speech recognizer in the very near future. The system outputs a string of words. We could either run the NLP component on that, just as we have been using the text string, or we could rewrite the grammar so that it could be used during recognition.

Our current plans therefore are:

- to integrate a cleaner version of the current system with the TI speech recognizer,
- to perform a more rigorous analysis of the syntactic, semantic and discourse structures required in this domain.

APPENDIX A.1

Annotated Dialogues

Call One

S: Hello, this is South Western Bell's phone service.
U: Call Forwarding <REQUEST>
(pause)
U: Help. <REQUEST>
S You want your calls forwarded from 334-0561 to 224-0099? <QUESTION>
U: Yes <ACCEPT>
S: You want your calls forwarded to 224-0099? <QUESTION>
U: Yes <ACCEPT>
S: Calls are being forwarded to 224-0099 <ACCEPT>
(pause)
(hang-up)

Call Two

S: Hello, this is South Western Bell's phone service.
U: *Dial Mom*. <REQUEST>
S Now dialing Mother <ACCEPT>

Call Three

S: Hello, this is South Western Bell's phone service.
U: *Cancel Call-Forwarding* <REQUEST>
S: Call-Forwarding is off <ACCEPT>

Call Four

S: Hello, this is South Western Bell's phone service.
U: *Stop Call-Forwarding* <REQUEST>
S: Call-Forwarding is off <ACCEPT>

Call Five

S: Hello, this is South Western Bell's phone service.
U: *Call my mother*. <REQUEST>
S: Now dialing Mother <ACCEPT>

Call Six
S: Hello, this is South Western Bell's phone service.
U: Um, lets see. *Call forward* <REQUEST>
S: What number do you want your calls forwarded to? <QUESTION>
U: *224-0099* <INFORM>
S: Calls are being forwarded to 224-0099 <ACCEPT>

Call One

S: Hello, this is South Western Bell's phone service.
U: Yes, I would like my home phone *call-forwarded* <REQUEST>
to a neighbor (medium pause) <RELEASE TURN>
and I need some directions on how to do that.

S: What number do you want your calls forwarded to? <QUESTION>
U: Um the phone number is *
S: Calls are forwarded to 224-0099. <ACCEPT>
U: That is *correct* (long pause) <ACCEPT>
Do you take care of this for me? Do I need *confirmation*?
<QUESTION>
S: *No* <INFORM>
U: OK *Thank you* <ACCEPT>
S: Thank you for participating in this experiment. Please hang up.

Call Two

S: Hello, this is South Western Bell's phone service.
U: Yes my calls are being *forwarded* to a friend's home and I'd like to *cancel* that so I'm receiving the calls. <REQUEST>
S: You want call-forwarding off? <QUESTION>
U: *Correct* <ACCEPT>
S: Call forwarding is off. <ACCEPT>
U: OK Thank you very much.

Call Three

S: Hello, this is South Western Bell's phone service.
U: Yes, um I call a number quite frequently and need to know if um, speed-calling and or *dialing by name or number* is the same type of service? <QUESTION>
S: You can dial by speaking a phone number, a speed dial code, or a speed dial name. <INFORM>
U: OK I would like to have this *speed-call* number set up by dialing by name. How can I do that? <QUESTION>
S: With Speed Calling, you can reach important numbers more quickly and easily. You store a person's name and phone number under a-one or two-digit code. Then you can call using either the person's name or their code instead of their phone number. <INFORM>
U: OK so I would need to put that person's phone number in and then I can just list it by name, and and that works the same way, correct? Or or is it the same thing by ... mmh I'm confused <RELEASE TURN>
(system talks over user)
U: Can you
S: Please say what service you want. <REQUEST>
U: I want *dialing by name* <INFORM>
S: Whom shall I dial? <QUESTION>
U: Mmm I want to dial *Jane*. <INFORM>
S: You want to dial Jane at 225-0099? <QUESTION>
U: *Correct*. <INFORM>
S: Now dialing Jane <ACCEPT>
U: OK Mmm Now does that mean my service is set up? <QUESTION>
S: Yes <INFORM>
U: OK *Thank you* very much. <ACCEPT>

Call Four

S: Hello, this is South Western Bell's phone service.
U: Yes, I would like to have um my phone set up that I can call my friend *Jane* by *dialing* her name. <REQUEST>
S: You want to dial Jane at 224-0099? <QUESTION>

U: *Yes* I do <INFORM>
S: Now dialing Jane <ACCEPT>
U: OK *Thank you* <ACCEPT>

Call Five

S: Hello, this is South Western Bell's phone service.
U: Yes um I need to *cancel* my *call-forwarding* um to, which is at a friend's house and I just need to cancel that. <REQUEST>
S: Which service do you want off? <QUESTION>
U: *Call- forwarding* <INFORM>
S: Call- forwarding is off <ACCEPT>
U: *Thank you* <ACCEPT>

Call Six

S: Hello, this is South Western Bell's phone service.
U: Yes, I need my *calls forwarded to 224-0099* <REQUEST>
S: Calls are forwarded to 224-0099 <ACCEPT>
U: *Thank you* very much. <ACCEPT>

Call One

S: Hello, this is Southwestern Bell's phone service.
(pause) <RELEASE TURN>
S: Do you need help? <QUESTION>
U: *Yes* <INFORM>
(pause)
S: Please say what service you want. <REQUEST>
U: *Call Forwarding* <INFORM>
S: Call Forwarding, Your phone automatically forwards all calls to a location of your choice. <INFORM>
U: *Call Forwarding* <REQUEST>
S:What number do you want your calls forwarded to? <QUESTION>
U: *224-0099* <INFORM>
S: Calls are forwarded to 224-0099 <ACCEPT>
S: I'm still listening. So you need more information? <QUESTION>
U: *no* <INFORM>
(pause)
S: Thank you for your participation. Please hang-up now.

Call Two

S: Hello, this is Southwestern Bell's phone service.
U: *224-0099* <REQUEST>
S: Now Dialing 224-0099 <ACCEPT>
(long pause)
(hang-up)

Call Three

S: Hello, this is Southwestern Bell's phone service.
U: *Call Waiting* <REQUEST>
(pause)
S: You want Call Waiting off? <QUESTION>
U: *yes* <INFORM>

S: Call Waiting is off. <ACCEPT>
(hang-up)

Call Four
S: Hello, this is Southwestern Bell's phone service.
U: *Call Waiting* <REQUEST>
S: Call Waiting is off. <ACCEPT>
(Hang-up)

Call FIve
S: Hello, this is Southwestern Bell's phone service.
U: *224-0099* <REQUEST>
S: You want to dial 224-0099? <QUESTION>
U: *Yes*. <INFORM>
S: Now Dialing 224-0099. <ACCEPT>
(long pause)
(hang-up)

Call Six

S: Hello, this is Southwestern Bell's phone service.
U: *Call Forwarding* <REQUEST>
S: What number do you want your calls forwarded to? <QUESTION>
U: *224-0099* <INFORM>
S: Calls are forwarded to 224-0099. <ACCEPT>
(hang-up)

Call One

S: Hello, this is Southwestern Bell's phone service.
(pause)
U: Hello <RELEASE TURN>
S: Do you need help? <QUESTION>
U: *no* <INFORM>
(pause) <RELEASE TURN>
S: Please say what service you want. <QUESTION>
U: *Call Forwarding*. <INFORM>
S: What number do you want your calls forwarded to? <QUESTION>
U: *224-0099* <INFORM>
S: Calls are forwarded to 224-0099 <ACCEPT>
U: *Thank you*. <ACCEPT>
S: Thank you.
(hang-up)

Call Two

S: Hello, this is Southwestern Bell's phone service.
U: Hello (pause) I need to *call* my *mother*. <REQUEST>
S: Now dialing Mother. <ACCEPT>
(pause)
U: *Thank you*. <ACCEPT>
S: Thank you.
(hang-up)

Call Three

S: Hello, this is Southwestern Bell's phone service.
U: Hello, uhh, I need to *turn off call waiting*. <REQUEST>
S: You want call waiting off? <QUESTION>
U: *yes* <INFORM>
S: Call Waiting is off. <ACCEPT>
U: *Thank you* <ACCEPT>
S: Thank you.
(hang-up)

Call Four

S: Hello, this is Southwestern Bell's phone service.
U: Hi, I need to *turn off* my *call waiting* while I make a long distance call.
<REQUEST>
S: What service do you want off? <QUESTION>
U: *Call Waiting*. <INFORM>
S: Call Waiting is off. <ACCEPT>
U: *Thank you*. <ACCEPT>
S: Thank you.
(hang-up)

Call Five

S: Hello, this is Southwestern Bell's phone service.
U: Hi , I forgot my mother's number. I would like to call her.
S: Please rephrase the command. <REQUEST>
U: Hi, I want to *call* my *mother*. <REQUEST>
S: Now dialing mother. <ACCEPT>
U: *Thank you*. <ACCEPT>
(hang-up)

Call Six

S: Hello, this is Southwestern Bell's phone service.
U: Hi, I want to *forward* all my calls to another number. <REQUEST>
S: What number do you want your calls forwarded to? <QUESTION>
U: *224-0099*. <INFORM>
S: Calls are forwarded to 224-0099. <ACCEPT>
U: *Thank you*. <ACCEPT>
(hang-up)

Call One

S: Hello, this is Southwestern Bell's phone service.
U: Hello, hold on . The number where I am staying is 224, I don't care.
S: Do you need help? <QUESTION>
U: *Yes* <INFORM>
(pause)
U: Hey, Jennifer What is going on here, hold on... mumble
S: Please say what service you want. <REQUEST>
U: mumble
U: *Call Forwarding* please <REQUEST>
S: What number do you want your calls forwarded to? <QUESTION>
U: *224-0099* <INFORM>

S: Calls are forwarded to 224-0099 <ACCEPT>
(pause)
U: *Thank you* <ACCEPT>
(pause)
(Hang-up)

Call Two

S: Hello, this is Southwestern Bell's phone service.
U: Thu, let's see... Hi, I would like to get a movie theatre number please.
U: mumble, I don't get this one.
S: Please say what service you want. <REQUEST>
(pause)
U: mumble ... *dialing by name or number*. <REQUEST>
S: Whom shall I dial? <QUESTION>
U: mumble
U: I'm confused. I need *help* please. <REQUEST>
S: You can dial by speaking a phone number, a speed dial code,
or a speed dial name. <INFORM>
U: Please *dial 224-0099*. <REQUEST>
S: Now dialing 224-0099. <ACCEPT>
(pause)
S: Thank you
U: Okay, now do I hang-up
(pause)
(hang-up)

Call Three

S: Hello, this is Southwestern Bell's phone service.
U: I would like to, uhh, let's see, uhh, mumble...
I'm trying to use a modem. Unn, let's see what service do I need.
S: Please say what service you want. <REQUEST>
U: Tru, tru, tru, uhh mumble
U: *Dial by name or number* <REQUEST>
S: Whom shall I dial? <QUESTION>
U: Is there a *help* available? <QUESTION>
S: *yes* <INFORM>
U: Please give me *help* information. <REQUEST>
S: You can dial by speaking a phone number, a speed dial code,
or a speed dial name. <INFORM>
U: I'm trying to use a computer modem to place a call.
What service would you recommend?
S: Please rephrase the command. <REQUEST>
U: Would you repeat that please? <REQUEST>
S: Please rephrase the command. <REQUEST>
U: Uhh, I would like to use the computer modem to place a call.
Which features should I use other than dialing by name.
(long pause) <RELEASE TURN>
S: I'm sorry. Let me get someone who can help you. Please wait.
U: Hey, that was good.
S: Thank you for participating in this experiment. Please hang-up now.
U: Please ..mumble
(Hang-up)

Call Four

S: Hello, this is Southwestern Bell's phone service.
U: Umm, I would like to make a call from a computer modem.
So you have any features available?
U: mumble
S: Please say what service you want. <REQUEST>
U: I don't know..
I'm not really sure. Is there a help available?
U: Please give me *help*. <REQUEST>
S: Call Forwarding, Your phone automatically forwards
all calls to a location of your choice. <INFORM>
Speed Calling, You can reach important numbers more quickly and easily.
You store a person's name and phone number under a one digit code.
The you can call using either the person's name or their code
instead of their phone number.
You can find out what speed codes you have available,
or whose number you have stored.
Cancel Call Waiting stops important calls from being interrupted.
Before you dial ask for call waiting to be cancelled.
You can dial by speaking a phone number, a speed dial code,
or a speed dial name.
(pause)
U: Umm, I would like to have *call waiting* service, please. <REQUEST>
S: You want call forwarding off? <QUESTION>
U: *yes* <INFORM>
S: Call forwarding is off. <ACCEPT>
U: What about *call waiting*? <QUESTION>
S: You want call waiting off? <QUESTION>
U: *yes* <INFORM>
S: call waiting is off. <ACCEPT>
U: Now, please,uhh, umm, *dial* the number. <REQUEST>
(pause)
S: Whom shall I dial? <QUESTION>
(pause)
U: No number is available. <INFORM>
(hang-up)

Call Five

S: Hello, this is Southwestern Bell's phone service.
U: mumble
U: I would like to have a *dialing by name or number* feature please. <REQUEST>
S: Whom shall I dial? <QUESTION>
U: Umm, please *dial 224-0099*. <REQUEST>
S: Now dialing 224-0099. <ACCEPT>
(pause)
S: Thank you.
(hang-up)

Call Six

S: Hello, this is Southwestern Bell's phone service.
U: Umm, Please give me *help*. <REQUEST>
S: Please say what service you want. <REQUEST>
U: May I have *help* on the feat, uhh, features *call forwarding

and remote call forwarding*, please? <REQUEST>
S: Call Forwarding, your phone automatically forwards all calls to a location of your choice. <INFORM>
U: *Help* on *remote call forwarding*, please. <REQUEST>
S: Call Forwarding, your phone automatically forwards all calls to a location of your choice. <INFORM>

Call One

S: Hello, this is Southwestern Bell's phone service.
(long pause)
U: I want to be reached this evening at 224-0099. Thank you.
(hang-up)

Call Two

S: Hello, this is Southwestern Bell's phone service.
U: Hi, I want to call my mother but I can't remember the code I used. Thank you.
(hang-up)

Note: The subject did not appear to understand the interactive ability of the system. She was retrained and began the task over again.

Call One

S: Hello, this is Southwestern Bell's phone service.
U: Yes, I want my *calls forwarded* from 5 to 7, uhh, to the number of *224-0099*. Thank you. <REQUEST>
S: Calls are forwarded to 224-0099. <ACCEPT>
U: *Thank you*, *goodbye* <ACCEPT>
(Hang-up)

Call Two

S: Hello, this is Southwestern Bell's phone service.
U: Yes, I want to *call* my *mother*. I've stored her number, but I can't remember the code. <REQUEST>
S: Whom shall I call? <QUESTION>
U: My *mom*. <INFORM>
S: Now dialing mother. <ACCEPT>
U: Thank you. <ACCEPT>
S: Thank you.
(hang-up)

Call Three

S: Hello, this is Southwestern Bell's phone service.
U: Yes, please *stop forwarding* my calls. I'm now at home. Thank you. <REQUEST>
S: You want call forwarding off? <QUESTION>
U: *Yes*, please. Thank You. <INFORM>
S: Call forwarding is off. <ACCEPT>
U: *Thank you*, *goodbye* <ACCEPT>
S: Thank you

(hang-up)

Call Four

S: Hello, this is Southwestern Bell's phone service.
U: Hi, I'm at home. You can *stop forwarding* my calls. Thank you. <REQUEST>
S: Call forwarding is off. <ACCEPT>
U: *Thank you* <ACCEPT>
S: Call forwarding is off.
(hang-up)

Call Five

S: Hello, this is Southwestern Bell's phone service.
U: Hi, I want to call my *mom*. What is her *code*? <QUESTION>
(pause)
U: What is the code..(system interrupts)
S: Stored names are mother at 224-0099 under speed calling code 1. <INFORM>
U: *okay, thank you*. <ACCEPT>
(hang-up)

Call Six

S: Hello, this is Southwestern Bell's phone service.
U: Please *forward* my calls from 7 to 9 to *224-0099*. *Repeat* please. <REQUEST>
S: Calls are forwarded to 224-0099. <ACCEPT>
U: *Thank you*. <ACCEPT>
(Hang-up)

Call One

S: Hello, this is Southwestern Bell's phone service.
S: Please say what service you want. <REQUEST>
U: Remote, I'm sorry, *cancel call forwarding*. <REQUEST>
S: To what number do you want your calls forwarded? <QUESTION>
U: *224-0099*. <INFORM>
S: Calls are forwarded to 224-0099. <ACCEPT>
(hang-up)

Call Two

S: Hello, this is Southwestern Bell's phone service.
U: I'd like to access my *speed calling dial by name*. <REQUEST>
S: Whom shall I dial? <QUESTION>
U: *Mother* <INFORM>
S: Now dialing mother. <ACCEPT>
(pause)
(hang-up)

Call Three

S: Hello, this is Southwestern Bell's phone service.

U: I'd like to access *cancel call waiting and dial by name*. <REQUEST>
S: You want call waiting off? <QUESTION>
U: *Yes*. <INFORM>
S: Call waiting is off. <ACCEPT>
U: I'd like to access *dial by name*. <REQUEST>
S: Whom shall I dial? <QUESTION>
U:My *daughter*. <INFORM>
S:Now dialing Jane. <ACCEPT>
(pause)
(hang-up)

Call Four

S: Hello, this is Southwestern Bell's phone service.
U: I'd like to, uhh, *cancel call waiting and dial by name*. <REQUEST>
S: Call waiting is off <ACCEPT>
U: I'd like to *dial by name*. <REQUEST>
S: Whom shall I dial? <QUESTION>
U: Uhh, my daughter *Christine*. <INFORM>
(long pause)
S: I'm sorry. Let me get someone who can help you.
Thank you for participating in this experiement. Please hang-up now.
(hang-up)

Call Five

S: Hello, this is Southwestern Bell's phone service.
U: *Dial by name*, please <REQUEST>
S: Whom shall I dial? <QUESTION>
U: *Mother* <INFORM>
S: Now dialing mother. <ACCEPT>
(hang-up)

Call Six

S: Hello, this is Southwestern Bell's phone service.
U: *Call forwarding* <REQUEST>
S: To what number do you want your calls forwarded? <QUESTION>
U: *224-0099*. <INFORM>
S: Calls are forwarded to 224-0099. <ACCEPT>
(hang-up)

Call One

S: Hello, this is Southwestern Bell's phone service.
(pause) <RELEASE TURN>
S: Do you need help? <QUESTION>
U: Uhh, *yes* (throat clearing) <INFORM>
S: These features are available: call forwarding, speed calling,
call waiting, and dial by name or number. <INFORM>
U: *Dialing by name or number* <REQUEST>
S: Whom shall I dial? <QUESTION>
U: Uhh, the *movie theatre*, or Robin Hood. <INFORM>
(pause)
S: Whom shall I dial? <QUESTION>

U: Umm, *224-0099*. <INFORM>
S: Now dialing 224-0099. <ACCEPT>
(Pause)
U: Well, *okay* (throat clearing) <ACCEPT>
S: Thank you.
(long pause)
U: Well., what's happening <RELEASE TURN>
S: Thank you for participating in this experiment. Please hang-up now.
U: *Hang-Up*? <QUESTION>
S: yes <INFORM>
U: *What*? <QUESTION>
S: yes <INFORM>
(hang-up)

Call Two

S: Hello, this is Southwestern Bell's phone service.
(pause) <RELEASE TURN>
S: Please say what service you want. <REQUEST>
U: Uhh, uhh *stop*, uhh, *call waiting* <REQUEST>
S: You want call waiting off? <QUESTION>
U: *yes* <INFORM>
S: Call Waiting is off. <ACCEPT>
U: *okay* <ACCEPT>
S: Thank you
U: Well, what next do I need to do?
(hang-up)

Call Three

S: Hello, this is Southwestern Bell's phone service.
U: Well, I want my uhh *speed calling directory changed*. <REQUEST>
S: What do you want to change it to? <QUESTION>
U: I want to *drop*, uhh, *Mike*. <REQUEST>
S: You want to delete Mike's number? <QUESTION>
U: *Correct*. <INFORM>
S: Now deleting Mike's number. <ACCEPT>
(pause)
S: Thank you.
U: Okay, what else do I need to do?
S: Please say what service you want. <REQUEST>
(pause)
(hang-up)

Call One

S: Hello, this is Southwestern Bell's phone service.
(pause) <RELEASE TURN>
S: Please say what service you want. <REQUEST>
U: *Speed Calling* <REQUEST>
S: What is the person's name? <QUESTION>
U: Uhh, *Mike* <INFORM>
(long pause)
U: Mike
(pause)
S: I'm sorry. Let me get someone who can help you. Please wait.

Thank you for participating in this experiment. Please hang-up now.
(hang-up)

Call One

S: Hello, this is Southwestern Bell's phone service.
(long pause) <RELEASE TURN>
S: Please say what service you want. <REQUEST>
U: *Speed Calling* <REQUEST>
(pause)
S: Whom shall I dial? <QUESTION>
U: *Mike* <INFORM>
(pause)
S: Now dialing 224-0099 <ACCEPT>
S: Thank you (pause)
S: Thank you for participating in this experiment. Please hang-up now.

Call Two

S: Hello, this is Southwestern Bell's phone service.
(long pause) <RELEASE TURN>
S: Do you need help? <QUESTION>
U: *Yes* <INFORM>
S: These features are available: Call Forwarding, Call Waiting,
Speed Calling, and Dial by name or number. <INFORM>
S: Which calling feature do you want help with? <QUESTION>
U: *Call Waiting* <INFORM>
S: Cancel Call Waiting stops important calls being interrupted.
Before you dial, you can ask for call waiting to be cancelled.
<INFORM>
(pause) <RELEASE TURN>
S: Please say what service you want. <REQUEST>
U: *Cancel Call Waiting* <INFORM>
(tape ends)

Call Three

S: Hello, this is Southwestern Bell's phone service.
(long pause) <RELEASE TURN>
S: Hello, this is Southwestern Bell's phone service.
U: *Call Forwarding* <REQUEST>
S: What number do you want your calls forwarded to? <QUESTION>
U: *224-0099* <INFORM>
S: Calls are forwarded to 224-0099 <ACCEPT>
S: Thank you
(hang-up)

Call Four

S: Hello, this is Southwestern Bell's phone service.
U: *Call forwarding* <REQUEST>
S: What number do you want your calls forwarded to? <QUESTION>
U: *224-0099* <INFORM>
S: Calls are forwarded to 224-0099 <ACCEPT>
(pause)

S: Thank you
(hang-up)

Call Five

S: Hello, this is Southwestern Bell's phone service.
U: *Cancel call waiting* <REQUEST>
S: Call waiting is off <ACCEPT>
(pause)
S: Thank you
(hang-up)

Call One

S: Hello, this is Southwestern Bell's phone service.
U: *help* <REQUEST>
S: Which calling feature do you want help with? <QUESTION>
U: *Speed Calling* <INFORM>
S: Speed Calling. You can reach important numbers more quickly and easily.
You store a person's name and phone number under a one digit code.
Then you can call using either the person's name or their code
instead of their phone number. <INFORM>
(pause)
U: *Cancel speed calling for Mike* <REQUEST>
S: Which service do you want off? <QUESTION>
U: *Speed Calling* <INFORM>
(pause)
S: Which entry do you want to change it to? <QUESTION>
U: *Mike* <INFORM>
S: What do you want to change it to? <QUESTION>
U: *Erase* it <INFORM>
S: You want to delete Mike's number? <QUESTION>
U: *Yes* <INFORM>
S: Now deleting Mike's number. <ACCEPT>
(pause)
S: Thank you
(hang-up)

Call Two

S: Hello, this is Southwestern Bell's phone service.
Please say what service you want.
U: *Speed Dialing* <REQUEST>
S: What is the person's name? <QUESTION>
U: Mrs. *Poppelstein* <INFORM>
(long pause)
(hang-up)

Call Two

S: Hello, this is Southwestern Bell's phone service.
Please say what service you want
U: *Help* <REQUEST>
S: Which calling feature do you want help with? <QUESTION>

U: *Speed Calling* <INFORM>
S: Call Forwarding, your phone automatically forwards all calls to a location of your choice. <INFORM>
S: Speed Calling, you can reach important numbers more quickly and easily. You store a person's name and phone number under a one digit code. Then you can call using either the person's name or their code instead of their phone number. <INFORM>
(pause)
U: *Change Speed Calling* <REQUEST>
S: Which entry do you want to change? <QUESTION>
U: Mrs. *Poppelstein* <INFORM>
S: What do you want to change it to? <QUESTION>
U: *Poopsey* <INFORM>
(laughter)
S: You want to change Poppelstein at 224-0099 to Poopsey? <QUESTION>
U: *Correct* <INFORM>
S: Thank you
(pause)
(hang-up)

Call Three

S: Hello, this is Southwestern Bell's phone service. Please say what service you want
U: *Call forwarding* <REQUEST>
S: What number do you want your calls forwarded to? <QUESTION>
U: *224-0099* <INFORM>
S: Calls are forwarded to 224-0099 <ACCEPT>
S: Thank you
(hang-up)

Call Three

S: Hello, this is Southwestern Bell's phone service.
U: *Remote Call forwarding* <REQUEST>
S: What number do you want your calls forwarded to? <QUESTION>
U: *224-0099* <INFORM>
S: What number do you want your calls forwarded from? <QUESTION>
U: *334-0561* <INFORM>
S: You want your calls forwarded from 334-0561 to 224-0099? <QUESTION>
U: *Yes* <INFORM>
S: Thank you
(hang-up)

Call Five

S: Hello, this is Southwestern Bell's phone service.
U: *Call Forwarding* <REQUEST>
S: What number do you want your calls forwarded to? <QUESTION>
U: *224-0099* <INFORM>
S: Calls are forwarded to 224-0099 <ACCEPT>
(Hang-up)

Call Six

S: Hello, this is Southwestern Bell's phone service.
Please say what service you want
U: *Change* (pause) *speed calling* <REQUEST>
S: Which entry do you want to change? <QUESTION>
U: Mrs. *Poppelstein* <INFORM>
S: What do you want me to change it to? <QUESTION>
U: *Poopsey* <INFORM>
S: You want to change Popplestein to Poopsey? <QUESTION>
U: *Correct* <INFORM>
S: thank you
(hang-up)

Call One

S: Hello, this is Southwestern Bell's phone service.
U: Could you tell me who is in my *speed call directory*, please?
<REQUEST>
S: Stored names are Poppelstein at 224-0099 under speed calling code 1.
<INFORM>
U: I'm sorry. I didn't get that. Could you *repeat* that please? <REQUEST>
S: Stored names are Poopsey at 224-0099 under speed calling code 2. <INFORM>
U: Umm, I'm sorry I need some *help*. I ... I need to find out who
are in my speed calling directory, and uhh I... I don't understand
the voice that is coming through to me. <REQUEST>
S:You can reach important numbers more quickly and easily.
You store a person's name and phone number under a one digit code.
Then you can call using either the person's name or their code
instead of their phone number. <INFORM>
U: Could you *repeat* that please? <REQUEST>
S:You can reach important numbers more quickly and easily.
You store a person's name and phone number under a one digit code.
Then you can call using either the person's name or their code
instead of their phone number. <INFORM>
U: Uhh, could you tell me whose *numbers* I have *stored*. <REQUEST>
S: Stored names are Poppelstein at 224-0099 under speed calling code 1.
<INFORM>
U: Uhh, could you *repeat* that name please? <REQUEST>
S: Stored names are Poppelstein at 224-0099 under speed calling code 1.
<INFORM>
U: Is that the *only name* in my *speed calling directory*? <QUESTION>
S: *Yes* <INFORM>
(hang-up)

Call Two

S: Hello, this is Southwestern Bell's phone service.
U: I'd like to *call Jane* could you please give me her number?
S: Whom shall I dial? <QUESTION>
U: *Jane*. The first name is Jane, J-A-N-E. <INFORM>
S: Now dialing Jane. <ACCEPT>
(pause)
S: thank you
(pause)
(hang-up)

Call Three

S: Hello, this is Southwestern Bell's phone service.
U: I'd like to my *calls forwarded* please <REQUEST>
S: To what number do you want your calls forwarded? <QUESTION>
U: *224-0099* <INFORM>
S: Calls are forwarded to 224-0099 <ACCEPT>
(hang-up)

Call Four

S: Hello, this is Southwestern Bell's phone service.
U: I'd like to my *calls forwarded* to *224-0099* <REQUEST>
S: Calls are forwarded to 224-0099 <ACCEPT>
(hang-up)

Call FIve

S: Hello, this is Southwestern Bell's phone service.
U: Umm, please, please *dial Jane* for me. <REQUEST>
S: Now dialing jane <ACCEPT>
(hang-up)

Call Six

S: Hello, this is Southwestern Bell's phone service.
U: can you tell me, what all, *what* are the *serivces* I can obtain?
<QUESTION>
S: These featured are available: Call Forwarding, Call Waiting,
Speed Calling and Dialing by name or number. <INFORM>
U: How do I *include a name and a number* in my *speed call directory*?
<QUESTIOn>
S: Do you want to delete, add, change, or dial an entry? <QUESTION>
U: Would you tell me *how many entries* I have in my *speed calling directory*?
<QUESTION>
(tape ends)
ed Calling, and Dial by

Call One

(first part of conversation missing)
U: I want to use *call forwarding* to *224-0099*. <REQUEST>
S: Calls are forwarded to 224-0099 <ACCEPT>
(Hang-up)

Call Two

S: Hello, This is Southwestern Bell's phone service.
U: Umm, I'm *calling* my *mother*. <REQUEST>
S: Whom shall I dial? <QUESTION>
(pause)
(hang-up)

Call Three

S: Hello, This is Southwestern Bell's phone service.
U: Umm, *cancel* my *call waiting* for my next phone call please. <REQUEST>
S: You want call waiting off? <QUESTION>
U: *Yes* <INFORM>
S: Call waiting is off. <ACCEPT>
(hang-up)

Call Four

S: Hello, This is Southwestern Bell's phone service.
U: What was the *code* I used for my *mother* for *speed calling*? <QUESTION>
S: Stored name is mother at 224-0099 under speed calling code 1. <INFORM>
(hang-up)

Call Five

S: Hello, This is Southwestern Bell's phone service.
U: Please *cancel* the *call waiting*. <REQUEST>
S: Call waiting is off. <ACCEPT>
(hang-up)

Call One

S: Hello, This is Southwestern Bell's phone service.
U: Okay, umm, I guess I need to determine, uhh the best way to
transfer my *calls*. <REQUEST>
S: To what number do you want your call forwarded? <QUESTION>
U: To *224-0099*. <INFORM>
S: Calls are forwarded to 224-0099 <ACCEPT>
U: *Thank you* <ACCEPT>
S: Thank you
(hang-up)

Call Two

S: Hello, This is Southwestern Bell's phone service.
U: Yes, I need to place a call, but I don't want to be interrupted what,
umm, so I need to do?
S: Do you need help? <QUESTION>
(pause)
U: *Yes* <INFORM>
(pause)
S: Which calling feature do you want help with? <QUESTION>
U: That was my question. I guess, I guess it might be *Call waiting*. <INFORM>
S: Cancel Call Waiting, stops important calls from being interrupted.
Before you dial, ask for call waiting to be canceled. <INFORM>
U: Good, that's what I need.
(pause)
U: I want to have *call waiting cancelled*. <REQUEST>
S: Call waiting is off <ACCEPT>
U: *thank you* <ACCEPT>
S: Thank you
(hang-up)

Call Three

S: Hello, This is Southwestern Bell's phone service.
U: I need *speed calling help*. <REQUEST>
S: Speed calling, You can reach important numbers more quickly and easily.
You store a person's name and phone number under a one digit code.
Then you can call using either the person's name or their code instead
of their phone number. <INFORM>
U: I need my mother's number. Could you please *dial* it? <REQUEST>
S: Whom shall I dial? <QUESTION>
U: My *Mother* <INFORM>
S: Now dialing 224-0099. <ACCEPT>
U: *Thank you* <ACCEPT>
S: Thank you
(hang-up)

Call One

S: Hello, This is Southwestern Bell's phone service.
(pause) <RELEASE TURN>
S: Please say what service you want. <REQUEST>
(pause)
U: *Call Forwarding* <REQUEST>
S: To what number do you want your calls forwarded? <QUESTION>
U: *cancel* <REJECT>
S: Which service do you want off? <QUESTION>
(pause)
U: opps
U: *cancel*, previous menu, whatever <REJECT>
S: Do you need help? <QUESTION>
U: *Yes* <INFORM>
S: Which calling feature do you want help with? <QUESTION>
U: *Call waiting* <INFORM>
S: Cancel call waiting stops important calls from being interrupted.
Before you dial, you can ask for call waiting to be cancelled. <INFORM>
U: okay, uhh, *main menu* <REQUEST>
S: These features are available: call waiting, call forwarding,
speed calling, and dialing by name or number <INFORM>
U: *call waiting* <REQUEST>
S: you want call forwarding off? <QUESTION>
U: *no* (cough) <INFORM>
S: Which service do you want off? <QUESTION>
U: *none* <INFORM>
(pause)
U: *help* <REQUEST>
S: Which calling feature do you want help with? <QUESTION>
U: *call waiting* <REQUEST>
S: Cancel call waiting stops important calls from being interrupted.
Before you dial, you can ask for call waiting to be cancelled. <INFORM>
U: Alright, let's try it again
(hang-up)

Note: Subject repeats call one to try to accomplish task

Call One

S: Hello, This is Southwestern Bell's phone service.
(pause)
U: *call waiting* <REQUEST>
S: You want call waiting off? <QUESTION>
U: *yes* <INFORM>
S: call waiting is off. <ACCEPT>
S: thank you
(long pause)
(hang-up)

Call Two

S: Hello, This is Southwestern Bell's phone service.
(pause)
U: *help* <REQUEST>
S: Which calling feature do you want help with? <QUESTION>
U: *Dialing by name or number*. <INFORM>
S: You can dial by speaking a phone number, a speed dial code,

(orator cuts off this help message)
U: *Change* <REQUEST>
S: Which entry do you want to change? <QUESTION>
U: *Mr. and Mrs. Poppelstein* <INFORM>
S: What do you want to change it to? <QUESTION>
U: The name is *Poopsey*. <INFORM>
S: You want to change Poppelstein at 224-0099 to Poopsey? <QUESTION>
U: *Yes* <INFORM>
S: Thank you
(pause)
(hang-up)

Call Three

S: Hello, This is Southwestern Bell's phone service.
U: (cough) *Call forwarding* <REQUEST>
S: To what number do you want your calls forwarded? <QUESTION>
U: *224-0099* <INFORM>
S: Calls are forwarded to 224-0099 <ACCEPT>
(hang-up)

Call Four

S: Hello, This is Southwestern Bell's phone service.
U: *Call forwarding* <REQUEST>
S: To what number do you want your calls forwarded? <QUESTION>
U: *224-0099* (cough) <INFORM>
S: Calls are forwarded to 224-0099 <ACCEPT>
(pause)
(hang-up)

Call Five

S: Hello, This is Southwestern Bell's phone service.
U: *Dial by name or number* <REQUEST>
S: Whom shall I dial? <QUESTION>

U: *Change* <REQUEST>
S: Which entry do you want to change? <QUESTION>
U: *Poppelstein* <INFORM>
S: What do you want to change it to? <QUESTION>
U: *Poopsey*. <INFORM>
S: You want to change Poppelstein at 224-0099 to Poopsey? <QUESTION>
U: *Change* <REQUEST>
S: Thank you
(hang-up)

Call Six

S: Hello, This is Southwestern Bell's phone service.
U: *Change call waiting* <REQUEST>
S: You want call waiting off? <QUESTION>
U: *yes* <INFORM>
S: call waiting is off. <ACCEPT>
(hang-up)

Call One

S: Hello, this is Southwestern Bell's phone service.
U: Hi
(pause)
U: I'd like to have my *calls forwarded* to *224-0099*. <REQUEST>
S: Calls are forwarded to 224-0099. <ACCEPT>
U: *Thank you* <ACCEPT>
S: Thank you
(hang-up)

Call Two

S: Hello, this is Southwestern Bell's phone service.
U: Hi, I'd like to know who's in my *speed calling directory*. <REQUEST>
S: Stored name is Dad at 224-0099 under code 1. <INFORM>
(pause)
S: Thank you
(hang-up)

Call Three

S: Hello, this is Southwestern Bell's phone service.
U: Yea. I *no* longer want my *calls forwarded*. <REQUEST>
(Pause)
S: Which service do you want off? <QUESTION>
U: *Call forwarding* <INFORM>
S: Call forwarding is off. <ACCEPT>
U: *Thanks* <ACCEPT>
S: Thank you
(hang-up)

Call Four

S: Hello, this is Southwestern Bell's phone service.
U: Hi, I'd like to *turn call forwarding off*. <REQUEST>

S: Call forwarding is off. <ACCEPT>
U: *thanks* <ACCEPT>
S: Thank you
(hang-up)

Call Five

S: Hello, this is Southwestern Bell's phone service.
U: Who's in my *speed calling directory*? <QUESTION>
S: Stored name is Dad at 224-0099 under code 1. <INFORM>
S: Thank you
U: *Thanks* <ACCEPT>
(hang-up)

Call Six

S: Hello, this is Southwestern Bell's phone service.
U: Yea, I want my *calls forwarded* (pause) to *224-0099*. <REQUEST>
S: Calls are forwarded to 224-0099. <ACCEPT>
U: *Thanks* <ACCEPT>
S: Thank you
(hang-up)

Call One

S: Hello, this is Southwestern Bell's phone service. Please say what service you want.
U: *Forwarding*, Call forwarding <REQUEST>
S: What number do you want your calls forwarded to? <QUESTION>
U: *224-0099* <INFORM>
S: Calls are forwarded to 224-0099. <ACCEPT>
(hang-up)

Call Two

S: Hello, this is Southwestern Bell's phone service.
Please say what service you want.
U: *Remove Call forwarding* <REQUEST>
(pause)
S: You want call forwarding off? <QUESTION>
U: *yes* <INFORM>
S: call forwarding is off <ACCEPT>
(hang-up)

Call Three

S: Hello, this is Southwestern Bell's phone service.
Please say what service you want.
U: *Call Jane* <REQUEST>
S: Now dialing Jane. <ACCEPT>
(pause)
S: Thank you
(hang-up)

Call Four

S: Hello, this is Southwestern Bell's phone service.
Please say what service you want.
U: *Call Jane* <REQUEST>
S: Now dialing Jane. <ACCEPT>
(pause)
S: Thank you
(hang-up)

Call Five

S: Hello, this is Southwestern Bell's phone service.
Please say what service you want.
U: *Remove Call forwarding* <REQUEST>
S: call forwarding is off <ACCEPT>
(hang-up)

Call Six

S: Hello, this is Southwestern Bell's phone service.
Please say what service you want.
U: *Call forwarding* <REQUEST>
S: What number do you want your calls forwarded to? <QUESTION>
U: *224-0099* <INFORM>
S: Calls are forwarded to 224-0099 <ACCEPT>
(hang-up)

APPENDIX A.2

PATTERNMATCHER

```
:- ensure_loaded(library(basics)).
:- consult('lexicon').
:- consult('parser').
:- consult('postparse').
:- consult('input').

process(A,B) :-
        present(help,A),!,
        getfw(A,Fw),!,
        help(A,B,Fw).

process(A,B) :-
        present(cancel,A),!,
        getfw(A,Fw),!,
        cancel(A,B,Fw).

process(A,B) :-
        present(remove,A),!,
        getfw1(A,Fw),!,
        cancel(A,B,Fw).

process(A,B) :-
        present(wait,A),!,
        waiting(A,B).

process(A,B) :-
        present(remote,A),!,
        remotefor(A,B).

process(A,B) :-
        present(forward,A),!,
        forw(A,B).

process(A,B) :-
        present(change,A),!,
        getfromcode(A,F),!,
        gettocode(A,T),!,
        ch(A,B,F,T).

process(A,B) :-
        present(add,A),!,
        getaddcode(A,C),!,
        add(A,B,C).

process(A,B) :-
        present(delete,A),!,
        getreph(A,P),!,
        delete(A,B,P).

process(A,B) :-
        present(show,A),!,
        show(A,B).
```

```
process(A,B) :-
        present(dial,A),!,
        getph(A,P),!,
        dial(A,B,P).

process(A,B) :-
        present(menu,A),!,
        B = surface_request(u,s,help(s,u,menu)).

process(A,B) :-
        present(repeat,A),!,
        B = surface_request(u,s,repeat(s,u)).

process(A,B) :-
        present(hello,A),!,
        B = surface_request(u,s,help(s,u,menu)).

process(A,B) :-
        present(thank, A),!,
        B = surface_acknowledge(u,s).

process(A,B) :-
        present(goodbye,A),!,
        B = surface_acknowledge(u,s).

process(A,B) :-
        present(yes,A),!,
        B = surface_informif(u,s,_,true).

process(A,B) :-
        present(no,A),!,
        B = surface_informif(u,s,_,false).

process(A,B) :-
        present(is,A),!,
        getonlyph(A,P),
        \+ (P = nonexistent),
        B = surface_informref(u,s,P,parameter(P,_)).

process(A,B) :-
        getonlyph(A,P),
        \+ (P = nonexistent),!,
        B = surface_request(u,s,dial(s,u,P)).

process(_,B) :-
        B = surface_inform(u,s,noinfo).

help(A,B,Fw) :-
        present(how,A),!,
        B = surface_wh_question(s,u,Fw).

help(A,B,Fw) :-
        present(confused,A),!,
        B = surface_inform(s,u,want(u,help(s,u,Fw))).
```

```
help(A,B,Fw) :-
        A = [can|_],!,
        helpcan(A,B,Fw).

help(A,B,Fw) :-
        present(want,A),!,
        B = surface_inform(u,s,want(u,help(s,u,Fw))).

help(_,B,Fw) :-
        !,B = surface_request(u,s,help(s,u,Fw)).

cancel(A,B,Fw) :-
        A = [can|_],!,
        cancelcan(A,B,Fw).

cancel(A,B,Fw) :-
        present(want,A),!,
        B = surface_inform(u,s,want(u,cancel(s,u,Fw))).

cancel(_,B,Fw) :-
        !,B = surface_request(u,s,cancel(s,u,Fw)).

waiting(A,B) :-
        A = [can|_],!,
        waitingcan(A,B).

waiting(A,B) :-
        present(want,A),!,
        B = surface_inform(u,s,want(u,wait(s,u))).

waiting(_,B) :-
        !,B = surface_request(u,s,wait(s,u)).

remotefor(A,B) :-
        A = [can|_],!,
        gettoph(A,Pto),!,
        getfromph(A,Pfrom),!,
        reforwcan(A,B,Pto,Pfrom).

remotefor(A,B) :-
        present(want,A),!,
        gettoph(A,Pto),!,
        getfromph(A,Pfrom),!,
        reforwwant(A,B,Pto,Pfrom).

remotefor(A,B) :-
        gettoph(A,Pto),!,
        getfromph(A,Pfrom),!,
        reforw1(A,B,Pto,Pfrom).

forw(A,B) :-
        A = [can|_],!,
        getph(A,P),!,
        forwcan(A,B,P).

forw(A,B) :-
```

```
        present(want,A),!,
        getph(A,P),!,
        forwwant(A,B,P).

forw(A,B) :-
        getph(A,P),!,
        forw1(A,B,P).

ch(A,B,F,T) :-
        A = [can|_],!,
        getforph(A,P),
        chcan(A,B,F,T,P).

ch(A,B,F,T) :-
        present(want,A),!,
        getforph(A,P),
        B = surface_inform(u,s,want(u,change(s,u,P,F,T))).

ch(A,B,F,T) :-
        getforph(A,P),!,
        B = surface_request(u,s,change(s,u,P,F,T)).

add(A,B,C) :-
        A = [can|_],!,
        getatph(A,P),
        addcan(A,B,C,P).

add(A,B,C) :-
        present(want,A),!,
        getatph(A,P),
        addwant(A,B,C,P).

add(A,B,C) :-
        getatph(A,P),!,
        add1(A,B,C,P).

delete(A,B,P) :-
        A = [can|_],!,
        deletecan(A,B,P).

delete(A,B,P) :-
        present(want,A),!,
        deletewant(A,B,P).

delete(A,B,P) :-
        !,delete1(A,B,P).

show(A,B) :-
        A = [can|_],!,
        showcan(A,B).

show(A,B) :-
        present(want,A),!,
        B = surface_inform(u,s,want(u,showcodes(s,u))).

show(_,B) :-
```

```
        !,B = surface_request(u,s,showcodes(s,u)).

dial(A,B,P) :-
        A = [can|_],!,
        dialcan(A,B,P).

dial(A,B,P) :-
        present(want,A),!,
        dialwant(A,B,P).

dial(A,B,P) :-
        !,dial1(A,B,P).

helpcan(A,B,Fw) :-
        present(please,A),!,
        B = surface_request(u,s,help(s,u,Fw)).

helpcan(_,B,Fw) :-
        B = surface_yn_question(u,s,cando(s,help(s,u,Fw))).

cancelcan(A,B,Fw) :-
        present(please,A),!,
        B = surface_request(u,s,cancel(s,u,Fw)).

cancelcan(_,B,Fw) :-
        B = surface_yn_question(u,s,cando(s,cancel(s,u,Fw))).

waitingcan(A,B) :-
        present(please,A),!,
        B = surface_request(u,s,wait(s,u)).

waitingcan(_,B) :-
        B = surface_yn_question(s,u,cando(s,wait(s,u))).

forwcan(A,B,P) :-
        present(please,A),!,
        B = surface_request(u,s,forward(s,u,P)).

forwcan(_,B,P) :-
        B = surface_yn_question(s,u,cando(s,forward(s,u,P))).

forwwant(_,B,P) :-
        B = surface_inform(u,s,want(u,forward(s,u,P))).

forw1(_,B,P) :-
        !,B = surface_request(u,s,forward(s,u,P)).

reforwcan(A,B,Pto,Pfrom) :-
        present(please,A),!,
        B = surface_request(u,s,remote(s,u,Pto,Pfrom)).

reforwcan(_,B,Pto,Pfrom) :-
        B = surface_yn_question(u,s,cando(s,remote(s,u,Pto,Pfrom))).

reforwwant(_,B,Pto,Pfrom) :-
        B = surface_inform(u,s,want(u,remote(s,u,Pto,Pfrom))).
```

```
reforw1(_,B,Pto,Pfrom) :-
        !,B = surface_request(u,s,remote(s,u,Pto,Pfrom)).

chcan(A,B,F,T,P) :-
        present(please,A),!,
        B = surface_request(u,s,change(s,u,P,F,T)).

chcan(_,B,F,T,P) :-
        B = surface_yn_question(u,s,cando(s,change(s,u,P,F,T))).

addcan(A,B,C,P) :-
        present(please,A),!,
        B = surface_request(u,s,add(s,u,C,P)).

addcan(_,B,C,P) :-
        B = surface_yn_question(u,s,cando(s,add(s,u,C,P))).

addwant(_,B,C,P) :-
        B = surface_inform(u,s,want(u,add(s,u,C,P))).

add1(_,B,C,P) :-
        B = surface_request(u,s,add(s,u,C,P)).

deletecan(A,B,P) :-
        present(please,A),!,
        B = surface_request(u,s,delete(s,u,P)).

deletecan(_,B,P) :-
        B = surface_yn_question(u,s,cando(s,delete(s,u,P))).

deletewant(_,B,P) :-
        B = surface_inform(u,s,want(u,delete(s,u,P))).

delete1(_,B,P) :-
        B = surface_request(u,s,delete(s,u,P)).

showcan(A,B) :-
        present(please,A),!,
        B = surface_request(u,s,showcodes(s,u)).

showcan(_,B) :-
        B = surface_yn_question(u,s,cando(s,showcodes(s,u))).

dialcan(A,B,P) :-
        present(please,A),!,
        B = surface_request(u,s,dial(s,u,P)).

dialcan(_,B,P) :-
        B = surface_yn_question(u,s,cando(s,dial(s,u,P))).

dialwant(_,B,P) :-
        B = surface_inform(u,s,want(u,dial(s,u,P))).

dial1(_,B,P) :-
        B = surface_request(u,s,dial(s,u,P)).
```

```
present(Elem,List) :-
        member(Elem,List).

present(Elem,List) :-
        synonym(Elem,Syn),
        member(Syn,List).

getfw(A,Fw) :-
        fw(Fw),
        member(Fw,A),!.

getfw(A,Fw) :-
        fw(Fw),
        synonym(Fw,Syn),
        member(Syn,A),!.

getfw(_,_).

getfw1(A,Fw) :-
        fw1(Fw),
        member(Fw,A),!.

getfw1(A,Fw) :-
        fw1(Fw),
        synonym(Fw,Syn),
        member(Syn,A).

getph(A,P) :-
        A = [to | Tail],
        getph1(Tail,P).

getph(A,P) :-
        A = [dial | Tail],
        getph1(Tail,P).

getph(A,P) :-
        A = [Head | Tail],
        synonym(dial,Head),
        getph1(Tail,P).

getph(A,P) :-
        A = [_ | Tail],
        getph(Tail,P).

getph(_,_).

gettoph(A,P) :-
        A = [to | Tail],
        Tail = [P | _],
        number(P),!.

gettoph(A,P) :-
        A = [_ | Tail],
        gettoph(Tail,P).
```

```
gettoph(_,_).

getfromph(A,P) :-
        A = [from | Tail],
        Tail = [P | _],
        number(P),!.

getfromph(A,P) :-
        A = [_ | Tail],
        getfromph(Tail,P).

getfromph(_,_).

getph1(B,P) :-
        B = [P|_],
        number(P),!.

getph1(B,P) :-
        B = [H|_],
        speed(H,P),!.

getph1(B,P) :-
        B = [_ | T],
        getph1(T,P).

getonlyph(A,P) :-
        A = [P | _],
        number(P),!.

getonlyph(A,P) :-
        A = [H|_],
        speed(H,P),!.

getonlyph(A,P) :-
        A = [_ | Tail],
        getonlyph(Tail,P).

getonlyph(_,P) :-
        P = nonexistent.

getforph(A,P) :-
        A = [for | Tail],
        Tail = [P | _],
        number(P),
        checkph(_,P),!.

getforph(A,P) :-
        A = [_ | Tail],
        getforph(Tail,P).

getforph(_,_).

getatph(A,P) :-
        A = [at | Tail],
        Tail = [P | _],
        number(P),!.
```

```
getatph(A,P) :-
        A = [_ | Tail],
        getatph(Tail,P).

getatph(_,_).

getfromcode(A,F) :-
        A = [from | Tail],
        Tail = [F | _],
        checksp(F),!.

getfromcode(A,F) :-
        A = [_ | Tail],
        getfromcode(Tail,F).

getfromcode(_,_).

gettocode(A,C) :-
        A = [to | Tail],
        Tail = [C | _],!.

gettocode(A,C) :-
        A = [_ | Tail],
        gettocode(Tail,C).

gettocode(_,_).

getaddcode(A,C) :-
        A = [add | Tail],
        Tail = [C|_],!.

getaddcode(A,C) :-
        A = [Head | Tail],
        synonym(add,Head),
        Tail = [C|_],!.

getaddcode(A,C) :-
        A = [_|Tail],
        getaddcode(Tail,C).

getaddcode(_,_).

getreph(A,P) :-
        A = [delete | Tail],
        getph1(Tail,P).

getreph(A,P) :-
        A = [Head | Tail],
        synonym(delete,Head),
        getph1(Tail,P).

getreph(A,P) :-
        A = [_|Tail],
        getreph(Tail,P).
getreph(_,_).
```

```
checksp(F) :-
        speed(F,_),!.

checksp(F) :-
        F = nonexistent,!.

checkph(_,P) :-
        speed(_,P),!.

checkph(_,P) :-
        P = nonexistent,!.

different(X,Y):-
        X = Y,!,
        fail
          ;
        true.

PARSER
------
informif-->um,leadphrase,um,[if].
informif-->um,leadphrase,um,[is].
informif-->um,modal,um,selector.
informif-->um,modal,um,firstperson.
informif-->um,modal,um,feature.

informref-->um,leadphrase,um,wword.
informref-->um,wword.

leadphrase-->modal,um,[you],um,[tell],um,[me].
leadphrase-->firstperson,um,desire,um,[to],um,[know].
leadphrase-->contraction,um,[like],um,[to],um,[know].
leadphrase-->[well].

feature-->[call],[waiting].
feature-->[call],[forwarding].
feature-->[forwarding].
feature-->[dialing].
feature-->[speed],[dialing].
feature-->[remote],[call],[forwarding].
feature-->[remote],[forwarding].

modal-->[can].
modal-->[could].
modal-->[will].
modal-->[would].
modal-->[is].
modal-->[are].

firstperson-->[i].
firstperson-->[we].

contraction-->[id].
contraction-->[wed].
```

```
desire-->[need].
desire-->[want].
desire-->[would],um,[like].

wword-->[who].
wword-->[whos].
wword-->[whose].
wword-->[whom].
wword-->[what].
wword-->[why].
wword-->[how].

selector-->[that].
selector-->[the].
selector-->[there].
selector-->[this].

um-->[um],um.
um-->[ah],um.
um-->[uhh],um.
um-->[so],um.
um-->[fine],um.
um-->[but],um.
um-->[ok],um.
um-->[yes],um.
um-->[hi],um.
um-->[].

POSTPARSE
---------
% concatenate two lists.  also used to determine the difference of two lists.
conc([], L, L).
conc([X | L1], L2, [X | L3]) :-
        conc(L1, L2, L3).

% reverse list arg1 and put it in arg2
reverse([], []).
reverse([First | Rest], Reversed) :-
        reverse(Rest, ReversedRest),
        conc(ReversedRest, [First], Reversed).

% attempts to parse the utterance as either an informref or an
% informif.  It first attempts to parse the entire utterance and
% then if unsuccesful tries deleting the last word.  This is continued
% until a succesful parse or all words have been deleted.
parse(I1,Type) :-
        reverse(I1,I2),
        conc(L1,_,I2),
        reverse(L1,L4),
        informref(I1, L4),!,
        Type = ref.

parse(I1,Type) :-
        reverse(I1, I2),
        conc(L1,_, I2),
        reverse(L1, L4),
```

```
        informif(I1, L4),
        Type = if.

% process the utterance as a question if appropriate.  if not
% then process it as before.
processq(A,B) :-
        parse(A,T),
        T = if, !,
        processif(A,B).

processq(A,B) :-
        parse(A,T1),
        T1 = ref, !,
        processref(A,B).

processq(A,B) :-
        process(A,B).

processif(A,B) :-
        present(available, A), !,
        B = surface_yn_question(u,s,available(help)).

processif(A,B) :-
        present(waiting, A), !,
        getstatus(A,S),
        waitingif(A,B,S).

processif(A,B) :-
        present(forwarding,A), !,
        getstatus(A,S),
        forwardingif(A,B,S).

processif(A,B) :-
        present(directory,A), !,
        directoryif(A,B).

processref(A,B) :-
        present(how,A), !,
        getfw(A,Fw), !,
        help(A,B,Fw).

processref(A,B) :-
        present(directory,A), !,
        directoryref(A,B).

processref(A,B) :-
        present(stored,A), !,
        present(numbers,A), !,
        directoryref(A,B).

processref(A,B) :-
        A = [what|_],!,
        getfw(A,Fw),
        B = surface_wh_question(u,s,Fw).

processref(A,B) :-
```

```
        getfw(Fw),
        help(A,B,Fw).

processref(A,B) :-
        present(services,A), !,
        B = surface_wh_question(u,s,services).

processref(A,B) :-
        present(code, A), !,
        getperson(A,P), !,
        dialref(A,P,B).

waitingif(_,B,Status) :-
        B = surface_yn_question(u,s,isstatus(waiting,Status)).

forwardingif(_,B,Status) :-
        B = surface_yn_question(u,s,isstatus(forwarding,Status)).

directoryif(A,B) :-
        present(any,A),
        present(entries,A),
        B = surface_yn_question(u,s,directory(non_empty)).

directoryif(A,B) :-
        present(empty,A),
        B = surface_yn_question(u,s,directory(empty)).

directoryif(A,B) :-
        present(only,A),
        present(that,A),
        B = surface_yn_question(u,s,directory(onlyone)).

directoryref(A,B) :-
        A = [who|_],!,
        B = surface_wh_question(u,s,entries).

directoryref(A,B) :-
        present(who,A),
        present(want,A),!,
        B = surface_inform(u,s,want(u,showcodes(s,u))).

directoryref(A,B) :-
        present(who,A),
        B = surface_request(u,s,showcodes(s,u)).

getperson(A,P) :-
        A = [Head|_],
        speed(Head,_),
        P = Head.

getperson(A,P) :-
        A = [_ | Tail],
        getperson(Tail, P).

dialref(_,P,B) :-
        B = surface_wh_question(u,s,code(P)).
```

```
LEXICON
-------
synonym(help,directions).
synonym(help,information).
synonym(help,know).
synonym(help,how).
synonym(help,confused).
synonym(help,confuse).

synonym(cancel,disable).
synonym(cancel,stop).
synonym(cancel,stopped).
synonym(cancel,off).

synonym(wait,waiting).
synonym(wait,interrupt).

synonym(forward,forwarding).
synonym(forward,forwarded).
synonym(forward,placed).
synonym(forward,place).
synonym(forward,move).
synonym(forward,moved).

synonym(dial,speed-dialing).
synonym(dial,speed-calling).
synonym(dial,speed-call).
synonym(dial,speed-dial).
synonym(dial,call).
synonym(dial,dialing).
synonym(dial,dialling).

synonym(change,modify).
synonym(add,insert).
synonym(delete,remove).

synonym(show,display).
synonym(show,hear).

synonym(want,would).
synonym(want,need).
synonym(want,like).

synonym(yes,correct).
synonym(yes,ok).
synonym(yes,sure).
synonym(yes,fine).
synonym(yes,right).

synonym(thank, thanks).

synonym(hello,hi).
synonym(hello,anyone).

synonym(services,service).
```

```
fw(remote).
fw(forward).
fw(wait).
fw(speed).
fw(dial).

fw1(remote).
fw1(forward).
fw1(wait).

speed(mom,7654321).
speed(moms,7654321).
speed(mother,7654321).
speed(mothers,7654321).
speed(jane,1234567).
speed(janes,1234567).
speed(mike,1233456).
speed(mikes,1233456).
```

APPENDIX A.3

Sample Output

```
Input = [call,forwarding,help]
Output = surface_request(u,s,help(s,u,forward))

Input = [yes]
Output = surface_informif(u,s,_6702,true)

Input = [dial,mom]
Output = surface_request(u,s,dial(s,u,7654321))

Input = [cancel,call,forwarding]
Output = surface_request(u,s,cancel(s,u,forward))

Input = [stop,call,forwarding]
Output = surface_request(u,s,cancel(s,u,forward))

Input = [call,my,mom]
Output = surface_request(u,s,dial(s,u,7654321))

Input = [um,lets,see,call,forward]
Output = surface_request(u,s,forward(s,u,_6707))

Input = [1234567]
Output = surface_request(u,s,dial(s,u,1234567))

Input = [yes,i,would,like,my,home,phone,call,forwarded,to,a,neighbor,and,i,need,
some,directions,on,how,to,do,that]
Output = surface_wh_question(s,u,forward)

Input = [um,the,phone,number,is,1234567]
Output = surface_informref(u,s,1234567,parameter(1234567,_6716))

Input = [that,is,correct]
Output = surface_informif(u,s,_6706,true)

Input = [ok,thank,you]
Output = surface_acknowledge(u,s)

Input = [yes,my,calls,are,being,forwarded,to,a,friends,house,and,id,like,to,canc
el,that,so,im,receiving,the,calls]
Output = surface_inform(u,s,want(u,cancel(s,u,forward)))

Input = [correct]
Output = surface_informif(u,s,_6702,true)

Input = [ok,thank,you,very,much]
Output = surface_acknowledge(u,s)

Input = [yes,um,i,call,a,number,quite,frequently,and,need,to,know,if,um,speed-ca
lling,and,dialling,by,name,or,number,is,the,same,type,of,service]
Output = surface_inform(u,s,want(u,help(s,u,dial)))
```

```
Input = [ok,i,would,like,to,have,this,speed-call,number,set,up,by,dialling,by,na
me,how,can,i,do,that]
Output = surface_wh_question(s,u,dial)

Input = [ok,so,i,would,need,to,put,that,persons,phone,number,in,and,then,i,can,j
ust,list,it,by,name,and,that,works,the,same,way,correct,or,is,it,the,same,thing,
by,mmh,im,confused]
Output = surface_inform(s,u,want(u,help(s,u,_6773)))

Input = [i,want,dialling,by,name]
Output = surface_inform(u,s,want(u,dial(s,u,_6707)))

Input = [mmm,i,want,to,dial,jane]
Output = surface_inform(u,s,want(u,dial(s,u,1234567)))

Input = [ok,does,that,mean,my,service,is,set,up]
Output = surface_informif(u,s,_6718,true)

Input = [yes,i,would,like,to,have,um,my,phone,set,up,that,i,can,call,my,friend,j
ane,by,dialling,her,by,name]
Output = surface_inform(u,s,want(u,dial(s,u,1234567)))

Input = [yes,i,do]
Output = surface_informif(u,s,_6706,true)

Input = [ok,thank,you]
Output = surface_acknowledge(u,s)

Input = [yes,um,i,need,to,cancel,my,call,forwarding,um,to,which,is,at,a,friends,
house,and,i,just,need,to,cancel,that]
Output = surface_inform(u,s,want(u,cancel(s,u,forward)))

Input = [yes,i,need,my,calls,forwarded,to,1234567]
Output = surface_inform(u,s,want(u,forward(s,u,1234567)))

Input = [forward,my,calls,to,mom]
Output = surface_request(u,s,forward(s,u,7654321))

Input = [add,sue,at,1111111]
Output = surface_request(u,s,add(s,u,sue,1111111))

Input = [delete,mom]
Output = surface_request(u,s,delete(s,u,7654321))

Input = [delete,7654321]
Output = surface_request(u,s,delete(s,u,7654321))

Input = [change,my,code,for,7654321]
Output = surface_request(u,s,change(s,u,7654321,_6708,_6707))

Input = [change,my,code,for,7654321,from,mom,to,mother]
Output = surface_request(u,s,change(s,u,7654321,mom,mother))

Input = [change,the,code,for,7654321,to,mother]
Output = surface_request(u,s,change(s,u,7654321,_6711,mother))
```

```
Input = [hello]
Output = surface_request(u,s,help(s,u,menu))

Input = [no]
Output = surface_informif(u,s,_6702,false)

Input = [call,forwarding]
Output = surface_request(u,s,forward(s,u,_6701))

Input = [2240099]
Output = surface_request(u,s,dial(s,u,2240099))

Input = [thank,you]
Output = surface_acknowledge(u,s)

Input = [hello,uhh,i,need,to,turn,off,call,waiting]
Output = surface_inform(u,s,want(u,cancel(s,u,wait)))

Input = [yes]
Output = surface_informif(u,s,_6702,true)

Input = [hi,i,need,to,turn,off,my,call,waiting,while,i,make,a,long,distance,call]
Output = surface_inform(u,s,want(u,cancel(s,u,wait)))

Input = [call,waiting]
Output = surface_request(u,s,wait(s,u))

Input = [thank,you]
Output = surface_acknowledge(u,s)

Input = [hi,i,forgot,my,mothers,number,i,would,like,to,call,her]
Output = surface_inform(u,s,want(u,dial(s,u,_6721)))

Input = [hi,i,want,to,call,my,mother]
Output = surface_inform(u,s,want(u,dial(s,u,7654321)))

Input = [uhh,yes,i,need,information,on,call,waiting]
Output = surface_inform(u,s,want(u,help(s,u,wait)))

Input = [remote,call,forwarding]
Output = surface_request(u,s,remote(s,u,_6704,_6703))

Input = [2240099]
Output = surface_request(u,s,dial(s,u,2240099))

Input = [3340561]
Output = surface_request(u,s,dial(s,u,3340561))

Input = [hi,my,name,is,keith,uhh,petty,i,plan,to,be,over,at,my,friend,mikes,hous
e,this,evening,and,i,want,my,calls,to,be,placed,over,there,please]
Output = surface_inform(u,s,want(u,forward(s,u,1233456)))

Input = [dial,an,entry]
Output = surface_request(u,s,dial(s,u,_6703))
```

```
Input = [jane]
Output = surface_request(u,s,dial(s,u,1234567))

Input = [i,want,to,hear,the,directory]
Output = surface_inform(u,s,want(u,showcodes(s,u)))

Input = [what,is,dialing,by,name,or,number]
Output = surface_wh_question(u,s,dial)

Input = [yes,id,like,to,know,who,is,in,my,speed,call,directory]
Output = surface_inform(u,s,want(u,showcodes(s,u)))

Input = [i,want,my,calls,remote,call,forwarded,from,2240099,to,7879090]
Output = surface_inform(u,s,want(u,remote(s,u,7879090,2240099)))

Input = [i,need,help,with,call,forwarding,and,call,waiting]
Output = surface_inform(u,s,want(u,help(s,u,forward)))
```

REFERENCES

Allen, J. F, and L. K. Schubert. "The TRAINS project."*TRAINS Technical Note 91-1, Department of Computer Science, University of Rochester*. (1991).

Balentine, J., Berry, U., and A. Johnstone. "The DIM system : WOZ Simulation Results - Phase I." *Technical Report WUCS-92-20, Computer Sciience, Washington University, St. Louis*. (1992).

Becker, J. "The Phrasal Lexicon." *Proceedings of the workshop on theoretical issues in Natural Language Processing* (1975): 70-73.

Kelley, J. F. "An empirical methodology for writing User-Friendly Natural Language computer applications." *Proceedings from CHI 1983* (1983): 193-196.