CS381K Artificial Intelligence
Assignment 4, due 12/6/00

Natural Language Access to Databases

Natural languages such as English can provide interfaces to computer systems that can be used by non-programmers with little or no training. Such an interface can easily be constructed provided that the English coverage is limited to a narrow domain; this requirement is satisfied by database systems. Several commercial systems for natural language access to databases exist.

For this exercise, you are to construct a natural language interface to a small relational database using the *semantic grammar* technique. A semantic grammar differs from a general English grammar in that phrases in the grammar have *meaning in the domain* rather than being purely syntactic descriptions of English phrases. For example, a general English grammar would have categories such as "noun phrase", while a semantic grammar would have categories such as "date" or "part-name".

A good first step in developing your program is to write a grammar that will describe the legal sentences that may be used in querying this database. The grammar might be in the form of productions or an ATN. Note that the grammar only needs to represent *sensible* questions about this one particular database; that makes the program *ad hoc*, but also makes it easier to write. Your grammar should meet two goals: avoidance of ambiguity by accepting only the "correct" expression of any query, and generality in the sense that a significant number of kinds of database queries should be handled by the grammar.

Examples of the kinds of queries your system should handle include the following:

```
(WHO ORDERED WIDGETS IN APRIL)

(WHAT CUSTOMERS IN AUSTIN ORDERED WIDGETS)

(SHOW CUSTOMER AND CITY FOR ORDERS OF TIRES)

(WHAT PRODUCTS ARE SUPPLIED BY GOODYEAR)

(WHAT DID MCC ORDER IN APRIL)
```

Some natural language systems produce a *parse tree* as output. In this case, the desired output is a database query. The output from parsing a given phrase will in general be a part of a query; by collecting the parts of the query, the parsing algorithm can output a completed query that can be executed by the database system. Use the supplied database system and database in the file `/u/cs381k/db.lsp` to answer the questions. Note that phrases in the English question will in general do one of two things to the database query:

1. Restrict the set of records applicable to the query (e.g., “in April”).
2. Request information to be reported from the chosen records (e.g., “Show customer and city...”).

One problem that needs to be handled in your parser is the position of the “scanner” that points to the current position in the sentence. The position must be moved as the sentence is read, but it must be saved at strategic points to allow for backup in case of a parsing error. A set of simple functions to do this has been provided in the file `/u/cs381k/atn.lsp` .

An easy way to write this program is to write a function to handle each type of phrase. Each such function will do two things: recognize and consume its particular type of phrase, and collect and synthesize the results of any sub-phrases to produce the total result. It may be convenient to use global variables to collect the results. Returning `NIL` from a grammar function may be used to indicate failure.

You can use the file `/u/cs381k/nldb.lsp` as a starting point for your lexicon and grammar. Expand it so that it allows logical and comparison expressions such as and, or, before, after:

```
(WHO ORDERED TIRES AND PLIERS OR WIDGETS FROM SEARS OR GOODYEAR)

(WHAT DID MCC ORDER AFTER APRIL AND BEFORE JUNE)
```

This assignment is rather open ended. You can go on endlessly expanding your grammar trying to cover more and more esoteric cases, and still would not get them all. You don’t have to. Take care of the obvious cases, and stop when things start to get hairy (e.g. don’t worry about things like ambiguous queries and precedence of AND and OR). Your final grammar will not be able to parse all meaningful queries, and it will also parse some that are not meaninful (to a human). That’s ok. That’s what NLP research is like: you can almost never solve the problem completely, so you should look for an elegant solution that works well in most important cases.

Report:

As usual, write a paragraph or two explaining and evaluating what you’ve done and what you learned from the assignment. Turn in a listing of your grammar, the program with detailed comments, and execution traces. EXTRA POINTS: Expand the coverage of your program to allow more complex queries than the examples shown above.