(1)

MECHANICAL INTELLIGENCE: RESEARCH AND APPLICATIONS

First Semiannual Report
Covering the Period October 1, 1978 to March 31, 1979

SRI Project 7910
Contract N00039-79-C-0118
ARPA Order No. 3175.28

October 8, 1979

Co-authors:
Gary G. Hendrix, Program Manager Natural Language
William H. Lewis, Systems Programmer

Co-principal Investigators:
Earl D. Sacerdoti, Associate Director
Daniel Sagalowicz, Computer Scientist
(415) 326-6200 x4840

Artificial Intelligence Center
Computer Science and Technology Division

Prepared for:

Defense Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, Virginia 22209

Attn: Lcdr. A. J. Dietzler

The views and conclusions contained in this document are those of the authors and should not be interpreted as representative of the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency of the U.S. Government.

SRI International
333 Ravenswood Avenue
Menlo Park, California 94025
(415) 326-6200
Cable: SRI INTL MPK
TWX: 910-373-1246

SRI International

MECHANICAL INTELLIGENCE: RESEARCH AND APPLICATIONS

First Semiannual Report
Covering the Period October 1, 1978 to March 31, 1979

SRI Project 7910
Contract N00039-79-C-0118
ARPA Order No. 3175.28

October 8, 1979

Co-authors:
Gary G. Hendrix, Program Manager Natural Language
William H. Lewis, Systems Programmer

Co-principal Investigators:
Earl D. Sacerdoti, Associate Director
Daniel Sagalowicz, Computer Scientist
(415) 326-6200 x4840

Artificial Intelligence Center
Computer Science and Technology Division

Prepared for:

Defense Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, Virginia 22209

Attn: Lcdr. A. J. Dietzler

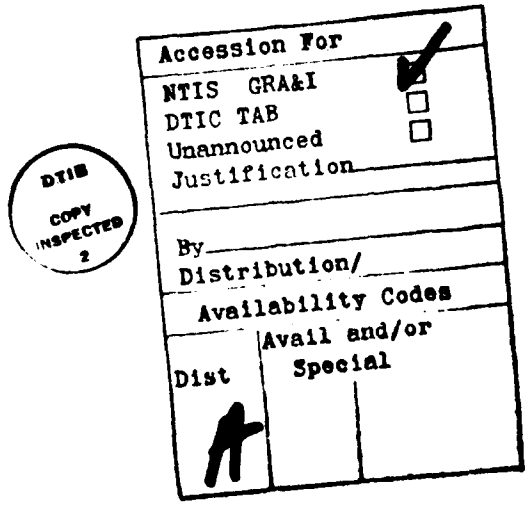
Approved:

Peter E. Hart, Director
Artificial Intelligence Center

David H. Brandin, Executive Director
Computer Science and Technology Division

CONTENTS

ILLUSTRATIONS



I INTRODUCTION

A. Overview

This document describes Since October 1, 1978, we have been engaged in research and development activity in support of the ARPA/IPTO Advanced Command and Control Architectural Testbed (ACCAT) program. The Our work has focused on three areas of interest: comprehension of English-language queries about information in a data base, intelligent data base retrieval, and the representation of knowledge within a computer's memory. The project also involves We are simultaneously constructing high-performance systems that run reliably in real time, and experimenting with and developing more advanced systems that possess extended capabilities but poorer performance characteristics.

Supported by a previous contract, a milestone version of the high-performance system, LADDER (Language Access to Distributed Data with Error Recovery), was conveyed to the ACCAT facility at the Naval Ocean Systems Center (NOSC) in January 1977. As new features have been checked out and proved viable, interim versions of LADDER with increasing levels of sophistication have been installed in the ACCAT facility. Under the current contract, such a milestone version was installed in August 1979, and a subsequent milestone version will be

installed in March 1980. Progress on the advanced systems is facilitating development of the high-performance systems and will provide supporting data for evaluation of the long-term role of mechanical intelligence in the ACCAT program.

B. Major Accomplishments

1. Standard LADDER System

LADDER, our high-performance system for answering English-language queries about information in multiple remote data bases, is being continually updated on the ACCAT facility. The system incorporates a natural-language front end, the Sophisticated Data Access (SODA) program for intelligent data base access, and the current version of the File Access Manager (FAM). The system now includes a limited interface with the situation display program developed at the Information Science Institute of the University of Southern California. Recently, improvements to the natural-language front end have been added in the area of paraphrase and ellipsis. For example, it is now possible to obtain a list of all existing paraphrases, and to delete some of them. As regards ellipsis, two main new capabilities have been added: it is now possible to use "What about X?" as an elliptical input. It is also possible to use elliptical fragments that are added at the end of the previous sentence, as in the sequence: "What are the U.S. carriers?" followed by "In the Mediterranean?".

To complete this partial list of improvements, the user can now request the system to present all ambiguous interpretations of an input. This latter capability however, is still quite primitive in its current implementation.

The natural-language front end now produces queries that ask explicitly for relations to be joined. For example, for the user query: "What is the distance from the Kennedy to its home port?" the natural-language component used to issue three SODA queries that in essence requested:

- What is the home port of the Kennedy?
- Where is that home port?
- What is the distance from the Kennedy to that position?

Now INLAND issues a single SODA query encompassing those three subqueries. This improvement has been implemented for most of the patterns parsed by the natural-language front end.

Other improvements in INLAND have been in the area of pronouns and paraphrasing. For example, it is possible to use pronouns inside macroparaphrases, where they are now understood correctly. Also the paraphrase capability has been extended to include the "IE" feature. For example, if a user issues a query such as:

```
L* FOX  (which is not parsed)
```

followed by:

```
IE WHAT IS THE LENGTH OF THE FOX
```

the IE query will be understood as if the user had issued the queries:

```
DEFINE "L* FOX" LIKE "WHAT IS THE LENGTH OF THE FOX"
L* FOX
```

New diagnostic capability has been added to increase the user's control over and understanding of the paraphrase facility. "What are the <definitions> ?" will list all the substitutions currently in effect. Feedback to the user when he defines a paraphrase has been greatly improved.

A delete command, "UNSUBSTITUTE", or "DON'T USE <new pattern> has been added, which allows the user to eliminate string substitution definitions.

We have now tentatively designed the patterns to be used with the Situation Display. They will allow the LADDER user full access to the Situation Display capabilities on several displays. The implementation has now been completed, although we may still modify this interface in the future to satisfy the needs of the ACCAT facility as well as the needs of the implementers of the Situation Display.

LADDER, producing map displays via the Situation Display and Graphics System (SDGS), has been utilized with both the stationary and mobile packet radio demonstrations. Users can now ask questions in English of a data base in Massachusetts and have the results displayed graphically, in real time, from the back of a van while driving down the highway in Menlo Park.

2. SODA

We have implemented a new version of SODA, the Sophisticated Data Access component of our demonstration system. The new SODA accepts quantified queries and provides a mechanism using "virtual relations" that gives a user the option of stating more precisely which access paths he wants SODA to follow in answering his query. The new SODA builds significantly more complex programs in the languages of the data base management system (DBMS) than did the old Intelligent Data Access (IDA) program. Inputs to the new SODA may specify operations involving nested quantification (e.g., "Give the last reported position of each carrier.") and multiple references to the same field in a single query (e.g., "Which subs are faster than the fastest cruiser?"). Furthermore, SODA's input language allows two types of maximization/minimization queries to be differentiated: those returning all the records that satisfy the maximum/minimum test, and those that return only one such record. The first type of query involves generation of a much more complex DBMS program.

On SRI's KL-10 computer, we have installed DBMS-20, a data base management system that supports a network model of data. (IDS, the DBMS used in the WWMCCS is such a system.) We have now generalized LADDER to access data distributed over two inhomogeneous DBMSs: the Datacomputer and DBMS-20. We are working to extend our current coverage of DBMS-20 to include using the links between record sets that have an owner/owned relationship.

3. DIAMOND

Progress continues on D-LADDER, our new language-processing subsystem based on the DIAMOND parser. D-LADDER will be able to produce all grammatically acceptable interpretations of a query, thereby providing the information necessary for the system to enter into a limited dialogue with the user for the purpose of resolving ambiguous queries. Limited tests have been conducted during the current period; we have succeeded in parsing "syntactically" 235 out of 240 queries with which we regularly test our LADDER demonstration system. Even though the "semantic" portion of the DIAMOND-based system is not yet complete, we have also processed completely about 50 questions, generating the correct SODA queries.

We have run experiments to compare the DIAMOND-based and LIFER-based systems. In general, the LIFER-based system is more robust and faster (by a factor of two). However, in the area of syntactic coverage the DIAMOND-based system is clearly superior.

4. TED: A Transportable English Data Manager

We are developing an interactive facility for defining a new data base for the data access part of our system, SODA. An initial implementation of this program now exists and we are working on improvements of its user interface. The program, a Transportable English Data Manager (TED), was released in August 1979 for installation

in the ACCAT facility. We are also currently developing an extension of this program that includes definition of the data base for the language front end, i.e., that will help the user to define the grammar needed for accessing the new data base in natural language. The system to be delivered in June 1979 will include such a capability, albeit limited in scope. TED is described in detail in section II.

II TED: A TRANSPORTABLE ENGLISH DATA MANAGER

A. Introduction

Historically, the acquisition of any computer system implied the need for the acquisition of a staff of specialists to provide an interface between the machine and the people who wanted results from it. Properly speaking, these specialists were as much a part of the system as the hardware itself, as both were absolute requirements if one were to get any results whatsoever.

As the relative costs of machine and human time changed (the computers became more reliable and less expensive, and the humans became more expensive and were given harder jobs), more emphasis was placed on simplifying the man/machine interface in most application areas. This has meant that the specialists can produce more for a given amount of their time, but it hasn't affected the status of the specialist as an integral part of any computer system. With the exception of a variety of "canned" programs with very limited abilities, the specialist must still be there to get anything done.

One of the long-range goals of Artificial Intelligence is to eradicate the need for the specialist by endowing machines with the "intelligence" to understand what is desired of them through

communication in natural language. This is no trivial task by any means, and the attainment of that goal, even in small measure, is still remote in all but a few limited domains.

One of those domains where practical results are presently being realized and where production systems have actually been released is in the domain of data base management. Computers have been used for years in data base management to provide the raw (or slightly processed) information by which decisions are regularly made. Here the computer specialist is being asked to obtain data from the machine in various forms, but NOT to provide answers to important questions.*

Because no one is asking the DBMS to answer the hard questions, it is feasible to eliminate the intermediate step of routine coding by specialists if the machine is capable of giving the same answers by understanding questions posed in English.

Many hours have been spent in designing systems that do a good job of understanding English questions about a given data base. Examples of these include LUNAR [9], LADDER, [6] and PLANES [8].

As for the case of novel data bases, systems that allow the naive user to set up the terms necessary for interaction in English are just appearing on the horizon. These systems are intended to allow a user

---

* By raw data we mean the answers to such questions as:
HOW MANY AIRCRAFT ARE WITHIN 200 MILES OF LUANDA?
HOW LONG WILL IT TAKE TASK FORCE 3 TO REACH LUANDA?
while the important question might be:
SHALL WE PREVENT THE CUBAN INVASION OF ANGOLA?

who has a fair idea about what he wants from his data base, but who lacks detailed knowledge of computer science and DBMSs, to construct a data base and define terms about it, which will allow him to ask questions in a subset of English and receive the desired answers. TED is one such system.

B. Overview of TED

TED, written in INTERLISP, is a complete package of functions which needs no additional programs or subsystems to run. Using TED, the user is immediately able to construct a data base; to provide the English terms desired; to populate that data base; to edit the terms, the data base structure, and the data itself; and to query the data base in either a subset of English or in the SODA query language [ ]soda} . In addition, TED provides the user with the capacity to access external data bases stored on the local machine or stored on a remote machine accessible over the ARPANET.

When using its internal data base, TED works with a simply structured relational data base (DB).* The DB is constructed of any number of relations (or files). Each relation is essentially an array in which entries are either single numbers or single words. Each row in the array is called a tuple (or record). Each column corresponds to an attribute (or field). A typical example is shown in Figure 1

---

* When using an external data base, TED can use any DB accessible via SODA [4], currently the Datacomputer [1] and DBMS-20 [2].

Relation: PERSONNEL*

NAME	AGE	SEX	SPOUSE	DEPT
JOHN	26	MALE	JOAN	SHOE
JOAN	29	FEMALE	JOHN	TOY
EMILY	39	FEMALE	*	SHOE

Relation: DEPARTMENT*

NAME	FLOOR	CHIEF	STORE
SHOE	3	KURT	BOB'S
TOY	2	JANE	BOB'S

Figure 1 Example of Relational Data Base.

Secondary bindings--such as indices, hash tables, and pointers--may be used for efficiency reasons, but are not of any particular importance to the functioning of TED. There are NO links between relations that are inherently defined by the data base itself. When making a query against the data base, the user may decide to join any relations over any fields, but it is not part of the DBMS's responsibility to decide which ones may and may not be joined. There are no inherent limitations placed on what data can be put in any of the fields by the user. That is, the DBMS does nothing more than to put information into the data base and retrieve information from it as requested by the user.

After the user has decided what relations he desires and what fields each relation will contain, TED proceeds to coax information from

the user concerning the contents of the fields and the terms by which each field is to be referred. The user is presented with a simple set of yes/no and multiple-choice questions, which require no particular linguistic knowledge, and can be answered by any user with a reasonable background in the English language. Answers to these questions are used to add specific information to a general English grammar designed to work with DBs. The grammar is defined in a top-down parser, LIFER [3], along with a set of response expressions that formulate the queries in a formal data base query language (SODA [4]). Once the user has answered the questions regarding the terms to be used with the data base and its structure, he is free to begin asking questions about the data base in the subset of English which he has helped define.

TED views each tuple (record) of a relation as being a description of some entity. In the first relation in Figure 1, each row describes an EMPLOYEE of some department. TED asks the user for terms that describe the entity (here EMPLOYEE, PERSON, WORKER), which will later be used when the user refers to them (e.g., how many EMPLOYEES are there?).

In addition, TED asks which fields should be returned as default answers to questions that do not mention any specific fields. For example, in answer to the question "LIST ALL THE PEOPLE," TED is not told which fields to return, so it uses the fields in the RESPONSE KEY which the user defined initially. In the PERSONNEL relation, NAME would be a good default response.

When acquiring language information, TED requests the user to categorize each field into one of three types.

* ARITHMETIC for any field containing numeric values that may be used in calculations. This includes AGE, and WEIGHT for example, but not ZIP-CODE. TED further asks about comparative and superlative terms to go along with the field name and its synonyms--such as YOUNGER, YOUNGEST, OLDER and OLDEST for AGE.

* FEATURE for any field that contains Boolean values referring to the existence of a certain feature. For example, there may be a field, say RADARFLAG, which has the value R whenever the corresponding ship has a radar, and the value * otherwise.

* SYMBOLIC for any field, the values of which may be arbitrary strings. These fields are divided into two subcategories:

  Those which are commonly used as modifiers such as the fields COLOR (where one may refer to a BLUE BOX) and SEX (where one may refer to the FEMALE EMPLOYEE).

  Those which are not used as modifiers such as the fields DATE.OF.BIRTH (one cannot refer to the 9/5/52 EMPLOYEE), and SPOUSE (one cannot refer to the JOAN EMPLOYEE -- the employee whose spouse is JOAN).

C. TED's File Structure

In any DB containing more than one relation, the user will want to define links between them. In relational DBs there do not exist any links between relations defined by the DBs themselves; rather, every link is stated explicitly in the data language query,* as a comparison

---

* The word QUERY is used here to refer to a formal query language such as in the next footnote, while the word QUESTION is used to refer to an English question that will get translated into a query.

between values in the pairs of fields being joined. Thus, in TED the linking is done as a direct result of the parsing, and TED therefore looks for the linguistic motivation behind the linking. Two separate cases are considered.

The first is the case where the user specifies in the question the fields he wants joined. For example, in the data base introduced in Figure 1, the question, "HOW OLD IS JOAN'S SPOUSE?"* specifies that a join of the relation PERSONNEL* to itself is to be made using the SPOUSE field. It is evident that SPOUSE is intended to be linked with the NAME field of the same relation to obtain the answer that JOHN is 26. TED is not clever enough to be able to guess this link, so it must have been told that this is indeed the intended join before the user can ask the question. This is done when the data base is initially defined.

The second kind of linking occurs when the fields to be joined are not mentioned in the question. Consider the question: "WHO IS EMILY'S CHIEF?"** A join is to be made between the two relations, PERSONNEL* and DEPARTMENT*, with a link taken over the DEPT and NAME fields, neither of which is mentioned in the question. TED is expected to figure out that

---

* This is translated into SODA as

```
[(IN P1 PERSONNEL* ((NAME P1) EQ "JOAN"))
 (IN P2 PERSONNEL* ((SPOUSE P1) EQ (NAME P2)))
 (? (AGE P2))]
```

** This is translated into SODA as:

```
[(IN P1 PERSONNEL* ((NAME P1) EQ "EMILY"))
 (IN D1 DEPARTMENT* ((NAME D1) EQ (DEPT P1)))
 (CHIEF D1))]
```

the field CHIEF is to be INHERITED from the DEPARTMENT* relation by the PERSONNEL* relation. Once again it is necessary for the user to have predefined a link between the two relations over those fields (see Appendix lines 34 - 38)

The second case concerns itself solely with links between relations that allow field values to be inherited from one relation by another. A field to be inherited may require joins to be made between any number of relations. For example, consider the additional relation in Figure 2

Relation: STORE*

NAME	CITY	MANAGER
BOB'S	RIPON	ALBERT
JOE'S	WAPAUN	MARTHA

Figure 2 Extension to Data Base.

This is linked to the relation DEPARTMENT*, but one may also want to find out who was JOAN'S MANAGER, which would require joining PERSONNEL* to DEPARTMENT* via a link between DEPT and NAME, and then joining DEPARTMENT* to STORE* via a link between STORE and NAME. In TED, when a new link is defined, that link is added to the INHERITANCE CHAINS* of every relation that can be reached from either of the two relations being linked.

These inherited chains in TED are directional (the CHIEF field in DEPARTMENT* is inherited by the relation PERSONNEL*, so that you can ask

---

* We use the word CHAIN to specify that it is possible to find a path from one relation to another via previously established individual links.

"WHO IS JOAN'S CHIEF?"; but the AGE field of PERSONNEL* is not inherited by DEPARTMENT*, so that the question "HOW OLD IS THE SHOE DEPARTMENT?" will not get parsed), but there is no requirement that inheritance links all be in the same direction. It is possible to create a rather convoluted file structure in TED which is circular, or even has links from every relation to every other relation. This brings up a form of the multipath problem [7] which TED handles in the following manner:

Assume that a user has asked a question, "WHO IS THE BOSS OF JOE?" which refers to a field that doesn't exist in the same relation as the subject (see Figure 3).

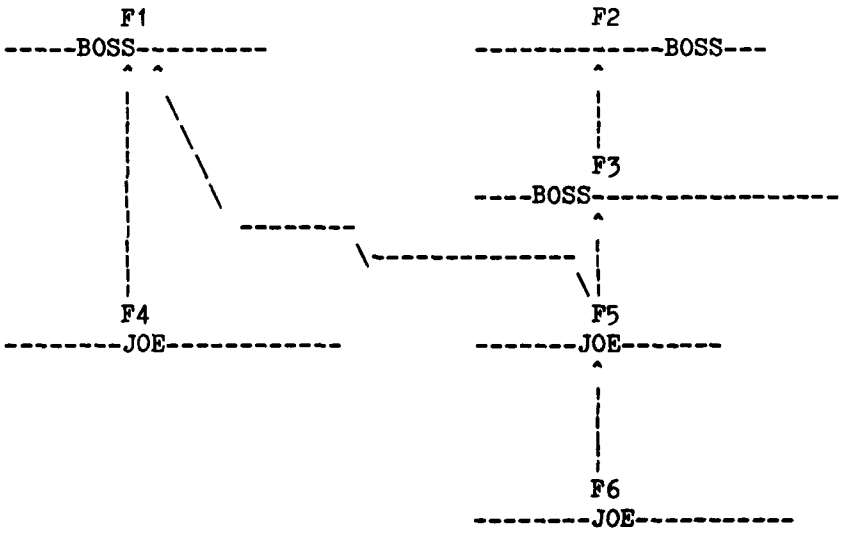


Figure 3 A Multipath Problem.

Some path from the subject to the field must be established if a query is to be created. Assume further that the subject and the field

both exist in several different relations. All possible chains are computed, and then eliminated if

1) The destination field (BOSS) exists in a relation in the middle of the chain.

2) The source field (the subject JOE) exists in a relation in the middle of the chain.

The remaining chains are all retained to the end of the parsing process, during which some may be eliminated by other restrictions of the question. Then, the user is presented with a menu of the various possible queries he may choose from, each translated back into a stilted English paraphrase showing the possible paths.

In the data base shown in Figure 3, the possible paths from JOE to his BOSS are shown in Figure 4. Paths 1, 2, and 3 are eliminated by rule 1; Paths 3 (again) and 6 are eliminated by Rule 2, leaving paths 4, 5, and 7 for the user to decide among.

```
1 (F6 F5 F3)    2 (F6 F5 F1)    3 (F6 F5 F3 F2)
4 (F5 F1)       5 (F5 F3)       6 (F5 F3 F2)
7 (F4 F1)
```

Figure 4 Possible Paths From JOE to His BOSS

D. Use of TED to Access Data on External Data Bases

In addition to operating in a locally created data base, TED has the capacity to access data on DBs that have been previously created on normal DBMSs. To accomplish this, TED calls an external version of SODA, which takes care of translating the SODA query into the query language for the particular DBMS. SODA also takes care of coordinating the efforts of however many different machines it deems necessary to answer the question, and watches out for data base problems (systems crashing, and so forth.)

From the user's point of view, there is no difference between formulating a question (or a SODA query) intended for processing on a foreign computer and one to be processed locally. As in one of the examples (lines 57 - 59 in the Appendix), the user is able to submit the same query first to the local data base, and then to a foreign data base by setting a flag and repeating the question.

A certain amount of information about the external relations and machines must be initialized for IDA [7] (the part of SODA that deals with the data base navigation) before a query is sent off. Most of this is set up in parallel with the language acquisition part of TED, but there is some very specific information that the user must also supply to FAM [5] (the file manager of IDA). To do this, the user calls the FAM editor and tells it:

The file status of the file (FILE or PORT)

The kind of DBMS the computer has (several known machines are predefined)

The login and directory accounts

The local name of the file

After this, the user can access the data base on an external machine using TED. From here on, depending upon the setting of a special flag, TED will either access the local or the remote data base. The FAMTABLE editor is shown in lines 43 - 45 of the Appendix.

E. TED's Schema Editor

Once the user has finished defining a relation (or relations), he is free to change his mind about anything in the relation (names, synonyms, fields, spellings, values, and so on). The schema editor will present all the terms the user defined and allow him to delete them, respell them, or leave them as they are. After going through all the defined terms and fields, the user is presented with the structure (inter-relation links and joins) of the data base and again given the option of deleting them or leaving them. He may also add new links or more joins to an existing link. If the user pleases, there is nothing stopping him from starting out with a data base about the U.S. navy and editing it into a data base concerning NATO land units. An example of the schema editor is shown in lines 46 - 47 of the Appendix.

F. TED's Data Base Editor

Once a relation is defined, the user will presumably desire to populate it with data from his particular domain. The internal editor will add and edit records in the relation created via a relatively simple set of commands. The user is able to move about freely in the relation, changing values by typing over the existing ones. A limited search function is available, along with a print function that prints a table of the contents of any particular internal relation. Two editors are available: one designed for use with terminals with cursor control, and a similar one that will operate on hard-copy terminals. In lines 48 and 49 of the Appendix is an example of an interaction with the hard-copy terminal editor.

G. Conclusion

TED marks a major milestone on the road to transportability: it can readily be used by nonprogrammers and nonlinguists to describe new data bases so that users will subsequently be able to query the data base in a subset of English. The experience of writing a system such as TED has provided many insights into what is needed to enable a noncomputer scientist to create his own natural-language interface to data bases. Moreover, the simplicity of design and the engineering aspects that are already part of the TED system indicate that it will be feasible to install such a system as part of a small "intelligent terminal."

The existence of small systems, such as TED, that allow naive users to do all of their own data base management without the assistance of highly paid professionals should have a very positive effect on both the business world and on the computer programmers, who will be freed from the mundane tasks presently imposed upon them.

Appendix

TED's USER INTERFACE

Appendix

TED's USER INTERFACE

This appendix describes how TED operates from the user's perspective, showing examples from one interaction. First, the following procedural notes are of interest to users of the system. TED employs the character ">" as its prompt. The user may always type "?" in response if he desires a more detailed explanation of what any question is. The system responds to the normal TOPS-20 control character set, so that typing a <DEL> will rubout the last character typed, and <crtl-U> will rubout the entire line being typed. The character "¦" is used at the front of every line that is part of the explanation given in response to the "?". The question is always repeated after it has been explained.

The user must complete every statement with either a "<CR>" or a right square bracket "]" (the internal data base editor has an exception to this, which will be covered later). Whenever the user ends a line with a single right parenthesis ")", or a <CR> preceded by a space, LISP prints three dots on the line after that input to give the user the opportunity to continue the line.

The only time a user should use parentheses is when he is defining a multiword, e.g., >BOAT SHIP (SEA GOING VESSEL].

In the following transcript, all user input is in upper case and TED outputs in both, using capitals for emphasis. All relation names have stars at the end (SHIP* UNIT*) to distinguish them from generic names, which could otherwise be the same and potentially confusing. Additional comments are bounded by equal signs "==========". Numbers have been added in front of questions and prompts. In actual use, one can type in lower case (it will be translated into upper case), or choose not to add stars at the end of relation names.

****

```
NEW file, EDIT, OTHER, or query     For help, type "?" after any ">"
1  > ?

 TED is a system  for defining the structure of a data base,
                  for populating the data base with data,
                  for editing the data and the data base structure,
                  and for posing questions about the data
                       in the SODA query language,
                       and in ordinary English.

 TED uses a multiple-file, relational data base structure, which may
 be viewed as a set of interconnected data tables.

 After TED types its one-line header message
 "NEW file, EDIT, ?, or query          For help, type '?' after any '>'"
 the user may

 1] Type "NEW <CR>" (the word NEW followed by a carriage return)
    to initiate a dialogue through which the structures of new files
    (tables) are defined.

 2] Type "EDIT <CR>" to initiate a dialogue through which
      new data may be entered into defined files,
      and existing data entries may be altered.

 3] Type queries expressed in the SODA language about the data in the
     data base, viz:
```

```
    >SODA( ((IN S1 SHIP) (? (S1 NAM))) )

4] Type queries in ordinary English about the data in the data base.
   No punctuation is required, and no distinction is made between
   upper and lower case.

5] Type "OTHER <CR>" to perform infrequent tasks, such as saving a
   data base for subsequent use, or invoking an existing data base.

Whenever the system asks a question, help may be obtained by typing a
"?" followed by a carriage return.

To abort processing:
              Type control-E to return to the top level of TED.
              Type control-D to return to the top level of INTERLISP.
              Type control-C to return to the EXEC level of TOPS-20.


NEW file, EDIT, OTHER, or query      For help, type "?" after any ">"
2 > NEW
The name of the new file is (type file name)
3 > ?

TED uses a multiple-file, relational data base structure, which may be
viewed as a set of interconnected data tables.  Each TED file (table)
and each field (i.e., column of a table) has a name, which must begin
with a letter and be no more than 15 characters in length.


An example file is

                          EMP*
-------------------------------------------------------
NAME      SALARY  SEX  AGE  SPOUSE    STORE  DEPARTMENT
----      ------  ---  ---  ------    -----  ----------
J.FISHER  17000    M   38   M.FISHER  PLAZA  HARDWARE
L.KING    12000    F   24   B.KING    PLAZA  SHOE
M.FISHER  15000    F   37   J.FISHER  PARK   SHOE


Its file name is EMP* and its fields are named NAME, SALARY, SEX, AGE,
etc.

The name of the new file is (type file name)
4 > SHIP*
The fields of file SHIP* are (type sequence of fields)
5 > NAM TYPE MCS UICVCN MED  <======These are the actual names
                                          from the blue file======
The RESPONSE KEY for file SHIP*
```

```
is composed of the fields (type sequence of fields)
6 > NAM UICVCN  <=======These will be used as default answers==========
What names do you want to use to refer to a subject of the SHIP* file?
7 > SHIP
8 Is a SHIP human? (YES or NO) > NO

-------------------------------------------------------------------------
9  NAM is:
1       an arithmetic field (values may be added, subtracted, etc.)
2       a feature field (values are Boolean, T or F, YES or NO ...)
3       a symbolic field (values are usually nouns or adjectives)
(Please type 1, 2, or 3) > 3
10 Can the values of this symbolic field be used as modifiers? (YES or
NO) > NO
If there are other names for the attribute NAM, please list them.
(type a sequence of names, or a <CR> if there are none)
11  > NAME

-------------------------------------------------------------------------
12  TYPE is:
1       an arithmetic field (values may be added, subtracted, etc.)
2       a feature field (values are Boolean, T or F, YES or NO ...)
3       a symbolic field (values are usually nouns or adjectives)
(Please type 1, 2, or 3) > 3
13  Can the values of this symbolic field be used as modifiers? (YES
or NO) > YES
14  If there are other names for the attribute TYPE, please list them.
(type a sequence of names, or a <CR> if there are none)
>

-------------------------------------------------------------------------
15  MCS is:
1       an arithmetic field (values may be added, subtracted, etc.)
2       a feature field (values are Boolean, T or F, YES or NO ...)
3       a symbolic field (values are usually nouns or adjectives)
(Please type 1, 2, or 3) > 1
16  Please type in the minimum and maximum values for DCLIMITS
> ?

| This question deals with the specific information required to access
| information on foreign computers. If you don't have any predefined
| data bases on any datacomputers
| just type a <CR> and don't worry about it any more.
| When using SODA to access information in external data bases, the
| data base query language needs to know what the limits on numeric
| fields are if it is to be able to do any arithmetic calculations.
| If this is an integer field, then just type in the minimum and maximum
| values that can be found in the field, e.g.,  > 0 999
```

```
If the values of this field are to be floating-point numbers, then the
data base will turn them into strings (ie 99.9 becomes '99.9'). Please
type the floating point numbers that form the limits of the field, but
be sure not to put any leading or trailing spaces on the maximum
value.
      > 0.0 9.99
That is 9.001 is fine, but 9.990 isn't as it will get interpreted as
'9.99'

17  Please type in the minimum and maximum values for DCLIMITS
> 0 999
18  If there are other names for the attribute MCS, please list them.
(type a sequence of names or a <CR> if there are none)
> SPEED (CRUSING SPEED]  <=======Note the misspelling===============
19  If there is a word wwww such that the question
        HOW wwww IS THE SHIP ?
is equivalent to
        WHAT IS THE MCS OF THE SHIP ?
please type wwww (else type <CR>). > FAST
20  If there are synonyms for FAST, please list them
(type a sequence of words and multiwords)
>
21  Which is correct:
        1) FASTER
        2) MORE FAST
        3) other
(Type 1, 2, or 3 followed by a carriage return)  > 1
22  If there are antonyms for FAST, please list them
(type a sequence of words and multiwords)
> SLOW                                    ====TED will figure out that it
                                          should use FASTEST============

23  Which is correct:
        1) SLOWER
        2) MORE SLOW
        3) other
(Type 1, 2, or 3 followed by a carriage return)  > 1

-------------------------------------------------------------------------
24  UICVCN is:
1       an arithmetic field (values may be added, subtracted, etc.)
2       a feature field (values are Boolean, T or F, YES or NO ...)
3       a symbolic field (values are usually nouns or adjectives)
(Please type 1, 2, or 3) > 3
25  Can the values of this symbolic field be used as modifiers? (YES
or NO) > NO
26  If there are other names for the attribute UICVCN, please list them.
(type a sequence of names or a <CR> if there are none)
>
```

```
------------------------------------------------------------------------
27  MED is:
1       an arithmetic field (values may be added, subtracted, etc.)
2       a feature field (values are Boolean, T or F, YES or NO ...)
3       a symbolic field (values are usually nouns or adjectives)
(Please type 1, 2, or 3) > 2
28  A feature field must have a single positive value that TED can look
for Typically this would be T, TRUE, YES, POSITIVE, etc. What will this
value be for MED (type a single word) > D
29  If there are other names for the attribute MED, please list them.
(type a sequence of names or a <CR> if there are none)
> MEDICAL

------------------------------------------------------------------------
30  Does the file SHIP* have any links to itself? (YES or NO) > NO

NEW file, EDIT, OTHER, or query      For help, type "?" after any ">"
31  > NEW
The name of the new file is (type file name)
32  > UNIT*
The fields of file UNIT* are (type sequence of fields)
33  > ANAME UICVCN CONAM HOGEO

************************************************************************
*               TED runs through the same questions                    *
*                        as above.                                     *
************************************************************************


UNIT* has links to or from which of the following files:
 (SHIP* UNIT*)
34  > SHIP*

************************************************************************
35  For the link between UNIT* and SHIP*
which of the following is true:
        1) For each SHIP which is the subject of one of
the records of file SHIP* , SHIP* provides information
associating that SHIP with a unique UNIT of file UNIT*
        2) For each UNIT which is the subject of one of
the records of file UNIT* , UNIT* provides information
associating that UNIT with a unique SHIP of file SHIP*

> 1
The fields for the two files in question are:
     UNIT*                              SHIP*
     ANAME                              NAM
     UICVCN                             TYPE
     CONAM                              MCS
     HOGEO                              UICVCN
                                        MED
```

```
36  The fields of file SHIP* which identify a UNIT of file UNIT*
are (type a sequence of fields from file SHIP* ) > UICVCN

37  The field of UNIT* associated with field UICVCN of file SHIP*
is > UICVCN
The following are properties of UNIT which are the subjects of
the UNIT* file:
 ANAME
UICVCN
CONAM
HOGEO
38  The current link associates with each SHIP of file SHIP* a
 UNIT of file UNIT* . Which properties of the UNIT (from file UNIT*)
are inherited through this link as properties of the associated SHIP
(from file SHIP* )?
> CONAM HOGEO

==========================================================================
39  In addition to the links previously declared, file UNIT* is linked
to which of the following files:
 (SHIP* UNIT*)
(Type the name of a previously declared file or <CR>)? >

NEW file, EDIT, OTHER, or query      For help, type "?" after any ">"
40  > OTHER

The other commands which are available to you are:
1  ES     to edit a file schema and the terms associated with it
2  EL     to edit only the links between files
3  EF     set up and/or edit a FAM table for a file
4  IF     to create the language component for a file defined by FILEDATA
5  IA     to create the language component for a file defined by ACCAT
6  SHOW   to display both the language and structure of a file
7  SS     to display only the structure of a file
8  SAVE   to save the language, structure, and internal data base
9  CLEAN  to clean out grammar internal data base and FAMTABLE
10 XSODA  to use SODA to access an external data base
11 ISODA  to use SODA to access an InterLisp internal data base
12 PT     to print a table of the information in an internal file
13 SETL   to set up the lexicon for a known file on an external machine
14 NONE   to do none of the above, and return to top level TED
15 PP     to call the paraphrase function from LIFER
Please type either the number or the command name. (At top level the
command name may be typed alone)
41  > 6
42  Which of these files:
 (SHIP* UNIT*)
do you wish to see? (CR for all) >
```

```
==========================================================================
                              File:- SHIP*

NAM     <LIT.ATTR>      ***INDEXED*** <==This means that the attribute==
          NAME                        ===is in the RESPONSE KEY for the=
TYPE    <LIT.ATTR>                    ===file and is inverted on.=======
MCS     <NUM.ATTR>
          (CRUSING SPEED) SPEED         <==Misspelling is still here===
        <+NUM.ADJ>
          FAST
         <COMP.ADJ>
          LT: SLOWER
          GT: FASTER
         <SUPER>
          MAX: FASTEST
          MIN: SLOWEST
UICVCN  <LIT.ATTR>      ***INDEXED***
MED     <FEATURE>
          MEDICAL
        Positive feature value: D

              LINKING AND CHAINING INFORMATION

 SHIP*
==>
File------via-chain--------------------joining on fields--------------
UNIT*     (UNIT* SHIP*)                 (UICVCN UICVCN)

 SHIP*
 <==
File------via-chain--------------------joining on fields--------------


                    LINKS VIA SINGLE FIELDS

                SHIP*.UICVCN    ==>     UNIT*.UICVCN

==========================================================================
==========================================================================
                              File: UNIT*

ANAME   <LIT.ATTR>      ***INDEXED***
          NAME
UICVCN  <LIT.ATTR>      ***INDEXED***
CONAM   <LIT.ATTR>
          COMMANDER
HOGEO   <LIT.ATTR>
          (HOME PORT) PORT
```

```
                    LINKING AND CHAINING INFORMATION

 UNIT*
==>
File------via-chain--------------------joining on fields--------------

 UNIT*
 <==
File------via-chain--------------------joining on fields--------------
SHIP*     (UNIT* SHIP*)                  (UICVCN UICVCN)


                         LINKS VIA SINGLE FIELDS

                    SHIP*.UICVCN     ==>      UNIT*.

================================================================================

********************************************************************************

NEW file, EDIT, OTHER, or query      For help, type "?" after any ">"
43  > EF
44  Of the tables defined:
 (SHIP* UNIT*)
which ones do you wish to edit? (<CR> for all) >
45  Every file must be defined on the datacomputers as either:
1)      a FILE
2)      a PORT
which one is SHIP* (please type either 1 or 2) > 1
What computers is this file located on (even if the file has a different
name on that computer)? > SRI-KL CCA1
Please type the login directory for SHIP* on the machine SRI-KL
if it is NOT %TOP.ACCAT.GUEST >
Please type the file directory for SHIP* on the machine SRI-KL
if it is NOT %TOP.ACCAT. >
Please type the local name for SHIP* on the machine SRI-KL
if it is NOT SHIP* > SHIP
Please type the login directory for SHIP* on the machine CCA1
if it is NOT %TOP.ACCAT.GUEST >
Please type the file directory for SHIP* on the machine CCA1
if it is NOT %TOP.ACCAT. >
Please type the local name for SHIP* on the machine CCA1
if it is NOT SHIP* > SHIP
Finished defining file and ready to write it out? > ?
```

```
 If you have created all the files that you intend to, and have defined
 their structure, and are satisfied with it, and now wish to use SODA
 to access files that are on these computers, THEN the answer is YES.
Finished defining file and ready to write it out? > N
Every file must be defined on the datacomputers as either:
1)      a FILE
2)      a PORT
which one is UNIT* (please type either 1 or 2) > 1
What computers is this file located on (even if the file has a different
name on that computer)? > SRI-KL CCA1
Please type the login directory for UNIT* on the machine SRI-KL
if it is NOT %TOP.ACCAT.GUEST >
Please type the file directory for UNIT* on the machine SRI-KL
if it is NOT %TOP.ACCAT. >
Please type the local name for UNIT* on the machine SRI-KL
if it is NOT UNIT* > UNIT
Please type the login directory for UNIT* on the machine CCA1
if it is NOT %TOP.ACCAT.GUEST >
Please type the file directory for UNIT* on the machine CCA1
if it is NOT %TOP.ACCAT. >
Please type the local name for UNIT* on the machine CCA1
if it is NOT UNIT* > UN IT

**********PLEASE RESPOND WITH A SINGLE WORD, e.g.   >ONEWORD*******

Please type the local name for UNIT* on the machine CCA1
if it is NOT UNIT* > UNIT
Finished defining file and ready to write it out? > Y
If you desire to use a file name other than FAMTABLE.TMP, please type
it in (otherwise just <CR>) >
Writing out FAMTABLES on the file: <LEWIS>FAMTABLE.TMP.1

and now reading them into FAMINIT

==========FAMINIT is the function that opens the port===========
==========to the datacomputers with the data from above==========

*********************************************************************
```

```
**********************************************************************

NEW file, EDIT, OTHER, or query      For help, type "?" after any ">"
46  > ES
47  Of the tables defined:
 (SHIP* UNIT*)
which ones do you wish to edit? (<CR> for all) > SHIP*
----------------------------------------------------------------------
This editor allows you to change the terms used to describe subjects,
field names, field values, and the structure of a file. The system will
type the term presently in use and give a ">" as a prompt. You may:

a)      type a <CR> if the term is OK and you don't want to change it
b)      type a "\" if you want to delete the term
c)      type another word or multiword if you wish to correct spelling


               File name: SHIP* >
Generic name: SHIP >
Generic name: THING >
What names do you want to use to refer to a subject of the SHIP* file?
>
Literal field name: NAM >
        NAME >
If there are other names for the attribute NAM, please list them.
(type a sequence of names or a <CR> if there are none)
>
Literal field name: TYPE >
If there are other names for the attribute TYPE, please list them.
(type a sequence of names or a <CR> if there are none)
>
Numeric field name: MCS >
Other names for the field MCS
        (CRUSING SPEED) > CRUISING SPEED    <===Correction of
        SPEED > "SP"                        ====misspelling===
YOU MUST RESPOND WITH A CR, \, AN ATOM, OR A LIST OF ATOMS

        SPEED >
Numerical adjectives for the field MCS
        FAST >
Comparative adjectives for the field MCS
        SLOWER >
Polarity: LT >
        FASTER >
Polarity: GT >
Superlatives for the field MCS
        SLOWEST >
Polarity: MIN >
        FASTEST >
Polarity: MAX >
Do you wish to add any other terms for MCS (YES or NO) >
```

```
Literal field name: UICVCN >
If there are other names for the attribute UICVCN, please list them.
(type a sequence of names or a <CR> if there are none)
>

Feature field name: MED >
        MEDICAL >
Positive feature value: D >
If there are other names for the attribute MED, please list them.
(type a sequence of names or a <CR> if there are none)
> (DOCTOR ON BOARD]
If you wish to add some more fields to this file, please type them >
-------------------------------------------------------------------
This is the structure editor, which allows you to change the inter-file
linkage information. The system will type out the linking information
and you have two options:

a)      type a <CR> if the link is OK and you don't want to change it
b)      type a "\" if you want to delete the link

Links between files are indicated by a double arrow "==>" whereas the
fields which are to be joined to make these links appear below the
respective file names with single arrows "-->" connecting them.
        SHIP*       ==>  UNIT* >
         UICVCN      -->         UICVCN >
Other fields to be joined? (YES or NO) >

        EXPLICIT LINKS
        SHIP*.UICVCN          ==>        UNIT*.UICVCN >
SHIP* has links to or from which of the following files:
 (SHIP* UNIT*)
>




*********************************************************************

NEW file, EDIT, OTHER, or query      For help, type "?" after any ">"
48  > EDIT
49  Of the tables defined:
 (SHIP* UNIT*)
which ones do you wish to edit? (<CR> for all) >

SPACE, > = Move right     DEL, < = Move left          ^ = Move up
LINE-FEED = Move down     RETURN = Go to the front of this line
ctrl-S = Find a value     ctrl-A = Add new records     ctrl-Q = quit
! = Reprint the field names                    ? = retype this header
A number (followed by CR) directly after a single : = Go to that record
RETURN after a : = Repeat the previous search
```

```
The number of :'s indicates which row of the record that the editor is
on. To change a value, move the cursor under that value and type in the
new value followed by either a space or CR
:    NAM            TYPE           MCS            UICVCN         MED


ADD RECORD. Finish with a ctrl-Q

1    ? JFK          ? CVA          ? 34           ? N1           ? D
CVA SHIP? >  ?

Will you refer to the CVA SHIP or is there an adjective you
will use in place of CVA ? (If CVA is OK, use <CR>, otherwise
type all of the adjectives to be used for CVA including CVA
if desired ) >

CVA SHIP? >
2    ? FOX          ?  CCV         ? 30           ? N2           ? *
CCV SHIP? > CARRIER
3    ? ^Q
2    FOX            CCV            30             N2             *
:^Q

==========And now the editor gets called with the second file==========


NEW file, EDIT, OTHER, or query      For help, type "?" after any ">"
50  > PT        <=======See the options list above================

51  The files presently defined are:
 (SHIP* UNIT*)
Which one do you wish to work with? >
```

FILE SHIP*

NAM	UICVCN	TYPE	MCS	MED
JFK	N1	CVA	34	D
FOX	N2	CCV	30	*

FILE UNIT*

ANAME	UICVCN	CONAM	HOGEO
CTG.1	N1	JOE	LISB
CTG.2	N2	JEAN	NEWY

****

```
NEW file, EDIT, OTHER, or query      For help, type "?" after any ">"
52  > LIST THE SHIPS
PARSED!
What are the UICVCNs (SHIP* 34) and NAMs (SHIP* 34) of the SHIPs
(SHIP* 34) ?

<==========TED translates the user's question into a SODA query,========
===========then translates that back into English so that the user======
===========can see what TED has done==================================>

(SHIP* 34 NAM = FOX)(SHIP* 34 UICVCN = N2)

(SHIP* 34 NAM = JFK)(SHIP* 34 UICVCN = N1)


NEW file, EDIT, OTHER, or query      For help, type "?" after any ">"
53  > HOW MANY SHIPS ARE THERE WITH MEDS
PARSED!

Count the SHIPs (SHIP* 37) with MED (SHIP* 37) equal to D

(= 1)


NEW file, EDIT, OTHER, or query      For help, type "?" after any ">"
54  > WHAT IS THE FASTEST MEDICAL SHIP
PARSED!

What are the UICVCNs (SHIP* 49) and NAMs (SHIP* 49) of the FASTEST
SHIPs (SHIP* 49) with MED (SHIP* 49) equal to D ?

(SHIP* 49 NAM = JFK)(SHIP* 49 UICVCN = N1)


NEW file, EDIT, OTHER, or query For help, type "?" after any ">"
55  >WHA  ARE THE HOME PORTS OF THE FASTEST MEDICAL SHIP
PARSED!
```

```
What are the UICVCNs (SHIP* 52) and NAMs (SHIP* 52) and HOGEOs
(UNIT* 53) of the FASTEST SHIPs (SHIP* 52) with MED (SHIP* 52) equal
to D with UICVCN (SHIP* 52) equal to the UICVCN (UNIT* 53) of the
UNITs (UNIT* 53) ?

(UNIT* 53 HOGEO = LISB)(SHIP* 52 NAM = JFK)(SHIP* 52 UICVCN = N1)


NEW file, EDIT, OTHER, or query      For help, type "?" after any ">"
56  > WHICH SHIPS HAVE HOME PORT = NEWY
PARSED!

What are the UICVCNs (SHIP* 54) and NAMs (SHIP* 54) of the SHIPs
(SHIP* 54) with UICVCN (SHIP* 54) equal to the UICVCN (UNIT* 55) of
the UNITs (UNIT* 55) with HOGEO (UNIT* 55) equal to NEWY ?

(SHIP* 54 NAM = FOX)(SHIP* 54 UICVCN = N2)



NEW file, EDIT, OTHER, or query      For help, type "?" after any ">"
57  > COUNT THE SHIPS
PARSED!

Count the SHIPs (SHIP* 32)

(= 2)

NEW file, EDIT, OTHER, or query      For help, type "?" after any ">"
58  > XSODA     <=======See OTHER options again===============

NEW file, EDIT, OTHER, or query      For help, type "?" after any ">"
59  > COUNT THE SHIPS
PARSED!

Count the SHIPs (SHIP* 30)

(= 214)
```

REFERENCES

1. Computer Corporation of America, "Datacomputer Version 1 User Manual," Cambridge, Massachusetts (August 1975).

2. DEC System 20, DBMS Programmer's Procedures, Manual No. DEC-20-APPMB-A-D.

3. G. G. Hendrix, "The LIFER Manual: A Guide to Building Practical Natural Language Interfaces," SRI Artificial Intelligence Center Tech. Note 138, SRI International, Menlo Park, California (February 1977).

4. R. C. Moore, "Handling Complex Queries in a Distributed Data Base," SRI Artificial Intelligence Center Tech. Note 170, SRI International, Menlo Park, California (September 1979).

5. P. Morris and D. Sagalowicz, "Managing Network Access to a Distributed Data Base," Proc. Second Berkeley Workshop on *Distributed Data Management and Computer Networks*, Berkeley, California (May 1977).

6. E. D. Sacerdoti, "*Language Access to Distributed Data with Error* Recovery," Proc. 5th International Joint Conference on Artificial Intelligence, Cambridge, Massachusetts, (August 1977).

7. D. Sagalowicz, "IDA: An Intelligent Data Access Program," Proc. Third International Conference on Very Large Data Bases, Tokyo, Japan (October 1977).

8. D. Waltz, "Natural Language Access to a Large Data Base: an Engineering Approach," Proc. 4th International Joint Conference on *Artificial Intelligence*, Tbilisi, U.S.S.R (September 1975).

9. W. A. Woods, "Transition Network Grammars for Natural Language *Analysis*," *Communications of the ACM*, pp. 591-606, (October 1970).