TEAM: A TRANSPORTABLE NATURAL-LANGUAGE INTERFACE SYSTEM

Barbara J. Grosz
Artificial Intelligence Center
SRI International
Menlo Park, CA 94025

A. Overview

A major benefit of using natural language to access the information in a database is that it shifts onto the system the burden of mediating between two views of the data: the way in which the data is stored (the "database view"), and the way in which an end-user thinks about it (the "user's view"). Database information is recorded in terms of files, records, and fields, while natural-language expressions refer to the same information in terms of entities and relationships in the world. A major problem in constructing a natural-language interface is determining how to encode and use the information needed to bridge these two views. Current natural-language interface systems require extensive efforts by specialists in natural-language processing to provide them with the information they need to do the bridging. The systems are, in effect, handtailored to provide access to particular databases.

This paper focuses on the problem of constructing transportable natural-language interfaces, i.e., systems that can be adapted to provide access to databases for which they were not specifically handtailored. It describes an initial version of a transportable system, called TEAM (for Transportable English Access Data Manager). The hypothesis underlying the research described in this paper is that the information required for the adaptation can be obtained through an interactive dialogue with database management personnel who are not familiar with natural-language processing techniques.

B. Issues of Transportability

The insistence on transportability distinguishes TEAM from previous systems such as LADDER [Hendrix et al., 1978] LUNAR [Woods, Kaplan, and Webber, 1972], PLANES [Waltz, 1975], REL [Thompson, 1975], and has affected the design of the natural-language processing system in several ways. Most previously built natural-language interface systems have used techniques that make them inherently difficult to transfer to new domains and databases. The internal representations in these systems typically intermix (in their data structures and procedures) information about language with information about the domain and the database. In addition, in interpreting a query, the systems conflate what a user is requesting (what his query "means") with how to obtain the information requested. Moving such systems to a new database requires careful handcrafting that involves detailed knowledge of such things as parsing procedures, the particular way in which domain information is stored, and data-access procedures. To provide for transportability, TEAM separates information about language, about the domain, and about the database.

The decision to provide transportability to existing conventional databases (which distinguishes TEAM from CHAT [Warren, 1981]) means that the database cannot be restructured to make the way in which it stores data more compatible with the way in which a user may ask about the data. Although many problems can be avoided if one is allowed to design the database as well as the natural-language system, given the prevalence of existing conventional databases, approaches which make this assumption are likely to have limited applicability in the near-term.

The TEAM system has three major components: (1) an acquisition component, (2) the DIALOGIC language system [Grosz, et al., 1982], and (3) a data-access component. Section C describes how the language and data-access components were designed to accommodate the needs of transportability. Section D describes the design of the acquisition component to allow flexible interaction with a database expert and discusses acquisition problems caused by the differences between the database view and user view. Section E shows how end-user queries are interpreted after an acquisition has been completed. Section F describes the current state of development of TEAM and lists several problems currently under investigation.

C. System Design

In TEAM, the translation of an English query into a database query takes place in two steps. First, the DIALOGIC system constructs a representation of the literal meaning or "logical form" of the query [Moore, 1981]. Second, the data-access component translates the logical form into a formal database query. Each of these steps requires a combination of some information that is dependent on the domain or the database with some information that is not. To provide for transportability, the TEAM system carefully separates these two kinds of information.

1. Domain- and Database-Dependent Information

To adapt TEAM to a new database three kinds of information must be acquired: information about words, about concepts, and about the structure of the database. The data structures that encode this information--and the language processing and data-access procedures that use them--are designed to allow for acquiring new information automatically.

Information about words, lexical information, includes the syntactic properties of the words that will be used in querying the database and semantic information about the kind of concept to which a particular word refers. TEAM records the lexical information specific to a given domain in a lexicon.

Conceptual information includes information about taxonomic relationships, about the kinds of objects that can serve as arguments to a predicate, and about the kinds of properties an object can have. In TEAM, the internal representation of information about the entities in the domain of discourse and the relationships that can hold among them is provided by a conceptual schema. This schema includes a sort hierarchy encoding the taxonomic relationships among objects in the domain, information about constraints on arguments to predicates, and information about relationships among certain types of predicates.

A database schema encodes information about how concepts in the conceptual schema map onto the structures of a particular database. In particular, it links conceptual-schema representations of entities and relationships in the domain to their realization in a particular database. TEAM currently assumes a relational database with a number of files. (No language-processing-related problems are entailed in moving TEAM to other database models.) Each file is about some kind of object (e.g., employees, students, ships, processor chips); the fields of the file record properties of the object (e.g., department, age, length).

2. Domain-Independent Information

The language executive [Grosz, et al., 1982; Walker, 1978], DIALOGIC, coordinates syntactic, semantic, and basic pragmatic rules in translating an English query into logical form. DIALOGIC's syntactic rules provide a general grammar of English [Robinson, 1982]. A semantic "translation" rule associated with each syntactic phrase rule specifies how the constituents of the phrase are to be interpreted. Basic pragmatic functions take local context into account in providing the interpretation of such things as noun-noun combinations. DIALOGIC also includes a quantifier-scoping algorithm.

To provide access to the information in a particular database, each of the components of DIALOGIC must access domain-specific information about the words and concepts relevant to that database. The information required by the syntactic rules is found in the lexicon. Information required by the semantic and pragmatic rules is found in the lexicon or the conceptual schema. The rules themselves however do not include such domain-dependent information and therefore do not need to be changed for different databases.

In a similar manner, the data-access component separates general rules for translating logical forms into database queries from information about a particular database. The rules access information in the conceptual and database schemata to interpret queries for a particular database.

D. Acquisition

TEAM is designed to interact with two kinds of users: a database expert (DBE) and an end-user. The DBE provides information about the files and fields in the database through a system-directed acquisition dialogue. As a result of this dialogue, the language-processing and data-access components are extended so that the end-user may query the new database in natural-language.

1. Acquisition Questions

Because the DBE is assumed to be familiar with database structures, but not with language-processing techniques, the acquisition dialogue is oriented around database structures. That is, the questions are about the kinds of things in the files and fields of the database, rather than about lexical entries, sort hierarchies, and predicates.

The disparity between the database view of the data and the end-user's view make the acquisition process nontrivial. For instance, consider a database of information about students in a university. From the perspective of an end-user "sophomore" refers to a subset of all of the students, those who are in their second year at the university. The fact that a particular student is a sophomore might be recorded in the database in a number of ways, including: (1) in a separate file containing information about the sophomore students; (2) by a special value in a symbolic field (e.g., a CLASS field in which the value SOPH indicates "sophomore"); (3) by a "true" value in a Boolean field (e.g., a * in an IS-SOPH field).

For natural-language querying to be useful, the end-user must be protected from having to know which type of representation was chosen. The questions posed to the DBE for each kind of database construct must be sufficient to allow DIALOGIC to handle approximately the same range of

linguistic expressions (e.g., for referring to "students in the sophomore class") regardless of the particular database implementation chosen. In all cases, TEAM will create a lexical entry for "sophomore" and an entry in the conceptual schema to represent the concept of sophomores. The database attachment for this concept will depend on the particular database structure, as will the kinds of predicates for which it can be an argument.

In designing TEAM we found it important to distinguish three different kinds of fields--arithmetic, feature (Boolean), and symbolic--on the basis of the range of linguistic expressions to which each gives rise. Arithmetic fields contain numeric values on which comparisons and computations like averaging are likely to be done. (Fields containing dates are not yet handled by TEAM.) Feature fields contain true/false values which record whether or not some attribute is a property of the object described by the file. Symbolic fields typically contain values that correspond to nouns or adjectives that denote the subtypes of the domain denoted by the field. Different acquisition questions are asked for each type of field. These are illustrated in the example in Section D.3.

2. Acquisition Strategy

The major features of the strategy developed for acquiring information about a database from a DBE include: (1) providing multiple levels of detail for each question posed to the DBE; (2) allowing a DBE to review previous answers and change them; and (3) checking for legal answers.

At present, TEAM initially presents the DBE with the short-form of a question. A more detailed version ("long-form") of the question, including examples illustrating different kinds of responses, can be requested by the DBE. An obvious extension to this strategy would be to present different initial levels to different users (depending, for example, on their previous experience with the system).

Acquisition is easier if each new piece of information is immediately integrated into the underlying knowledge structures of the program. However, we also wanted to allow the DBE to change answers to previous questions (this has turned out to be an essential feature of TEAM). Some questions (e.g., those about irregular plural forms and synonyms) affect only a single part of TEAM (the lexicon). Other questions (e.g., those about feature fields) affect all components of the system. Because of the complex interaction between acquisition questions and components of the system to be updated, immediate integration of new information is not possible. As a result, updating of the lexicon, conceptual schema, and database schema is not done until an acquisition dialogue is completed.

3. Example of Acquisition Questions

To illustrate the acquisition of information, consider a database, called CHIP, containing information about processor chips. In particular, the fields in this database contain the following information: the identification number of a chip (ID), its manufacturer (MAKER) its width in bits (WIDTH), its speed in megahertz (SPEED), its cost in dollars (PRICE), the kind of technology (FAMILY), and a flag indicating whether or not there is an export license for the chip (EXP).

In the figures discussed below, the DBE's response is indicated in uppercase. For many questions the DBE is presented with a list of options from which he can choose. For these questions, the complete list is shown and the answer indicated in boldface.

Figure 1 shows the short-form of the questions asked about the file itself. In response to question (1), the DBE tells TEAM what fields are in the file. Responses to the remaining questions allow TEAM to identify the kind of object the file contains information about (2), types of linguistic expressions used to refer to it [(6) and (7)], how to identify individual objects in the database (4), and how to specify individual objects to the user (5). These responses result in the words "chip" and "processor" being added to the lexicon, a new sort added to the taxonomy (providing the interpretation for these words), and a link made in the database schema between this sort and records in the file CHIP.

Figure 2 gives the short-form of the most central questions asked about symbolic fields, using the field MAKER (chip manufacturers) as exemplar. These questions are used to determine the kinds of properties represented, how these relate to properties in other fields, and the kinds of linguistic expressions the field values can give rise to. Question (4) allows TEAM to determine that individual field values refer to manufacturers rather than chips. The long-form of Question (7) is:

```
Will you want to ask, for example,
        "How many MOTOROLA processors are there?"
to get a count of the number of PROCESSORS
with CHIP-MAKER=MOTOROLA?
```

Question (8) expands to:

```
Will you want to ask, for example,
        "How many MOTOROLAS are there?"
to get a count of the number of PROCESSORS
with CHIP-MAKER=MOTOROLA?
```

In this case, the answer to question (7) is "yes" and to question (8) "no"; the field has values that can be used as explicit, but not implicit, classifiers. Contrast this with a symbolic field in a file about students that contains the class of a student; in this case the answer to both

questions would be affirmative because, for example, the phrases "sophomore woman" and "sophomores" can be used to refer to refer to STUDENTS with CLASS=SOPHOMORE. In other cases, the values may serve neither as explicit nor as implicit classifiers. For example, one cannot say *"the shoe employees" or *"the shoes" to mean "employees in the SHOE department".

For both questions (7) and (8) a positive answer is the default. It is important to allow the user to override this default, because TEAM must be able to avoid spurious ambiguities (e.g., where two fields have identical field values, but where the values can be classifiers for only one field.).

Following acquisition of this field, lexical entries are made for "maker" and any synonyms supplied by the user. Again a new sort is created. It is marked as having values that can be explicit, but not implicit, classifiers. Later, when the actual connection to the database is made, individual field values (e.g., "Motorola") will be made individual instances of this new sort.

Figure (3) presents the questions asked about arithmetic fields, using the PRICE field as exemplar. Because dates, measures, and count quantities are all handled differently, TEAM must first determine which kind of arithmetic object is in the field (2). In this case we have a unit of "worth" (6) measured in "dollars" (4). Questions (8) and (9) supply information needed for interpreting expressions involving comparatives (e.g., "What chips are more expensive than the Z8080?") and superlatives (e.g., "What is the cheapest chip?"). Figure 4 gives the expanded version of these questions.

As a result of this acquisition, a new subsort of the (measure) sort WORTH is added to the taxonomy for PRICE, and is noted as measured in dollars. In addition, lexical entries are created for adjectives indicating positive ("expensive") and negative ("cheap") degrees of price and are linked to a binary predicate that relates a chip to its price.

Feature fields are the most difficult fields to handle. They represent a single (arbitrary) property of an entity, with values that indicate whether or not the entity has the property, and they give rise to a wide range of linguistic expressions--adjectivals, nouns, phrases. The short-form of the questions asked about feature fields are given in Figure 5, using the field EXP; the value YES indicates there is an export license for a given processor, and NO indicates there is not. Figures 6, 7, and 8 give the expanded form of questions (4), (6), and (8) respectively. The expanded form illustrates the kinds of end-user queries that TEAM can handle after the DBE has answered these questions (see also Figure 9). Providing this kind of illustration has turned out to be essential for getting these questions answered correctly.

Each of these types of expression leads to new lexical, conceptual schema, and database schema entries. In general in the conceptual schema, feature field adjectivals and abstract nouns result in the creation of new predicates (see Section E for an example); count nouns result in the creation of new subsorts of the file subject sort. The database schema contains information about which field to access and what field value is required.

TEAM also includes a limited capability for acquiring verbs. At present, only transitive verbs can be acquired. One of the arguments to the predicate corresponding to a verb must be of the same sort as the file subject. The other argument must correspond to the sort of one of the fields. For the CHIP database, the DBE could specify that the verb "make" (and/or "manufacture") takes a CHIP as one argument and a MAKER as the second argument.

E. Sample Queries and Their Interpretations

After the DBE has completed an acquisition session for a file, TEAM can interpret and respond to end-user queries. Figure 9 lists some sample end-user queries for the file illustrated in the previous section. The role of the different kinds of information acquired above can be seen by considering the logical forms produced for several queries and the database attachments for the sorts and predicates that appear in them. The following examples illustrate the information acquired for the three different fields described in the preceding section.

Given the query,

What are the Motorola chips?

DIALOGIC produces the following logical form:

```
(Query (WHAT t1 (THING t1)
         (THE p2 (AND (PROCESSOR p2)
                      (MAKER-OF p2 MOTOROLA))
                 (EQ p2 t1))))
```

where WHAT and THE are quantifiers;[1] t1 and p2 are variables; AND and EQ have their usual interpretation. The predicates PROCESSOR and MAKER-OF and the constant MOTOROLA were created as a result of acquisition.

The following information in the database schema:

```
PROCESSOR: file=CHIP
           keyfield=ID
MAKER-OF:  file=CHIP
           field(arg1)=ID
           field(arg2)=MAKER
```

---

[1] Because the current version of DIALOGIC takes no account of the singular/plural distinction, the uniqueness presupposition normally associated with "the" is not enforced.

is used, along with sort hierarchy information in the conceptual schema, to generate the actual database query.

Similarly, the end-user query

What are the exportable chips?

would lead to the logical form:

```
(Query (WHAT t1 (THING t1)
          (THE p2 (AND (PROCESSOR p2)
                       (EXP-POS p2))
                  (EQ p2 t1))))
```

where EXP-POS is a predicate created by acquisition; it is true if its argument is exportable. In this case the relevant database schema information is:

```
PROCESSOR: file=CHIP
           keyfield=ID
EXP-POS:   file=CHIP
           field=EXP
           fieldvalue=T
```

Finally, to illustrate how TEAM handles arithmetic fields, and in particular the use of comparatives, consider the query:

What chip is cheaper than 5 dollars?

The logical form for this query is

```
(Query (WHAT p1 (PROCESSOR p1)
                ((MORE CHEAP) p1 (DOLLAR 5))))
```

The conceptual schema encodes the relationship between the predicates CHEAP and PRICE-OF (again, both concepts created as a result of acquisition), with the following information

```
CHEAP:     measure-predicate=PRICE-OF
           scale=negative
```

And the relevant database schema information is:

```
PROCESSOR: file=CHIP
           keyfield=ID
PRICE-OF:  file=CHIP
           field(arg1)=ID
           field(arg2)=PRICE
```

F. Status and Future Research

An initial version of TEAM was implemented in a combination of Interlisp (acquisition and DIALOGIC components) and Prolog (data access component) on the DEC2060, but address space limitations made continued development difficult. Current research on TEAM is being done on the Symbolics LISP machine. The acquisition component has been redesigned to take advantage of capabilities provided by the bitmap display. The new acquisition component allows the user more flexibility in answering questions and provides a wider range of default answers.

TEAM currently handles multiple files and provides transportability to a limited range of databases. As mentioned previously, a relational database model is assumed. Currently, TEAM also assumes all files are in third normal form. The acquisition of verbs is limited to allowing the DBE to specify transitive verbs, as described in Section D.3. We are currently extending TEAM to

(1) Provide for interpretation of expressions involving such things as mass terms, aggregates, quantified commands, and commands that require the system to perform functions other than querying the database.

(2) Provide for efficient processing of the most common forms of conjunction.

(3) Generalize the verb acquisition procedures and extend TEAM to handle more complex verbs, including such things as verbs with multiple delineations, verbs that require special prepositions, and verbs that allow sentential complements.

(4) Handle databases encoding time-related information and extend DIALOGIC to handle expressions involving time and tense.

G. Acknowledgments

The development of TEAM has involved the efforts of many people. Doug Appelt, Armar Archbold, Bob Moore, Jerry Hobbs, Paul Martin, Fernando Pereira, Jane Robinson, Daniel Sagalowicz, and David Warren have made major contributions.

This research was supported by the Defense Advanced Research Projects Agency with the Naval Electronic Systems Command under Contract N00039-80-C-0645.

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

REFERENCES

Grosz, B. et al. [1982] "DIALOGIC: A Core Natural-Language Processing System," Proceedings of the Ninth International Conference on Computational Linguistics, Prague, Czechoslovakia (July 1982).

Moore, R. C. [1981] "Problems in Logical Form," in Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics, pp. 117-124. The Association for Computational Linguistics, SRI International, Menlo Park, California (June 1981)..

Robinson, J. [1982] "DIAGRAM: A Grammar for Dialogues," Communications of the ACM, Vol. 25, No. 1, pp. 27-47 (January 1982).

Thompson, F. B. and Thompson, B. H. [1975] "Practical Natural Language Processing: The REL System as Prototype," M. Rubinoff and M. C. Yovits, eds., pp. 109-168, Advances in Computers 13, Academic Press, New York, (New York 1975).

Walker, D. E. (ed.) [1978] Understanding Spoken Language, Elsevier North-Holland, New York, New York, (1978).

Waltz, D. [1975] "Natural Language Access to a Large Data Base: An Engineering Approach," Proc. 4th International Joint Conference on Artificial Intelligence, Tbilisi, USSR, pp. 868-872 (September 1975).

Warren, D. H. [1981] "Efficient Processing of Interactive Relational Database Queries Expressed in Logic," Proc. Seventh International Conference on Very Large Data Bases, Cannes, France, pp. 272-283, (September 1981).

Woods, W. A., R. M. Kaplan, and B. N-Webber [1972] "The Lunar Sciences Natural Language Information System," BBN Report 2378, Bolt Beranek and Newman, Cambridge, Massachusetts (1972).

File name - **CHIP**
(1) Fields - (ID MAKER WIDTH SPEED PRICE FAMILY EXP)
(2) Subject - PROCESSOR
(3) Synonyms for PROCESSOR - **CHIP**
(4) Primary key - **ID**
(5) Identifying fields - MAKER **ID**
(6) Can one say Who are the PROCESSORS? - YES **NO**
(7) Pronouns for file subject - HE SHE **IT** THEY
(8) Field containing the name of each file subject - **ID**

Figure 1: Questions About File **CHIP**

Field - **MAKER**
(1) Type of field - **SYMBOLIC** ARITHMETIC FEATURE
(2) Are field values units of measure? YES **NO**
(3) Noun subcategory - **PROPER** COUNT MASS
(4) Domain of field value's reference - SUBJECT **FIELD**
(5) Can you say Who is the CHIP-MAKER? **YES** NO
(6) Typical value - MORTOROLA
(7) Will values of this field be used as classifers? **YES** NO
(8) Will the values in this field be used alone as implicit classifiers? YES **NO**

Figure 2: Questions for Symbolic Field **MAKER**

Field - **PRICE**
(1) Type of field SYMBOLIC **ARITHMETIC** FEATURE
(2) Value type - DATES **MEASURES** COUNTS
(3) Are the units implicit? **YES** NO
(4) Enter implicit unit - DOLLAR
(5) Abbreviation for this unit? -
(6) Measure type of this unit - TIME WEIGHT SPEED VOLUME LINEAR AREA **WORTH** OTHER
(7) Minimum and maximum numeric values - (1,100)
(8) Positive adjectives - (EXPENSIVE COSTLY)
(9) Negative adjective - (CHEAP)

Figure 3: Questions for Arithmetic Field **PRICE**

Please specify any adjectives that can be used in their comparative or superlative form to indicate how much each PROCESSOR is in a positive direction on the scale measured by the values of CHIP-PRICE.

In a file about machine-tools with a numeric field called PRICE, one could ask:

How EXPENSIVE is each tool?

to mean

What is the price of each tool?

EXPENSIVE, COSTLY, AND (HIGH PRICED) are positive adjectives designating the upper range of the PRICE scale.

CHEAP and (LOW PRICED), which designate the lower range of the PRICE scale, are negative adjectives.

Please enter any such adjectives you will want to use in querying the database.

Figure 4: Expanded Version of Adjective Questions (Arithmetic Field)

Field - **EXP**
(1) Type of field - SYMBOLIC ARITHMETIC **FEATURE**
(2) Positive value - YES
(3) Negative value - NO
(4) Positive adjectives - EXPORTABLE
(5) Negative adjectives - UNEXPORTABLE
(6) Positive abstract nouns - EXPORT AUTHORIZATION
(7) Negative abstract nouns -
(8) Positive common nouns -
(9) Negative common nouns -

Figure 5: Questions for Feature Field **EXP**

What adjectivals are associated with the field values
YES in this field?

In general these are words wwww such that you
might want to ask:
Which PROCESSORS are wwww?
to mean
Which PROCESSORS have a CHIP-EXP of YES?

For example, in a medical file about PATIENTs with a
feature field IMM having a positive field value Y
and a negative filed value N,
you might want to ask:
Which patients are IMMUNE (or RESISTANT,
PROTECTED)?

Figure 6: Feature Field Adjectivals

List any abstract nouns associated with the positive
feature value YES.

In general this is any word wwww such that you
might want to ask a question of the form:
Which PROCESSORS have wwww?
to mean
Which PROCESSORS have CHIP-EXP of YES?

For example, in a medical database about PATIENTs
with a feature field IMM having a positive field
value Y and a negative field value N,
you might want to ask:
Which patients have IMMUNITY?
instead of
Which patients have an IMM of Y?

Figure 7: Feature Field Abstract Nouns

List any count nouns associated with positive
field value YES.

In general, this is any word wwww such that
you might want to ask:
What PROCESSORS are wwww-s?
to mean
What PROCESSORS have a CHIP-EXP of YES?

For example, in a file about EMPLOYEEs with a
feature field CITIZEN having a positive
field value Y and negative field value N,
you might want to ask:
Which employees are citizens?
instead of
Which employees have a CITIZEN of Y?

Figure 8: Feature Field Count Nouns

What 8 bit chips are cheaper than the fastest
exportable chip made by Zilog?
Who makes the fastest exportable *NMOS* chip
costing less than 10 dollars?
By whom is the most expensive chip made?
Who is the cheapest exportable chip made by?
Who is the most expensive chip made?
What is the fastest exportable chip that Motorola makes?
What 16 bit chips does Zilog make?
Who makes the fastest exportable *NMOS* chip?
Who makes the fastest exportable chip?
Does Zilog make a chip that is faster than every
chip that Intel makes?
Are there any 8 bit Zilog chips?
Is some exportable chip faster than 12 mhz?
Is every Zilog chip that is faster than 5 mhz exportable?
How fast is the fastest exportable chip?
How expensive is the fastest *NMOS* chip?

Figure 9: Sample questions for *CHIP* database