Transportability and Generality in a Natural-Language Interface System

Paul Martin, Douglas Appelt, and Fernando Pereira
Artificial Intelligence Center
SRI International

Abstract

This paper describes the design of a transportable natural language (NL) interface to databases and the constraints that transportability places on each component of such a system. By a transportable NL system, we mean an NL processing system that is constructed so that a domain expert (rather than an AI or linguistics expert) can move the system to a new application domain. After discussing the general problems presented by transportability, this paper describes TEAM (an acronym for Transportable English database Access Medium), a demonstratable prototype of such a system. The discussion of TEAM shows how domain-independent and domain-dependent information can be separated in the different components of a NL interface system, and presents one method of obtaining domain-specific information from a domain expert.

1. Transportable Systems

To build a transportable system, the designer must distinguish between domain-specific and general rules at all levels in the design process. The additional constraints imposed by transportability force the builders of such systems to address several basic issues that can be overlooked in designing special purpose interfaces (e.g., LUNAR [Woods et al. 72], LADDER [Hendrix 77]). As a result, research on transportable systems has implications beyond practical utility.

Specifically, transportability forces the designer to address the following broad issues:

- The syntax used must be a general grammar of English, rather than a domain-specific set of rules.
- Because the semantics cannot depend on some domain-specific syntax chosen for ease of semantic translation, a general mechanism must be developed to acquire and attach semantics to a wide variety of syntactic constructions.
- The lexicon must also be acquired; thus, any knowledge that might otherwise appear as "ad hoc" lexical entries must be structured to fit into the general grammar and to be specified in terms of user-oriented concepts.
- Both the pragmatics and database access processes must be able to use acquired information to turn the representation of what was said into a query appropriate for the database underlying the application. Whether a relation is actually stored in the database or is derived from other relations (a virtual relation) should be hidden from the user and from the semantic interpretation mechanisms.
- Finally, designing a system able to acquire all the kinds of information sketched above is in itself a challenge; through interacting with a person assumed to be unschooled in lingusitics and AI, it must elicit both the conceptual structure and the linguistic content of the domain.

2. Overview of TEAM

TEAM runs in two distinct phases: first an expert on the domain (the database expert or DBE) answers questions about the database and the linguistic expressions used to refer to the information it contains, and then end-users can use it to answer natural-language queries from the application.

Figure 1 shows the major components of TEAM — acquisition, DIALOGIC, and database access. Acquisition obtains the description of an application from the DBE, and uses it to expand and update the internal specification of the user world. DIALOGIC translates English sentences into a logical form that represents their literal content. DIALOGIC uses two kinds of information acquired from the DBE: the lexicon, a table of vocabulary words and their definitions; and the conceptual schema, the specification of the predicates the language can use. The database access component converts this logical form to a formal database query using the conceptual schema and the database schema, a set of structured statements about the database.

Section 3 describes the acquisition process, Section 4 shows how a logical form is produced from an input sentence, and Section 5 describes the production of an appropriate database query.

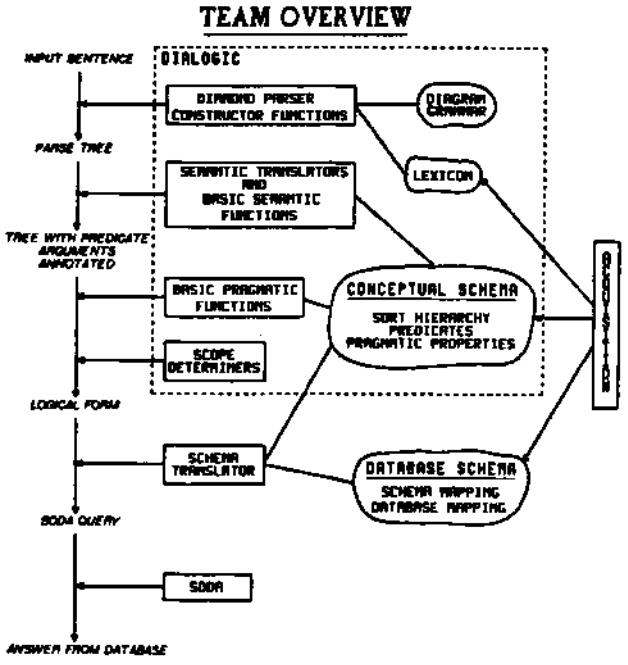

Figure 1: Overview of TEAM

3. Acquisition of a New Database

The acquisition component of TEAM is responsible for gathering information from the DBE about the structure of the database, the words that refer to objects in the database, and the relations among them. This information is incorporated into the lexicon and into the conceptual and database schemata. The acquisition component must meet several specific requirements:

- TEAM must acquire new concepts. Acquisition of a new database involves acquiring information about the structure of the database and modifying the existing conceptual schema.

- Detailed linguistic knowledge is acquired implicitly. The DBE is not required to have specialized knowledge of linguistics. Therefore, the linguistic information that DIALOGIC needs must be inferred from answers to questions that tap a layman's linguistic competence without recourse to the terminology of a trained linguist.

- TEAM must associate NL concepts with a wide range of database representations because a disparity generally exists between the database' view and the end-user's view of the world (cf. the notion of view in database theory).

- TEAM must provide the DBE with a powerful but clean interface to the acquisition processes. Acquisition of a multifile database is complex. The TEAM acquisition component thus seeks to present the DBE with an interface that allows him to see the different relations of the database, move freely back and forth among different relations, and freely change answers to previous questions.

An Overview of Acquisition

The interface through which the DBE imparts his knowledge to TEAM is shown in Figure 2. At the top of the layout is a menu of the operations the DBE can perform. Below that is the file menu listing all the relations the system knows about, a field menu containing the fields in each relation, and a word menu containing all the vocabulary items that TEAM has acquired in addition to its basic lexicon. The question answering area is the space reserved for the user to answer detailed questions about files, fields, and words. The user selects one of the items in the menus with the mouse, and then continues to interact in the question-answering window.[1]

FILE MENU
EMPLOYEE

FIELD MENU
EMPLOYEE-DEPT EMPLOYEE-EXEMPT EMPLOYEE-ID EMPLOYEE-NAME
EMPLOYEE-SALARY

WORD MENU

DEPT (n)	**EXEMPT** (n)	**ID** (n)
NAME (n)	**EMPLOYEE** (n)	**SALARY** (n)

Question Answering Area
File name - EMPLOYEE
Type of relation - **ACTUAL** VIRTUAL
LMFS pathname - EMPLOYEE.DB
Fields - NAME ID SALARY DEPT EXEMPT
Subject - EMPLOYEE
Primary key set - NAME ID SALARY DEPT EXEMPT
Identifying fields - **NAME ID** SALARY DEPT EXEMPT
Pronouns for file subject - **HE SHE** IT THEY
Don't forget to insert *employee* into the sort hierarchy.

Figure 2: File Menu Questions

The TEAM acquisition component also contains the following support functions, which are accessed through a fixed menu at the top of the display (not shown in the figure):

- The sort-editor allows the DBE to add new concepts to the sort hierarchy, a central data structure in the representation of the concepts of the application (conceptual schema). The sort hierarchy is a network that expresses the taxonomy of all the objects in the TEAM world. When TEAM is started, it contains certain core concepts; the acquisition task entails fitting all new domain-specific conceptual objects into the taxonomy.

- Specify-rel allows the DBE to specify the connections between virtual relations and the actual files of the database.

[1] Bold face is used in menus to represent items for which enough information has been given, and in questions to mark the currently selected answer (default).

An Example of Acquisition

To illustrate how the DBE specifies the information needed for natural language access, we consider the example of a simple database consisting of two files, EMPLOYEE and DEPARTMENT, and a virtual relation composed of their join, MANAGER.2 The database consists of an EMPLOYEE file with fields NAME (the employee's name), SS (the employee's social security number), SALARY (the employee's salary), DEPT (the employee's department), and EXEMPT (a Boolean feature field indicating whether the employee is exempt from overtime regulations). The DEPARTMENT file has fields, NAME (the name of the department) and MGR (the name of the department's manager). A virtual MAN AG FOR relation is defined that relates employees to their managers.

When a new relation is created (with the new-rel command as has already been done in Figure 2), the DBE must supply its name and fields, which then appear in the appropriate menus.

Other questions provide TEAM with a primary key set (see section 6), a set of identifying fields (used for answers), and gender information. TEAM creates a noun for the subject of the file and one for the name of each field.

Field Types

TEAM distinguishes among three kinds of fields — arithmetic, feature (Boolean), and symbolic — on the basis of the kinds of linguistic expressions that are used to express relationships about those fields. Symbolic fields are the simplest; linguistic access is restricted to naming the field or its values. Arithmetic fields contain numbers for which it makes sense to compare and to associate comparative and superlative adjectives. Feature fields correspond to the presence or absence of some arbitrary property of the subject. TEAM supports a variety of linguistic constructions that refer to such a property: adjectives modifying the subject, nouns representing the property that the subject has or lacks, nouns representing the subset of the subjects that have or lack the property, and intransitive verbs applied to the subject that has or lacks the property.

During the acquisition of each field, the DBE specifies information about the lexical items associated with the field and how the field fits into the conceptual schema. These questions resolve the difference between the user's view and the database view of the world. Figures 3, 5 and 6, illustrate the questions asked for each type of field.

We concentrate here on noun relations, for which we assume that there is a subject field (e.g., ships, ship classes, employees, instances of an ownership relation). We also assume that each relation is in third-normal form [Codd 70); that is, each row represents an instance of the subject of the relation, and each column value is a function of the subject value.

Symbolic Field

Figure 3 shows the questions asked for the symbolic field DEPT in the EMPLOYEE file. Question (a), if answered affirmatively, would enter the database field values into the word menu so that the DBE can specify synonyms and irregular forms for database field entries. Questions (b) and (c) establish the modifier usages of the field values.

```
Question Answering Area
Is field actual or virtual? ACTUAL VIRTUAL
Associated file - EMPLOYEE
Name of field - DEPT
Type of field - SYMBOLIC ARITHMETIC FEATURE
Don't forget to insert the field in the sort hierarchy
Edit lexicon for words in this field? YES NO                    (a)
'Unknown' convention for this field - *
'Not applicable' convention for this field - **
Are field values units of measure? YES NO
Noun subcategory - PROPER COUNT MASS
Typical value - PAYROLL
Will values of this field be used as classifiers?
      YES NO                                                    (b)
Will the values in this field be used alone
      as implicit classifiers? YES NO                           (c)
```

Figure 3: Symbolic Field Acquisition

In our example the answer to question (b) is "Yes," and the answer to (c) is "No." Depending on the domain, other answers are possible. For example, in a database about ships, we could neither ask "How many Jones ships are there?" nor "How many Joneses are there?" to obtain a count of the number of ships commanded by Jones. However, in a database about automobiles, we could ask "How many Ford cars are there?" or "How many Fords are there?" to find out how many automobiles are made by Ford.

Although TEAM derives the placement of arithmetic and feature fields in the sort hierarchy from the answers to the field questions, this is not possible for symbolic fields. The DBE must place symbolic fields in the 9ort hierarchy using the sort-hierarchy editor; we illustrate this process in Figure 4.

Arithmetic Fields

Figure 5 illustrates the questions asked about arithmetic fields. Because dates, measures, and counts are used differently, TEAM must be told, by answering question (a), what type of arithmetic field is being acquired. In this case, we have a unit of economic worth that is measured in dollars. Questions (b) and (c) discover adjectives (or other adjectival modifiers) that will be used in their comparative and superlative forms in queries (e.g., "Who is the highest paid employee?").

Feature Fields

Feature fields are the most difficult to handle because of the wide range of linguistic expressions used to express their values. Figure 6 illustrates feature field acquisition for the EXEMPT field in the EMPLOYEE file. TEAM needs to know the positive and negative values in the fields, and the nominals and adjectivals associated with each value. In this example, given the answers in Figure 6, one could ask "What employees are exempt?" or "How many nonexempt employees earn more than 20000

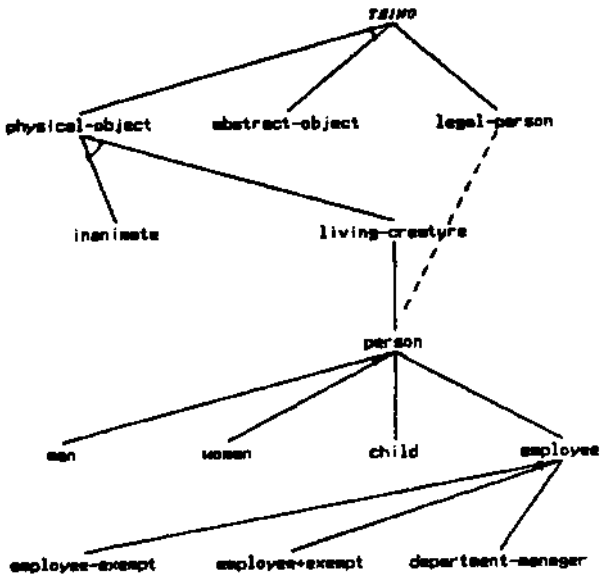

Figure 4: Sort Editor

```
Question Answering Area
Is field actual or virtual? ACTUAL VIRTUAL
Associated file - EMPLOYEE
Name of field - SALARY
Type of field - SYMBOLIC ARITHMETIC FEATURE
Value type - DATES MEASURES COUNTS                    (a)
Are the units implicit? YES NO
Enter implicit unit - DOLLAR
Abbreviation for this unit?
Measure type of this unit -
TIME WEIGHT SPEED VOLUME LINEAR AREA WORTH OTHER
Minimum and maximum numeric values - 0. 100000.
Positive adjectives - (HIGH PAID)                     (b)
Negative adjectives - (LOW PAID)                      (c)
```

Figure 5: Arithmetic Field Acquisition

```
Question Answering Area
Is field actual or virtual? ACTUAL VIRTUAL
Associated file - EMPLOYEE
Name of field - EXEMPT
Type of field - SYMBOLIC ARITHMETIC FEATURE
Positive value - Y
Negative value - N
Positive adjectivals - EXEMPT
Negative adjectivals - NONEXEMPT
Positive abstract nouns -
Negative abstract nouns -
Positive count nouns -
Negative count nouns -
```

Figure 6: Feature Field Acquisition

dollars?"

Each type of expression leads to new lexical, conceptual schema, and database schema entries. In general, in the conceptual schema, feature field adjectivals create a new predicate, abstract nouns create contrasting new sorts that are subsorts of some abstract quality, and count nouns create contrasting subsorts of the file subject.

Volunteered Information

```
User has selected new-word from menu.
Question Answering Area
Enter word - DEPARTMENT
Synonya - DEPT
Syntactic category - ADJECTIVE NOUN VERB
Plural - DEPARTMENTS
```

Figure 7: Word Menu

```
Question Answering Area
Enter word - EARN
Syntactic category - ADJECTIVE NOUN VERB
Third person singular present tense (he she it) - EARNS
Past tense - EARNED
Past participle - EARNED
Sentence - AN EMPLOYEE EARNS A SALARY
•A SALARY EARNS.• <=>
        'Soaething EARNS a SALARY. ' YES NO
•A EMPLOYEE EARNS.' <=>
        •A EMPLOYEE EARNS soaething.' YES NO
'A SALARY is EARNED.' <=>
        *Soaething EARNS a SALARY.* YES NO
```

Figure 8: Verb Acquisition

The DBE may volunteer lexical items to TEAM and assign them a meaning in terms of database relations (actual or virtual). For example, in Figure 7, the user volunteers the word DEPARTMENT and says it is a synonym of DEPT; in Figure 8 the DBE volunteers the verb 'earn.' In both cases, TEAM extracts both the syntactic and semantic properties of the new verb without recourse to technical linguistic terms.

Virtual Relations

```
Specification of virtual fields for MANAGER
MANAGER-EMP:  EMPLOYEE-NAME
MANAGER-MGR:: DEPARTMENT-MANAGER
```

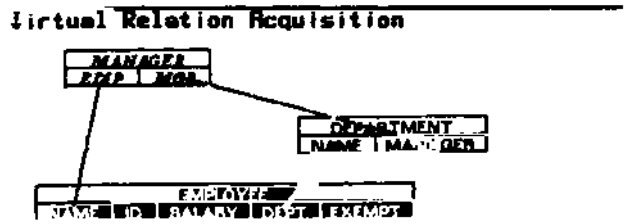

Figure 9: Virtual Relation

Figure 9 shows the DBE specifying the fields of a new virtual relation MANAGER, where an employee's manager is the manager of the department in which the employee works. The DBE can use the file menu to call up diagrams of any relation the system has acquired. Relations are specified graphically by making connections between the fields in the relation he is specifying and other relations in the database. The graphic interface serves both to display and manipulate the relationships of the database. The heavy line indicates that a database join is to be performed between the two fields, and the dotted lines indicate which fields in the virtual relation

are carried over from the relations being joined.

4. DIALOGIC

The component of the TEAM system that takes an input sentence and produces a logical form is called DIALOGIC [Grosz et al. 82]. DIALOGIC comprises several phases, illustrated in Figure 1. All of the domain-dependent information it needs is in the conceptual schema and the lexicon (domain-independent information for core vocabulary such as pronouns and prepositions is already in these data structures before the acquisition).

We can illustrate the operation of DIALOGIC by considering an example sentence that could be asked in the employee database acquired in our previous example:

What employees earn more than their manager's salary?

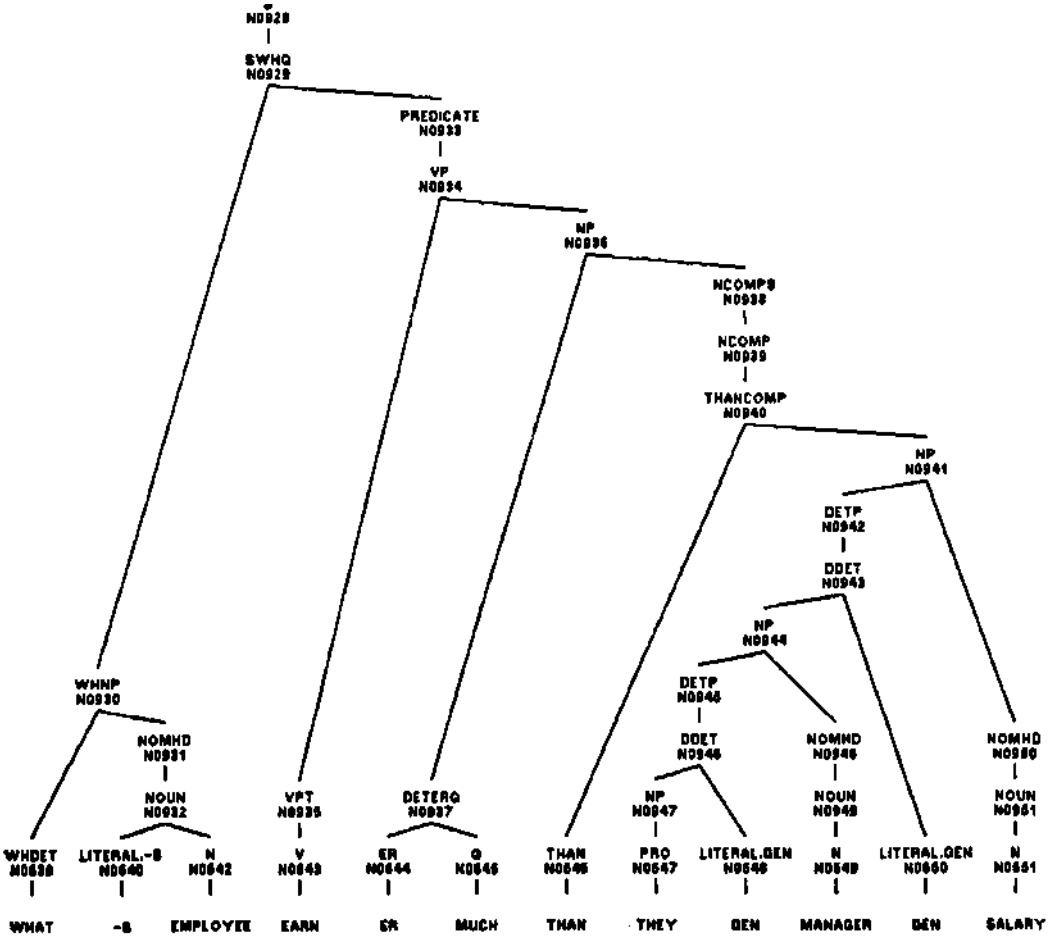

Figure 10: Sample Parse Tree

Syntactic Operations

The DIAMOND parser and the DIAGRAM grammar [Robinson 82] constitute the syntactic portion of DIALOGIC. Together they can analyze all common sentence types in English (with the exception of sentences using conjunctions). They are applied to the morphemes retrieved from the lexicon to produce a set of syntactically acceptable parses.

The grammar is a set of context-free rules annotated with constructors that enforce context-sensitive syntactic constraints, and supply scores for the parses based on the *a priori* likelihood of a construction and on the composed likelihood of certain combinations. These scores are composed upward to the sentence node, yielding an overall sentence score. The score is used to rank the resulting parses so that the syntactically "best" ones are tried first by the semantic translation process. Figure 10 shows the top-ranked syntactic parse for our example sentence.

Semantic Translation

In addition to the constructor functions, each rule of the grammar also has another associated function called its translator. The translators are domain-independent, but they derive much of their behavior from the acquired features and predicate structures associated with lexical entries. They compute the predicate relations of the

sentence by using its syntactic structure and a set of semantic operations called the basic semantic functions (BSFs). This approach of decoupling these commonly conflated issues is important: By separating the parsing from the semantic translation, we allow independent development of the grammatical coverage and semantic capabilities. Separating the semantic phase into translators that are close to the grammar and BSFs that are tied to the target formal theory achieves a similar independence [Hendrix 78].

The BSFs annotate the parse tree with noun groups, predicates, and quantified variables. The predicate relations are specified as logical form fragments (LFFs). These are quantifier-free open formulas in the target logical form [Moore 81]. The complete logical forms are a higher-order predicate calculus in which the restriction and assertion clauses of a quantified expression are asserted separately.

For our example query, translator calls to the BSFs build variables representing employees, managers, and salaries as the heads of the noun phrases. The translator calls for pronouns and definite noun phrases also collect the set of syntactically feasible coreferents of such phrases [Hobbs 76]. In our example sentence, the pronoun 'them,' which is derived morphologically from 'their,' is assigned a variable that is marked as ambiguous among the employee and salary variables (the manager variable was previously ruled out on syntactic grounds). Calls to other BSFs build the predications that relate these variables, expressing information such as the relationship between an individual employee and a salary.

The BSFs also resolve certain classes of ambiguities, and detect and reject some kinds of semantically inappropriate parses. The requirements on the argument types of most predicates arc used by the BSFs to check the semantic suitability of the proposed role fillers. The acquisition of domain-dependent predicates implicitly obtains the delineations that specify the sorts of their arguments. In our example sentence, the EARN predicate requires an EMPLOYEE in the first argument position and a SALARY in the second. Because a MANAGER is a kind of EMPLOYEE (as indicated by the sort hierarchy), the delineation checking mechanism allows a manager to EARN a salary too. If a parse had been chosen that reversed the order of these arguments, it would have been rejected.

When delineation checking detects an inappropriate sort, the BSFs have several options other than simply rejecting the parse:

- If either the predicate or the role filler is derived from words that have alternative semantic interpretations, the remaining alternatives can be tried.

- Alternative candidates for the referent of a pronoun are likewise treated as semantic alternatives. In our example sentence, the 'them,' which is the pronoun part of the genitive 'their' is constrained to be some EMPLOYEE (rather than a SALARY, for example) by the delineation testing of the MANAGER relation.

- Noun groups may be converted from their initial sort to a related sort that satisfies the delineation (backward coercion). Backward coercion would recognize that 'salesman' is of sort JOB-TITLE and hence could not be used with the SALARY-OF predicate (which requires EMPLOYEE), and thus would replace it with the semantic equivalent of "an EMPLOYEE whose JOB-TITLE is salesman."

Pragmatics

Two further transformations must be applied to the parse tree: Remaining ambiguities caused by vague predicates (e.g., OF) or semantic alternatives must be resolved, and quantifiers and similar operators must be assigned relative scopes.

The basic pragmatic functions replace vague predicates with more meaningful predicates derived from properties of the sorts of the arguments. During acquisition, the sorts of the fields of every relation are automatically entered in the conceptual schema as representing functions of the the subject sort of the relation. Thus, because 'salary' is a field name of the employee file, its semantic sort SALARY is marked with the implicit predicate SALARY-OF. Inasmuch as the implicit predicate is not always the right replacement, delineation checking and other tests are used to make the decision.

Scope of Quantifiers

Quantifiers and operators such as negation and superlatives are distinguished from normal predicates because their arguments cannot generally be determined by simple composition. To determine the embedding of their arguments (or, equivalently, the correct relative scopes), TEAM uses the structure of the parse tree to rank each permutation of the operators, eliminating those permutations that produce an ill-structured logical form (e.g., one that uses a variable outside the region in which it is quantified). The choice between the existential and universal readings of 'any' in questions and negative contexts is also handled at this stage. Legal interpretations are scored by a set of scope critics that judge specific aspects of the scoping. Left-right ordering, the relative scoping strengths of the operators, and the tendency of 'any' to exceed the scope of at least one structure that syntactically surrounds it are exemplary of the information embodied in the scope critics. The summed scoping scores are used to order the possibilities, and the user is shown the result in the form of a very literal English rendering of the logical form. If TEAM has succeeded, this rendering is a paraphrase of the original input sentence.

A post-processor applies a few minor transformations, and the result is a logical form that represents an unambiguous interpretation of the natural language sentence. The final process of TEAM uses this logical form (the logical form for our example sentence is shown in the next section), and the acquired conceptual schema,

and the database schema information to produce a database query.

5. The Schema Translator

The schema translator translates the logical forms of English sentences into database queries for the host database system. The current database system is SODA, a relational database system embedded in LISP [Moore 79].

The schema translator performs the following four tasks in sequence:

1. It replaces logical form quantifiers by machine-oriented quantification operators.

2. It simplifies the logical form by expanding the definitions of derived predicates (virtual relations) and by removing predications implied by other predications in the logical form.

3. It identifies and eliminates redundant occurrences of the same relation with the same key field.

4. It translates the simplified logical form into the host query language, SODA.

The schema translator uses a schema mapping that defines the predicates used in the logical form in terms of the actual database relations. The schema mapping is analogous to the virtual relations in a relational database that are used to make a database application independent of the actual relations in the database. The schema mapping is expressed in a variation of the definite clause subset of first-order logic. This subset has convenient computational properties [Roussel 75] and allows the acquisition of new relations to be implemented by using an extension of the techniques of Query-by-Example [Zloof 75, Neves et al. 82], as shown in Section 3.

Schema Mapping

The following formulas give a fragment of the schema mapping produced by the acquisition processes for the relations in Figure 9 (the actual notation in TEAM is somewhat different)

```
(MANAGER emp mgr) <=
  (EMPLOYEE emp ? ? dept ?) &
  (DEPT dept mgr)

(EARN name salary) <=
  (EMPLOYEE name ? salary ? ?)

(SALARY-OF name salary) <=
  (EMPLOYEE name ? salary ? ?)

(EMPLOYEE-SORT name) <=
  (EMPLOYEE name ? ? ? ?)

(SALARY-SORT salary) <=
  (EMPLOYEE ? ? salary ? ?)

(MANAGER-SORT manager) <=
  (DEPT ? manager)
```

The formulas above are merely definite clauses, with upper-case words denoting predicates, lower-case words denoting named variables, and '?' denoting anonymous (don't care) variables. The notation is positional, with the arguments of predicates corresponding to the relation fields in the same positions in Figure 9.

The first rule defines the virtual relation MANAGER in terms of the two database relations EMPLOYEE and DEPT. These two relations are joined (in the relational sense) by the shared variable dept.

The second and third rules define the implicit virtual relations EARN (a verb) and SALARY-OF by projecting the relation EMPLOYEE over the 'name' and 'salary' fields. These virtual relations are not defined explicitly during acquisition, but correspond to the association of verbs and nouns to particular combinations of fields of a single relation.

The last three rules trivially define sorts associated with fields of EMPLOYEE and DEPT as projections of appropriate relations.

The first fields of EMPLOYEE (name) and of DEPT (dept) are the key fields of those relations.

Schema Translation

We will show now how the schema translator processes the Logical Form from the example sentence in Section 4.

```
What employees earn more than their manager's salary?

(QUERY (WH e
         (EMPLOYEE-SORT e)
         (SOME m
           (AND
             (MANAGER-SORT m)
             (MANAGER m e))
           (SOME s1
             (AND
               (SALARY-SORT s1)
               (SALARY-OF m s1))
             (SOME s2
               (SALARY-SORT s2)
               (AND
                 (EARN e s2)
                 (GT s2 s1)))))))
```

The schema translator would transform this logical form in steps (1) to (4) which follows:

1. Translate into Horn clause format removing redundant quantifiers.

```
(ANSWER e) <=
  (EMPLOYEE-SORT e) &
  (MANAGER-SORT m) &
  (MANAGER m e) &
  (SALARY-SORT s1) &
  (SALARY-OF m s1) &
  (SALARY-SORT s2) &
  (EARN e s2) &
  (GT s2 s1)
```

2. Expand the definitions of virtual relations, and remove the predications implied by other predications.

```
(ANSWER e) <=
  (EMPLOYEE-SORT e)
  (MANAGER-SORT m) &
  (EMPLOYEE e ? ? d ?) &
  (DEPT d m) &
  (SALARY-SORT s1) &
  (EMPLOYEE m ? s1 ? ?) &
  (SALARY-SORT s2) &
  (EMPLOYEE e ? s2 ? ?) &
  (GT s2 s1)

(ANSWER e) <=
  (EMPLOYEE e ? ? d ?) &
  (DEPT d m) &
  (EMPLOYEE m ? s1 ? ?) &
  (EMPLOYEE e ? s2 ? ?) &
  (GT s2 s1)
```

3. Identify relation instances with equal key fields.

```
(ANSWER e) <=
  (EMPLOYEE e ? s2 d ?) &
  (DEPT d m) &
  (EMPLOYEE m ? s1 ? ?) &
  (GT s2 s1)
```

4. Translate to SODA.

```
(IN $E EMPLOYEE)
 (IN $D DEPT)
  (EQ ($E DEPT) ($D NAME))
   (IN $M EMPLOYEE)
    (EQ ($M NAME) ($D MANAGER))
     (GT ($E SALARY) ($M SALARY))
      (? ($E NAME))
```

Note that all but the last step apply the inference rules created by acquisition to transform a formula into a simpler one that implies it. The last step merely entails a change of representation between two essentially equivalent languages. This deduction process effectively decouples the format of the logical form from the database query language and from the actual database.

6. Conclusion

We have constructed a natural-language interface that cleanly separates domain-dependent from domain-independent information. The domain-independent portion includes the parser and grammar, the semantic translators, the pragmatic and scope determining processes, the schema translator, and the basic vocabulary and taxonomy that form the initial state of the data structures subsequently modified by the acquisition process.

The information that is automatically acquired for each new domain includes: the lexicon; the conceptual schema, comprising taxonomy, predicate argument relations, and pragmatic markings; and the database schema, describing the mapping from predicates to database relations.

The current version of TEAM already supports fluent interaction with multiple file databases and work is underway to extend its linguistic and database coverage, especially in the following areas:

- Acquisition of more complex verbs, such as verbs with multiple delineations, verbs that require special prepositions, verbs that do not translate into a projection of an existing noun relation, and verbs with sentential complements.
- Interpretation of aggregates, quantified commands, and commands to perform functions other than database access.
- Treatment of common forms of sentences that use conjunctions.
- Time and tense in DIALOGIC and in the database.

Earlier versions of TEAM ran on a DEC 2060, implemented in a combination of INTERLISP (acquisition and DIALOGIC) and PROLOG (schema translator). The current version runs on a Symbolics LM-2 LISP machine, and includes its own PROLOG interpreter for the schema translation. Although TEAM is still under development and therefore has not yet been formally tested, it has successfully acquired a variety of databases.

Acknowledgments

The development of TEAM has profited from the efforts of Armar Archbold, Barbara Grosz, Norman Haas, Gary Hendrix, Jerry Hobbs, Bob Moore, Jane Robinson, Daniel Sagalowicz, and David Warren.

This research was supported by the Defense Advanced Research Projects Agency under Contract N00039-80-C-0645.

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 Projects Agency or the U.S. Government.

References

E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM,* Vol. 13, No. 6, pp. 377-387 (June 1970).

B. Grosz, N. Haas, G. G. Hendrix, J. Hobbs, P. Martin, R. Moore, J. Robinson and S. Rosenschein, "DIALOGIC: A Core Natural-Language Processing System." Technical Note 270, Artificial Intelligence Center, SRI International, Menlo Park, California (November 1082).

G. G. Hendrix, "Human Engineering for Applied Natural Language Processing," *Proe. 5th International Joint Conference on Artificial Intelligence,* pp. 183-101, IJCAI, Cambridge, Massachusetts (August 1977).

G. G. Hendrix, "Semantic Aspects of Translation/' in *Understanding Spoken Language,* D. E. Walker, ed., pp. 193-226 (Elsevier, New York, New York, 1978).

J. R. Hobbs, "Pronoun Resolution.'[1] Research Report 76-1, Department of Computer Sciences, City College, City University of New York, New York, New York (August 1076).

R. C. Moore, "Handling Complex Queries in a Distributed Database." Technical Note 170, Artificial Intelligence Center, SRI International, Menlo Park, California (October 1979).

R. C. Moore, "Problems in Logical Form," Proc. of the 19th Annual Meeting of the Association for Computational Linguistics, ACL, Stanford, California (1981).

J. Neves, R. Backhouse and S. Anderson, A Prolog Implementation of Query-by-Example. Forthcoming., 1982.

J. J. Robinson, "Diagram: a Grammar for Dialogues," Communications of the ACM, Vol. 25, No. 1, pp. 27-47 (1982).

P. Roussel, "Prolog: Manuel de Reference et Utilisation." Technical Report, Groupe d'Intelligence Artificielle, Universite d'Aix-Marseille II, Marseille, France (1975).

W. A. Woods, R. M. Kaplan and B. Nash-Webber, "The Lunar Sciences Natural Language Information System: Final Report." Report 3438, Bolt Beranek and Newman Inc. (June 1972).

M. M. Zloof, "Query-by-Example," Proc. AFIPS 1975 NCC, vol. 44, pp. 331-348, AFIPS Press, Montvale, New Jersey (1975).