ENGLISH WORDS AND DATA BASES: HOW TO BRIDGE THE GAP

Remko J.H. Scha

Philips Research Laboratories
Eindhoven
The Netherlands

ABSTRACT

If a q.a. system tries to transform an English question directly into the simplest possible formulation of the corresponding data base query, discrepancies between the English lexicon and the structure of the data base cannot be handled well. To be able to deal with such discrepancies in a systematic way, the PHLIQA1 system distinguishes different levels of semantic representation; it contains modules which translate from one level to another, as well as a module which simplifies expressions within one level. The paper shows how this approach takes care of some phenomena which would be problematic in a more simple-minded set-up.

I INTRODUCTION

If a question-answering system is to cover a non-trivial fragment of its natural input-language, and to allow for an arbitrarily structured data base, it cannot assume that the syntactic/semantic structure of an input question has much in common with the formal query which would formulate in terms of the actual data base structure what the desired information is. An important decision in the design of a q.a. system is therefore, how to embody in the system the necessary knowledge about the relation between English words and data base notions.

Most existing programs, however, do not face this issue. They accept considerable constraints on both the input language and the possible data base structures, so as to be able to establish a fairly direct correspondence between the lexical items of the input language and the primitives of the data base, which makes it possible to translate input questions into query expressions in a rather straightforward fashion.

In designing the PHLIQA1 system, bridging the gap between free English input and an equally unconstrained data base structure was one of the main goals. In order to deal with this problem in a systematic way, different levels of semantic analysis are distinguished in the PHLIQA1 program. At each of these levels, the meaning of the input question is represented by an expression of a formal logical language. The levels differ in that each of them assumes different semantic primitives.

At the highest of these levels, the meaning of the question is represented by an expression of the English-oriented Formal Language (EFL); this language uses semantic primitives which correspond to the descriptive lexical items of English. The primitives of the lowest semantic level are the primitives of the data base (names of files, attributes, data-items). The formal language used at this level is therefore called the Data Base Language (DBL). Between EFL and DBL, several other levels of meaning representation are used as intermediary steps. Because of the space limitations imposed on the present paper, I am forced to evoke a somewhat misleading picture of the PHLIQA set-up, by ignoring these intermediate levels.

Given the distinctions just introduced, the problem raised by the discrepancy between the English lexicon and the set of primitives of a given data base can be formulated as follows: one must devise a formal characterization of the relation between EFL and DBL, and use this characterization for an effective procedure which translates EFL queries into DBL queries. I will introduce PHLIQA's solution to this problem by giving a detailed discussion of some examples[1] which display complications that Robert Moore suggested as topics for the panel discussion at this conference.

II THE ENGLISH-ORIENTED LEVEL OF MEANING REPRESENTATION

The highest level of semantic representation is independent of the subject-domain. It contains a semantic primitive for every descriptive lexical item of the input-language[2]. The semantic types of these primitives are systematically related to the syntactic categories of the corresponding lexical items. For example, for every noun there is a constant which denotes the set of individuals which fall under the description of this noun: corresponding to "employee" and "employees" there is a constant EMPLOYEES denoting the set of all employees, corresponding to "department" and "departments" there is a constant DEPARTMENTS denoting the set of all departments. Corresponding to an n-place verb there is an n-place predicate. For instance, "to have" corresponds to the 2-place predicate HAVE. Thus, the input analysis component

---

[1] There is no space for a definition of the logical formalism I use in this paper. Closely related logical languages are defined in Scha (1976), Landsbergen and Scha (1979), and Bronnenberg et al.(1980).

[2] In previous papers it has been pointed out that this idea, taken strictly, leads not to an ordinary logical language, but requires a formal language which is ambiguous. I ignore this aspect here. What I call EFL corresponds to what was called EFL⁻ in some other papers. See Landsbergen and Scha (1979) and Bronnenberg et al. (1980) for discussion.

of the system translates the question

"How many departments have more than 100 employees ?" (1)

into

Count({x ε DEPARTMENTS | Count({y ε EMPLOYEES | HAVE(x,y)}) > 100}). (2)

III THE DATA BASE ORIENTED LEVEL OF MEANING REPRESENTATION

A data base specifies an interpretation of a logical language, by specifying the extension of every constant. A formalization of this view on data bases, and its application to a CODASYL data base, can be found in Bronnenberg et al.(1980). The idea is equally applicable to relational data bases. A relational data base specifies an interpretation of a logical language which contains for every relation $R[\underline{K}, A_1, \ldots, A_n]$ a constant K denoting a set, and n functions $A_1, \ldots, A_n$ which have the denotation of K as their domain.

Thus, if we have an EMPLOYEE file with a DEPARTMENT field, this file specifies the extension of a set EMPS and of a function DEPT which has the denotation of EMPS as its domain. In terms of such a data base structure, (1) above may be formulated as

Count({x ε (for: EMPS, apply: DEPT) | Count({y ε EMPS | DEPT(y)=x}) > 100}). (3)

I pointed out before that it would be unwise to design a system which would directly assign the meaning (3) to the question (1). A more sensible strategy is to first assign (1) the meaning (2). The formula (3), or a logically equivalent one, may then be derived on the basis of a specification of the relation between the English word meanings used in (1) and the primitive concepts at the data base level.

IV THE RELATION BETWEEN EFL AND DBL

Though we defined EFL and DBL independently of each other (one on the basis of the possible English questions about the subject-domain, the other on the basis of the structure of the data base about it) there must be a relation between them. The data base contains information which can serve to answer queries formulated in EFL. This means that the denotation of certain EFL expressions is fixed if an interpretation of DBL is given.

We now consider how the relation between EFL and DBL may be formulated in such a way that it can easily serve as a basis for an effective translation from EFL expressions into DBL expressions. The most general formulation would take the form of a set of axioms, expressed in a logical language encompassing both EFL and DBL. If we allow the full generality of that approach, however, it leads to the use of algorithms which are not efficient and which are not guaranteed to terminate. An alternative formulation, which is attractive because it can easily be implemented by effective procedures, is one in terms of translation rules. This is the approach adopted in the PHLIQA1 system. It is described in detail in Bronnenberg et al. (1980) and can be summarized as follows.

The relation between subsequent semantic levels can be described by means of local translation rules which specify, for every descriptive constant of the source language, a corresponding expression of the target language[1]. A set of such translation rules defines for every source language query-expression an equivalent target language expression. An effective algorithm can be constructed which performs this equivalence translation for any arbitrary expression.

A translation algorithm which applies the translation rules in a straightforward fashion, often produces large expressions which allow for considerably simpler paraphrases. As we will see later on in this paper, it may be essential that such simplifications are actually performed. Therefore, the result of the EFL-to-DBL translation is processed by a module which applies logical equivalence transformations in order to simplify the expression.

At the most global level of description, the PHLIQA system can thus be thought to consist of the following sequence of components: Input analysis, yielding an EFL expression; EFL-to-DBL translation; simplification of the DBL expression; evaluation of the resulting expression.

For the example introduced in the sections II and III, a specification of the EFL-to-DBL translation rules might look like this:

DEPARTMENTS → (for: EMPS, apply: DEPT)
EMPLOYEES → EMPS
HAVE → (λx,y: DEPT(y)=x)

These rules can be directly applied to the formula (2). Substitution of the right hand expressions for the corresponding left hand constants in (2), followed by λ-reduction, yields (3).

V THE PROBLEM OF COMPOUND ATTRIBUTES

It is easy to imagine a different data base which would also contain sufficient information to answer question (1). One example would be a data base which has a file of DEPARTMENTS, and which has NUMBER-OF-EMPLOYEES as an attribute of this file. This data base specifies an interpretation of a logical language which contains the set-constant DEPTS and the function #EMP (from departments to integers) as its descriptive constants. In terms of this data base, the query expressed by (1) would be:

Count({x ε DEPTS | #EMP(x) > 100}). (5)

If we try to describe the relation between EFL and DBL for this case, we face a difficulty which did not arise for the data base structure of section III: the DBL constants do not allow the construction of DBL expressions whose denotations involve employees. So the EFL constant EMPLOYEES cannot be translated into an equivalent DBL expression - nor can the relation HAVE, for lack of a suitable domain. This may seem to force us to give up local translation for certain cases: instead, we would have to design an algorithm which looks out for sub-expressions of the form

---

[1] I ignore the complexities which arise because of the typing of variables, if a many-sorted logic is used. Again, see Bronnenberg et al. (1980), for details.

$(\lambda y$: Count($\{x \in$ EMPLOYEES $|$HAVE(y,x)$\}$)), where y is ranging over DEPARTMENTS, and then translates this whole expression into: #EMP. This is not attractive - it could only work if EFL expressions would be first transformed so as to always contain this expression in exactly this form, or if we would have an algorithm for recognizing all its variants.

Fortunately, there is another solution. Though in DBL terms one cannot talk about employees, one can talk about objects which stand in a one-to-one correspondence to the employees: the pairs consisting of a department d and a positive integer i such that i is not larger than than the value of #EMP for d. Entities which have a one-to-one correspondence with these pairs, and are disjoint with the extensions of all other semantic types, may be used as "proxies" for employees. Thus, we may define the following translation:

EMPLOYEES → $\cup$(for: DEPTS,
apply: (λd:(for: INTS(#EMP(d)),
apply:
(λx:id_{emp}(<d,x>)))))

DEPARTMENTS → DEPTS

HAVE → (λy: rid(y[2])[1] = y[1])

where id_{emp} is a functionwhich establishes a one-to-one correspondence between its domain and its range (its range is disjoint with all other semantic types); rid is the inverse of id_{emp}; INTS is a function which assigns to any integer i the set of integers j such that $0<j\leq i$.

Application of these rules to (2) yields:

Count($\{x \in$ DEPTS $|$
Count($\{y \in \cup$(for: DEPTS,
apply:(λd:(for: INTS(#EMP(d)),
apply:
(λx:id_{emp}(<d,x>)))))$|$
rid(y)[1] = x$\}$) > 100$\}$) (6)

which is logically equivalent to (5) above.

It is clear that this data base, because of its greater "distance" to the English lexicon, requires a more extensive set of simplification rules if the DBL query produced by the translation rules is to be transformed into its simplest possible form. A simplification algorithm dealing succesfully with complexities of the kind just illustrated was implemented by W.J. Bronnenberg as a component of the PHLIQA1 system.

VI EXTENDING THE DATA BASE LANGUAGE

Consider a slight variation on question (1): "How many departments have more than 100 people ?" (7) We may want to treat "people" and "employees" as non-synonymous. For instance, we may want to be able to answer the question "Are all employees employed by a department ?" with "Yes", but "Are all people employed by a department ?" with "I don't know". Nevertheless, (7) can be given a definite answer on the basis of the data base of section III. The method as described so far has a problem with this example: although the answer to (7) is determined by the data base, the question as formulated refers to entities which are not represented in the data base, cannot be constructed out of such entities, and do not stand in a one-to-one correspondence with entities which can be so constructed. In order to be able to construct a DBL translation of (7) by means of local substitution rules of the kind previously illustrated, we need an extended version of DBL, which we will call DBL*, containing the same constants as DBL plus a constant NONEMPS, denoting the set of persons who are not employees. Now, local translation rules for the EFL-to-DBL* translation may be specified. Application of these translation rules to the EFL representation of (7) yields a DBL* expression containing the unevaluable constant NONEMPS. The system can only give a definite answer if this constant is eliminated by the simplification component.

If the elimination does not succeed, PHLIQA still gives a meaningful "conditional answer". It translates NONEMPS into $\emptyset$ and prefaces the answer with "if there are no people other than employees, ...". Again, see Bronnenberg et al. (1980) for details.

VII DISCUSSION

Some attractive properties of the translation method are probably clear from the examples. Local translation rules can be applied effectively and have to be evoked only when they are directly relevant. Using the techniques of introducing "proxies" (section V) and "complementary constants" (section VI) in DBL, a considerable distance between the English lexicon and the data base structure can be covered by means of local translation rules.

The problem of simplifying the DBL* expression (and other, intermediate expressions, in the full version of the PHLIQA method) can be treated separately from the peculiarities of particular data bases and particular constructions of the input language.

VIII ACKNOWLEDGEMENTS

Some of the ideas presented here are due to Jan Landsbergen. My confidence in the validity of the translation method was greatly enhanced by the fact that others have applied it succesfully. Especially relevant for the present paper is the work by Wim Bronnenberg and Eric van Utteren on the translation rules for the PHLIQA1 data base. Bipin Indurkhya (1981) implemented a program which shows how this approach accommodates the meaning postulates of Montague's PTQ and similar fragments of English.

IX REFERENCES

W.J.H.J. Bronnenberg, H.C. Bunt, S.P.J. Landsbergen, R.J.H. Scha, W.J. Schoenmakers and E.P.C. van Utteren: The Question Answering System PHLIQA1. In: L. Bolc (ed): Natural Language Question Answering Systems. München, Wien: Hanser. London, Basingstoke: Macmillan. 1980.

B. Indurkhya: Sentence Analysis Programs Based on Montague Grammar. Unpubl. Master's Thesis. Philips International Institute. Eindhoven. 1981.

S.P.J. Landsbergen and R.J.H. Scha: Formal Languages for Semantic Representation. In: S. Allén and J.S. Petöfi (eds): Aspects of Automatized Text Processing. Hamburg: Buske. 1979.

R.J.H. Scha: Semantic Types in PHLIQA1. Preprints of the 6th International Conference on Computational Linguistics. Ottawa. 1976.