THE REMIT SYSTEM FOR PARAPHRASING RELATIONAL QUERY EXPRESSIONS INTO NATURAL LANGUAGE.

B.G.T. LOWDEN and A.N. DE ROECK

University of Essex - Department of Computer Science
COLCHESTER - ENGLAND

1. ABSTRACT.

REMIT - Relational Model Interpreter and Translator - is a formal query language to natural language interpreter designed to aid query verification in a relational database environment. The system has been developed to work in conjunction with the ICL natural language query interface, NEL, which translates English query expressions into the formal query language QUERYMASTER. Funding for this research project has been provided by International Computers Limited.

2. INTRODUCTION.

Of the many problems facing the casual user of a database enquiry system probably the most difficult is gaining a competent understanding of the query language the system expects him to use. Even if he manages to formulate a syntactically correct query expression, there is no guarantee that it will reflect the question he intended to ask. In a study of Query by Example (Thomas and Gould 1975), it was found that 27% of the queries analysed were syntactically correct but they did not correspond to the questions the users thought they had asked.

Natural language (NL) processing is seen by some as a promising solution to these difficulties. However, NL interfaces, allowing users to ask questions in their own language, can create problems of their own. They encourage the user's often inflated ideas about what is a reasonable question to ask. Furthermore, NLs are typically ambiguous and, although transparent to the querent, they are opaque to the machine. The interpretation which the interface must place on a NL query in order to allow for its evaluation may therefore itself be misleading, especially if it is established outside the user's control.

We can improve this situation by producing a NL paraphrase of what the system has taken the query to mean, allowing a user to check whether its interpretation of his question coincides with what he wants to ask and in the case of ambiguous input to select the alternative that does.

This paper describes such a paraphraser designed at the University of Essex and implemented in Prolog. The system has been constructed to deliver paraphrases for queries formulated in the formal query language QUERYMASTER (ICL 1983, 1985) which are used to retrieve data from an ICL example database called SCOPE. Consequently, unlike most NL feedback systems, it can also help those with no access to NL input facilities and who must use a formal language. Furthermore, the paraphraser assumes an extended Relational Calculus (RC) as an underlying representation, so it can quite easily be made to work for most current query languages.

The system can also paraphrase NL questions interpreted by the database query sytem NEL (Natural Enquiry Language), an ongoing ICL research project formerly known as QPROC (Wallace and West 1983). NEL consists of a front end which maps NL text onto expressions in the ICL formal query language QUERYMASTER.

2. DESIGN APPROACH.

A paraphraser is a mechanism which maps an underlying formal representation onto a NL text. Both representation and text must be suited to the task on hand. Given that our aim is to provide casual database querents with a useful paraphrase of their question, the representation must capture how the system understands the input. This understanding must then be translated into clear, unambiguous and grammatical textual output.

2.1 Selection of the underlying representation.

One can classify paraphrasers in two groups, based on the kind of underlying representation they assume. One type will specifically work alongside a NL front end (McKeown 1979). The user's question is introduced in a NL and parsed into a structure which makes linguistic facts explicit about the input. This representation, which is motivated linguistically, then serves as the starting point for the paraphrase. The mapping between formal representation and text established by this kind of system is "close" as the synthethiser can obtain most of the linguistic information needed from the parse tree of an equivalent NL question.

Nevertheless, not all NL questions users may formulate are evaluable against a database. Since systems of this kind map the NL query into a formalism which does not necessarily reflect the limitations of the Database Management System, the result may be a paraphrase of a question the system cannot ultimately handle. Also, paraphrasers working from linguistically motivated representations cannot work independently from a parser that will build the necessary structure. They do not help the user who has no access to a NL front end and must therefore use a formal query

Proceedings of the Twelfth International Conference on Very Large Data Bases

Kyoto, August, 1986

language.

The other kind of system assumes representations which capture exactly that information which can be evaluated against a database, usually a formal query expression. Used with a NL interface it can report only on relevant ambiguities in the input, ie. in so far as they correspond to alternative evaluable formal queries. However, since these expressions are linguistically underspecified, the mapping it must establish is more difficult to achieve. No useful linguistic facts for building the paraphrase can be retrieved by reference to a linguistically motivated parse of an equivalent question. Although this approach is more constrained than its alternative, and its results probably less spectacular, it does allow for the paraphraser to be used both with, and without, a NL front end, and as such, is the one adopted for this project.

2.2 Portability.

The preceding discussion seems to suggest QUERYMASTER as the obvious candidate for the underlying representation. However, the choice of QUERYMASTER expressions as the paraphraser's starting point would restrict the system's use to those Database Management Systems capable of supporting that query language.

In order to retain all the advantages of a paraphraser working from representations which capture exactly the information present in a formal query and, at the same time, to increase its portability, the mapping process between formal query and NL text has been split into two stages using an intermediary formalism. Two considerations have guided the choice of that formalism. Firstly, it must be able to express exactly what can be captured by any QUERYMASTER expression. Secondly, it must be possible to define an exact mapping from other query languages into expressions of that formalism.

The use of an applied relational calculus (RC) as an intermediary representation satisfies these two considerations. As defined by (Codd 1971, 1972), the RC is well defined and relationally complete. Extending it by a range of library functions (Date 1977), gives it at least the retrieval power of most query languages currently available. Furthermore, (Ullman 1980) has shown that an exact mapping exists between an expression in any relationally complete language and an expression in the RC (and vice versa), provided that expression defines a derivable relation.

Therefore, one part of the project involved the development and implementation of a transducer which maps QUERYMASTER statements into expressions of the RC (Shephard 1985). The function of this program (which is written in Prolog) is independent of the main body of the paraphraser. Any further reference to the latter will assume that it works directly from the RC.

This modular concept means that the process of adapting the paraphraser to work from other relational query languages is relatively straightforward.

2.3 Grammaticality.

It is important that the text produced by the paraphraser should be wellformed according to the grammar rules of the human language in which the query is described (in this case English). Generally speaking, grammaticality of a text can best be ensured if the text generation process refers to a linguistic theoretical framework which can accomodate such grammar rules. However, many such frameworks exist and some guidelines for making a choice between them were drawn up. First of all, the framework should be implementable. This means that it should be formally specified to a level where an equivalent program can be written. Secondly, the syntax of the RC and the syntax of English are very dissimilar. As a consequence, the mapping between RC formulae and English texts can be called "distant", and should, as far as possible, be left to the linguistic part of the implementation. In practical terms, this means that a linguistic theory which assumes an underlying representation that is poorly specified with respect to NL syntax will be preferred.

The choice made was Lexical Functional Grammar (LFG) (Kaplan & Bresnan 1983). LFG is a generative linguistic theory allowing for the specification of NL grammar rules. It has the advantage of being highly implementable - in fact is was designed from a computational linguistic point of view. As extended by (Halvorsen 1983) it defines a mapping between sentences of English and underlying, syntactically poorly specified, predicate/argument structures. For our purposes, it has an additional advantage in that the mapping it defines is stratified, using several intermediary representations at different levels of linguistic description. This gives an indication of which linguistic information needs to be specified at a given stage in the process.

2.4 Non-ambiguity.

Whereas grammaticality can be ensured by reference to a grammar, non-ambiguity cannot. The problem arises from the fact that all human languages are ambiguous. Grammars try to account for ambiguity but do not seek to avoid it. In short, if the definition of a paraphrase only requires it to be a grammatical text in some human language, then it follows that it is potentially ambiguous.

Ambiguity is a phenomenon that is difficult to control. No measure for a degree of ambiguity exists. One may attempt to parse the output text and thus try to gain some such measure, but many different sorts of ambiguity occur and it is not clear whether any grammar can account for all of them. Lexical ambiguity in particular is problematic and, in the extreme case, words may mean a variety of different things to different users. This is often dependent upon the users' backgrounds, and totally beyond the control of any grammar formalism. The solution adopted by REMIT was to concentrate on ambiguities which MUST be avoided at all cost in order to render the formal query's meaning accurately.

Since the aim of a paraphrase is query verification, the ambiguities which must be avoided are those significant with respect to

query evaluation. These mainly relate to the scope of logical connectives and quantifiers occurring in an expression of the unambiguous, formal query language. For example, consider the following sequence of logical conjunctions and disjunctions:

$$a \wedge (b \wedge (c \vee (d \wedge e)))$$

"a and b and c or d and e" is not an adequate, nor a helpful, rendering of the above bracketed expression since all indication of scope is lost. We found that the written form of a human language, stripped of expressive devices such as intonation and stress patterns is extremely ill-suited to express scope relationships of this kind. If one tries to describe the scope information by means of punctuation and special words (eg. either, both, all three of the following) then the resulting linear text becomes illegible and unhelpful as a paraphrase. REMIT solved this problem by abandoning the idea of a paraphrase as a linear text. It adopted the view that scope is best represented hierarchically. As a result, the paraphraser retains some degree of explicit structure in the output text which is then used to display the result on the screen using indentation as a means of conveying scope. The formal sequence above would for REMIT result in a paraphrase displayed in the following format:

```
        a
and b
and either c
        or d and e
```

2.5 Readability.

The requirement that the paraphrases delivered must be "readable" has in part been satisfied by the solution adopted to avoid scoping ambiguity. A side effect of structuring the output text has been that it becomes indeed more readable and thus more friendly as vital scoping information is passed on visually to the user.

However, there is more to "readability" than producing a grammatical text and displaying it in a particular format. The text must also be coherent, not just syntactically, but also conceptually. What we mean by this is explained in more detail in the next section.

3. A MODEL FOR THE SCOPE DATABASE.

Paraphrasing expressions in a query language comes down to selecting and organising the appropriate lexical material for describing, in a human language, what the query stands for in terms of the information that must be retrieved. Query languages and the RC are formal languages, i.e. their semantics with respect to retrieval is defined unambiguously on the basis of their syntax. As a consequence, the syntactic structure of a formal expression can be viewed as a shorthand for what it "means" and can be used in order to guide the paraphrasing process.

Nevertheless, paraphrasing expressions of these formal languages can be problematic. Their syntax bears no resemblance to the syntax of a human language and a parphraser must perform more than a simple syntactic transduction. Also, these formal expressions are poor in conceptual information about the domain or field a particular database covers. Although one can produce literal paraphrases relying solely on the information present in a formal query, the result will be a stunteled incoherent rephrasing of the formal expression. In general, human language text is rich in conceptual information. If a paraphraser is to deliver texts that are acceptable and helpful to naive users then it must be able to produce output that is rich in conceptual terms.

This conceptual information cannot be collected from the database itself. Databases are implementations of formal objects that allow for storing and manipulating large bodies of knowledge. Although the administrative organisation of a relational database will often, to a large extent, be compatible with the conceptual structure of the field it covers, this is largely due to the "common sense" of database engineers and such an organisational correspondence cannot always be guaranteed.

In addition, formal query languages are totally devoid of any such conceptual information. Consider, for instance, the following RC formula:

```
{ (CUSTOMER.NAME, WAREHOUSE.CODE) : true }
```

as might be expressed over the example SCOPE database, reproduced in Fig.1 from (ICL 1983). This is a perfectly wellformed RC expression. It has an equivalent in all relational query languages and the result will be the Cartesian product of all customer names with all warehouse codes. Although this is certainly a legal query, it is hard to see what it might "mean" in conceptual terms and why anybody might want to formulate it. Paraphrasing queries of this kind is extremely difficult, even for people, short of saying "Give me the Cartesian product of all values for ..."

Still, most questions which users care to ask do make sense and usually carry conceptual content. They centre around a focal point or FOCUS which is not explicitly marked as such in the original formal expression, but which can be derived from it given conceptual information about the field the database covers.

This conceptual information will be contained within a MODEL of the database in question. Such a model must be constructed for any database with which the paraphraser will operate. Note that models designed for this particular application do not seek to settle the conceptual structure of the domain a database draws upon. They merely intend to provide the linguistic/conceptual information which is necessary for the delivery of coherent and elegant paraphrases. The model for SCOPE in the REMIT system contains three kinds of information as described in the sections below.

3.1 Information for Focus selection.

For a query to be conceptually coherent in terms relevant to this project means that all relations involved in that query must be linked. At a database level, these links can be pointers, or value based relationships. In this sense, a conceptually coherent query relates to a

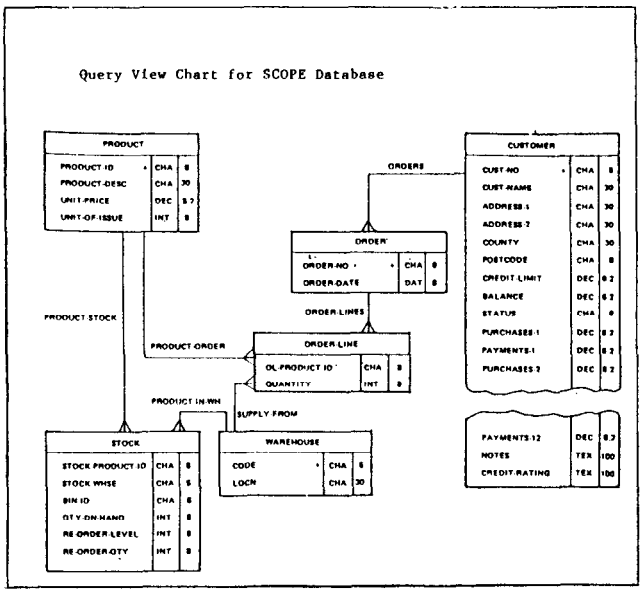


FIGURE 1

consistent subset of the database. To give an example for the SCOPE database: a query involving the relations ORDERLINE and CUSTOMER is conceptually coherent only in the case where it also refers to the relation ORDER, otherwise it is impossible to establish a link between ORDERLINE and CUSTOMER. Intuitively this can be seen as defining a notion of "paraphrasable query" over a particular database. Note, however, that this situation is not a consequence of what the formal language will allow, since there are wellformed formal expressions which are not conceptually coherent in the sense described above.

The intuition behind the notion of a paraphrasable query as expressed over a consistent subset of database relations linked by relationships, is that the subset corresponds to a "network" which the query lifts off the database and which can be used by the paraphraser to guide the building of a coherent conceptual structure underlying the output text. The system thus uses two different control structures: the parse tree of the RC expression for rendering the query's content and the "network" the query defines over the database for building a coherent conceptual structure underlying the output.

However, structuring the output text on the basis of a "network" of relations and relationships raises a question about selecting an appropriate starting point. That starting point will be the FOCUS of the query.

Three assumptions underly the selection of an appropriate focus:

1. Every paraphrasable query has a focus.
2. A focus is a single relation in the current database which occurs in the query's associated "network".
3. Every relation other than the focus, involved in a particular query, must be linked directly or indirectly to that focus. This means that there is a path from the focus to every other relation mentioned in the query and such that none of the links (database relationships) making up that path refers to a relation which is not specified explicitly in the query.

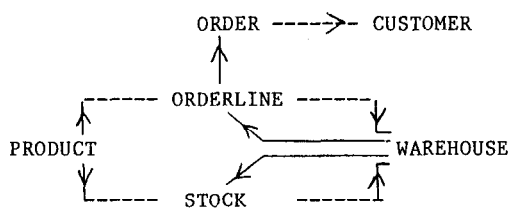
To be useful for focus selection, the third assumption needs constraining. Since the query defines a network of relations and relationships over the database, and since database relationships are not directed, every relation in that network becomes a candidate for focus according to the above criteria. The model therefore makes the distinction between the directions in which a relationship can be traversed and associates lexical material (usually an English predicate) for describing that relationship with respect to the direction of traversal. For example, the database relationship between ORDER and CUSTOMER can be associated with the English predicate "to place" in the following way:

```
ORDER ------------------------> CUSTOMER
                  !
        [to place: [arg1: CUSTOMER]
                   [arg2: ORDER] ]
```

where the specification of the arguments indicates that the customers place the orders.

Under these circumstances, the network over the database which is associated with a paraphrasable query becomes a tree, and the third assumption means that the focus relation is the root of the tree (with relations as nodes and relationships as arcs) which the query defines over the database. This now leads us to formulate a paraphrasing strategy which starts building a description of the derived relation by first paraphrasing the focus. Subsequently, it moves along the paths of the tree, stepping through each of the links that makes up such a path. The notion of focus, as described above, thus enables us to specify a recursive paraphrasing strategy.

Two practical points must be made here. The model adopted by REMIT stands in an elementary form. Only one flow of directionality has been imposed, with two exceptions, as illustrated below.



As a consequence, a query involving only ORDER and CUSTOMER will always have the ORDER relation as focus. Furthermore, given the recursive paraphrasing strategy, we predict that if two way directionality is imposed on the database relationships then, in order to avoid circularity, the two flows of direction must be kept separate.

3.2 Information for describing database objects.
In addition to allowing for the selection of an appropriate focus and for the description of

directed database relationships, the model also provides for the description of other database objects. Both relations and attributes are associated with alternative descriptions (usually English nouns or complex nouns). The paraphraser will pick one of the alternatives thus specified depending on what focus has been selected for the query that is being paraphrased. For instance, the attribute CUSTOMER.CUST-NAME will be described as "name" in a situation where the current focus is CUSTOMER and as "customer name" when another focus has been picked.

3.3 Information for describing RC constructs.
The model also contains linguistic information for describing elements of the RC syntax. Boolean operators, for instance, will be associated with English descriptions relative to the conceptual type of the attributes they are used to compare. For example, the operator "<" will be paraphrased as "cheaper than" when it compares prices, "alphabetically classified before" for names, "smaller than" for numbers, "before" for dates, etc. The conceptual type of the attributes compared is derived from the NEL "End User View" which has been incorporated into the REMIT model.

4. PARAPHRASER OVERVIEW.

Given an RC expression, the paraphraser will perform its task in four steps. First of all, the RC expression will be parsed into a structure which makes the syntactic build-up of the formula explicit. After completion of a basic parse, the resulting structure is converted into a list and some of its components are flattened out and simplified so that they can be handled more easily by the rest of the program. This is implemented as a prolog DCG on the Essex Dec-10 based on a context free grammar that specifies the calculus syntax and whose rules are applied top down depth first.

Secondly, the focus of the input query is determined on the basis of the relations occurring in that query, according to the stages described in the previous section. As expected, the model for the database plays an important role in this part of the process.

In a third step, the parsed RC expression is paraphrased relative to the focus discovered in the previous step. This part of the paraphraser, to be described in more detail in the next section, produces the conceptual/linguistic predicate/argument structure which underlies the final paraphrasing text.

During the fourth step, the predicate/argument structure is assembled into an English text which retains some degree of explicit structure used to determine the format of what will appear on the screen. This small degree of structuring allows for the output to reflect the scope of logical operators. The original aim was to develop this component as a full LFG generator. However, although the predicate/argument structure which is input to this module of the system is compatible with LFG (as extended by (Halvorsen 1983)), it was felt that the work should, in the short term, concentrate on the third stage described above, since completion of the latter was judged more critical for the success of the project as a whole. For these reasons, only a basic linguistic component has been implemented, which concentrates largely on agreement and word order. However, the lack of a fully implemented theoretically sound linguistic component, seems not to have impaired the quality of the paraphrases delivered. This suggests to us that, for synthesising human language text from formal languages, the implementation of a sophisticated syntactic component is subsidiary to the development of a mechanism that settles the conceptual structure underlying the final text.

5. PARAPHRASER STRATEGY.

The main body of the paraphraser utilises three categories of information in order to guide its actions. First of all, it analyses the syntactic structure of the incoming formal expression. Secondly, information is provided regarding the focus relation of that expression. Thirdly, both the previous items of information are used to specify the tree of database relationships and relations defined by the query.

The syntax of the RC expression is used to determine the overall format of the paraphrasing text. First, those parts of the query which specify a number of options open to the user, eg. user defined functions, ordering requirements on the retrieved information, etc., are singled out. They are paraphrased as separate sentences which either preceed or follow the text describing the main body of the formal query.

Then the main body of the query is described. A wellformed query must have a left hand side, specifying the information to be retrieved, and a right hand side constraining that information. For instance, in:

```
{ (CUSTOMER.CUST-NAME) :
      /     ( (CUSTOMER.ADDRESS2 = 'LONDON') ) }
   LHS                        \RHS
```

the left hand side specifies that customer names must be retrieved. The right hand side restricts that retrieval to the names of those customers who live in London.

This basic syntactic structure is reflected in the format of a standard paraphrase which is the following:

For <DERIVED RELATION> <ACTION VERB> <TARGET LIST>

<DERIVED RELATION> is the paraphrase of the right hand side of the input query, and <TARGET LIST> of the left hand side. <ACTION VERB> is some English predicate selected on the basis of what items are contained in the left hand side (eg. "show" for attributes, "calculate" for functions, etc.).

The description of <DERIVED RELATION> relies upon the tree of database relationships and relations which the model has assigned to the query as a control structure to guarantee that the output text will be conceptually coherent. The paraphraser uses the syntactic structure of the RC query to settle the content of the query description. Overall, it distinguishes between

different kinds of comparisons that can occur on the right hand side of the formal expression. These include:

- ORDINARY comparisons, comparing the value of a database attribute with a constant.
- LINKING comparisons, comparing by means of "=" key attributes of relations between which a relationship exists in the database. These correspond to "links" along paths in the conceptual tree as defined by the model.
- COMPLEX comparisons involving attributes of different relations without being linking comparisons.
- DISCONTINUOUS comparisons which are groups of comparisons bundled together under a different logical operator from that of the previous level.

The paraphraser starts by describing the focus of the query. This involves not only a paraphrase of the focus relation itself, but also of ordinary comparisons involving an attribute of that relation. In the next step, all linking comparisons between the focus and other relations one step along the paths in the conceptual tree are paraphrased. When one such link is described, the relation newly linked to the old focus is propagated as a subsidiary focus. This new focus is described in turn, including links to other relations along the path, which will in time also become subsidiary foci. When all links along a path have been described, the old foci are (recursively) restored. The lexical material used to describe a particular database object or relationship will depend upon what is the current (possibly subsidiary) focus at that stage of the process. All partial paraphrases are linked together by means of the appropriate logical operators. Paraphrasing is thus done recursively, relative to the syntactic structure of formal expression components, the focus of the query and the conceptual tree delivered by the model.

For the top level focus, all four types of comparison are described in turn. However, for subsidiary foci, complex comparisons are omitted. They typically involve two relations and it is difficult to decide at which stage they should be paraphrased. All complex comparisons are therefore paraphrased relative to the overall, top level focus.

The elements of the left hand side are described by retrieving from the model the appropriate lexical material with which they are associated relative to the overall focus of the query. The descriptions of similar objects are conjoined and grouped with an appropriate verb. Such verb phrases, if applicable, can also be conjoined.

6. AN EXAMPLE.

To illustrate the operation of REMIT we give a comprehensive query example, defined on the SCOPE database, showing each stage of the transduction and paraphrasing process. This is one of many such examples compiled jointly by ICL and Essex to test the different features of the software.

QUERYMASTER:

List stock.stock-whse, bin-id, stock-value is qty-on-hand * unit-price sorted by ascending warehouse.locn where warehouse.locn > 'London' and re-order-qty < 100 and product-stock and product-in-whse starting stock

RELATIONAL CALCULUS:

```
{ stock-value (stock.qty-on-hand,
                        product.unit-price) :=
('stock.qty-on-hand * product.unit-price')
w (stock.stock-whse, stock.bin-id,
stock-value (stock.qty-on-hand,
                  product.unit-price) )  :
(∃ wh ∈ warehouse)
( ( (wh.locn > 'London') ∧
    (stock.reorder-qty < 100) ) ∧
  ( (stock.whse = wh.code) ∧
   (stock.product-id = product.product-id) ) ) )
up (wh.locn) }
```

COMPUTED FOCUS:

PRODUCT (Note that PRODUCT is not referred to in the target list)

PARAPHRASE:

For products
which are physically stocked
and whose reorder quantity is less than 100
and which are stored in warehouses
whose location is alphabetically
listed after 'London'

(1) show
a. the warehouse codes
b. the bin numbers
and
(2) calculate and display stock value

where stock value is defined as stocked quantity on hand * product unit price.

Sequence the result by ascending warehouse location.

7. CONCLUSION.

This paper has described a prototype paraphraser developed as an ICL funded research project at the University of Essex. It has been fully implemented in Prolog on the University's Dec-10. The paraphrasing process, as described, has been split into two steps using an extension of the relational calculus as an intermediary representation. This design feature enhances the potential portability of the paraphraser over relationally complete query languages. The system has been successfully tested for a wide range of sample queries and results have both justified the extensive efforts spent in defining a suitable model and also underlined the importance of selecting an appropriate focus to guide the paraphrasing process. Furthermore, it has been shown that the provision of a sophisticated NL grammar formalism is subsidiary to the development of a mechanism for defining the underlying coherent and unambiguous conceptual structure of the output paraphrase. Overall, the system has demonstrated that it is feasible to deliver paraphrases of formal query language expressions which are helpful to the user in verifying whether his question reflects his intention.

ACKNOWLEDGEMENT.

The authors are indebted to V. West of International Computers Ltd for many helpful discussions on this work.

Funding for this research has been provided by International Computers Limited.

8. REFERENCES.

CODD, E.F. (1971)
"A Database Sublanguage founded on the Relational Calculus", in Proceedings of the ACM SIGFIDET workshop on Data Description, Access and Control.

CODD, E.F. (1972)
"Relational Completeness of Database Sublanguages in Database Systems" in Courant Computer Series, Vol 6, Prentice Hall, Englewood Cliffs.

DATE, C.J. (1977)
An Introduction to Database Systems, Addison Wesley,

HALVORSEN, P.K. (1983)
"Semantics for Lexical Functional Grammar", in Linguistic Inquiry, Vol 14, No 4, pp 567-615.

INTERNATIONAL COMPUTERS LTD (1983)
Using Querymaster (200 Level), VME 2000, Publication R00260/00.

INTERNATIONAL COMPUTERS LTD (1985)
Using Querymaster (QM.250), Publication R00433/01.

KAPLAN, R. and J. BRESNAN (1982)
"Lexical Functional Grammar. A Formal System for Grammatical Representation", in J. BRESNAN (ed), The Mental Representation of Grammatical Relations, MIT Press, Cambridge (Mass.).

McKEOWN, K. (1979)
"Paraphrasing using Given and New Information in a Question Answering System", in Proceedings of the 17th ACL, La Jolla.

SHEPHARD, I. (1985)
Implementation of a Transduction Algorithm to Convert Querymaster Language Statements into Relational Calculus, MSc Dissertation, University of Essex.

THOMAS, J.C. and J.D. GOULD (1975)
"A psychological Study of Query by Example", in Proceedings NCC 44.

ULLMAN, J.D. (1980)
Principles of Database Systems, Computer Science Press.

WALLACE, M. and V. WEST (1983)
"A Natural Language Database Enquiry System Implemented in Prolog", in ICL Technical Journal, November 1983.