# GENERATING ENGLISH PARAPHRASES FROM FORMAL RELATIONAL CALCULUS EXPRESSIONS

A.N. De Roeck and B.G.T. Lowden
University of Essex
Wivenhoe Park
Colchester - United Kingdom

## 0. ABSTRACT

This paper discusses a system for producing English descriptions (or "paraphrases") of the content of formal relational calculus (RC) formulae expressing a database (DB) query. It explains the underlying design motivations and describes a conceptual model and focus selection mechanism necessary for delivering coherent paraphrases. The general paraphrasing strategy is discussed, as are the notions of "desirable" paraphrase and "paraphrasable query". Two examples are included. The system was developed and implemented in Prolog at the University of Essex under a grant from ICL.

## 1. INTRODUCTION.

Querying databases (DB) is often problematic for casual users, who cannot be expected to master fully the art of expressing themselves in a query language. Much has been said about how natural language (NL) front ends (FE) would help them, but NLFEs create problems of their own. In mapping from an ambiguous NL to an unambiguous formal one they must make a number of decisions regarding interpretation of the input which remain outside the user's control. This may introduce new misconceptions which cannot always be detected from the format of the retrieved information. [13]

One solution to the problem is to present the casual user with a NL description of what his query has been taken to mean. He can then verify whether its interpretation corresponds to what he intended to ask. This paper describes such a description generator, or "paraphraser" as we will call it.

## 2. DESIGN PRINCIPLES.

The system is intended to work alongside NEL (formerly QPROC [14]), an NLFE which is a current research project at ICL. NEL uses "Descriptions and Qualifiers" (D&Q) as an intermediary representation (IR) in mapping English sentences on Querymaster (QM - an ICL query language). However, IR such as D&Q which are to an extent linguistically motivated can often represent queries which cannot be evaluated against the DB. Since the aim is to inform the user about how his query has been ultimately understood, a text derived from such IRs may be misleading. This influenced our choice of QM as the input to the paraphraser, which results in a system which will equally benefit users who have no access to a NLFE. The QM question is translated into the RC before paraphrasing which makes the system independent of QM syntax. Since a trivial mapping exists between all relationally complete query languages and the RC, this step enhances the system's portability.

The paraphrase must be grammatical, and this can best be ensured by reference to a linguistic theory which can contain a grammar. Our choice of Lexical Functional Grammar (LFG - [9,7]) as an underpinning is to some extent arbitrary, as most implementable generative theories will accomodate our needs. Yet, the high degree of stratification in the theory was thought an advantage as it predicts which linguistic information must be calculated at every stage in the process. This feature gains importance relative to the distance covered by the mapping process, which in this case is large.

The system design is as follows. Input QM queries are syntactically transduced into RC expressions which are then parsed. Their syntactic structure, in formal languages a shorthand for their semantics, is fed to the main body of the paraphraser for which it also acts as control mechanism. The system consults a model of the current DB to map the RC parse trees into LFG compatible predicate/argument structures (as they are defined in [7]), which can be passed to an LFG syntactic generator. A QM expression, even if obtained via a NLFE, cannot carry linguistic information; the text produced sustains no other relation to a possible NL input query than what is carried by a QM command.

## 3. WHAT IS A "DESIRABLE" PARAPHRASE.

Apart from being grammatical, the paraphrase should be unambiguous to be useful for query verification. Yet, NLs are ambiguous. Though there is no general solution to this clash of interests, the problem can be contained by concentrating on those ambiguities which must be avoided at all cost in this context, i.e. involving scope of quantifiers and logical connectives. Scoping information is hard to convey in a linear text which must remain easy to read, but easy to express in a tree. The requirement for this system to produce linear text was not considered paramount and consequently the paraphrases will use indentation to express logical connective scope.

## 4. COHERENCY, FOCUS AND THE MODEL.

RC formulae are poor in conceptual information about the field a DB draws on (and rightly so). They refer to DB objects but do not qualify the relationships between them in conceptual terms. NL texts, however, are rich in conceptual information and if a coherent helpful paraphrase is to be generated rather than a stuntled literal description of the original formal query, an underlying conceptual structure must be defined. The necessary information cannot be derived from the DB itself, but must come from a model of the domain the DB covers. Such a model must be constructed for each DB the paraphraser is to be used with: a task for which no clear cut formal guidelines exist and which may be possible only for DBs whose organisation is in broad intuitive terms compatible with the conceptual build of the domain it covers. The current propotype works from the SCOPE DB, developed at ICL as a tool for research, which has the required properties. It holds data on customers, products, stock, orders and warehouses. Its overall structure is given below:

$$\begin{tikzcd}
 & \text{orders} \arrow[r] & \text{customer} \\
\text{product} \arrow[ur] \arrow[r] \arrow[dr] & \text{order-line} \arrow[r] & \text{warehouse} \\
 & \text{stock} \arrow[ur] &
\end{tikzcd}$$

The model written over SCOPE contains three kinds of conceptual/linguistic information. Each DB object is associated with several lexical items by which it can be described, relative to the current focus. It attaches English predicates to relationships between these objects: e.g. "to place" with customer as its 1st, order its 2nd argument marks the link order-customer. Secondly, it holds lexical material to describe RC syntactic operators relative to the conceptual type of their arguments. Thirdly, it contains information for selecting a focus for each RC expression. The notion of focus, which is crucial for producing conceptually coherent text, is rooted in the assumption that queries "make sense" and must be elaborated.

Three principles underpin focus selection. The focus of a RC query is a relation in the current DB. Any other relation in the RC expression is linked to it, directly or indirectly, in such a way that the network of links and relations in the query form a tree over the DB, with the focus as the root and where the nodes in the tree cover all and only the other relations in the expression. Consequently, if the paraphraser relies on the conceptual information associated with the nodes and branches of the tree, the result will be a coherent text. Finally, each "paraphrasable" query has a focus. Note that the last principle is not a matter of wellformedness as queries such as {(order, product):true} can be legally expressed in the RC, retrieving the cartesian product of all customers and products in the DB because the link between the two relations is not specified. Apart from being hard to describe, it is also difficult to see what the point of such a query is. To count as paraphrasable, a query over orders and products in this DB should also specify order-line.

## 5. PARAPHRASING STRATEGY.

In mapping the RC query into a predicate/argument structure underlying a coherent English text the paraphraser is guided by the syntactic structure of the query. It singles out user defined functions and sorting requirements on the retrieved data and describes them in separate sentences to be added to the front or end of the main text. In describing the main body of the query the system also relies on the focus of the expression and the network over the DB it entails. The paraphraser first introduces the focus and describes all comparisons relevant to the focus relation as relative clauses to that (nominal) description. It then walks down the branches of the conceptual tree over the DB making each node in turn the subsidiary focus and describing parts of the formal expression relevant to it. This part of the system thus works recursively. Care is taken not to modify the scope of logical connectives, as reflected in the RC parse tree. The lexical items necessary for paraphrasing DB objects and the links between them are retrieved from the model.

The success with which such a recursive strategy can be adopted directly depends upon the selection of an appropriate focus. The definition of an adequate conceptual model over a DB is crucial to the ease with which queries on it can be paraphrased.

## 6. TWO EXAMPLES.

The following examples are drawn from the testing results of the prototype. The first shows how scope of logical connectives is rendered by indentation. The second illustrates how user defined functions are described (which is problematic as there is no restriction on the name or arity they can be given). Note how the same boolean operator (<) is rendered differently in the two examples, depending on the conceptual type of its arguments. The RC identifiers with "." in the middle are attribute names.

**EXAMPLE 1**
**RC expression:**
```
 {(customer.cust-name), order :
   ((∃ od ∈ order-line) (∃ pe ∈ product)
   ((pe.product-id=od.product-id) &
    ((od.order-no=order.order-no) &
     ((order.order-no<458879) &
     ((order.cust-no=customer.cust-no) &
      ((customer.cust-name="Vegetables-Assoc") v
       (customer.cust-name="Machines-Ltd")))))))}
```
**Selected focus:** PRODUCT
**Paraphrase:**
```
For products
   which are ordered
and
  which are contained in orders
           whose number is smaller than 458879
       and
           which are placed by customers
                   whose name is Vegetable Assoc
                 or
                   whose name is Machines LTD
```
(1) give details of each order involved
(2) show the customer names.

**EXAMPLE 2**
**RC expression:**
```
{stock-value(stock.qty-on-hand,product.unit-price)
  :=('stock.qty-on-hand * product.unit-price')
stock-value(stock.qty-on-hand,product.unit-price):
   ((product.product-id=stock.product-id) &
    (product.unit-price < $1.5))}
```
**Selected Focus:** PRODUCT
**Paraphrase:**
```
For products
   whose unit price is cheaper than $1.5
and
   which are stocked
```
calculate and display stock value,
where stock value is defined as stocked quantity available * product unit price.

## 7. SHORTCOMINGS.

The prototype is the result of a single year's work, which forced us to take account of priorities. First of all, the LFG syntactic generator has not been fully implemented. The surface string is collected from the predicate/argument structure by an ad hoc procedure. Adding a full LFG generator, which can be done easily because of the modular system design, will improve the quality of the output text.

Also, as a prime target the system aimed at covering that subset of the RC corresponding to QM. Since QM does not allow for universal quantification it has not been introduced in the present prototype, although provision have been made for its inclusion.

## 8. CONCLUSION.

The test results for the prototype have justified the extensive effort spent in defining an adequate DB model for the application on hand. They have also demonstrated the importance of defining a suitable mechanism for selecting a focus to guide the paraphrasing process and around which the conceptual structure of the text can be centered. The current system delivers paraphrases of a high quality in spite of it not incorporating a full syntactic generator. This fact supports our claim that, for synthesising NL text from formal expressions, the presence of an elaborate NL grammar formalism is subsidiary to the development of a mechanism that defines a coherent underlying conceptual structure. The prototype has demonstrated that it is possible to deliver paraphrases of query language expressions which are helpful to a user who wishes to verify his question. It has successfully countered the reservations that paraphrasers of this type necessarily deliver text which is "a mere 'syntactically sugared' variant of the original formal expression" [1], and that their output must, for complex queries, "in fact become virtually unreadable" [ibid.].

## 9. BIBLIOGRAPHY.

[1] B.K.BOGURAEV and K. SPARCK JONES, "A Natural Language Front End to Databases with Evaluative Feedback", in GARANDIN and GELENBE (eds), New Applications of Databases, Academic Press, London, 1984.

[2] W.F.CLOCKSIN and C.S.MELLISH, Programming in Prolog, Springer Verlag, Berlin, 1981.

[3] E.F. CODD, "A Database Sublanguage founded on the Relational Calculus", in Proceedings of the ACM SIGFIDET Workshop on Data Description, Access and Control, 1971.

[4] E.F. CODD, "Relational Completeness of Database Sublanguages in Database Systems", in Courant Computer Science Series, Vol 6, Prentice Hall, Englewood Cliffs, 1972.

[5] C.G. DATE, An Introduction to Database Systems (2nd edit.), Addison Wesley Publishing Co, Reading (Mass), 1977.

[6] R. GRISHMAN, "Response Generation in Question - Answering Systems", in Proceedings of the 17th ACL, La Jolla, 1979.

[7] P.K. HALVORSEN, "Semantics for Lexical Functional Grammar", in Linguistic Inquiry, Vol 14, No 4, 1983.

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

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

[10] B.G.T. LOWDEN and A.N. DE ROECK, "Generating English Paraphrases from Relational Calculus Expressions", to appear in Behaviour and Information Technology.

[11] K. McKEOWN, "Paraphrasing using Given and New Information in a Question Answering System", in Proceedings of the 17th ACL, La Jolla, 1983.

[12] E. MUECKSTEIN, "Q-Trans: Query Language Translation into English", in Proceedings of the 8th IJKAI, Karlsruhe, 1983.

[13] J.C. THOMAS and J.D. GOULD, "A Psychological Study of Query by Example", in Proceedings NCC 44, 1975.

[14] M. WALLACE and V. WEST, "QPROC: A Natural Language Database Enquiry System Inplemented in Prolog", in ICL Technical Journal, November 1983.