An English Paraphraser
for
Logical Query Languages

Abstract

For most users of database systems, formulating and understanding queries expressed in logical query languages (e.g. SQL) is a distinctly non-trivial task. It would therefore be advantageous to be able to provide a component capable of producing automatic natural language paraphrases of logical queries. Such a component would have a place in both traditional database environments in order to confirm that queries formulated by end users do perform the expected task, and in database systems which have a natural language front end in order to allow the user to disambiguate ambiguous queries. In this paper, we describe a system which attempts to address this problem.

B.G.T. Lowden
B.R. Walls
A.N. De Roeck
C.J. Fox

Department of Computer Science
The University of Essex
1. INTRODUCTION

The aim of this paper is to describe work that has been carried out at the University of Essex† with the aim of building a paraphraser capable of delivering English paraphrases of queries addressed to relational databases and expressed in logical query languages. As such, this represents a consolidation and partial extension of work carried out under a previous project (see [Lowden and De Roeck, 1986a and 1986b] for details). The ability to produce paraphrases of queries expressed in a logical query language is desirable for two main reasons. Firstly, we consider a traditional database environment in which the user is expected to formulate queries directly in some logical query language. This is actually a non trivial task, and studies have shown that many queries formulated in this way do not retrieve the information which the user expects ([Thomas and Gould, 1975]). In such a system, it would obviously be an asset to be able to produce an English paraphrase of the query so that the user could verify that the logical query does perform the required task. Secondly, we can envisage a system (for example [Lowden et al, 1991]) which converts queries expressed in English into a logical query language. By its very nature, certain English queries will in fact turn out to be ambiguous with respect to certain databases. In such cases, it would be desirable for a system that is able to generate competing logical expressions derived from English sentences to be able to paraphrase these expressions in order to allow the user to perform the necessary disambiguation. This paper will concern itself with the way in which such paraphrases can be produced.

2. BACKGROUND

Clearly, for any given logical expression there is some way of paraphrasing that expression in English. The simplest and least interesting way of producing such a paraphrase is to simply echo the structure of the logical expression. Applying this method to the logical expression

\[ x. \text{man}(x) \rightarrow y. \text{woman}(y) \land \text{loves}(x, y) \]

† Supported by SERC grant GR/E/69485
would produce a paraphrase something like

"For every individual that is a man, there exists
an individual that is a woman such that he loves her"

Applying this principle to more complicated logical expressions will produce more complicated paraphrases, but it seems reasonable to assume that paraphrases of this nature are always possible.

However, we are interested in producing English paraphrases that are helpful to users, and as such we would prefer to be able to generate paraphrases which are closer to the way in which an English speaker would phrase the query. It is quite obvious that the logical expression above would be more naturally rendered in English as

"Every man loves a woman"

The system we will be describing represents an attempt to produce paraphrases of this sort. In order to achieve this effect we will be making use of two key observations.

Firstly, we note that sentences expressed in "normal" English seem to correspond to some kind of tree structure. If we consider the sentence

"Every employee works in a department on floor 2
and earns a salary that is greater than 20000"

we can see that we can express this conceptually as a tree of the form
Every node in this tree represents some kind of entity (employees, departments etc.), while the links represent the relationships between these entities. In order to paraphrase such a conceptual structure, it is necessary firstly to determine the root of the tree (in this case "Every employee"), and then to traverse the tree in a depth first fashion. Usually, there will be several different ways of expressing the same information. For example, we can see that we could rearrange the above information into the following tree.
This tree is rooted in the node "A salary", and is different from our original example in that we have used the passive form of the employee-salary relationship (i.e. "is earned by" rather than "earns"). By traversing this structure depth first we would obtain the paraphrase

"A salary that is greater than 20000 is earned by every employee that works in a department that is on floor 2"

While this is arguably a less elegant way of expressing the same information, it nevertheless seems acceptable. It therefore seems to be the case that there exist criteria by which we are able to rank paraphrases in some way. In the example we have just seen, it seems to be the case that the first paraphrase is superior since it avoids the use of the passive.

The second observation that we will use is the fact that it is possible to form logical queries which are well formed and which are executable with respect to the given database, yet which do not make sense with respect to the underlying conceptual structure upon which the database is drawn. An example of
such a query is one which simply forms the cartesian product of two unrelated database objects. In such cases, it would not be possible to massage the query order to match the underlying conceptual structure, in which case the strategy outlined above would fail. This corresponds to the notion of coherency introduced in [Lowden and De Roeck, 1986a]. Queries are only coherent with respect to a database if the various logical components of the query can be linked together in to a single structure which respects the conceptual structure of the underlying database.

The system which we will be describing attempts to make use of these observations in the following manner. Firstly, we will make use of some specification of the conceptual structure of the database in order to try to arrange the query into a coherent conceptual form. Secondly, we will determine whether this conceptual form is a tree structure. If so, we locate the root of the tree and paraphrase this in the way described. (The root of the tree corresponds to the notion of the topic of the query as described in [Lowden and De Roeck, 1986a].) If we are unable to form a tree structure from the query, then we would envisage that some other method for paraphrasing would be required (for example simply echoing the logical structure of the query). Finally, if we are unable to link the query according to the database's conceptual structure, then the query in question is incoherent and can be rejected.

3. THE ALGORITHMS

The query language we will be attempting to paraphrase is a version of the domain relational calculus (DRC). The syntax of this language is given below.
\( \theta \ ::= \{ \nu \in \delta, \xi \in \varepsilon, \ldots ! \phi \} \)

\( \phi \ ::= \neg \phi \)

\( \Gamma [ \phi \land \psi] \)

\( \Gamma [ \phi \lor \psi] \)

\( \Gamma [ \phi \rightarrow \psi] \)

\( \Gamma \nu \in \delta \phi \)

\( \Gamma \nu \in \delta \phi \)

\( \Gamma <\alpha, \beta, \ldots> \in \tau \)

\( \Gamma \alpha = \beta \)

\( \Gamma \alpha < \beta \)

\( \Gamma \alpha > \beta \)

\( \alpha \ ::= \kappa \)

\( \Gamma u \)

\( \Gamma * \)

The term "*" is not essential, it merely provides an anonymous implicit existential quantification, e.g.

\( * \leftrightarrow \rho \in \text{LOC} \leftrightarrow x \in \delta \ y \in \varepsilon <x, y> \in \text{LOC} \)

DRC is broadly equivalent to a sorted first order logic, with the tuple membership clauses corresponding to logical predicates. The ideas that will be introduced in this paper are therefore applicable to such a logic. The query language that we are interested in paraphrasing here is different to that described in [Lowden and De Roeck, 1986a], which uses a restricted version of the tuple relational calculus (TRC). The restrictions on this language are firstly that only existential quantification is allowed, and secondly that only one tuple variable is allowed per relation. In order to paraphrase DRC, we will therefore encounter problems which did not occur in the original project, in particular these will be associated with the use of the universal quantifier.

We will be illustrating our ideas by example. Our example queries will be the DRC translations which are assigned to English sentences by the front end described in [Lowden et al, 1991], and these queries
will be addressed towards a relational database which has the following structure.

\[
\text{EMP( name, age, sal, dept )}
\]

\[
\text{LOC( dept, floor )}
\]

\[
\text{BOSS( name, mgr )}
\]

Having a full set of logical connectives introduces a certain amount of redundancy (for example $\phi \rightarrow \psi$ is equivalent to $\neg \phi \vee \psi$). We will be avoiding any problems associated with this redundancy by working from the clausal form version of the query to be paraphrased. This will also help to further ensure that the paraphrasing procedure is not dependent upon the original structure of the query. In essence, we will be working from a canonical form of the query.

For our first query, we will be paraphrasing the logical representation assigned by our front end to the English query "Who works in admin and works for Malcolm?". The DRC representation assigned to this query is

\[
\text{DRC: } \{ \text{fv} \in \text{EMP}[\text{name}] \mid <\text{fv}, \text{*}, \text{admin}> \in \text{EMP} \& <\text{fv}, \text{malcolm}> \in \text{BOSS} \}
\]

We now translate this into clausal form (the algorithm can be found in [Bundy, 1983]). This yields two clauses :

\[
<\text{fv}, \text{*}, \text{admin}> \in \text{EMP} \&
\]

\[
<\text{fv}, \text{malcolm}> \in \text{BOSS}
\]

The problem now is to map these clauses onto the underlying conceptual structure of the database. In order to define this conceptual structure we must extend the idea of an extended data model as introduced in [Lowden et al, 1991]. For each relationship that can occur between entities in the database we have a corresponding entry in the extended data model. In order to produce paraphrases of the above query, we require the following entries :
data_model( <X,*,*,Y>∈ EMP, dept_of(X,Y), active ).
data_model( <X,*,*,Y>∈ EMP, dept_has(X,Y), passive ).
data_model( <X,Y>∈ BOSS, mgr_of(Y,X), active ).
data_model( <X,Y>∈ BOSS, works_for(X,Y), passive ).

This information tells us that whenever we encounter a tuple of the form <X,*,*,Y> in the EMP relation, then this corresponds to the conceptual dept_of relationship, i.e. X works in department Y, or alternatively such a tuple also corresponds to the dept_has relationship (Y is a department which contains employee X). Note that the ordering of the arguments in the conceptual relation is important, since this indicates the directionality of the relationship. The data model entries also indicate whether the relationship is an active or a passive one (this information is used to help rank alternative paraphrases). The first stage of the paraphrasing process involves linking up the logical components with respect to these definitions. In our example, there are four possible ways in which this can be done, i.e.

1. 

```
   fv
  /\      /
 dept_of dept_of
 /    \  /    /
 admin admin
```

2. 

```
   fv
  /\      /
 dept_has dept_has
 /    \  /    /
 admin admin
```
These trees represent every possible way of linking up the query according to the conceptual structure specified for the database. Since we have been able to link all the components, we now know that our query is coherent. However, our requirement was also that the resulting structure formed a tree, and we can see that according to the directionality of the relations involved, this is not true of our second structure. We are therefore able to rule this out, leaving structures 1, 3 and 4 as competing possibilities for paraphrases. We will now demonstrate how each of these possibilities are paraphrased.

Basically, we require that our extended data model contains suitable pieces of canned English which will allow us to produce an appropriate sentence corresponding to our tree structure. These pieces of English are of two kinds. If we look at our logical fragment, we see that all variables are associated with an underlying domain. In our system, domains are defined as being the projection of some attribute from some relation in our database. We now require that each of these domains be associated with some English noun (possibly complex) which describes it. Our example query contains only one variable, `fv`, which is drawn upon a domain defined as being the projection of the name attribute on the
EMP relation. The following data model entry will suffice:

\[
\text{english\_domain}(\text{EMP[name]}, \text{"employee"}).
\]

This states simply that the domain EMP[name] is described by the noun "employee". The second kind of entry we require contains English verbs which correspond to the conceptual relationships underlying the database. In our example, we have introduced four such relationships. We can link these to their English descriptions as follows:

\[
\begin{align*}
\text{english\_relation}(\text{dept\_of}, \text{"works in"}). \\
\text{english\_relation}(\text{dept\_has}, \text{"contains"}). \\
\text{english\_relation}(\text{mgr\_of}, \text{"manages"}). \\
\text{english\_relation}(\text{works\_for}, \text{"is managed by"}).
\end{align*}
\]

We now have sufficient information to allow us to paraphrase our example. We will begin by paraphrasing structure 1, i.e.

1. \[
\begin{array}{cc}
\text{fv} & \text{dept\_of} & \text{works\_for} \\
\text{admin} & \text{malcolm}
\end{array}
\]

The root of this tree is the free variable fv. Free variables we associate with the determiner "Which". We also know that fv is drawn on the domain EMP[name], which is associated with the English description "employee". Paraphrasing therefore starts with the noun phrase "Which employee". We now proceed to paraphrase depth first. fv is linked to admin by the relationship dept\_of, which is associated with the English "works in". We therefore now have "Which employee works in". The node admin is a constant, and all constants we paraphrase as themselves. This gives us "Which employee works in admin". Returning to the top level of the tree we must now paraphrase the second branch. At the top level of the tree, multiple branches are paraphrased by conjunction. We can easily see that the second branch is paraphrased as "is managed by malcolm", while conjoining this with the first branch...
produces the overall paraphrase "Which employee works in admin and is managed by malcolm".

We will now paraphrase structure 3, which was

```
3.  
   f
   v
       dept_of
       mgr_of
         admin
         malcolm
```

This structure is rooted in the node malcolm. If we begin at this node and follow the procedure outlined above, we see that the link between f v and malcolm is paraphrased by "malcolm manages which employee". We must now deal with the link between f v and admin. Links attached to non-root nodes are paraphrascable by relative clauses, and so we proceed by attaching a suitable relative pronoun ("which" or "that"). The remainder of the structure is paraphrased as before, giving the final paraphrase "malcolm manages which employee that works in admin".

Structure 4 was rooted in the node admin, i.e.

```
4.  
   f
   v
       dept_has
       works_for
         admin
         malcolm
```

paraphrasing this tree produces the paraphrase "admin contains which employee that is managed by malcolm". We have now shown how our original query produces three competing paraphrases:
a. which employee works in admin and is managed by malcolm

b. malcolm manages which employee that works in admin

c. admin contains which employee that is managed by admin

In order to choose between these competing paraphrases, we can employ some simple heuristics. As we have seen, it seems in general to be better to avoid the passive form of a relationship (these are distinguished in the data model). It also seems to be best to heavily favour paraphrases which begin with a wh word (i.e. those whose tree structure is rooted in a free variable. Using these heuristics, we would choose the first paraphrase as the best. We will now briefly consider another simple example derived from the English sentence "Every employee earns less than a manager". The DRC translation which our front end assigns to this query (after some optimisation) is as follows:

DRC: \( x \in \text{EMP}[\text{name}] (\forall \text{v1} \in \text{EMP}[\text{sal}] (\forall \text{v2} \in \text{EMP}[\text{sal}] (\forall \text{v3} \in \text{EMP}[\text{name}]) (\langle*,\text{v3}\rangle \in \text{BOSS} \land \langle*,\text{v2},*\rangle \in \text{EMP} \land \langle\text{v3},*,\text{v1},*\rangle \in \text{EMP} \land \text{v2}<\text{v1}))) \)

This can be reduced to the following clause set:

\[
\begin{align*}
\langle*,\text{v3}(x)\rangle & \in \text{BOSS} \land \\
\langle\text{x},*,\text{v2}(x),*\rangle & \in \text{EMP} \land \\
\langle\text{v3}(x),*,\text{v1}(x),*\rangle & \in \text{EMP} \land \\
\text{v2}(x) & < \text{v1}(x)
\end{align*}
\]

In order to paraphrase this, we require extra information in our data model, i.e.

```prolog
data_model( <X,*,Y,*> \in EMP, earns(X,Y), active ).
data_model( <X,*,Y,*> \in EMP, earned_by(Y,X), passive ).
data_model( X<Y, less(X,Y), active ).
data_model( X<Y, gt_eq(Y,X), active ).
data_model( <*,X> \in BOSS, manager(X), property ).
```
english_relation( earns, "earns" ).
english_relation( earned_by, "is earned by" ).
english_relation( less, "is less than" ).
english_relation( gt_eq, "is greater than or equal to" ).
english_relation( manager, "is a manager" ).

english_domain( EMP[sal], "salary" ).

These new definitions include the unary relation manager. Because of the database structure, this does not correspond to a domain (managers are drawn from the employee domain). This information is therefore used to specify an additional property of some individual. We will see how this works when we paraphrase our example. One way of matching the query with the conceptual structure of our database is shown below:
Paraphrasing this structure proceeds as before, with the exception that we associate individuals introduced by universally quantified variables with the determiner "every", and those introduced by existentially quantified variables with "a" (or "an"). This structure therefore yields the paraphrase "Every employee earns a salary that is less than a salary that is earned by an employee who is a manager". There are, of course, other ways in which this clause set maps onto the database structure. One of these is shown below.
Paraphrasing this structure yields the somewhat unsatisfactory sentence "A salary is less than a salary that is earned by an employee who is a manager and is earned by every employee". Firstly, we note that in this sentence there is ambiguity as to where the conjunction attaches. This could be improved by formatting the sentence in some way that would make this clear. The previous project, described in [Lowden and De Roeck, 1986a], avoided this type of ambiguity by suitably indenting the paraphrases produced, and indeed this solution would be applicable here. However, it is also clear that this paraphrase is not very good for other reasons. The heuristics we have described earlier would favour the first paraphrase since it contains fewer passives, but the second paraphrase arguably has a problem with the scoping of the determiners. The extent to which this is a problem is unclear, but it might be possible to build some mechanism into the heuristic function which would take the scopings of the logical variables into account when evaluating competing paraphrases.

We have now seen how we are able to produce paraphrases of logical expressions which, when reduced to clausal form, consist entirely of conjunctions of single literals. However, many queries do not have this property. We will now show how our system is able to cope with some queries which contain disjunction. Initially we will consider the sentence "Every manager in admin employs every employee
who is older than 30'. This sentence is converted by our front end into the following DRC expression:

\[
\text{DRC: } x \in \text{EMP[name]}(y,(\text{moEMP[name]}(z \in \text{EMP[age]}(\neg\neg, x) \in \text{BOSS} \land \neg\neg, x, *, *), \text{admin}) \in \text{EMP} \land y, z, *, * \in \text{EMP} \land z > 30) \Rightarrow
\]

\[
\langle y, x \rangle \in \text{BOSS}>)
\]

In clausal form, this becomes

\[
\langle y, x \rangle \in \text{BOSS} \lor \langle y, x \rangle \in \text{BOSS} \lor \langle x, *, *, \text{admin} \rangle \in \text{EMP} \lor \langle y, z, *, * \rangle \in \text{EMP} \lor \neg z > 30
\]

Firstly, we require some extra data model definitions in order to be able to process this expression, i.e.

\[
\text{data_model}(\langle X, Y, *, * \rangle \in \text{EMP}, \text{is_aged}(X, Y), \text{active})
\]

\[
\text{data_model}(\langle X, Y, *, * \rangle \in \text{EMP}, \text{age_of}(Y, X), \text{passive})
\]

\[
\text{data_model}(X > Y, \text{gt}(X, Y), \text{active})
\]

\[
\text{data_model}(X > Y, \text{less_eq}(Y, X), \text{active})
\]

\[
\text{english_relation( is_aged, "is")}
\]

\[
\text{english_relation( age_of, "is the age of")}
\]

\[
\text{english_relation( gt, "is greater than")}
\]

\[
\text{english_relation( less_eq, "is less than or equal to")}
\]

\[
\text{english_domain( \text{EMP[age]}, "age")}
\]

In order to paraphrase this query, we will make the assumption that all of the negated literals within the clause form antecedents. The algorithm involves that we identify the consequent (the positive literal) and match this with the conceptual structure. We then match the literals from the antecedent (the negated literals) with the database structure. These should form one or more distinct tree structures. We then attach these structures to the structure associated with the consequent. This overall structure should again be a single tree. In order to paraphrase this structure, we again start at the root and proceed depth first. For nodes which have an antecedent, we must paraphrase the antecedent first. This is best illustrated with reference to our example. The consequent is the literal
\(<y,x> \in \text{BOSS}\)

This can be mapped on to the conceptual structure as either

\[ x \xrightarrow{\text{mgr_of}} y \]

or

\[ y \xrightarrow{\text{works_for}} x \]

We now process the antecedent, i.e. the remaining negated literals, and we find that they form two disjoint trees:

\[ x \xrightarrow{\text{manager}} \xrightarrow{\text{dept_of}} \xrightarrow{\text{admin}} \]
\[ y \xrightarrow{\text{is_aged}} \xrightarrow{\text{gt}} \xrightarrow{30} \]

(Actually, these are not the only ways of linking up the antecedent literals, however as we will see they are the only ways that will allow the antecedents to be joined to the consequent so that the resulting structure still forms a tree).

Now we simply attach each of the antecedent trees to the consequent. If we choose the active version of the consequent, we get the following structure:
The root of this tree is the variable x, and we start paraphrasing at this point. As we have said, if there is an antecedent, we paraphrase this first. This gives us "Every employee who is a manager and who works in admin". We now paraphrase the main link from the consequent, which yields "Every employee who is a manager and who works in admin manages every employee". Finally, we must paraphrase the antecedent associated with variable y. In order to do this, we must mention one final detail. We can see that variable z is universally quantified (i.e. it is not a Skolem function). Now, whenever we are paraphrasing an individual introduced by a variable in the antecedent associated with some other node, we must switch the kind of the determiner that we associate with that individual. In our example, z will therefore be associated with the determiner "a". Similarly, if there were Skolem functions in the antecedent, we would associate them with the determiner "every". Bearing this in mind, the final paraphrase we achieve is "Every employee who is a manager and who works in admin manages every employee who is an age that is greater than 30". If we choose the passive version of the consequent, we get the structure
This structure is rooted in variable y, and produces the paraphrase "Every employee that is an age that is greater than 30 works for an employee who is a manager and who works in admin". Both of these paraphrases are reasonable, with the first being favoured by the simple heuristics which were outlined earlier.

We have now seen the manner in which we are able to paraphrase a certain restricted class of logical queries. Queries which reduce to a clausal form which consists entirely of single conjoined literals are easiest to process, since this involves a simple mapping of the literals onto the data model descriptions of the conceptual relations. We have also seen how we are able to paraphrase certain queries which contain disjunction and negated literals. These queries, we paraphrase by making the assumption that the negated literals form antecedents. However, ideally we would like a method which allows us to paraphrase arbitrary expressions. For example, if we consider the English sentence "Every manager in every department that is on floor 2 earns 2500". This is translated by our front end into the DRC expression

\[
\text{DRC: } x \in \text{EMP[name]}(y \in \text{LOC[dept]}((<*,2>\in \text{LOC} \lor <x,*.2500,*>\in \text{EMP}) \land

(<y,2>\in \text{loc} \lor <x,*.2500,*>\in \text{EMP}) \land (<*,x>\in \text{BOSS} \land

<x,*,*,y>\in \text{EMP} \Rightarrow <x,*.2500,*>\in \text{EMP}))
\]
In clausal form, this is

\[(<*\text{.}2>\in \text{LOC} \lor <x,\text{.}2500,\text{.}>\in \text{EMP}) \land \]
\[(<y(x),2>\in \text{LOC} \lor <x,\text{.}2500,\text{.}>\in \text{EMP}) \land \]
\[(<x,\text{.}2500,\text{.}>\in \text{EMP} \lor \neg <x,\text{.}>\in \text{BOSS} \lor \neg <x,\text{.}>\in \text{EMP}) \]

It is less clear how this set of clauses can be mapped onto the conceptual structure associated with the database. This clause set contains both existentially and universally quantified variables, both conjunction and disjunction, as well as both positive and negative literals. Ideally, we would like to be able to map these clauses onto our conceptual structure, retaining all the information about conjunction, disjunction, negation etc. on the structure being built up. Having built up some coherent overall structure, we could then use the equivalence between \(\neg \phi \lor \pi \lor \phi \rightarrow \psi\) in order to locate implication whenever possible (this is in effect what is happening in our earlier examples involving antecedents). However, so far no systematic way has been found for doing this. Our problem is that whereas it is easy to determine the meaning of simple expressions involving only the conjunction of single literals, it is not at all easy to establish the meaning of some diagrammatic representation involving disjunction, negation etc. However, it remains an open research question as to whether such a method may be found.

4. CONCLUSIONS

We have consolidated work done on an earlier project by demonstrating that in order to produce reasonable paraphrases of logical queries the idea of coherence is paramount. Queries are coherent if we are able to map them onto some structure which reflects the underlying conceptual structure of the domain. We have also shown how coherent queries may be paraphrased into "normal" English if the resulting structure forms a tree. In such cases, paraphrases can be produced by a depth first traversal. We have shown how, for a certain class of query, i.e. those involving only the conjunction of single literals, this process is relatively straightforward. We have also demonstrated that more complicated queries can be paraphrased, but a completely general algorithm has not yet been found.
REFERENCES


De Roeck, A.N., A Natural Language Front End To
Jowsey, H.E., Relational Systems Based On Formal Semantics;
Lowden, B.G.T., International Conference on Information Technology,
Walls, B.R. (1990)

De Roeck, A.N., A Natural Language System Based on Formal Semantics;
Fox, C.J., The International Conference on Current Issues
Lowden, B.G.T., in Computational Linguistics,

Lowden, B.G.T., REMIT : A Natural Language Paraphrase for Relational
De Roeck, A.N. Query Expressions;

Lowden, B.G.T., The REMIT System for Paraphrasing Relational Query
De Roeck, A.N. Expressions into Natural Language;
(1986b) 12th International Conference on Very Large Databases,
Kyoto, Japan, 1986
Lowden, B.G.T., A Formal Approach to Translating English
Walls, B.R., into SQL;
De Roeck, A.N., The Ninth British National Conference on Databases

Thomas, J.C., A Psychological Study of Query By Example;
Gould, J.D. (1975) Proceedings NCC 44