Information Technology: Research and Development (1982), *1* (23-39)

A NATURAL LANGUAGE ANALYSER FOR DATABASE ACCESS

B. K. BOGURAEV AND K. SPARCK JONES

Computer Laboratory, University of Cambridge, Corn Exchange Street, Cambridge CB2 3QG, UK

(Received 22 April 1981, revised 16 July 1981)

ABSTRACT

Access to databases is conventionally by queries couched in a formal query language. However in many environments it would be more convenient if questions in natural language were allowed. Hitherto natural language access systems have relied on grammars making heavy use of database-specific information. Such systems are not readily transportable from one database to another. The paper describes a project aimed at improving transportability through the use of a powerful general natural language analyser, to be combined with a relatively restricted database-specific translator. The design of the analyser and its current state as developed and tested are detailed, its performance is illustrated, and the basis for the building of the translator is outlined.

In recent years increasing effort has been devoted to the design of intelligent retrieval systems. The proliferation of ever-larger databases, with more extended user communities, available via interactive and distributed computer systems has, in particular, stimulated research on systems for natural language access to databases. This paper describes the approach being adopted to the design of a natural language processing front end for database access aimed at maximising data independence in the analyser and grammar used, especially in semantic processing.

1. RATIONALE

Among the various types of information retrieval systems, conventional database systems are the most straightforward. Though there are many challenging problems associated with database management, as far as retrieval is concerned the combination of definite questions and direct access typical of current database systems implies a relatively simple matching operation for retrieval. In data retrieval systems questions are definite because they are accurate characterisations of the sought information; and access is direct because the items retrieved explicitly display

the sought information. Accuracy in question formulation, and explicitness in data description, are promoted by the use of formal query and data languages. Indefinite questions and indirect access, on the other hand, require more intelligent systems allowing for non-trivial, that is evaluative, matching of question against information store, i.e., systems able to put interpretive effort into manipulating questions, and inferential effort into manipulating the contents of the store.

Unfortunately, while formal query languages have retrieval advantages, they are not necessarily attractive to the system user, and especially to the so-called casual user, just because they force him to express his need in a highly controlled and hence artificial way. The obvious vehicle for initial question presentation is the user's natural language, and in the last few years in particular a variety of approaches to the design of natural language question processors have been put forward and implemented, in some cases for independent and even commercial user communities. In these cases, even though the inferential manipulation of the contents of the information store which is characteristic of much artificial intelligence activity is not undertaken, the requirements to be met by the input question interpreter mean that the system has to become much more sophisticated, i.e., intelligent, than the conventional database system. Except where the data in the store itself are extremely simple, this is true even though the questions are essentially definite in the sense introduced earlier, and the data store itself is precoded. The richness of natural language, and its consequent potential for ambiguity, make the transformation of input natural language questions into formal query language search specifications extremely complicated. The difficulties of the task are increased by the need to provide a hospitable system, i.e., one allowing incomplete inputs and extended dialogue, as well as providing various feedback facilities, for example for checking input question interpretations. But the difficulties remain even where the single question–answer interaction, for well-formed questions, is all that is assumed.

In general, the strategy adopted in order to control the interpretive process has been to work with a database-specific grammar, i.e., to assume that the concepts and concept structures of the input question texts are restricted to ones appropriate to the universe of discourse of the database (Hendrix *et al.*, 1978; Waltz, 1978). The detail has varied, but has essentially consisted of working with semantic categories and semantic patterns, or message forms, referring to the objects and relationships of the database, for example in a geological application to rocks and to samples containing minerals, or in a defence application to types of ship and their proximity to fuel supplies. In early systems, such as LUNAR (Woods *et al.*, 1972; Woods, 1977), such semantic categories and patterns were used to check and select structures derived by conventional syntactic analysis; more recently they have to a considerable extent replaced conventional syntactic categories and patterns. The approach is made more efficient if, as is natural, it is tied to a restricted vocabulary, i.e., a dictionary confined to those words, or more specifically, word senses, relevant to the database. Further, though a small or, more importantly, closed vocabulary cannot, as Harris (1977) has pointed out, be assumed for database systems, database characterisations of the vocabulary may still be produced through reference to the database itself.

The essential disadvantage of exploiting database-specific semantics is that it makes the system front end less generally applicable. It can be assumed that the front end will normally interface with a standard query language and database management system, e.g., a system using the relational model. A standard type of

database management system like this might well be expected to be applicable to different databases, especially in a highly portable implementation. It is therefore unsatisfactory if the front end natural language processor is not equally widely applicable. Unfortunately there may be a tradeoff between increased effectiveness in the question interpreter achieved by greater database specificity, and reduced generality. From this point of view, relying on database-specific lexical coding using database-oriented semantic categories, and on database-oriented semantic patterns minimising the use of conventional syntax, maximises the effort of dealing with a new database, quite apart from that involved in providing dictionary entries for new words and word senses. It has moreover to be recognised that the database-oriented approach may prove unduly restrictive even for a single database, if this is developing over a period of time, or being exploited by new kinds of user or for new types of use.

Given that a database-specific interpreter is efficient, one possible approach to portability is to provide an extensive apparatus for characterising the database-specific elements of the front end, i.e., for constructing (and hence for using) the database-specific word and text characterisations (Bronnenberg *et al.*, 1979; Konolige, 1979.) The alternative approach to reducing the rewriting effort is to increase the generality of the front end, and to compensate for the loss of focus resulting from the lack of database orientation by increasing the richness and power of the natural language processing elements of the interpreter.

Unfortunately, the price to be paid for a more general interpreter is that the transformation of the input question into the formal search specification becomes less direct. It follows that the organisation of the interpreter will have to be more complex than has been assumed so far: specifically the front end will cover two processing steps, natural language analysis proper deriving a meaning representation from the input question, followed by database-oriented processing to derive the query search specification from the meaning representation. Quite extensive rewriting of the input question representation may be required even when a database-oriented natural language analyser is used, since the organisation of the question may be very different from that of the searching query, but the rewriting is facilitated by the database-oriented elements and structures applied in analysis and featuring in the initial output of the analyser. If a general analyser is used, explicit connections have to be made between the semantic apparatus which is used for analysis and to provide the question meaning representation, and the semantic apparatus associated with the database. We may therefore suppose that the meaning representation provided by the analyser will be input to a distinct translation component which at least requires rules linking detailed meaning representation elements and structures with those of the database.

The separation of the analyser and translator may nevertheless be less costly than it appears to be. If the meaning representation derived by the analyser is a regularised, or 'normalised', expression of the content of the input text, the formulation and application of the translation rules may not be too complicated. Specifically, applying database-specific information in translation from normalised meaning representations may be less exigent than applying it in analysis, where un-normalised input texts have to be handled.

The approach being adopted in the present project for natural language access to databases is therefore that of working with a front end consisting of a natural language analyser exploiting a general syntax and semantics to derive a meaning representation for the input question, and a translator exploiting the database

syntax and semantics to derive a search specification from the meaning representation. The basic assumption is that the analyser semantics in particular are powerful enough to interpret questions without recourse to database-specific information, and to provide a rich enough meaning representation to drive the translator. The general structure of the proposed system is compared with that of a typical current system in Figure 1.

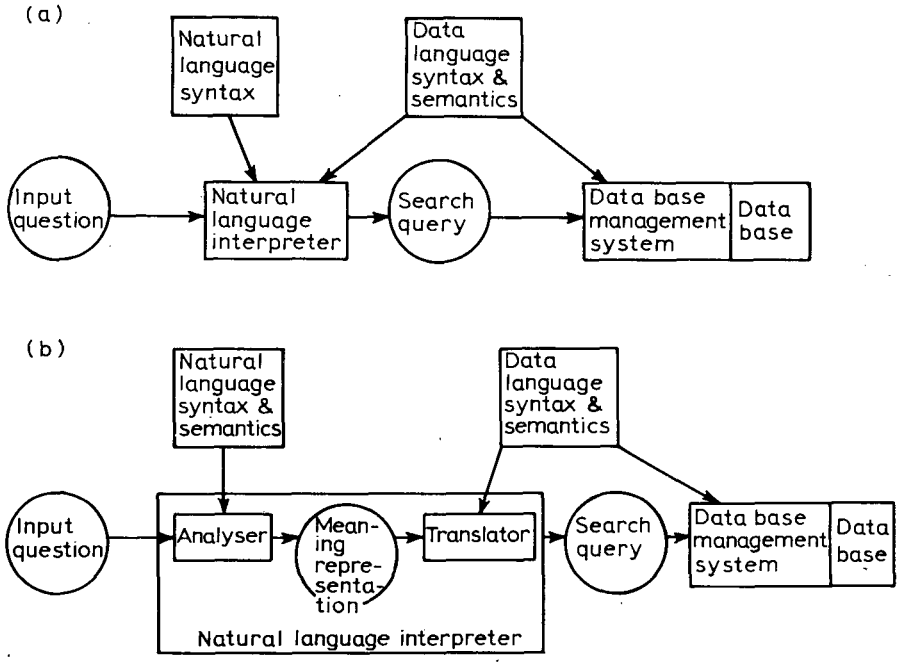

FIG. 1. Scheme for natural language analyser. (a) Conventional system; (b) proposed system

The remainder of this paper will be devoted to the proposed system analyser: the development of the translation component of the interpreter has only just begun, and no more than indicative remarks will be made about it, following the description of the analyser.

2. OUTLINE OF THE NATURAL LANGUAGE ANALYSER

The analyser is built on the ideas of Wilks (1975a, b) and Woods (Woods *et al.*, 1972; Woods 1977). As described in Boguraev (1979), it was explicitly designed to tackle lexical and structural disambiguation, i.e., to determine the meaning in context of words having a range of possible lexical functions and senses, and the meaning of word strings having a range of possible syntactic structures and readings. The analyser was tested by Boguraev for a non-trivial vocabulary and syntactic grammar, and evaluated by the generation of paraphrase sentences from the derived meaning representations.

The analyser is based on the use of semantic primitive categories and patterns, i.e., on the principle that the meaning of text elements and structures is determined by reference to general semantic categories which are shared by individual words, and to the ways in which these categories can be combined to construct message forms. In other words, an important role in text understanding is played by the assimilation of sentences to underlying generalised messages, or, to put it the other way round, familiar message prototypes can be applied to given text sentences to force the selection of specific meanings for them, and to supply characterisations of their essential character.

This basic idea has of course been widely exploited, both for language analysis in general and as indicated, via specialised categories and patterns, for access to databases. The points of interest here are the effectiveness of the given semantic apparatus, i.e., repertoire of primitives and patterns, both for analysis and representation, and the way in which it is applied in analysis, particularly in relation to conventional syntax.

The semantic apparatus is derived from Wilks, but has been considerably extended. It depends on a large number of primitives, and on several types of pattern ranging from the text-general to the word-specific. This apparatus is exploited in conjunction with conventional syntax within the framework of an augmented transition network (ATN) parser. The approach to syntax, and to the control of the parsing process as a whole, by the use of the ATN formalism, is derived from Woods. The syntactic grammar is designed to identify syntactic constituents, such as noun phrases, prepositional phrases, etc. These constituents are then semantically evaluated and combined to obtain the lexically and structurally selected meaning representation for the sentence as a whole.

The parser thus uses conventional syntax in the way that Wilks' system did not; but it does not seek to build a complete syntactic parse tree in LUNAR style, for subsequent semantic evaluation. However, while syntax and semantics are combined in processing, they are not exploited together for every input lexical item, as in Riesbeck's original MARGIE analyser (Riesbeck, 1975) or Small's Word Expert Parser (Small, 1980). The syntax is used to identify the basic building blocks of the sentence structure, which are then fitted together, and hence labelled, according to the way they satisfy the element and relationship preferences of the various semantic patterns. The semantic matching is therefore done at those points in the syntactic analysis which naturally supply sets of candidate pattern slot fillers, i.e., at noun phrase and, more importantly, clause levels. Since, realistically, the semantic pattern matching is not based on the idea of meeting absolute requirements, but on that of satisfying as many preferences as possible, the semantic interpretation of a sentence is the combined product of many different competing, and perhaps only partially successful, pattern matching operations. The parser output is a case-labelled dependency structure, i.e., a structure with semantic relation primitive labels and slots filled by the characterisations, using semantic category primitives, of selected word senses. The dependency structure itself, and the particular case labels involved, are derived from the various patterns supporting the preferred interpretation.

3. ANALYSER DETAILS

The syntactic apparatus is quite straightforward. It is largely derived from the LUNAR system and is quite comprehensive in scope. It has recently been extended

to deal with question forms and to allow for compound nominals, both types of structure of particular importance to the database application. The non-determinism inherent in natural language syntax analysis is not eliminated, but is much reduced, by the fact that no attempt is made to integrate all the syntactic constituents identified into a single syntactic parse tree. The constituents in turn are handed over as the contents of a set of registers to the semantic specialist routines. Inappropriate constituent analyses are blocked by semantic checks at the relevant assembly level.

The semantic apparatus, though it is based on Wilks', has been systematically and significantly developed, partly to permit cooperation with syntax, and partly to provide for a richer output structure characterisation. There are about 100 semantic category primitives and currently about 30 semantic structure primitives representing case relations (not all of which have been fully implemented). The primitives are used to characterise the senses of words, and a variety of discourse patterns. The category primitives, virtually directly taken over from Wilks (1977), are fundamental to both analysis and representation; the relation primitives appear explicitly only in sentence meaning representations, extracted from different syntactic and semantic structures identified by the analyser. Some examples of each type are given in Figure 2a and c. (Note that category primitives can be prefixed by a special primitive NOT.) For convenience in expressing discourse patterns, the category primitives have been grouped in classes, as illustrated in Figure 2b. As mentioned, the case relation primitives have not all been implemented: the analyser

a) examples of category primitives

```
ACT
ASK
BEAST
COUNT
GOOD
HAVE
LIFE
MUCH
PART
STUFF
WHERE
```

b) examples of category primitive classes

```
*PHYSOB = < THING, MAN, BEAST, SPREAD, PLANT, PART >

*ANI = < MAN, FOLK, BEAST, SIGN >

*MAR = < SIGN, ACT, STATE >
```

c) examples of relation primitives

```
attribute
destination
goal
instrument
location
mental-object
source
time-location
```

FIG. 2

was tested by Boguraev with a limited and slightly ad hoc set, and the current set is the product of a recent extensive review of the literature and of prepositional behaviour. In Wilks' system a number of category primitives also functioned as explicit case labels, but this approach proved difficult to sustain, and the separate case labels for the meaning representation dependency structure were therefore introduced.

The category primitives are used to build formulae defining word senses, i.e., to characterise word senses to the extent required for effective sense discrimination. This level is assumed to be also adequate for operations on the analyser's output meaning representation, though this assumption has not been tested beyond the point represented by Boguraev's generation experiments: the translation of meaning representations into search specifications for database access will constitute another test of their wider adequacy. The word sense formulae are complex structures with a well-defined syntax, laid down by Wilks (1977). Essentially they are binary trees with dependent-governor pairs at each node, the top-level (rightmost) governor being marked as the head of the formula. The rules for formula building are associated with groups of primitives and types of subformula: according to their groups, the primitives have specified governor and dependent potentialities for participating in different subformula types (full details are given in Wilks). Illustrative formulae for different senses of the noun "crook" (meaning criminal and shepherd's staff respectively) are given in Figure 3a.

As noted, the semantic patterns used are of two kinds, ones general to discourse, and ones tied to individual word senses. In principle this distinction is perhaps not absolute, but rather one of degree; it is in any case a practically convenient one.

The general-discourse text patterns are chiefly represented by a substantial set of templates, i.e., basic message or propositional forms. These patterns are designed to link constituents, via their primitive characterisations, and specifically their head primitives. The general form of a template is that of actor–action–object, i.e., syntactically they link the heads of verb groups with the heads of their subject and object arguments. The expression of patterns is made more convenient by the classes of primitives: thus the template for each action primitive in the illustrative set of Figure 3b represents a group of more specific alternative templates with individual category primitives substituted for the class names. The relations between the elements of the templates may lead to certain relation primitives in the meaning representation, such as *agent* labelling for the syntactic subject. The templates used in the system are essentially taken over from Wilks; but as their discriminating power is rather weak, they play a less important role than in Wilks' system. In addition, some other general text patterns have been introduced. Establishing the relation between adjective and noun *de facto* exploits what may be called a single qualifier structure 'qualplate' *ENT BE KIND; and there are also about 10 'complates', more elaborate patterns designed to deal with complement structures. An example is

MAN *DO *INAN "to" *ACT-embedded clause
supplying the output relation primitive *goal*, which would match onto
"Did Clark supply new parts to replace the faulty ones?"

Word based patterns are primarily represented by formulae associated with two types of word, verbs and prepositions. The patterns associated with verbs, called contextual verb frames, indicate the characteristics of the clause environment for the

a) formulae for senses of the noun "crook"

```
"crook"
(cat NOUN)
(ndef
 (crook1
  ((((NOTGOOD ACT) OBJE) DO) (SUBJ MAN)))
 (crook2
  ((((((THIS BEAST) OBJE) FORCE) (SUBJ MAN)) POSS) (LINE THING)))
)
```

b) examples of templates

```
*ANI FEEL *MAR        subsuming MAN  FEEL SIGN
                                FOLK FEEL ACT
                                ...
*ENT HAVE *ENT        ...

*POT NOTPLEASE *ANI subsuming THING NOTPLEASE FOLK
                                GRAIN NOTPLEASE FOLK
                                ...
```

c) verb frames for senses of "beat"

```
"beat"
(cat VERB)
(vdef
 (beat1
  ((*HUM SUBJ)
   ((*ANI OBJE)
    ((*INST INST)
     ((((NOTPLEASE BE) CAUSE) GOAL) STRIK)))))
 (beat2
  ((*ANI SUBJ)
   ((*ANI OBJE)
    ((((ACT OBJE) (GOOD WAY) DO)) GOAL) DO)))
  (preps
   (("in" "at") *MAR  activity )))
)
```

d) preplates for "with"

```
"with"
     with1 (*ENT attribute *ENT)
     with2 (STRIK manner *MAR)
     with3 (STRIK instrument *INST)
     with4 (*DO manner *MAR)
     with5 (*DO accompaniment *HUM)
     with6 (MOVE instrument THING)
     with7 (NOTHAVE manner *MAR)
     with8 (CHANGE instrument *INST)
     with9 (CAUSE instrument *INST)
     with10 ((SEE SENSE) instrument (SEE THING))
```

FIG. 3

particular verb sense. The verb frame reflects the importance of the verb both in analysis, as the key to the interpretation and connection of the other constituents, and in sentence representation as the focus of the output dependency structure. The verb frame specifies the semantic characteristics of the verb's case role fillers as these may be supplied by syntactic subject and object noun phrases (as in Wilks' system),

by prepositional phrases associated with compulsory and optional prepositions (so-called prepspecs), and by other syntactic constituents like complements, predicative adjectives, etc. The prepspecs have relation primitives for the output meaning representation associated with them. This frame specification supplements the basic word sense formula, so the full dictionary entry for a verb is as illustrated for "beat" (meaning hit and defeat respectively) in Figure 3c. The patterns for prepositions not dependent on verbs, called preplates, indicate the semantic character of the prepositionally linked items, usually the primitive class membership required of the heads of these constituents. Preplates, like the prepspecs, also indicate the case relation primitives to be used to label the output for the meaning representation linking prepositionally connected items. The system does not require any direct formula for the preposition itself; but recent study of an extensive preposition sample suggests that an attempt will have to be made to characterise prepositions other than by the relation primitives, and more fully than Wilks did. The preplates are derived from Wilks' paraplate structures, but are less constraining and allow a more flexible approach to the treatment of prepositional structures. Some representative preplates, for "with", are given in Figure 3d. The emphasis on the output case structure further led to the introduction of prepspecs, like the verb ones, for predicative adjectives.

These word based patterns are clearly more specific than the template type, though they could be common to the senses of different words.

Taken together, the types of pattern provide extensive resources for disambiguation, given that on the one hand they are associated with the quite detailed formula characterisations of individual word senses, and on the other are able to exploit the information about sentence units supplied by the syntax analysis. They account for both the system's capacity to deal with the more complex and extensive sentential structures than Wilks' original implementation, particularly by relating semantic processing to syntactic analysis, and its ability to provide a more tightly structured output meaning representation. The word formulae are the essential basis for both meaning determination and representation: they are exploited by the patterns in the former, and fill the case-relation labelled slots in the latter. The primitives, and especially the semantic category primitives, underpin the whole.

4. ANALYSER PROCESSING

As indicated earlier, this semantic apparatus is applied at potentially productive points in the syntactic traverse of the input sentence, i.e., at those points where enough information has been accumulated for it to be worth attempting ambiguity resolution. The most important of these points occurs when the parser has recognised a clause, either at top or any embedded level, for example when a relative clause has been recognised. Semantic processing is also done when a noun phrase is recognised, but the processing here is simpler since it is confined to semantic evaluation either of relatively straightforward strings like adjective–noun combinations, or of strings which have already been partly evaluated, like nouns with associated relatives: the relative clause will already have been subjected to semantic processing. Semantic processing in the first case exploits the qualplate, in the second essentially reconfigures the clause.

The more interesting and comprehensive clause processing involves matching the

various relevant types of pattern against the building blocks represented by the syntactic constituents identified by the ATN syntactic analysis, to see how far the constituent primitive characterisations match the pattern preferences, and hence select word senses and sentence structures for which appropriate dependency trees and case labels can be triggered. The processing cycle, briefly outlined below and summarised in Figure 4, is basically divided into two phases, the first aimed primarily at manipulating preferences to obtain the highest ones, the second at building the clause meaning representation suggested by the preferred interpretations for the clause constituents and their relationships.

step	activity	pattern
phase	1 - establish preferences	
1	check compulsory prepositions	verb frame prepspecs
2	check optional prepositions	verb frame prepspecs
3	check syntactic cues	verb frames, complates
4	check SUBJ OBJE	verb frames
5	match templates	templates
phase	2 - build and label representation	
6	identify Object as object or recipient	
7	identify Predicate Adjectives as state	
8	identify Modifiers as cases e.g. location	preplates
9	identify Subject as agent	
10	identify Complements as cases e.g. goal	complates

Processing cycle using syntactic registers Subject, Object, Verb, Modifiers, etc.

FIG. 4

The first phase of the cycle, obtaining preferences for alternative lexical and structural interpretations, is centred on the verb and applies conditions in order from most stringent to least stringent. Thus attempts are first made to satisfy any obligatory preposition requirements for the verb, then optional ones, then to match syntactic cues referring to, for example, complements, subsequently to satisfy subject and object requirements for the verb, and finally to apply templates. It must be emphasised that there is never any attempt to unequivocally accept or reject interpretations, only to rank them. This allows for revisions of earlier interpretations given more information, and for partially satisfied requirements, i.e., for the novelty of word use and primacy of context which are characteristic of ordinary discourse; and it leads to the output of a meaning representation embodying the most plausible sentence interpretation.

The second phase of the cycle builds the sentence representation, subject to assembly checks, and labels the case relationships involved. The order of processing here is from the most local or central environment of the verb to the more global or peripheral elements of the clause. The syntactic object is initially identified as *object* or *recipient* role filler, any predicative adjective as *state*, and other prepositional modifiers as other case role fillers, according to their preplate matches; syntactic subject is normally labelled as *agent*, and finally any complements are attached with further case roles. The cycle steps are listed in Figure 4, which shows the types of semantic pattern used along with formulae and other syntactic information at each stage. The assembly process also produces the dependency tree; so where complex sentence structures are input, the final tree represents the attachment of lower-level,

already-analysed clauses to their higher-level governors. Essentially the tree structure represents the hierarchical organisation of the analysis process.

The characteristic cumulative effect of the semantic processing can be illustrated by an example. In the account below the operations of the analyser are necessarily summarised, and irrelevant details (e.g., about syntactic procedures) are omitted, but the essential nature of the processing should be clear. Suppose we have the input sentence

"The crook beat the woman with violence"

(compare, e.g., "The crook beat the woman at chess," "The crook beat the woman with the cloak," "The shepherd caught the ewe with his crook"). According to the dictionary entries of Figures 3a and c we have two senses of "crook" and two senses of "beat" to sort out, assuming, for simplicity, only one sense each of "woman" and "violence". We also have alternative interpretations of the preposition "with", as shown in Figure 3d. The essential task of the clause semantic processing for the sentence is to select the correct (i.e., most likely) senses of "crook" and "beat", and to attach the prepositional phrase correctly, in the process giving it its case interpretation.

The processing starts with the constituent analyses for "the crook", "beat", "the woman" and "with violence" in the syntactic Subject, Verb, Object, and Modifiers registers respectively. As these constituents are simple, we can assume that the register contents consist essentially of pointers to all of the dictionary formulae for their head words. The contents of the registers are summarised in Figure 5a. In the first phase of the processing cycle, step 1, which considers compulsory prepositions for the verb, does nothing. Step 2, considering optional prepositions, deprefers 'beat2' as there is no preposition present, i.e., indirectly prefers 'beat1'. Step 3, which checks syntactic cues, for this sentence does nothing. Step 4, which looks to see how the primitive preferences associated with the SUBJ and OBJE subformulae in the verb frame are satisfied by the heads of the constituents in the Subject and Object registers, has no effect on the preferences for "beat" itself, while the OBJE preferences are both satisfied by "the woman" which is *ANI; but for SUBJ the step prefers 'crook1', as MAN matches both *HUM and *ANI, neither of which are matched by THING for 'crook2'. The two relevant templates, for the verb head primitives STRIK and DO respectively, both match the head primitives for "the crook" and "the woman" respectively, so there are no preference changes. Overall, therefore, the effect of the first phase of processing has been to assign higher preferences to 'crook1' and 'beat1'.

In the second phase, step 6 identifies the syntactic object as *recipient*, since it is characterised by *HUM. Step 7, predicative adjective, is irrelevant. Step 8, applying the preplates for "with", has the alternative possible attachment structures corresponding to "beat with violence" and "woman with violence" to consider; the step prefers 'with2' and 'with4', which match the two senses of "beat" respectively and link the prepositional phrase with the verb, to the other interpretations of "with": 'with2' and 'with4' both match "beat" with STRIK and *DO respectively, and "violence" with *MAR. 'With2' and 'with4' are thus preferred to 'with1' which, trying to attach the prepositional phrase to the object, succeeds in matching *ENT with "woman", but fails to match "violence" as this is ACT and not *ENT as required. 'With2' and 'with4' are also preferred to 'with3', even though this links the prepositional phrase with the verb by matching "beat" with STRIK, because

```
Sentence:  "The crook beat the woman with violence."

a) register contents

Subject
     crook1 (... MAN)
     crook2 (... THING)
Verb
     beat1 (... STRIK)
     beat2 (... DO)
Object
     woman1 (... MAN)
Modifiers
     with (preplates)
     violence1 (... ACT)

b) processing cycle

phase and step      activity                          result

1 1                 compulsory prepositions           -
  2                 optional prepositions             deprefer beat2
  3                 syntactic cues                    -
  4                 SUBJ & OBJE
                     beat1 (*HUM SUBJ) (*ANI OBJE)
                     beat2 (*ANI SUBJ) (*ANI OBJE)    prefer crook1
  5                 templates
                     beat1 *POT STRIK *PHYSOB
                     beat2 *POT DO    *ENT            -

  highest preferences by now are for crook1 and beat1

2 6                 Object is *HUM                    label as recipient
  7                 Predicative Adjectives            -
  8                 Modifiers                         preplates
                     with1 (*ENT attribute *ENT)
                     with2 (STRIK manner *MAR)
                     with3 (STRIK instrument *INST)
                     with4 (*DO manner *MAR)
                     with5 (*DO accompaniment *HUM)   prefer with2,with4
  9                 Subject                           label as agent
 10                 Complements                       -

final highest preferences are for crook1, beat1 and with2

c) output representation

   (clause
     (type dcl)
     (tns past)
     (v
       (beat1 (... STRIK)
         ( agent (crook1 (... MAN)))
         ( recipient (woman1 (... MAN)))
         ( manner (violence1 (... ACT)))  )  )  )
```

FIG. 5

"violence" is not *INST; and they are preferred to 'with5', which matches *DO for "beat" but fails with "violence", which is not *HUM. The remaining senses of "with" do not match at all. Step 9 labels the syntactic subject as *agent*, while step 10, treating complements, has no effect. The final highest preferences are therefore for 'crook1', meaning criminal, 'beat1', meaning physically strike, and, because of

'beat1', 'with2', linking "violence" with "beat" and labelling the relationship as one of *manner*. The complete sequence of steps is shown in Figure 5b, together with the pattern information and the step results, and the output meaning representation in Figure 5c.

5. DATABASE APPLICATION

This example shows how the analyser works in a lexically and structurally ambiguous environment of the kind for which it was designed. For the database application a number of system developments are necessary. These can be viewed primarily as required by the need to meet the task specification, but also as following from the need to tackle problems appearing in a particularly virulent form in this context. However, they can also be seen as representing the type of development which would have to be undertaken if the analyser was to be connected with those more extensive and powerful procedures for manipulating knowledge and making inferences which are associated with the idea of language understanding in the strong sense in which this expression is frequently used. Thus while the database task may be treated as an end in itself, it can also be treated as a not uninteresting or overly eccentric type of many processes which use knowledge in a complex or sophisticated way. It is clear that many of the details of the system front end must be determined by the individual and somewhat artificial characteristics of the precoded database; but the hypothesis is that a good deal of more general utility for language understanding can be learnt from the attempt to link natural language with some form of concept representation, even if this is only of the comparatively simple kind permitted by the average database. A good example of the type of development which is desirable from both points of view is that required to handle quantifier structures adequately.

As mentioned earlier, the translation interface between the analyser proper and the database management system has not yet been designed. All that can be done here, therefore, is to indicate the constraints placed upon the overall front end organisation and content by the combination of task requirement and philosophy underlying the front end design; and to suggest some of the lines to be followed.

An initial question is the extent to which the meaning representation output by the analyser should be normalised, i.e., what different surface texts should be mapped onto the same meaning representation. The analyser meaning representations are more normalised than Wilks' original ones, but do not aim at the detachment from the surface text typical of Conceptual Dependency, for example (Riesbeck, 1975). It is, however, obvious that many quite different input question texts will have to map onto the same formal search specification, for instance those illustrated in Figure 6. Normalisation is a matter both of the meaning representation structure and of the characterisation of its slot fillers. As these examples show, very considerable transformation of the input sentence may be required to achieve the canonical form of the search specification. Quite radical rewriting of the input expression was typical of the LUNAR question interpreter, for instance.

It is not, however, obvious that the analyser should be required to produce highly regularised meaning representations as its direct output. The essential function of the translation component of the front end is to produce the canonical form queries required by the database management system being used. In general the task of the translator will be made easier if its input meaning representation is highly

```
a) input questions

   What city is Smith in?
   What is Smith's city?
   Which city is Smith in?
   Which is Smith's city?
   What city does Smith have?
   In what city is Smith?
   What is the city for Smith?
   Where is Smith located?
   What is Smith's town?
   Where is Smith?


b) search query in DSL ALPHA

   GET W(S.CITY): S.SUPPLIER = 'SMITH'
```

FIG. 6

normalised, and also if the type of normalisation is oriented towards the ultimate query form; naturally, the degree and type of normalisation which is appropriate will be influenced by the particular 'lexical' and structural schematisation of the data language expressions determined by the content of the database. However, if the analyser proper is to be applicable to different or changing databases, and hospitable to different user views of an individual database, this suggests that it should initially produce relatively 'unoriented' meaning representations, but that it should have the capacity to further process initial meaning representations to obtain refined representations more appropriate in structure and/or degree of lexical standardisation to their ultimate application. It is perhaps dangerous to make the further normalisation too database-specific: a safer strategy would be reformulation based on the general principle that the whole interpreter is concerned with processing questions, i.e., with texts searching for unknowns. Thus the analyser currently reformulates question structures to simplify them and emphasise the queried unknown: for example, the reformulation of the initial meaning representation derived by the analyser for

"What is the colour of the parts which are supplied to London by S1?"

is shown in Figure 7. In this case a *recipient* is initially generated because the (query (DUMMY)) derived as syntactic object from "what" vacuously matches the *HUM requirement of step 6 in the processing cycle; but it can properly be removed as redundant because there is a more specific query pointer identifying the unknown item in the structure, namely the *state* value.

Allowing further normalisation in this fashion, if it does not represent database encroachment on the analyser, does imply complication of the front end interpreter as a whole. Complication also follows from a much more intractable problem in the context of the attempt to separate the language-general from the database-specific, namely that represented by compound nouns. Non-lexicalised compound nouns are a pervasive feature of ordinary language use in English for example, and are exceptionally conspicuous in the database context, where the front end analyser may encounter such phrases as "motorway gas station building regulations". Unfortunately, as the general literature on the subject makes clear, the interpretation of such compounds is chiefly determined by pragmatic considerations, only weakly

```
Sentence: "What is the colour of the parts which are supplied to London by S1?"

a) before post-editing

(clause
   (type question)
   (tns present)
   (v
      (be1
         ((*ANY SUBJ) ((*ANY OBJE) BE))
         (agent
            ((trace (clause v agent))
               (clause
                  (v
                     (be2
                        ((*ENT SUBJ) (((OWN STATE) OBJE) BE))
                        (agent
                           ((trace (clause v object))
                              (clause
                                 (type relative)
                                 (tns present)
                                 (aspect (passive))
                                 (v
                                    (supply1
                                       ((*ORG SUBJ)
                                          ((*INAN OBJE)
                                             ((*ORG RECIPIENT)
                                                (((((->RECIPIENT SUBJ) HAVE) CAUSE)
                                                   GOAL)
                                                  GIVE))))
                                       (agent (S1 (MAN)))
                                       (object
                                          (part1
                                             ((*INAN POSS)
                                                ((WORK GOAL) (SUBJ THING)))
                                             (number many)))
                                       (location
                                          (London (THIS (WHERE POINT)))) ))) ))
                        (state colour (val (*var* *dummy-val)))) ))
         (recipient (query (DUMMY))) ))

b) after  post-editing

(clause
   (type question)
   (tns present)
   (v
      (be2
         ((*ENT SUBJ) (((OWN STATE) OBJE) BE))
         (agent
            ((trace (clause v object))
               (clause
                  (type relative)
                  (tns present)
                  (aspect (passive))
                  (v
                     (supply1
                        ((*ORG SUBJ)
                           ((*INAN OBJE)
                              ((*ORG RECIPIENT)
                                 (((((->RECIPIENT SUBJ) HAVE) CAUSE) GOAL) GIVE))))
                        (agent (S1 (MAN)))
                        (object
                           (part1
                              ((*INAN POSS) ((WORK GOAL) (SUBJ THING)))
                              (number many)))
                        (location (London (THIS (WHERE POINT)))) ))) ))
         (state colour (val (query (DUMMY)))) )))
```

FIG. 7

supported by linguistic generalisations. The problem is removed, or significantly reduced, if the analyser depends directly on database-specific semantics, but this approach is not allowed here. However, since such compounds can only, in many cases, be interpreted by exploiting pragmatic knowledge, the current intention is to seek to proceed as far as possible with interpretation based on the ordinary analyser semantics, relying on the comparative richness of this apparatus, but to allow additional reference, in a strictly controlled way, to the independent semantics of the data language. The attempt must be made to maintain a genuine distinction between building the analyser on database-specific information and allowing it to call on

such information. At the same time, problems like the compound noun one make it quite clear that the system front-end processing cannot be expected to be a single pass exploiting first language, and then database, knowledge: some interactive feedback between the various components of the interpreter is to be expected. Interaction is also implied by the need to handle proper names.

The ease with which this interaction can be achieved must follow from the characterisation of the natural language–database connection which is central to the operation of the translation component. The translator has to replace expressions of the meaning representation language by those of the data specification language. It is not yet clear whether this can be more successfully done by a production rule strategy or by the invocation of some knowledge structure, say in the form of a semantic net. The natural starting point, however, for constructing the translator is the rich resources of the meaning representation language. The main challenge of the project is to build the translator; and though, as mentioned, work on this has barely begun, something needs to be said here about the proposed foundation for it.

The meaning representation language provides for the syntactically and semantically well-formed and explicit characterisation of question content. The database will be characterised both logically, i.e., syntactically and semantically, and administratively by the formal data language, and its content will be represented by the well-organised and explicit data descriptions allowed by the data language. The hypothesis is that, when viewed through the medium of the meaning representation language described above, the natural language and the data language will be found to have a substantial overlap: i.e., that while it may not be so easy to see the connection between the words and sentences of the natural language and the terms and expressions of the data language, it is much easier, given the conceptual and structural normalisation of the meaning representations output by the analyser, to connect these with the database; and in making this connection the semantic primitives have an important role. While it should be possible to characterise the message forms of the data language using some or all of the semantic apparatus described earlier, it may not be possible to characterise, for an expanding database, all of the lexical items represented by the values of attributes in the database. But it should be possible to characterise, using the semantic primitives, the names of entity types, attributes, and relations; and this may provide enough leverage to construct appropriate search queries. It is not being suggested that the words and forms of the database are exactly the same as those of the natural language, only that they may be characterised using the same semantic apparatus based on category and case primitives. For example while the English word "supplier" and the database name 'supplier' may not mean the same, they will have some conceptual overlap which can be captured by common primitives in their dictionary entries; and this may be sufficient to derive the appropriate query term from the question word. Similar considerations apply to the mapping of question word relationships onto query term relationships via case primitives. We are now beginning work on the attempt to provide these data language characterisations, and hence to build translation rules between meaning representations and search specifications, for a very simple database.

ACKNOWLEDGEMENT

This research is supported by a Science Research Council grant.

REFERENCES

Boguraev, B. K. (1979) *Automatic resolution of linguistic ambiguities.* Ph.D. Thesis, University of Cambridge (Technical Report No. 11, Computer Laboratory, University of Cambridge).

Bronnenberg, W. J. H. J., Bunt, H. C., Landsbergen, S. P. J., Scha, R. J. H., Schoenmakers, W. J. and van Utteren, E. P. C. (1979) The question answering system PHLIQA1. *Natural language question answering systems.* (L. Bolc, ed.) London: Macmillan.

Harris, L. R. (1977) User oriented data base query with the ROBOT natural language query system. *International Journal of Man-Machine Studies, 9,* 697–713.

Hendrix, D. G., Sacerdoti, E. D., Sagalowicz, D. and Slocum, J. (1978) Developing a natural language interface to complex data. *ACM Transactions on Database Systems 3,* 105–147.

Konolige, K. (1979) *A framework for a portable natural language interface to large data bases.* Menlo Park, Calif.: SRI International (Technical Note 197).

Riesbeck, C. K. (1975) Conceptual understanding. *Conceptual information processing.* (R. C. Schank, ed.) Amsterdam: North-Holland.

Small, S. (1980) *Word expert parsing: a theory of distributed word-based natural language understanding.* Ph.D. Thesis, University of Maryland (Report TR-954, Department of Computer Science, University of Maryland).

Waltz, D. L. (1978) An English language question answering system for a large relational database. *Communications of the ACM 21,* 526–539.

Wilks, Y. A. (1975a) An intelligent analyser and understander of English. *Communications of the ACM 18,* 264–274.

Wilks, Y. A. (1975b) A preferential, pattern-seeking semantics for natural language inference. *Artificial Intelligence 6,* 53–74.

Wilks, Y. A. (1977) Good and bad arguments about semantic primitives. *Communication and Cognition 10,* 181–221.

Woods, W. A., Kaplan, R. M. and Nash-Webber, B. (1972) *The lunar sciences natural language information system.* Cambridge, Mass.: Bolt, Beranek and Newman Inc. (Report 2378).

Woods, W. A. (1977) Lunar rocks in natural English: explorations in natural language question answering. *Linguistic structures processing.* (A. Zampolli, ed.) Amsterdam: North-Holland.