*Technical Report*

UCAM-CL-TR-64
ISSN 1476-2986

Number 64

UNIVERSITY OF
CAMBRIDGE

**Computer Laboratory**

# A framework for inference in natural language front ends to databases

Branimir K. Boguraev, Karen Spärck Jones

February 1985

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

*https://www.cl.cam.ac.uk/*

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

*https://www.cl.cam.ac.uk/techreports/*

ISSN 1476-2986
DOI *https://doi.org/10.48456/tr-64*

# A Framework for Inference in Natural Language Front Ends to Databases

Branimir K. Boguraev and Karen Sparck Jones

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

February 1985

**Abstract**

This report discusses the inference requirements to be met by a natural language database front end of the kind developed in Cambridge, with detailed examples, and proposes a framework, exploiting a specific type of network for knowledge representation, for developing the front end to support inference.

**Keywords**

inference, databases, natural language

**Acknowledgements**

This work was supported by the Science and Engineering Research Council under Grant GR/C/33321, and was carried out during Karen Sparck Jones' tenure of a GEC Research Fellowship.

# A Framework for Inference in Natural Language Front Ends to Databases

One prerequisite of the "cooperativeness" which will ultimately be required of natural language front ends to databases is their ability to perform inferences from the users' input, taking into account not only what has been explicitly requested from the database, but also a substantial body of implicit knowledge which is assumed to be mutually known. While some work has focussed on computing those inferences that are based on properties of language (namely presuppositions and entailments), issues of extra-linguistic inference, if tackled at all in current database access systems, usually have ad-hoc solutions which are tied down to a particular domain, database organisational structure and front end architecture. Questions of what knowledge to bring to bear on the reasoning process and how to control the potential explosion of inferences remain largely unsolved in the general case.

This report will present an analysis of the kinds of extra-linguistic inference required to support a database front end, and of the sorts of knowledge and inferential process required by an "ideal" front end inference component. The place of such a component in a portable natural language front end to databases will then be discussed together with its implications for the overall front end architecture and the structure of its underlying knowledge bases. The report subsequently proposes a framework for incorporating inferential mechanisms into portable front ends of the general kind considered.

Both the form of the inference problem and the approach adopted to its solution stem from the work on natural language access to databases carried out in Cambridge in the last three years: this provides the context for the whole discussion. The structure of

the report is therefore as follows. In the Introduction the general frame of reference for the report as a whole is established. Section 2 describes the background database project. Section 3 introduces a classification of inference requirements relating to the database front end structure outlined in Section 2. Relevant work on inference in the database context is summarised in Section 4. Section 5 presents the proposed strategy for providing the current database front end with inference capabilities, while more specific implementational issues are considered in Section 6. In the Conclusion an attempt is made to evaluate the contribution made by the previous problem analyses and development proposals, which are geared to a particular approach to the design of front ends, to the broader state of the art on front end inference.

# 1. Introduction

A not entirely surprising side effect of recent work on access to structured data via natural language interfaces is that data retrieval may be attempted by a wide user community, including a range of "naive" or "casual" users, on a random and infrequent basis. Each of the many users thus sharing the database has a different understanding of its physical and logical structure; in addition they are typically interested in different aspects of the stored data, and, furthermore, probably intend to use it for different purposes. They have no knowledge of the functionality of the formal query facilities of the underlying database management system (DBMS), and the freedom of expression offered to them by the medium of natural language encourages them to ask non-standard questions and gives rise to expectations that the database interface is indeed an intelligent agent with whom they are free to converse on an equal basis.

Thus the questions likely to be addressed to the database could well contain, or imply, considerably more than is stated in them explicitly. The only motivated way to fully understand such questions would involve the making of *inferences,* which will connect what has been said with what has been left implicit. An attempt to give a suitably formal definition of inference here will only get us into deep water; in the next section the forms that inference can take in the database context, or at least in one database context, will be illustrated and categorised. In the meantime it is sufficient to note that inference has both a **functional** interpretation — the process of drawing explicit conclusions from implicit assumptions — and an **operational** one — the process of suitably restructuring the original question to produce another question which is answerable in the database and so satisfies the user's requirement.

While it is doubtful whether, for primarily practical reasons, existing commercial level natural language interfaces would benefit substantially from the addition of an inference component, especially given the cost of constructing this and integrating it into

an already operational system (see for example Damerau, 1981), "there is no doubt that a truly natural system for language understanding requires such a capability" (ibid.:44).

Indeed there are many questions about the extent to which current conventional data models can be used as the basis for the representation of rich domain knowledge. Proposals for superior data models have been and are being made (e.g. Azmoodeh, 1984), and much is claimed for the formal power of logic databases (Gallaire and Minker, 1978; Kowalski, 1981; Gray, 1984); but such comparative investigations as Griethuysen (1981), and the difficulties of handling e.g. constraints and metadata (Robson, 1984; Storrs et al., 1984) show that conventional data models will either have to be radically developed or alternatively supplanted by powerful *conceptual schema* formalisms, to meet the kind of inference needs to be discussed. It certainly appears to be the case that providing a fully functional inference capability implies at least the provision of a conceptual schema embodied in the kinds of formalism studied by the AI community (e.g. KL-ONE (Brachman 1978)), along with a lot of disagreeable gearing to the existing DBMS data model, or the total downgrading of the DBMS to a purely routine service role, with the replacement of the data model by the domain knowledge representation apparatus.

Looking now at the functionality of natural language interfaces from the point of view of how cooperative they are — i.e. how far they go in their attempts to meet the users' demands, expectations and idiosyncracies — it is possible to demarcate a range of behaviours. At one end of the scale would be more or less "blind" translation of the user input (on the assumption that all that users require has been stated explicitly in their request). At the other would be full cooperation where the interface system is itself an expert in the domain and converses with the user on the basis of a dynamically constructed model of his expectations, beliefs, goals and plans (see Boguraev, 1985).

For the purposes of this paper, we shall concentrate on the role of inference in systems which do not go as far as analysing speech

acts (Perrault, 1980; Allen, 1982), inferring user goals (Pollack, 1983; Carberry, 1984), planning responses (Cohen, 1978; McKeown, 1982), or attempt to carry out substantial reasoning on the basis of data retrieved from the database, the way expert systems do. More specifically, we shall concentrate on those inferential processes that are required to interpret input rather than organise output; and further, in relation to input, we shall concentrate on the role of inference in what we may call 'straightforward' interpretation of the original input question, aimed at understanding it in the context of the database and deriving an answer from the database. Though this is only one role of inference, it is a critical one, particularly since it is a prerequisite for the extended interpretation of the input, for example to determine users' hidden goals. Providing computational support for inference needed in straightforward interpretation will thus constitute a "first-order approximation" to incorporating an inferential capability into a natural language interface system.

In addition, we shall not be concerned here with *linguistic* as opposed to *extra-linguistic* inference. The term "linguistic" is used to refer to a class of inferences which seem to be computationally relatively more tractable since they are based on properties of language. There is a substantial body of work which investigates the computation of *entailments* and *presuppositions,* (Weischedel, 1975; Kaplan, 1981; Webber, 1982) and algorithms to extract entailments and presuppositions have been successfully incorporated in systems like LADDER, (Hendrix et al., 1978) INTELLECT (Harris, 1982) and REL (Thompson, 1975). We shall therefore assume that such algorithms would be available to us.

Finally, this paper is concerned with a framework for incorporating common-sense reasoning capabilities into a particular front end design, namely that already implemented in Cambridge (Boguraev and Sparck Jones, 1984; also see next section). The motivation for this system is transportability over databases and domains. What we propose, therefore, is an investigation into the types of inferential processes and knowledge to support them, which will

allow the eventual incorporation into this system of an inference component without such common "tricks" as hard-wired domain knowledge, lexicon-driven short-cuts in the reasoning processes, or explicit takeover of the underlying DBMS by the interface, and which would stand up to new applications together with the rest of the system.

Perhaps the most important question then, and one that both requires careful study and implies trade-offs, is related to the functionality of inferential behaviour that we demand from the whole system. The representation of both linguistic and non-linguistic knowledge for the system depends substantially on the constraints imposed by the type of process that is utilising this knowledge in the course of text processing. Thus a system that aims at a certain level of performance would not necessarily require a completely general and very powerful representation formalism, nor should it support a completely versatile inference capability. We simply do not know enough today to build such a fully flexible and intelligent system. The question is really: what types of inferential processes would we initially like to be able to incorporate in a natural language front end, improving functionality while maintaining generality so that transportability is preserved.

We are going to exclude here the possibility that there is no way that the DBMS could handle the type of questions that the user wants to ask; and, equally, we shall assume that it is not necessary for the front end to go into major contortions to pervert the user's query into something that a very limited back end database could manage.

For the purposes of discussion we shall confine ourselves to single shot, single sentence questions, and not consider the additional complexities introduced by multi-sentence dialogue or discourse. Cooperative interaction will of course have to handle these, but equally the inference capabilities with which we will be particularly concerned are a necessary prerequiste for any more elaborate interaction capabilities. In particular, while it may sometimes be the case that the interpretation problems of individual sentences

can be eased or overcome by reference to the surrounding context, essentially the same mechanisms are required (if only as part of the apparatus) to exploit contextual information as to handle lone sentences.

Our investigation into requirements for inference in natural language front ends to databases begins with a detailed study and classification of the situations in which types of reasoning have to be performed. Before that, however, we shall present a brief description of the current state of the Cambridge interface system, to provide the specific context for the discussion in the rest of the paper.

# 2. Background

## 2.1. The front end

Our approach to building a natural language front end, and the system we have implemented, have been very fully described elsewhere — see Boguraev and Sparck Jones (1983, 1984). This section is thus intended simply to summarise the main features of the work for reference, in order to provide an adequate context for the subsequent discussion of how the front end could be enhanced with inference capabilities.

In this account we shall use "domain" to refer to the world of objects, properties and relationships of which the database is an instantiation, and we shall use "database" to refer both to the particular database schema constructed for this domain, as required by the DBMS, and to the actual data. "Schema" and "data" will therefore be used where the two aspects of the database have to be distinguished. "Data model" will be used to refer to formal types of model, for example the relational one, which may be adopted for the schema.[1] We shall use "domain description" (or "description", where appropriate) for the explicit characterisation of the domain.

The approach we have adopted has been one of maximising the front end's independence of the database and its domain, by making use not only of general syntactic information, but also of general semantic information about lexical items and their collocational relationships. The interpretation of the input question is thus carried out in two stages. In the first, *analysis*, the question is processed to determine its meaning as an ordinary expression of English. In the second, *translation*, the question is processed to determine its more specialised database reading. The function of the first component is to resolve, as much as possible, the ambiguity of natural language questions and to establish their essential meaning. The second component achieves the necessary mapping

[1] The relational model is the only one we have used so far, and is assumed throughout this report.

into the particular form of expression appropriate to the DBMS. The essential principle on which the front end is based is that its transportability is enhanced by the use of our general semantic apparatus, since this reduces the complexity of the semantic operations tied to the domain, and hence the effort of supplying the domain information on which these are based.

The detailed structure of the front end is in fact more complicated than the foregoing would suggest, since each major component has two subcomponents. The front end thus consists of four processors, each delivering its own characteristic representation of the input question to the next, as shown in Figure 1. The first processor within the domain-independent analysis component, the *analyser,* combines syntactic parsing using an ATN and conventional grammar with semantic procedures incorporating ideas derived from Wilks, to achieve lexical and structural disambiguation of the input question. The output *meaning representation* is a dependency tree whose components are case structures linking word senses, defined by formulae using semantic category primitives, through semantic relation primitives. The second analysis processor, the *extractor,* works over the dependency tree to pull out its 'atomic' propositions of object-link-object or object-possess-property form, and to determine the question's quantification structure. The resulting *logic representation* has the general form of the quantified expressions of LUNAR (Woods, 1972), with the detailed semantic information given by the formulae and case labels embedded in it as hooks for the subsequent domain-dependent operations. This subcomponent can be viewed as task-oriented in the sense that the item at the focus of the question is moved to the top of the tree; but this is only in the broad sense that question-answering in general, without reference to coded databases, is deemed a natural language processing task.

In the second major, translation, stage of interpretation, involving domain-dependent operations, the question is processed first to establish its domain meaning, and then to derive a suitable form for the database instantiation of this domain. The initial

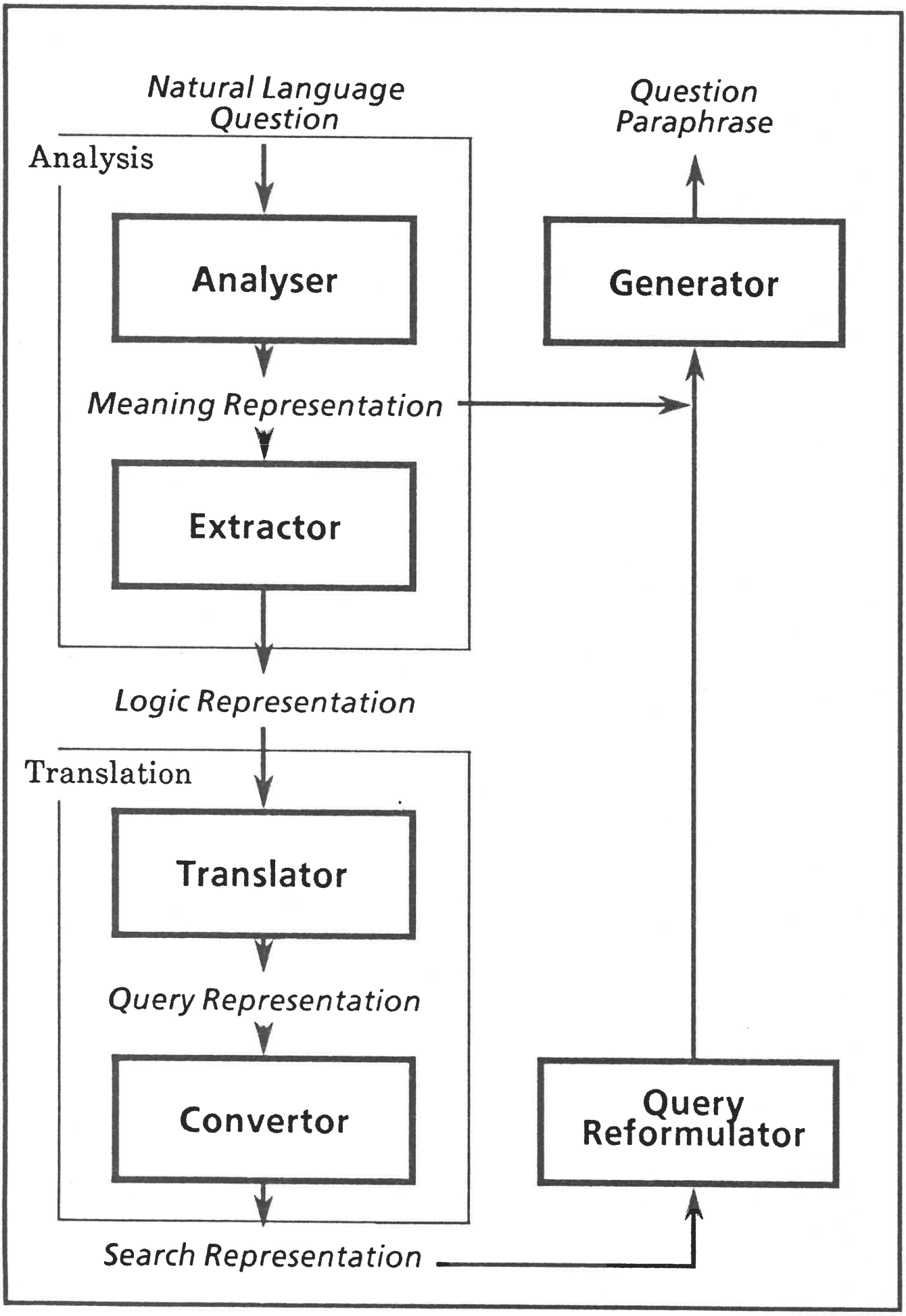
Natural Language Question
Analysis
Analyser
Meaning Representation
Extractor
Logic Representation
Translation
Translator
Query Representation
Convertor
Search Representation
Question Paraphrase
Generator
Query Reformulator

Figure 1

*translator* substitutes domain-referring terms and expressions for those referring to the ordinary world that were handed to it by the extractor. As explained in detail in Boguraev and Sparck Jones (1983), the grounds for this substitution are the semantic primitive characterisations of the elements and the structure of the question. As the terms and expressions of the high-level data language used for the translator's output *query representation* can in principle be similarly characterised, implicit commonalities in the characterisations of sentences in the two languages motivate the translation. The second translation processor and final front end module, the *convertor,* in turn relates domain-referring terms and expressions to those of the low-level data language used to search the actual application database. This processor in fact first produces a relational algebra form of the query, i.e. a representation suited to a relational database of any kind, and subsequently the final *search representation* of the input question in the specific language accepted by the local DBMS (in our case primarily SALT — see King, 1983 — and QUEL — Stonebraker, 1976).

The essential properties of this front end are a strongly modular design, with each processor undertaking a well-defined interpretive task and generating a correspondingly 'oriented' representation of the question; and the uniform application of *pattern matching* as the processing technique driving each module. Of course the application details vary in the different modules: the analyser, which has been used for other natural language systems, is a more comprehensive processor than the others; and the convertor is somewhat messy. Nonetheless the general philosophy of the whole front end is one of pattern matching. It might be more appropriate to use the expression "pattern mapping" rather than "pattern matching", reserving "pattern matching" as the term for the wholly abstract, formal operation. "Pattern mapping" in the present context would make the contrast, in linguistic interpretation, between transformations mapping one linguistic expression onto another without provision for inference and those for which inference may be invoked. However we will continue to use "pattern matching", albeit

in the sense of pattern mapping just described.

The sequence of representations produced for one input natural language question is illustrated in Figure 2. The details of the various forms of representation are given in Sparck Jones (1984). For present purposes it is sufficient to notice that in the meaning representation (or as it has sometimes been called, the *text representation*), items such as 'own1' are word sense labels; the expressions following these are the primitive formulae (with head elements at the right emphasised in bold) — thus

((*org subj) ((*ent obje) **have**))

is the formula for 'own1', denoting the "possess" sense of "own"; and items prefaced by @@, for example @@agent, are case labels, i.e. semantic relation primitives. In the logic representation the formulae and case labels associated with word senses are not given, but it must be emphasised that they are retained by the system. In the query representation domain-referring items in the high-level query language are prefaced by '&', as in '&own'. The various elements of the search representation are of course those of the DBMS query language: thus 'Owners' is a relation name and 'Surnam' an attribute. Note that for clarity in the text of this report, items in specialised languages, like the domain and data languages, are indicated by the use of bold, along with e.g. underlining for data attributes. Thus we write &own, **Owners** and **Surnam.**

The representations of Figure 2 are not merely the means of communication between one module and the next. As noted, they embody different views of the question which are of value in their own right and can be exploited, for example, as sources of feedback to the user showing how the question is being handled (see Boguraev and Sparck Jones, 1984). Thus generating an English equivalent of the initial meaning representation can show how lexical and syntactic ambiguities have been resolved, while generating from the final search representation can show how the database access path has been constructed (see also Figure 2). It is argued in Sparck Jones (1983) that representations of different types have

User Query: *Which Norwich Road properties do International Stores Ltd own?*

2.a Meaning Representation:

```
(clause (type question) (query identity) (tns present)
      (v (own1 ((*org subj) ((*ent obje) have)))
            (@@ agent (n (International Stores Ltd (this (*org ent))))
            (@@ object
                  (n (property1 (((*org subj) have) (obje ((where spread) thing))))
                        (@@ number many)
                        (@@ nmod
                              (((n (Norwich Road (this (line (where spread)))))))))))))
```

2.b Logic Representation:

```
(For Every $Var1/property1
      : (AND (For The $Var2/Norwich.Road - (dummy $Var1 $Var2))
            (For $Var3/International.Stores.Ltd - (own1 $Var3 $Var1)))
      - (Display $Var1))
```

2.c Query Representation:

```
(For Every $Var1/&property
      : (AND (For The $Var2/Norwich.Road - (&location $Var1 $Var2))
            (For $Var3/International.Stores.Ltd - (&own $Var3 $Var1)))
      - (Display $Var1))
```

2.d Search Representation:

```
(Range of (Ql-var2, Ql-var1) is (Parcel, Owners)
Retrieve into Terminal (Ql-var2.Pid)
      where (Ql-var2.Pid = Ql-var1.Pid)
      and   (Ql-var2.Strnam = 'Norwich Road')
      and   (Ql-var1.Surnam = 'International Stores Ltd').
```

2.e Reformulated Query:

*Show me the properties in Norwich Road which International Stores Ltd own.*

Figure 2.

functional roles in language use in general, and one issue implicit in this report is the mutual relationship between representation and inference.

## 2.2. Test Databases

The tests we have carried out with our front end have served to bring home to us the need for inference to support question interpretation, and since we are concerned with the way in which inference capabilities might be incorporated in our particular kind of front end, it is convenient to discuss this in the context of examples from our experimental material. This section is therefore intended to provide enough information about this material to motivate the subsequent analysis of inference requirements and discussion of our proposed inference mechanisms.

The initial tests with the system were with a **Suppliers and Parts** database. This was a toy in both complexity and scale, though it served its purpose as a non-trivial test of a front end not relying on domain-specific semantics. Experiments were subsequently begun with a database of planning information. This database, **Green Hills,** is an English reflection of the IBM TQA Project's White Plains application (Damerau, 1980). Actual White Plains data could not be made available to us for proprietary reasons, but we are using the same form of database description, i.e. set of attributes, applying them to an imaginary village in East Anglia. The database has been set up in relational form, rather more thoroughly than the TQA Project was able to do, given the practical constraints of their use of a large, existing database. A few of the attributes, especially identifying numbers, have also been treated slightly differently, to avoid some arbitrary low-level coding features of the White Plains implementation.

The detailed descriptions of the attributes and relations of the full Green Hills database are given in Figure 3. (We have used only a subset of the attributes, and some in a simple form, in our tests so far, but this is irrelevant here.) The essential structure of

| | |
|---|---|
| Wno | ward number |
| Bno | block number |
| Pno | parcel number |
| Lno | lot number |
| Strnum | street number |
| Strnam | street name |
| Zone | planning zone class code |
| Type | assessor's property type No. |
| Stcode | state code |
| Stor | number of stories |
| LUCi | land use code for i-th floor (i =< 3) |
| Park | number of parking spaces |
| Dwell | number of dwelling units |
| Resfl | number of residential floors |
| Comfl | number of commercial floors |
| Landv | land assessment (i.e. valuation) |
| Imprv | improvement (building) assessment |
| Exemv | assessment exemption |
| Cityv | assessment for city district |
| Schv | assessment for school district |
| Sewv | assessment for sewer district |
| Sqft | parcel area |
| Gflsqft | ground floor area |
| Censt | census track number |
| Censb | census block number |
| Plarea | planning area number |
| Schzone | school zone number |
| Nbarea | (1962 plan) neighbourhood planning area number |
| Xgrid | X grid coordinate |
| Ygrid | Y grid coordinate |
| Drarea | drainage area number |
| Splarea | subplanning area number |
| Trzone | traffic zone number |
| Neigh1 | neighbourhood association 1 number |
| Neigh2 | neighbourhood association 2 number |
| Bid | combines Bno and Pno |
| Pid | combines Wno, Bno and Pno |
| Lid | combines Wno, Bno and Lno |
| Inits | initials |
| Surnam | surname |

Figure 3.a Green Hills attributes

the domain is that it deals with parcels of land. These are located in the progressively more embracing areas of block and ward, and themselves may include lots. Most of the attributes are those distinguishing parcels: they include area properties like square feet, volume properties like number of stories in buildings, valuation properties like the assessment for school taxes, and use properties like the number of residential floors and land use codes (LUCs). Blocks also have a range of properties including physical area and a variety of administrative areas ones, for example census tract and traffic zone. The most difficult attribute is LUC (strictly attribute type, since there are separate LUC assignments for different floors). Individual building functions e.g. pets hospital, school, etc are subsumed under specific codes via an independent list, presenting many problems for the recognition of equivalent forms e.g. pets hospital/animal hospital/veterinary centre. There are also complex implicit interactions between different attributes, for example different area measurements; and there are some nasty incomplete interactions hinging on the fact that LUC3 can refer not only to floor 3 specifically, but to floor 3 and all the higher floors.

The kinds of question that the IBM TQA system could process are illustrated in Figure 4 (see also Damerau, 1981). The TQA implementation clearly involved a great deal of very specific program unique to that application, sufficient to resolve many of the problems we shall consider, albeit in an ad hoc way. The user could also be assumed to be very knowledgeable about the domain, and indeed the database, so questions leading to difficulties might well not get asked.

Many of the questions which would ordinarily be addressed to Green Hills, like those addressed to White Plains, would not require inference, even without an LUC lookup mechanism as part of the lexical apparatus for dealing with words and phrases referring to parcel functions. Nor do we assume (particularly as we are not users ourselves and have no access to the staff of planning offices) that all the types of question we shall use as illustrations would actually get asked. However Green Hills analogues of actual White

*How many two family houses are there in the Oak Ridge Residents Assn.?*
*What is the account number of the parcels that DelVecchio owns?*
*How many dwelling units are on Craven Lane?*
*What parcels in Planning Area 8 have more than 79 dwelling units?*
*Who owns the vacant land in Hillair Circle?*
*What is the average assessment in subplanning area 7.10?*
*What is the parcel area of Roane's property?*
*How many three family homes are there in Fisher Hill Assn.?*
*What is the area of the vacant land in the B1 zone which is in subplanning area 5.30?*
*What is the average assessment of the apartments on Lake St.?*
*Where are the nature trails in the city?*
*What is the average lot area of the 4 family houses in Fisher Hill?*
*Where are the gas stations in Fisher Hill?*
*What is the gross floor area of the commercial buildings?*

Figure 4: Sample TQA queries

Plains questions can produce inference problems, serving to emphasise the fact that inference is required, so we must be prepared for it. In particular, inference problems can be produced for Green Hills which would be finessed by the domain-dependent semantics and powerful lexical apparatus of the TQA implementation. Further, other applications of the front end could well be to other domains and databases making stronger demands on inference. The examples considered in the next section should therefore be seen as legitimate illustrations of the kind of thing that we should allow for, rather than questions actually put, or likely to be put, to the particular White Plains/Green Hills implementations.

# 3. Classification of Inference Needs

In seeking to categorise inference procedures, we can in principle do this according to their different modes, according to their different functions, i.e. the sorts of needs they have to meet, according to the different kinds of information to which they are applied, or according to their various locations, and specifically the various modules by which they may be called.

As noted earlier, we are concerned here only with non-linguistic inference, i.e. inference utilising non-linguistic information. Our prime concern, moreover, is with inference on application-specific information, in this case on domain information and database information — the latter being primarily, though not exclusively, that given by the database schema. In general we may assume that inference on 'ordinary', or non-specialised, world information will be needed to support full text interpretation, as it has been found necessary in other work on language processing. Thus we may suppose that the analyser in particular, and possibly the extractor, may call for inference on general world knowledge in the same way as these modules call on general linguistic knowledge. However the problems presented for our front end design stem essentially from the need to access information about the application. We will therefore, somewhat squashingly, take resources to support non-specialised inference for granted, in order to focus on the application-derived issues. There would of course be questions about how general and specific information would be related in representation and use: some indication of our answers to these questions will emerge in the detailed presentation of our proposal for handling application-specific knowledge and inference.

In the analysis which follows we consider how needs for inference on domain and database information arise in the successive front end modules. The essential point which emerges is that while one would like in principle to associate inference referring to the domain and schema with the translator and convertor respectively, it is called for earlier on. This is a major problem for our approach.

It is what the conventional form of front end design using domain-dependent information in analysis would avoid. (Note that we will be considering only inference during question interpretation: similar problems could arise in feedback and response.)

When we refer to a type of inference, therefore, this is very informal and simply locates the inference in a specific module in the front end, and relates it to the type of information used within that module.

## 3.1. Inference during analyser operations

This is concerned mainly with resolving ambiguities in the user input. Most of these ambiguities are linguistic, but analysis can only be completed through reasoning about the domain. Constructions like

1. compound nominals (*"... city school district gas station owners..."* — owners of gas stations in the district with the city school in it) and
2. post-modifier phrases attachment (*"Do any owners who own properties located in Market Place also own parcels in ward 2?"* — is it the owners or the properties that are located in Market Place?)

belong here.

There seem to be essentially two approaches to incorporating the necessary inferential reasoning into this first phase of question processing. One approach is to interface the parsing with both the domain description and the database schema, making it possible for the parser to query which of the alternative interpretations makes sense in the database context (this is the function of DIALOGIC's *pragmatic functions* — Grosz, 1983; see also Ginsparg, 1983). Alternatively, the set of possible analyses may be passed over to the translator which will attempt to establish which particular one is valid from the point of view of the database conceptual outlook. Thus Boguraev and Sparck Jones (1983) finesses inferential reasoning by invoking semantic pattern matching procedures

in the translator, which are aimed at establishing the appropriate relationships between domain objects implicitly encoded in the linguistic structure of the input question.

There is, however, a different class of problems arising in analysis, for which reference to the database domain cannot be postponed. This is connected with the explicit use of *value* words. A wide range of linguistic constructions may be used to convey the potentially quite complex semantics of what is intrinsically just a character string or a number. While it may be possible to derive a relatively uncommitted interpretation of an ambiguous noun-noun compound, hoping for subsequent filtering during translation, situations may still arise where there is the danger of complete failure to capture the correct meaning of the question. Consider the following request to the Green Hills database:

*Which 541 properties have parking lots?*

Unless the analyser is aware that properties in the database are identified, among other things, by Land User Codes and "541" is a valid LUC which means "supermarket", it will produce, by default, the obvious interpretation leading to the subsequent output of a list of 541 properties which satisfy the specified condition, i.e. have parking lots; it will not retrieve, as intended, only certain supermarkets.

The important point here is that unless the parsing process somehow takes into account features of the domain (and also database to which it interfaces), it may completely fail to deliver the correct, intended analysis of the input question. In this case no amount of subsequent domain reasoning will be guaranteed to remedy the error.

Similar problems are exemplified in the following requests:

*Print the parcel areas of Land User Codes*
*300 to 399.*

*Print the parcel areas of Norwich Road 300 to 399,*

where the semantic impact of the numbers, and therefore the struc-

tural analysis of the questions, can only be established on the basis of specific knowledge of the properties of the domain.

In general, there is a whole host of issues related to the problems associated with value words. As Grosz (1982) notes, a certain amount of domain reasoning (see next section), interleaved with the process of linguistic analysis of the input, may be required to determine the particular way in which what in final database terms is a field value (or any of its synonyms) figures in the user question. Thus it may appear as a noun phrase modifier (consider the problems of analysis of *"shop buildings"* vs. *"school buildings"*, where there is a complete LUC reserved for the concept of a "shop building", but the database offers separate encoding for "school" only); or as a head noun referring to entities which, viewed from the angle of a particular attribute, have this particular value in the corresponding field (e.g. *"Market Place"* really means *"properties for which the street-name attribute is equal to* Market Place*"*.

Without going into any further details, it is clear that inferential processes cannot be confined only to domain-dependent components, and some of them will have to be invoked as early as the linguistic analysis of the input. As a final point, it is worth noting that the questions of referring pronouns and supplying missing information cannot be easily dismissed even in a 'single shot' exchanges, as the following example shows:

> *List all the properties with parking spaces together with their assessed values.*

What makes this question unambiguous is the fact that in the database it is properties (in fact, parcels) which have assessed values. There is a different field specifying how many parking lots (if any) are associated with a parcel, but no assessment for these is available. (If, on the other hand, the question was

> *List all the properties with parking lots together with their assessed values,*

it would be genuinely ambiguous, since there are fields both for the value of the land and for its improvement, i.e. the building(s)

on it.)

## 3.2. Inference during extractor operations

It is evident that domain-referring inference may be required during the extractor processing to sort out quantifier scoping. This is particularly obvious with "each". While

*Who owns each property?*

is genuinely ambiguous, since it may properly mean either *for each property, who owns the property,* or *who is the owner such that he owns each property,* domain inference would choose only one scoping for

*Where is the school in 2/2/8?*

since "2/2/8" is a unique parcel identifier, and there can only be a single school in a parcel. The difficulties of achieving correct quantification have been discussed elsewhere — see, for example, Woods (1978) and Moore (1982). However as our treatment of quantifiers has been left in a fairly undeveloped state, we shall merely note that inference problems arise in extraction, without further elaborating on them. In this context, however, we should admit that our treatment of definite and indefinite determiners is wholly inadequate, and that inference problems will also appear in extraction in connection with definite descriptions (Berry-Rogghe 1984, 1985; see also Woods 1978) as for example in

*What is the assessment for Smith's properties?*

which may mean either what is the assessment for each of Smith's properties or what is the total of the assessments for all of them.

## 3.3. Inference during translator operations

Most discussions of inference connected with databases have focussed on domain reasoning. This is only natural, since "inference" in the AI context has been mostly synonymous with "common-sense reasoning", and it is in the domain description that all salient assumptions about the database world are (or ought to be) made

explicit. (For the purposes of the present discussion "common sense" knowledge and reasoning will thus be associated with the domain, though more generally, of course, it is not so tied.)

The most common inference *functions* to be met by the translator can be grouped under the general headings of *coercion* and *conceptual completion*. Consider the following model of question interpretation. The natural language input is analysed and normalised, so that somehow the gap between the natural language words and expressions and the domain objects and concepts has been bridged. Without loss of generality, it is possible to regard the result of this processing as a conceptual structure, centred round a *domain verb* whose case (or slot) fillers are objects defined in, and known to, the domain description. This structure expresses certain relationships between the objects in the domain and the translation process utilises strong preferences about the types of argument that the domain verb expects, and will accept, as slot fillers.

The need for coercion arises when the expected preferences are violated, though all the slots in the domain verb case frame have been filled, and their fillers are valid domain objects. One obvious reason for this is that the user has achieved brevity in his question by "short-cutting" a long-winded statement about some assumed relationship between certain domain objects, easily inferrable from the domain and therefore left implicit. A question like

*Where is the Potters Lane car park?*

in fact in Green Hills means *Where in Potters Lane is the property with a car park on it?* and a coercion process is required to infer the implied relationship between "car park" and "property". Similar reasoning has to be carried out to understand the meaning (in this database context) of questions like

*Are there any schools in Ward 2?*
*What is the average height of Norwich Road?*
*Which owners in Market place aren't in Ward 1?*

(The last example demonstrates that coercion cannot be reduced

to simple expansions of conceptual slot fillers — what the question really means by "owners" is "properties".)

While the need for coercion can be traced to violation of slot-filler constraints, a different class of problems arise when there is an empty slot in the conceptualisation of the domain verb frame, since this typically has to be filled to obtain a valid query. This calls for conceptual completion inference whose aim is to deduce a likely filler for the slot. Thus in order to answer a question like

*How many shops are there?*

the system would have to infer the block and ward on which to focus.

Both coercion and conceptual completion may require arbitrary long chains of domain reasoning, and specifically of *causal* reasoning; extended causal reasoning is much more likely when there is no obvious domain verb to hand to guide the whole process. In such cases there is no clear starting point for the inference needed to establish the relevant domain predicate, because the given unmapped verb and its collection of possible slot fillers for the sought domain verb are not very discriminating and/or enlightening. The problem of controlling long chains of (weak) causal reasoning (in the style of Rieger, 1975) then becomes acute. Consider, for example, the following question

*Who could afford to buy International Stores Ltd?*

An answer might be derived by evaluating a query against the database, with approximately the following content: *Whose property is assessed at more than the total assessed value of International Stores Ltd parcel.* The missing conceptual filler here is the "money" (or its equivalent) required for the purchase, and inferring this gives rise to a chain of reasoning connecting assets with personal worth and personal worth with buying power. Similar processes will be required to derive an answer to queries like

*How wealthy is Colonel Cribbin?*
*Can I refuel my car in the Market Place?*

The inference engine must be capable of handling 'vague' concepts (such as "profitability" and "wealth") and reasoning about their relationships in the real world — and this is where much of the centrol problem comes from.

Causal reasoning may indeed be stimulated as a means of achieving coercion or conceptual completion, in what may be described as second-order inference defined in terms of the kind of pragmatic relationships it exploits, rather than in terms of its system function. Of course the causal relation is a very general one, which may be given a range of more specific forms, for example in Rieger's style. However for the kinds of reasons just illustrated, causal reasoning may be required to allow question interpretation outside the specific functional contexts associated with coercion and conceptual completion, and indeed may be the main form of inference employed.

An adequate capability for dealing with vague concepts in causal style appears to be particularly important in connection with *negative* inferences. In contrast to the cases discussed so far, where "understanding" the question effectively requires the inferential processes to deduce the real question to be asked of the database, negative inference is useful for intercepting questions where common-sense reasoning suggests that there is no point in translating the input, because it does not make sense (at least in the world of the database). While the need for a negative inferential capability manifests itself most obviously at the next stage of processing — producing the actual search query — domain reasoning by the translator about e.g. time, place and part-whole relationships would conveniently block further processing of questions like

*How many lots have land values*
*larger than lot values?*

*Which garages don't have parking lots?*

(In practice, processing the second example would require linguistic inference to compute the presupposition that some garages may

not have parking lots — but common-sense knowledge is required to block the further processing of this question).

Note that while controlling negative inference may be more difficult than controlling positive inference, the general organisation of the system's reasoning processes should not depend on whether they are being applied in a positive or negative way.

Inference of this common-sense kind may also be required for a rather different reason, to cope with a fundamental drawback of the typical DBMS. This is that such a DBMS has no proper facility for answering "yes/no" questions if taken at their face value (there may be other linguistic forms which are not acceptable either).[2] The unacceptability of such questions is properly a feature of the high level query language utilised by the translator. The translator may therefore need to exploit common sense knowledge and reformulate the question in an appropriate way. Thus while a simple linguistic transformation as a routine operation of the translator would be adequate for

> *Do the owners of The Blue Sheep Hotel own any other real estate?*

(the transformation would involve the rephrasing of the question into

> *Show the real estate owned by the owners of The Blue Sheep Hotel,*

the answer to which, by extension, would satisfy the original question), causal inference would be required to obtain more tractable versions of

> *Can I refuel my car in the Market Place?*
> *Has the Antique Shop been sold to Colonel Cribbin?*

(These could be paraphrased, for example, as

> *Is there a garage in the Market Place?*
> *Does Colonel Cribbin own the Antique Shop?* )

[2]As mentioned earlier, 'speech act' type questions like *Do you know who owns...?*, which present similar problems, are not being considered in this report for more general reasons.

Complex coercive inference would be needed for

*Is there a lawyer in the village,*

(paraphrasable as

*Is there a property whose owner can be identified as a lawyer?* ),

while an arbitrary combination of causal and coercive inference, with reasoning about e.g. part/whole and time relationships thrown in for good measure, would be needed for

*Has the second school in Ward 2 been demolished yet?*

in order to derive, for example, something equivalent to

*How many properties with schools are there now in Ward 2?*

and ultimately to

*Is the number of properties with schools in Ward 2 equal to 1?*

In addition to the above needs for inference, which must be met by any natural language front end, issues of transportability add further problems for an inferential component. Thus our particular approach to transportability assumes a completely general purpose, domain-independent natural language analyser. One consequence of such a decision is that the lexicon should never be changed, only augmented with new vocabulary senses or items. What this means, of course, is that in order to accommodate the user's conceptual viewpoint, it will, on occasions, be necessary to infer the domain status of a concept referred to by a word which does not immediately or easily fit into the domain description. There are two aspects of this problem.

The first one is that a concept referred to by the same natural language word will, in different domains, be instantiated to different domain objects (or properties, etc.): thus a "city" may be an **OperationsBase** in a Suppliers and Parts database, a **Port** in a Naval database, **Residence** in a Personnel database, and so

forth. The second problem is that in any of these applications, it is possible to use esoteric and/or ambiguous words to refer to these domain objects: "habitat", "headquarters", "domicile", "residency" are but a few. While a domain-specific front end can either hard-wire the appropriate interpretation in the lexical entries, or simply prohibit the use of these words (by omitting to supply lexical entries), a transportable system cannot conveniently take either of these essentially engineering approaches, thus making the need for coercive inference during subsequent convertor operations even more pronounced.

## 3.4. Inference during convertor operations

Generally it is the case that a system which processes a text in order to support the performance of a non-linguistic task also needs knowledge about the task itself. It is therefore difficult in practice to draw a precise line between reasoning within the domain and reasoning involving the database (both schema and, in the limit, actual data). In fact, many of the cases discussed in the previous section can only be tackled on the assumption that the domain reasoning is supported by access to knowledge about the database organisation. Unless it is known that the **OwnerIdentifier** field is a string which may indicate the individual's occupation, there is no basis for reasoning about the availability of lawyers; unless it is known that the DBMS[3] can support string manipulation operations, there is no point in even constructing a formal query, because we will not be able to extract the required information from whatever its evaluation returns. However, we will consider database reasoning here as if it could be completely separated from domain inference, as we discussed domain inference without reference to its database connections. (We will not be concerned here with inference aimed at optimising the search query, once it has

[3]Note that the term "DBMS" here covers the complete suite of supporting back-end programs — those which will pre-process the query, for example to optimise it, evaluate it against the database, and will effect any subsequent manipulation of the result(s)

been constructed — see, e.g. Warren and Pereira, 1981).

Perhaps the task to which database knowledge is most usefully applied is dealing with a question which, even though it makes sense in the domain, cannot be answered because the actual database schema will not support the generation of a corresponding specific search query. Consider

> *What parking lots are there in the immediate vicinity of the hospital?*

Even assuming that some interpretation of "immediate vicinity" imposing an upper limit on distance has already been provided, distance between parcels is not encoded in the database. An alternative interpretation of "immediate vicinity" as adjacency would not be supported by the database either. Similarly, the concept of developed land may well be familiar to the domain schema, supplying the interpretation for

> *What is the total undeveloped area in the Market Place neighbourhood?*

However, since the database does not support a breakdown of each parcel into built and not-built areas, we cannot construct a corresponding search query. Essentially in such situations we are in a bind stemming from the fact that it is likely in practice (if not in principle) to be the case that a domain description rich enough to interpret many natural language formulations of questions to which the DBMS can provide an answer will be overhospitable.

As in the domain case, negative inference could block the generation of 'bogus' search queries, which would fail against the database because its structure offers no support for them. Thus in the planning database the Land User Code information is spread over three fields, whose contents correspond to the LUCs of, respectively, the first, second, and all floors of a building above and including the third. A question like

> *What is the Land User Code for the fourth floor of the Guildhall?*

ought to be trapped before a database query is constructed. That is, we should not construct a plausible-looking, but improper, query using LUC4, or indeed incorrectly construct one using LUC3.

In general, positive inference in the translation stage as a whole is concerned with a range of planning processes. Two distinct activities are involved: identifying the raw data to be extracted from the database, and determining what operations on it will derive the required answer from this data.

In relation to the convertor, inference may be required to establish the appropriate database schema mapping of the domain concepts and structures determined by the translator. For example, in the Suppliers and Parts case, it would be necessary to decide whether &Supplier should be mapped onto **Sno** in the relation **Supplier** or **Sno** in the relation **Shipments**, and for a given query inference might be needed to do this. (Determining that "city" should be interpreted as &Scity — i.e. city from which suppliers operate — as opposed to &Pcity — city where parts are stored — on the other hand is properly a function of the translator). The context of the Green Hills database provides similar problems, for example in deciding whether it makes sense to map &parcel identifier (**Pid**) onto relation **Parcel** or relation **Owners**.

Many of the kinds of problem discussed in connection with the translator have obvious parallels in conversion. In other cases difficulties arising in the translation stage will probably be only partly handled and will propagate forward to the convertor. Indeed the assignment of inference problems to one or the other is ultimately arbitrary as it depends on the specific characters of the domain description and database schema. (The same applies in general to allocation between the analyser and extractor, of course.)

Further, there may be inference requirements to be met in determining, for example, the exact role of a value word (and thus its mapping onto the database schema), or the appropriate relationship between several domain objects with associated information stored in different files (and thus requiring complex 'joins' — sometimes known as the "multipath problem" — see Moore, 1979).

There are, however, also convertor inference problems associated with determining specific operations on individual database entries to complete the interpretation of the user's question. Thus, assuming that the DBMS is capable of supporting some calculation capabilities, for example those required to handle [*age* = *today* − *birthday*], [*totalworth* = sum of all assets], [*distance* = *finish*−*start*], [*average* = sum of *n* over *n*]) then a certain amount of reasoning will be required in order to generate DBMS programs to carry out operations like giving data summaries:

*List all the properties according to*
*their parcel numbers!*

*What is the total area of Ward 2,*

or carrying out implied calculations (e.g. sums, averages, percentages):

*What is the average value of a property*
*in Norwich Road?*

*What percentage of Ward 1 does the Manor take?*

Needless to say, the issues of invoking the appropriate inference procedures, together with related issues of control, become more complex when it is the case that more than one inferential process, of the variety presented, is required to interpret the question. In fact, a substantial portion of the examples in this section exhibit such mixed properties. A question like

*What is the average height of Norwich Road?*

in addition to the coercion from "Norwich Road" to "properties on Norwich Road", will require a certain amount of common-sense reasoning to associate the height of a building with the number of floors in it, to deduce that the total number of floors is a sum of the residential and commercial floors, to plan a query which will retrieve a sequence of records (for the properties along Norwich Road), and to initiate post-retrieval analysis of data aimed at computing individual, followed by the average, heights. This will in general also imply inference in more than one module, for

example in this case in the translator and the convertor modules.

The emphasis in this section has been on the points where inference needs arise, though in the process some specific *forms* of inference have been introduced. Thus coercive and conceptual completion inference, defined functionally, have figured in the context of translator operations, along with causal inference, which is defined rather in terms of the type of relationship it exhibits. Causal reasoning, as the major (though not the exclusive) manifestation of common sense reasoning, was again clearly needed for the translator, but would equally be the appropriate form of inference for dealing with some of the problems associated with, for example, the analyser. Other analyser problems would appear to require forms of inference resembling coercion and conceptual completion, given that these were specifically related to the translator's representation language structures.

It is clear that the forms of inference needed are somewhat heterogeneous, and that no very useful purpose is served by an attempt to define inference narrowly in terms of just one of the categories introduced at the beginning of this section, e.g. purely functionally. We should rather expect to have a number of forms of inference, some relevant to more than one processing module, others in fact exploited by only one processor. The examples discussed clearly identify coercion, conceptual completion, and causal inference as forms the system must support, and these will be explicitly referenced in the proposals made later in the report; but this is not intended to imply that other forms of inference might not be called for in practice.

# 4. Work on Inference to Date

There has been surprisingly little work on incorporating inferential capabilities in a motivated way into front ends to conventional database systems. Both semantic and domain knowledge have been applied to a variety of reasoning problems arising in language processing for other purposes, for example accessing expert systems. Thus inference has been invoked to support more straightforward linguistic operations (lexical and structural disambiguation, referent identification); to handle extended uses of language (metaphor, ellipsis); and in the analysis of larger bodies of text in context (situation identification, summarising).

Clearly techniques developed for such tasks as expert systems interaction can, in principle, also be applied to the task of natural language access to databases. In practice, however, very few database projects explicitly address the questions of what types of reasoning their task would require and how to provide the necessary apparatus for inference. Notable exceptions here are TEAM and KNOBS, (Grosz, 1983; Pazzani and Engelman, 1983), as well as Ginsparg's transportable natural language front end to databases (Ginsparg, 1983).

One reason for the lack of inference work in the database context is undoubtedly the common limitations to single shot operation. But a more important one, given that inference is required even here, is that most of the inference requirements of current commercial interfaces are met by engineering methods, which, necessarily, rely heavily on domain descriptions interfaced at a rather low level to the language analyser proper, thus imposing domain- (and task-) specific constraints on its design and making the analyser not readily (if at all) portable. To take just one example, while Palmer (1983) has shown that a very substantial mileage indeed can be obtained by properly utilising the semantics of finite and well-defined domains in order to postulate domain-specific inference rules and integrate their application with the semantic processing of a natural language utterance, it is not at

all clear, and she offers no indication of, how the approach is generalisable for applications targeted to a wider range of more complex domains (Palmer's approach will be discussed in more detail later). The engineering approach is indeed in some cases pushed as far as incorporating pragmatic information and inference-type operations in the system's semantic descriptions and pattern-matching strategies.

There is a certain amount of argument as to whether an experienced user community, well acquainted with the structure of the stored data and the functionality of the underlying DBMS, actually requires a sophisticated inference capability in a natural language interface. It appears that in operational, commercial systems users rapidly develop a feel for the limits of the system and make it sufficiently habitable by, on the one hand, ad-hoc enlargement of the lexicon, and, on the other, learning what sorts of question are acceptable. (In many cases, a reasonable natural language interface is likely to be such an improvement over previous arrangements that defects associated with the lack of an inference capability are disregarded. This is particularly true where it is known that the underlying, independent DBMS is very restricted, for example in its capacity to deal with qualifiers like "very" or "most".)

This may be one of the reasons why problems of natural language inference have been addressed more rigorously by researchers whose goal is to interface natural language capabilitites to more more sophisticated 'back end' systems exhibiting greater degrees of both 'intelligence' and 'cooperativeness'. The RESEDA and Automatic Yellow Pages Assistant projects (Zarri, 1981; Gershman, 1981), Palmer's mechanics problem interpreter (Palmer, 1983), and the CONSUL/CUE mail system interface (Mark, 1981) are just a few examples.

Even then, the inferential capability built into such natural language front end systems is achieved primarily by making heavy use of domain-flavoured and domain-particular rules of inference, which either try to emulate the reasoning processes of a domain expert — "... a methodology has been created from scratch for

the formalization and generalization of the reasoning used by historians in concrete situations..." (Zarri, 1981:401), — or rely on an exhaustive correspondence between the users' model of the domain and the functionality of the underlying system within the same domain — "...the handling of a natural language request from the user is treated as the reformulation (via the application of inference rules) of a description in the conceptual framework of the user to a description in the conceptural framework of the service model (...an actual function which can be executed in order to satisfy the user's request...)" (Mark, 1981:376). It is clear that the generalisability of such systems leaves a lot to be desired, not least because they do not try to distinguish between the different types of knowledge — of language, of the world in general, of the domain, of the system's task, and of the back-end system function — required for a variety of inferential tasks during the processing of a user input. In other words, it is not clear whether the range of reasoning processes, as discussed in the previous section, are implemented in such a way that they will survive transporting across domains and tasks.

Apart from the general challenge of whether we can achieve the same results in a more domain-decoupled way, therefore, the most important points for us in relation to this work in other language processing contexts are first, whether we can or should exploit the formalisms or representational schemes applied there, and second, that we will have to address ourselves, in the longer run, to some of the processing needs calling for inference that they have tackled, but we have not so far addressed: pronoun interpretation is just one obvious example. The strategy we propose in the next section is in fact an application of a version of one of the common types of formalism, the network. We will not, however, address such issues as pronoun resolution in this report, though it should be noted that the formalism adopted has been used for this and some other reference resolution problems.

———— * ————

It may be noted here that suggestions that the kinds of inference problems discussed will in some way disappear with the application of logic programming techniques are somewhat misdirected. Thus some researchers have suggested that if a natural language interface system is developed within the logic programming paradigm, and more specifically, implemented in Prolog, then the Prolog inference engine may take over the task of performing deductive information retrieval, thus disposing of the need for an explicit inference capability. But the important point about incorporating inference in a database front end is not the execution of the individual inference operations. The important issue is what knowledge is required to justify these operations and how, in general, it is to be interfaced with the other knowledge the system exploits. Thus while a Prolog implementation would preserve the modularity and declarativeness of the system's knowledge base, giving it proper connectivity with the rest of the system is what counts. This involves, of course, establishing connectivity with the back end database. Current Prolog implementations would typically be in difficulties with a knowledge base too large for main memory, but a much more fundamental problem is that Prolog *de facto* implicitly embodies the "closed world assumption" (Clark, 1978), which is likely to limit the applicability of interface systems relying blindly on the Prolog inference engine to do all the reasoning work.

# 5. Support for Inference: the Proposed Strategy

The review in the previous section indicates that perhaps the most common feature encountered in current systems is heavy dependence on domain descriptions. The price for the significant gain in efficiency of the analysis operations is the range of considerable domain/task specific constraints on the design of the language processor, which effectively limit its portability. The current project, on the other hand, started from a different premise, that a large part of the work of question interpretation, and specifically, in our terminology, its analysis, can be completely domain- and task-independent. As the work done so far suggests, the assumption that such a "neutral" analysis program, utilising general semantics and providing rich structures for the subsequent translation processes, will enhance front end portability across domains, applications and tasks, is not entirely unfounded. However, as the discussion of types of inference suggests, the language analyser must have some access to an inferential capability. Starting with an analyser which was not designed to have this, thus seems like a limitation on the functionality of the whole front end.

Clearly, we would not like to regress to the over-constraining alternative of building domain awareness into the question analysis process by supplying appropriate hooks into it via specialised lexical entries and semantic grammar patterns. The only other alternative is to connect question analysis and, more generally, interpretation with the domain-specific knowledge dynamically, which implies some modification in the philosophy and gross architecture of the system. Extrapolating from the examples in section 3, it is quite clear that at any point in the processing of a question, the module in charge may well need access to all knowledge — linguistic, world-general, domain-specific, task-related, or back end-tied — independently of which particular processor is involved.

## 5.1. System architecture

The current system, which has a very loose structure, is not capable of supporting this variety or volume of information traffic. In its present state it makes use of several distinct knowledge bases, each coupled to a particular processor; the representational schemes employed are quite different, and dependent on the types of question processing carried out by the individual components. The data channels between a component and the corresponding knowledge base are therefore very narrow and do not interleave or overlap in any way. Thus, for example, the analyser can employ linguistic and general semantic knowledge only, but has no access to, say, any of the domain or database knowledge, while the translator can only utilise domain, but not database (or, for that matter, general world) knowledge, and so forth. This model of separate channels between a system component and its corresponding knowledge base is, in fact, the simplest way of organising a front end of the general type we are concerned with (see Figure 5.A; in the figure thin boxes denote knowledge bases, thick boxes denote system processing components, and arrows indicate knowledge access paths). (Note that each knowledge base treated as a unit here can contain both linguistic and non-linguistic information.)

Meeting inference requirements, with their corresponding needs for access to a variety of different types of knowledge, then implies that the structure of the system ought to be made progressively tighter — this will offer the flexibility necessary to support the richer exchanges between a processor and a range of specialised knowledge bases.

In the first instance, we may wish to maintain the segregation between separate knowledge bases; this implies, however, that arrangements for more complex data traffic have to be made. Given that the different knowledge sources employ different representational schemes for their (linguistic and pragmatic) knowledge, a range of specialised transformation components will be required, each one controlling the flow of information between a particular

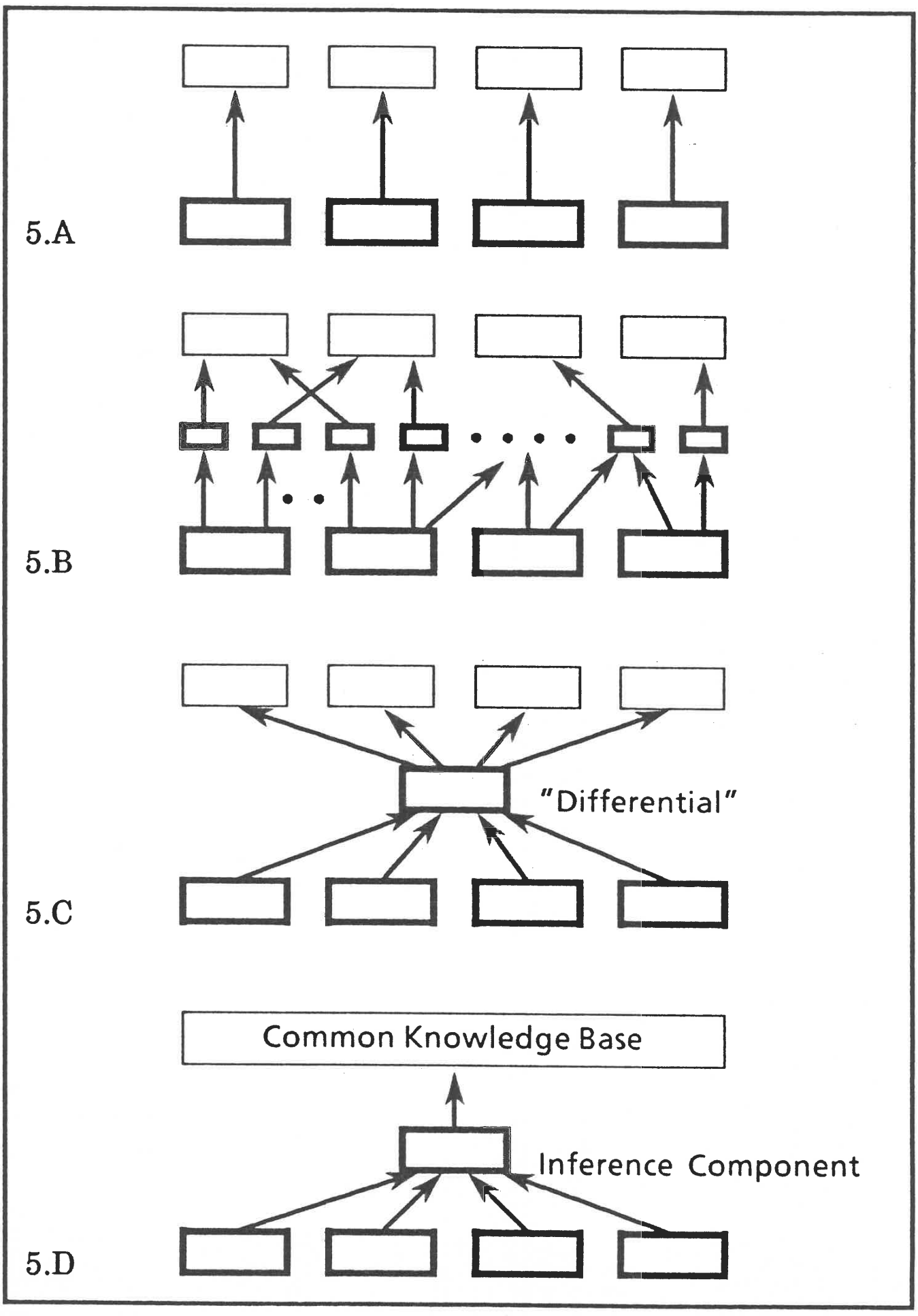
5.A
5.B
5.C
"Differential"
Common Knowledge Base
Inference Component
5.D

Figure 5

processor / knowledge source pair and capable of translation between the respective representational languages (Figure 5.B). A scheme like this involves, in the very least, a substantial amount of duplication of effort in the design and implementation of the individual transformation components, and is not entirely satisfactory, given that the whole system structure now appears to involve heterarchical control, with all its drawbacks.

An improvement on this would be a system which still maintains separate knowledge bases, but uses a "differential" component to mediate the exchange of information between a processor and the relevant knowledge source (Figure 5.C). This would act as a switch selecting both the route of the information flow and the particular method of conversion between representational schemes. The different represesentation formalisms would be retained as convenient means of encoding different kinds of fact. This model appears to fit nicely in the current system architecture, where the different knowledge representation formalisms have been selected and/or designed to support the different stages of question processing. Unfortunately, this model continues to suffer from problems associated with the previous ones. Firstly, knowledge just cannot be segmented and compartmentalised tidily: the border lines between the different knowledge bases tend to be somewhat fuzzy, and as the amount of knowledge involved increases, rational demarcation becomes more difficult. Secondly, the existence of distinct knowledge bases, even though they are intended to support different processes, inevitably leads to (occasional) unnecessary duplication of facts.

In an obvious attempt to avoid this, we arrive at the final, and in our view the best, option in structuring the system. This involves the same sequence of processing components, but these now access a common knowledge base encoding both facts about the domain and database, and also general knowledge about the world, <u>all</u> accessible to <u>all</u> modules in the front end (Figure 5.D). As implied earlier, this common knowledge base can in principle include linguistic as well as non-linguistic knowledge, though this

is not the important point in the present context: our primary concern is with the non-linguistic knowledge. In the single base model, decisions about borderline cases do not have to be made, duplication of facts is avoided, complex "differentials" do not have to be built, and, as we shall demonstrate below, traffic flow can be mediated in a natural way by a single component without the concomitant scheduling overheads. There is, however, a potential problem in conflating independent knowledge bases into a single knowledge base, in that knowledge appropriate to one processor may be mistakenly picked up by another. This potential for confusion is further discussed in section 6 below. There is also the problem of "straying".

Inference is also unified in this approach. In the first two models the inferential capabilities of the system are localised not within the linking component(s) — the individual transformation modules, or the centralised "differential gear box" — but are distributed among the original system components, essentially as additions to them. With the single knowledge base it is also rational to have a single inference engine, or perhaps inference driver, mediating the communications between the processing modules and the knowledge base, essentially by module-determined parametrisation. In fact this engine will have to control the set of inference *specialists* required to carry out the various forms of inference supported by the system: for example there would be specialists for coercive and conceptual completion reasoning, and for causal reasoning.

The attraction of having a common knowledge base is that (in theory) this does not lead to any change in the essential characteristics of the system, since the knowledge in the base is intended to support only the variety of inference procedures, and not the more basic system operations. Modularity is preserved, as are the different forms of representation which may be employed at different stages of processing. As long as the inference engine is confined to the mere mechanics of inference, whether more explicitly defined as the application of *modus ponens* or implicitly in terms of

standard, and constrained, retrieval operations on the knowledge base, there should be no problem in supporting and enhancing the operations of the four separate primary processors.

Whether, as allowed for earlier, the knowledge base should contain linguistic as well as non-linguistic information is essentially a practical question: putting some highly specialised morphological information, for example, into the knowledge base, whence it may be retrieved by very minimal inference procedures, may be more trouble than tying it to the relevant module. Certainly linguistic knowledge can in principle be put into the common knowledge base. Again, though the emphasis is on the single common inference engine, the model allows, if practically convenient, for specialised local inference procedures within individual modules, for which it might be necessary to restructure or reformat their internal information. Thus, for example, we may wish to exploit a (local) inference component to extract the presuppositions in the user's input (with view to expand the "cooperativeness" of the system in the style pioneered by Kaplan (1981) and Webber (1982)). Furthermore, modularity makes it easy to insert further components supporting wholly new functions, such as, for example, implementing sophisticated response generation strategies (à la McKeown, 1982).

But the main motivation for locating all the non-linguistic, pragmatic knowledge intended to support the whole range of inference procedures in the same, single, knowledge base, is that it makes it possible to establish connections between different types of knowledge. Of course, there is still the question of whether the varied types of knowledge in a single base can be effectively distinguished from the points of view of individual processing modules. We believe that this should not be a problem, since confusions are not likely to arise in a situation where the inference procedures are aware of the type of fact(s) they require at any point of processing, and where the different types of knowledge carry their differences, as it were, "on their sleeve".

For an inference component to carry out its task unobtrusively

and effectively, without taking over the working of the entire system, it is necessary to make sure that the propositional content of the analysed question,[4] (including all the associated inferences) is globally available to all system components on an "as needed" basis, while each module itself operates on a suitably encoded "view" of the input, carried over and translated into a "local" representation language. The permanent, or *static* common knowledge base itself would require a "neutral", and more powerful, representation formalism, i.e. one not biassed towards the operations of any particular component of the system, but still capable of supporting all of them, as discussed in more detail later. We also need a temporary, *dynamic,* knowledge base for recording all the facts inferred from, or globally true for, the specific input, in the same formalism. Apart from ready accessibility, this has the advantage of avoiding repeating the same inferences. Note that this is not necessarily in a different box, but the information is differentially marked within the common knowledge base.

Thus the front end architecture proposed here is a set of processing modules operating in linear sequence in the current way, each with an additional two-way connection to an inference component. This may be invoked at any point of the processing, entirely at the discretion of the currently active processor. The inference component acts as a buffer between the primary processing modules and the common knowledge base, and accesses and updates that according to its own internal operations. Moreover as there is only one inference component which is able to talk to all other modules as they may require its intervention and/or advice, this effectively mediates the exchange of information between them, in any direction other than that of the basic flow between adjoining modules. With the apparent alternative, considered earlier, of connecting the system components to the knowledge base via local inference specialists, the problem of scheduling intercomponent factual communication remains.

[4] This is in the general sense of "proposition" and does not refer only to the propositions of the question logic representation (as discussed in section 2.1.)

## 5.2. Requirements of a knowledge base

The advantages of adhering to well-founded principles of knowledge base organisation have been discussed at length in the AI literature (see e.g. Brachman, 1979); we shall briefly reiterate some of them below in the context of our inference task.

First, since it is not clear how far one can stretch the approach to question interpretation used in the current project — flexible pattern matching based on semantic primitives — it is clearly necessary to impose a more rigorous structure on the knowledge base than the hitherto unordered set of facts (propositions in some representational formalism) about the domain and database implicit in the pattern set. A systematic organisation of knowledge, and in particular a representational scheme which supports some sort of *classification* process, during which concepts are clustered together according to certain criteria, is perhaps the most critical requirement from the point of view of incorporating an inference component in the existing system. There is also the additional, equally critical, requirement for a *hierarchical* structuring of the concepts (nodes) in the knowledge base; however, a flexible and realistic system would almost certainly be able to perform only on the basis of a tangled, rather than simple hierarchy, and also one which admits more than mere "is-a" (subset) links as organisational principle.

There are at least two immediately obvious advantages of maintaining such a hierarchical structure. It provides a form of organisation of knowledge which is helpful for changes and/or updates, both of which will be inevitable in a system which is aimed at porting across domains. It also offers support for automatic categorisation or at least automatic characterisation / location of new knowledge. This itself is a very attractive and salient point for a variety of reasons. Firstly, in a changing environment (the knowledge base being altered, refined, and tuned for a new domain), with a hierarchical structure novel facts can be integrated in, rather than merely added to, the knowledge base. Secondly, the very fact that

the knowledge base is thus structured provides support for further procedures that will more precisely locate, characterise, and hence, classify the new knowledge. Since this process should be globally constrained by the semantics of the representation formalism, it should be relatively easy to maintain consistency as new facts are added to the knowledge base. A third advantage is that automatically deriving collocational constraints in a properly structured hierarchical knowledge base offers opportunities to perform deduction about certain relationships between concepts, thus furnishing a natural framework for carrying out some coercion inferences (for free), and providing considerable support for "higher-order" inference procedures (processes, exemplified in the AI context by the "classification" and "realisation", of Brachman (1982) and Schmolze (1983)).

The second requirement for the knowledge base, though one that is extremely hard to meet, is that it maintains some principled, or at least visible, distinction between domain-independent and domain-dependent facts. The latter then would have two important, somewhat different functions. One is supporting the transition from an input logic representation to a query representation, i.e. replacing natural language predications by domain ones; the other function is to support the derivation of further predications between elements.

For example, starting from *Which are Blake's stores in Market Place?,* there is a process whereby the natural language predication is translated into something like

(&own Blake &shop Market_Place).

(The translation process may involve some amount of coercive / completion inferencing, but this need not concern us here — we need only note that some domain-specific knowledge was used for the derivation of the domain predicate &own.) The question now is: what does it mean for Blake to be an agent in an ownership relationship with a shop for an object and the Market Square as location? The answer lies in the expansion of the representation

of the domain predicate to spell out its "implications".

Thus the example predicate implies, at least in the Green Hills domain, that the statements in Figure 6 can be assumed to be true.

```
(is_parcel X)
(eq (&luc X) 566)
(eq (&str_name X) Market_Place)
(is_property_owner Y)
(eq (&surname Y) Blake)
(&own Y X)
```

Figure 6: Explicit propositional content

In practice, some of these statements will be inferable by coercion (&shop is clearly coercible to &parcel without too much trouble, using both general semantic knowledge, and the categorisation hierarchy); some of them ought to be pre-specified as *value* words (e.g. the land-user code for a shop: "566"); and some of them ought to be interpretable as such, together with their generalisations in the concept hierarchy ("Market Place" and "Blake" may fall in this category). The remaining statements will be instantiated on matching a general inference rule for &own, and all of the derived predications will be recorded in the common knowledge base as dynamic facts associated with the current input.

There are at least two areas where the distinction between domain-independent and domain-dependent knowledge, if it can be maintained, is important. Firstly, it has been demonstrated (e.g. Palmer, 1983), that it alleviates the control problems (see further below); secondly, it would make the processes of transporting the system to new domains and databases and acquiring new domain-specific inference rules more tractable.

## 5.3. The knowledge base

We propose that the common knowledge base is implemented

using the techniques and mechanisms for knowledge representation and manipulation developed by Alshawi (1983). Before justifying our choice and discussing the application of Alshawi's work to the task of constructing a knowledge base to support inference procedures for our database purposes, and before characterising the inference procedures themselves, we shall present a brief summary of the most salient features of Alshawi's knowledge representation scheme. For more details the reader is referred to the original thesis (ibid.).

### 5.3.1. The MEMORY representation formalism

MEMORY is a knowledge representation system which utilises the idea of a semantic network. It is closely modelled on recent work in knowledge representation and in particular Brachman's KL-ONE (Brachman, 1978) and Fahlman's NETL (Fahlman, 1979). While MEMORY is not claimed to be a functionally complete knowledge representation formalism in the sense in which Brachman's Structured Inheritance Networks strive to be (Alshawi's aim was not that of satisfying an abstract design aim of completeness, but one of developing a knowledge representation scheme capable of supporting a non-trivial linguistic task), its implementation has been strongly guided and influenced by efficiency considerations, leading to a system which is capable of making the task of heavily knowledge-based language processing computationally tractable.

The result is thus a simple, yet quite powerful, representational formalism. Like other semantic networks, the knowledge base contains nodes which represent *concepts,* themselves primarily characterised as having *roles.* Anything at all — an object, a property, a relationship, etc. — can be viewed as a concept. However, MEMORY employs a much simpler ontology of links between concepts than other semantic net systems, since no attempt is made to prejudge issues and classify *entities* rigidly into concepts and roles. The idea is that it ought to be possible (and easy) to view an entity either as a concept having its own roles, or as a role in the

descriptive definition of other entities which may be concepts or roles. This natural way of having "multiple views" of entities offers opportunities both for the treatment of concepts as primitives which are then used to create composites, and for an analytical definition of complex concepts.

The typical MEMORY knowledge base can be viewed as a tangled hierarchy of small, overlapping frame-like structures, each of which represents some concept in terms of other concepts represented by other nodes elsewhere in the network. The organisational relationships — i.e. the links between the nodes, defining the global structure of the knowledge base — map onto two different types of assertions: *specialisations* and *correspondences*. The interpretation of specialisation statements imposes a hierarchical ordering of the concepts known to the system; the interpretation of the correspondence statements results in further classifying the associations between these concepts essentially in an analogical manner: the relationship that concept **A** has to concept **B** is like the relationship that concept **C** has to concept **D**, or **A** stands to **B** as **C** stands to **D**, where 'like' or 'stands to' are primitive notions.

Arguments to assertions like those illustrated below for the database access task in the Green Hills domain, can be of any kind. Some, e.g. `own1`, `agent` and `&owner`, are atoms denoting concepts already known to the analysis and translation components, while others, e.g. `PropertyOwner`, denote new concepts of general use to the inferencer.[5]

A specialisation assertion, such as

```
(Specialisation: PropertyOwner of LegalOrganisation)
(Specialisation: PropertyOwner of HumanIndividual)
```

states that, depending on which way the classification hierarchy is traversed, a `PropertyOwner` may be viewed as a specialisation of a `LegalOrganisation`, and, conversely, a `LegalOrganisation` may

[5] `PropertyOwner`, as we are writing it, would be written as `property/owner` by Alshawi himself. We prefer the former because it makes it clear that it is a single atom simply being presented in a (humanly) readable form.

be viewed as a generalisation of a **PropertyOwner**. The example clearly illustrates why the specialisation hierarchy is tangled.

A correpondence assertion, for example

```
(Corresponds:    OwnsAgent to own1 as
                 agent to VerbStatement)
```

causes certain associations to be established in the system's hierarchy of role-concept pairs, thus further augmenting the classification of the relationships between the known concepts. Starting from the most general association between two entities — that defined by [role, thing] — the example specifies that the association relationship between **OwnsAgent** and **own1** can be regarded as a refinement (where refinement is analogous to, but not the same as, specialisation) of the one between **agent** and **VerbStatement**. Note that even though it is easy to interpret this correspondence statement as "**OwnsAgent** fills the **agent** role of own1 frame", MEMORY does not impose any rigid distinction between "concepts" and "roles", thus allowing for a relatively easy re-interpretation of the ontological status of a MEMORY node as the knowledge base is used. In fact, a more faithful "semantics" of a correspondence assertion can be rendered by drawing an analogy with a specialisation assertion, where the arguments are implicit, and yet fairly well defined, as the relationships holding between the two pairs of concepts. Even though in general the range of relationships between two entities can be potentially quite large, ambiguities are avoided, as the second pair of entities in the assertion defines a context and established the particular angle from which the relationships ought to be viewed, as well as the direction(s) along which they are refined / generalised.

Using the MEMORY representational scheme, it is possible to encode a variety of different types of knowledge, and knowledge at very different levels of abstraction.

Thus in relation to the needs of our different front end processor components, one section of the knowledge base would contain statements regarding the concepts required to describe the

```
(Specialisation: TaggedStatement of Statement)
(Specialisation: VerbStatement of TaggedStatement)
(Corresponds:    agent to VerbStatement as
                 TagRole to TaggedStatement)
(Corresponds:    location to VerbStatement as
                 TagRole to TaggedStatement)
.....
```

Figure 7: Descriptions of analyser constructs

dependency structures that the analyser constructs and manipulates. For example, the assertions in Figure 7 define a portion of the network concerned with the representation of verb/argument structure. Another portion would presumably have to indicate some of the relationships between and properties of quantifiers, both informal and formal.

Moving from the more general, linguistic, concepts to the more specific domain world of the Green Hills database, further specialisation and correspondence assertions will specify the relationships between the types of structure handled by the analyser and extractor and their interpretation in the domain. The domain description would thus include domain concepts (objects and predicates), their specialisations to individual objects and facts, the ways in which the structure and content of the meaning and logic representations of the analyser and extractor can be related to these objects and facts, and any constraints that the domain may place on arguments of the domain predicates. The fragments in Figure 8 illustrate all of these.

It is not necessarily easy to draw the line between world-general and domain-specific knowledge, and indeed the formalism allows for a natural transition from one to the other. Thus domain-specific knowledge is most conveniently expressed as specialisation assertions whose 'roots' are, naturally, somewhere in the domain-independent part of the knowledge base. For example, while the general concepts of 'Rule' and 'Mapping', defined along the lines

```
(Specialisation: &own of own1)
(Specialisation: &owner of PropertyOwner)
.........
(Specialisation: own1 of VerbStatement)
(Corresponds:    OwnsObject to own1 as
                 object to VerbStatement)
.........
(Specialisation: OwnsObject of &property)
(Specialisation: OwnsAgent of &owner)
```

Figure 8: Connecting the analyser output with the domain

```
(Specialisation: Mapping of Statement)
(Specialisation: Rule of Mapping)
(Corresponds:    LhsRule to MappingSource as
                 RhsRule to MappingTarget)
.........
```

Figure 9: Fragment of a structural description of "Rule"

illustrated in Figure 9 may be used for a variety of different purposes, further memory assertions, illustrated in Figure 10, will encode particular facts related specifically to the world of the Green Hills database. These fragments state, in essence, that Land User Code, which is a property (attribute) of parcels, defines a mapping from the range of possible types of properties built on the parcels to numbers.[6]

In a similar fashion it is possible to describe a generic relation, DbRelp, as understood within the framework of relational databases, and then specialise this description to the particular relations in the Green Hills database. Thus assertions about the

[6] the MEMORY representation formalism allows certain additional features (flags on the assertions) which will make it possible, in this particular instance, to further specify that the mapping is unique (i.e. one-to-one). Since these do not affect the power of the formalism, nor bear any immediate relevance in the context of the inference task, we shall not concern ourselves with them here.

```
(Specialisation: LUC of Number)
(Corresponds:    LUC to &parcel as
                 PropertyOf to PropertyPossessor)
........
(Corresponds:    LUCCode to LUC as
                 MappingTarget to Mapping)
(Corresponds:    LUCPropertyType to LUC as
                 MappingSource to Mapping)
(Corresponds:    &hotel to 151 as
                 LUCPropertyType to Luc)
(Corresponds:    &school to 248 as
                 LUCPropertyType to Luc)
........
```

Figure 10: Structural description of a "Land User Code"

concept DbRelp — for example that a relation encodes a number of relationships (RelpStatement) between objects (RelpDbEntity), and is defined as a set of entries over a range of columns — can be "instantiated" to specific domain predications as illustrated in Figure 11.

It must be emphasised again, here, that though particular types of knowledge have been related to particular processors in the examples just presented, there is no implication that the association is an exclusive one: any form of knowledge can in principle be exploited, directly or indirectly, by any processor, though in practice a specific form of knowledge and portion of the network is likely to be more heavily used by one processor than another.

### 5.3.2. The MEMORY knowledge base

There are several good reasons for adopting Alshawi's representation framework.

Firstly, it offers a motivated and tested knowledge representation scheme, which allows the encoding of a variety of different types of knowledge. The comprehensive description of the task,

```
(Specialisation: OwnerRelation of relation)
(Specialisation: OwnerSurname of column)
.........
(Specialisation: OwnerRelp of DbRelp)
(Corresponds:    OwnerRelation to OwnerRelp as
                 RelpRelation to DbRelp)
(Corresponds:    OwnerDbEntity to OwnerRelp as
                 RelpDbEntity to DbRelp)
(Specialisation: OwnerDbEntity of PropertyOwner)
.........
(Corresponds:    RelpOwns to OwnerRelp as
                 RelpStatement to DbRelp)
(Specialisation: RelpOwns of &own)
(Corresponds:    OwnerDbEntity to RelpOwns as
                 OwnsAgent to &own)
.........
(Corresponds:    OwnerNameEntry to OwnerRelp as
                 RelpEntry to DbRelp)
(Corresponds:    NameValue to OwnerNameEntry as
                 RelpEntryValue to RelpEntry)
(Corresponds:    OwnerSurname to OwnerNameEntry as
                 RelpEntryColumn to RelpEntry)
(Corresponds:    NameValue to OwnerDbEntity as
                 DbEntityName to NamedDbEntity)
.........
```

Figure 11: Description of the Owner relation

domain and database of Green Hills we have drafted would be too bulky to include in this document; the examples of Figures 7–11 however, illustrate how this single formalism could represent a wide range of facts in a common knowledge base, making explicit the connections between linguistic, domain and database concepts. At the same time, the separation of domain-independent and domain-dependent knowledge required for transportability (cf. 5.2) should be fairly transparent and relatively conveniently achieved. Thus one can imagine separate declarations (say as physically separate files) of the different kinds of information required by the natural language front end, which, when executed by the knowledge representation interpreter / compiler will get merged into one network; in the process domain-imposed constraints on predicate fillers may override the common-sense defaults for example, or an explicit, short-cut, link may be established between a particular linguistic object (e.g. a compound nominal belonging to a specified semantic pattern) and the way it is encoded in the database (e.g. a named column in a named relation). Moving the front end to a different environment then would only involve (at least in principle) replacing the old application-dependent knowledge declarations with new data files, and recompiling the whole knowledge base from scratch.

Secondly, Alshawi's knowledge representation scheme offers a proper hierarchy, together with a mechanism for the specification of node *rolesets*. Thus, as the examples in the previous subsection demonstrated, the knowledge representation scheme makes it relatively easy to specify domain-imposed constraints on the arguments of domain verbs. As a result a considerable part of the information required to carry out some inference tasks — namely coercion and conceptual completion — can be encoded as a matter of course. The scheme also offers an easy way of incorporating additional information about relationships between arguments of domain predicates, thus making reasoning about e.g. part/whole, mass and measures, and so forth, somewhat more tractable, while still retaining its original simplicity and efficiency. Similarly, there would be no problems in specifying further relationships between

domain predicate arguments, inferable from the predicate itself and refining (or "spelling out") its meaning in the context of the database (as discussed in 5.2). It would also be quite easy to upgrade the knowledge base continuously, adding new facts on the fly during the course of the system's operations, as more of the propositional content of input text(s) is made explicit and so is available for incorporation in the dynamic part of the knowledge base.

Thirdly, Alshawi has developed techniques for the efficient representation of, access to, and manipulation of, the knowledge base which support unified memory and context mechanisms that can be applied to many common language problems. They make it easy in particular to take many different types of context factor into account in discourse operations. While it is not entirely clear exactly how much computation of contextual reference within the sentence is required for inference in the database case, the efficient implementation of knowledge access mechanisms is clearly one of the more important prerequisites for upgrading the functionality of the front end.

Fourthly, though the MEMORY system has only been applied to, and tested for, one task, namely data capture and database creation from text paragraphs, a philosophy very like the one proposed in this paper — that of all knowledge being available to all system components at all times — has been demonstrated, in this test, to be both feasible and successful for sophisticated language analysis problems.

Finally, the specific task to which MEMORY was applied, has much in common with our own, for example in the need for an explicit and elaborate representation of the data model.

We therefore believe that a common knowledge base, utilising a uniform underlying representation of the type of Alshawi's, should contribute significantly to a more powerful front end system, without a corresponding increase in complexity.

## 5.4. Inference within the MEMORY framework

There is nothing inherent in the representation formalism described above which forces the adoption of a particular way in which inferences ought to be drawn in the MEMORY system. (In fact, this was an additional point in favour of adopting a relatively neutral representational scheme for our task.) On the other hand certain features of MEMORY, both in its philosophy and implementation, make it a convenient starting point for the development of an inference component for our front end system.

Exploiting Alshawi's framework for our purposes, the action of drawing (simple) inferences is essentially one of establishing new links between entities in the semantic network — a process not entirely dissimilar to automatic categorisation. As noted above, this can be usefully exploited for effecting and controlling some inferential processes, namely coercion and conceptual completion.

Access to, and retrieval from, MEMORY utilise a marker propagation model enhanced with a novel indexing technique for efficient searches in the knowledge base for entities describable by a wide range of constraining information. Thus what amounts to simple inferences can, in effect, be achieved by a combination of standard retrieval operations. Consider an example we have already encountered in 3.3:

*Are there any schools in Ward 2?*

We shall sketch below the retrieval processes required to achieve the coercion from *schools* to *properties with LUC 248* required for the correct processing of the question.

In section 5.3.1 above we presented a fragment of the knowledge base describing the encoding of types of building associated with the parcels on which they stand. This fragment introduces the concept of a Land User Code by its structural description as a mapping from a finite, exhaustively defined range of building functions (including, in particular, &schools), into associated unique numbers. The fragment also indicates that there is a chain of roleset specifications, connecting the domain concepts of &school

and &parcel. Loosely translated into English, the example assertions jointly define LUC as a PropertyOf &parcel, while &school, &garage, &hotel, etc. all generalise to LUCPropertyType, which itself is a concept allowed to instantiate the MappingFrom "role" of LUC.

Thus a retrieval operation like "start from an entity and recursively traverse roleset specification chains backwards until you encounter another entity, marked in a certain way" (in this case — a domain entity capable of participating in the domain predicate &location) will easily recover the concept of a &parcel. The process amounts to starting from the translator derived predicate

(&location &school Ward_2),

and establishing a new link between &school and &parcel, the very existence of which completes the coercion with a result, explicitly recorded in the dynamic part of the knowledge base as

(is_parcel X)
(eq (&luc X) 248)
(&location X Ward_2) [7]

(cf. the detailed discussion of the example in 5.2.1).

In this way the standard retrieval operations available as part of the MEMORY representational scheme can be used to implement some inference procedures. Of course, not all types of inference, as discussed in section 3 above, can be accomplished in the same way. But we believe that the power of the formalism allows extensions to cover these other types along such lines as the following.

In particular the scheme should be capable of supporting causal reasoning. As the CONSUL project demonstrated (Mark, 1981), individual causal inference rules can be encoded and inference chains constructed, without losing generality, within a hierarchically organised knowledge base which contains structured definitions of

[7] This illustration only indicates the content of the propositions recorded in the knowledge base, and not their format. Also note that further inferences, of essentially similar character, will be required to establish the exact domain interpretation of "Ward_2".

individual concepts. In Alshawi's scheme individual causal rules are conveniently represented simply as specialisations of the general concept of mapping (as exemplified in Figure 9.) With rules encoded in the knowledge base in this way, the construction of causal inference chains (in the sense described in section 3.3) is triggered when a predicate without a direct domain interpretation is encountered, for example `wealth` derived from the question *How rich is Colonel Cribbin?*. Appropriate rules are applied (with relevant intermediate results recorded) until a related predicate, which is directly interpretable in the domain, is generated. (The example is too long to illustrate in full here.)

Furthermore, as the examples in 5.3.1 suggest, the MEMORY formalism supports the explicit construction of structures in the knowledge base that resemble frames, which themselves can be used to guide and control potentially explosive causal chains (an approach initially introduced in SAM (Cullingford, 1978) and subsequently applied to the database interface task in KNOBS (Pazzani, 1983)).

Thus it seems that adopting a representation formalism which is deliberately neutral with respect to both the domain and task, will, if anything, facilitate the addition of an inference component operating on a common knowledge base to augment the operations of the individual front end components.

## 5.5. Towards a less explicitly representational model of computation

Notwithstanding the global availability of the common knowledge base with its general purpose inference component, there may also be a proper place in the system for local inference procedures applied within a component to information directly available to it and represented in its own particular form. For example, the current system employs semantic pattern matching in the translator to perform the mapping from English to data language, as well as to infer implicit relationships between domain objects, while the

convertor uses a different style of semantic pattern matching to establish the way in which domain objects and their properties are modelled in the particular database. In both cases it is necessary, for example, to make explicit some implicit relational information. This type of restructuring and expansion, especially if extended, can legitimately be labelled inference, but is essentially local, and therefore can legitimately be retained within individual processors.

In general, these procedures must be capable of detecting and suppressing redundancy and checking for contradictions; and in dealing with the latter, and with incomplete data, they must be capable (to put it informally) of jumping to conclusions. Jumping to conclusions may lead to contradictions later on (encountered either by this processor, or subsequent ones), so the system's components must be able to revise their assumptions. This is a well known problem for natural language processing systems, which manifests itself as *ambiguity at all levels,* or in our case stages of processing.

Various system architectures have been proposed for solving this particular problem, or at least for alleviating the burden of unnecesssary computation, but all of them suffer from drawbacks (see Boguraev, 1984). The current system attempts to reduce the amount of blind work too early in processing by using a kind of a "meta" level of representation of an ambigous object, where the single structure delivered to the next processor encodes, in some sense, all possible detailed interpretations of the object. For example, as the structure and meaning of a compound noun can generally only be determined on the basis of domain constraints, the compound is passed from analysis to translation as a cluster of representations for the individual nouns, after minimal preprocessing to rule out only a very few obviously bad syntactic alternatives (see Tait, 1983 for details). However, as Sparck Jones (1985) points out, even a solution like this suffers from some problems, and so has ramifications for the entire system architecture.

We believe that the system ought to take such an uncommitted approach to its logical conclusion, as a general strategy. Given

that individual processors cannot always, because of their own intrinsic limitations, use all the knowledge of all types which is in principle available to them, it is essential that they should not overcommit on the basis of inadequate information. That is to say, we propose as a fundamental design principle that individual modules should not overspecify their outputs. However, while the current system does not go beyond producing (meta) representations which are uncommitted with respect to ambiguities they have failed to resolve, when there is an inference component this will have to be supplied, if it is to be effective, with (the locally deducible) constraints on the representations that the ultimate interpretation(s) must satisfy. These constraints associated with a processor's output, together with the relevant propositional information about the user input held in the knowledge base, will be exploited by subsequent modules to add more constraints, which will ultimately allow the required in-depth interpretation of the input. Of course to be realistic, we have to allow for the possibility that even a delayed decision will be wrong and that some means of error recovery will have to be supplied (see further 6.5).

It may seem that insisting on a globally available knowledge base, on the one hand, and staying agnostic during the interpretation and especially analysis process, on the other, are mutually inconsistent: why supply the information which could solve interpretation problems and then fail to take advantage of it? But this would not be sensible given the general modular structure and sequential processing of the front end, since each module contributes to the input interpretation as well as exploiting its predecessor's output. There is also the more general point that inference itself is not guaranteed to be correct, and there are many problems connected with it, like knowing precisely that inference is required, knowing precisely what to do when it fails and so on, discussed in the next section. Adopting a conservative strategy that allows for underspecified rather than demanding "overspecified" component output, and hence being satisfied with the more obvious inferences that can be comparatively naturally drawn given the information

available to a particular module, would seem to reduce some of the problems of inference management. It of course does not eliminate them altogether. But there are advantages in not overspecifying outputs. Thus if the system has failed to detect an inference trigger, it can still proceed, as long as the "guilty" module does not overcommit itself to a particular intermediate representation; similarly, failure to complete a reasoning process would not cause the whole system to grind to a halt, and as long as the partial results of the inferential procedures have been recorded into a set of constraints to be satisfied subsequently, not a lot will be lost.

A similar approach can be adopted in dealing with the issues of intercomponent communication. While we postulated that it would be convenient (and ultimately necessary) to allow for some communication between the language analyser and the database conceptual schema, the current status of data modelling means that we may not be able to do this in a sufficiently motivated way. Ensuring that the set of linguistic analyses encodes everything that the analyser managed to "deduce" locally, and does not prematurely close some analysis paths, is the only motivated way of maintaining the system integrity across a range of domains and applications.

Smoothly integrating an inference component into a database front end of the general kind with which we are concerned, and indeed into a modular front end even if this does not place so much emphasis on domain-independent operations, would thus seem to lead to a less explicitly representational model of processing than has been hitherto favoured. The strategy just outlined fits with the view of processing as constraint-based which has been put forward in other language processing contexts by, for example, Hobbs (1983) and Marcus (1983).

# 6. Problems for an Inference Component

This section focuses on some of the more critical, specific problems connected with the incorporation of an inference component of the type proposed into the existing system implementation, and concludes with prescriptions for the work required to develop a system based on our suggestions. The section discusses problems related to the functioning of the inference component itself, as well as to problems likely to arise in connection with, or as a result of, integrating such a component into an already existing system. We cannot offer detailed solutions to these problems in advance of some actual system building, but will outline the types of strategy we would attempt to follow.

The list below is not intended to be an exhaustive one. Rather, within the framework and system architecture proposed in this report — that of a separate inference component performing different forms of inference (e.g. coercion, conceptual completion, causal reasoning — see section 3), utilising a common knowledge base (see section 4.2 and 4.3), and accessible to the individual front end processors (section 4.1) — there appear to be certain issues related to the incorporation of such a component into the existing system structure which are particularly critical to the system building task. These include questions like when and how to perform a specific inferential task, how to control and constrain the inferential processes, and how to recover after errors. What follows is a more detailed discussion of these issues.

## 6.1. Detecting the need for inference

How we detect the need for inference is directly related to the question of what should trigger the invocation of inference procedures. The simple answer, "do it anyway" is not sufficient: unless we can perform some diagnostics on a particular inference trigger, we will be at a loss as to what type of inferential process to invoke and with what particular goal. There is, similarly, very

little point in waiting for an obvious failure at some stage of the input processing in order to realise that some inference process might have been usefully invoked earlier on.

A better alternative would be to determine a set of triggers, which react to certain characteristics of the question represention, or to phenomena encountered during the processing, and invoke appropriate specialists from the inference component (as indicated in sections 3 and 4, there are separate specialists for the different forms of inference — coercive, causal, etc.). For example some linguistic ambiguities like modifier placement encountered during the operation of the analyser may require resolution calling for inference. These ambiguities should trigger communication between the analyser and the relevant domain information encoded in the common knowledge base. Similarly, the presence of an incomplete or malformed domain verb structure, violating slot-filler preferences, in the translation phase ought to invoke conceptual completion or coercion, as appropriate, while a complete failure to map onto a domain verb may initiate a chain of causal reasoning aimed at suitably restructuring the question.

These are cases where the response to a need for an inference is driven by the processor that detected the need. In those cases where it may seem that invoking inference to resolve an ambiguity or clarify an implicit relationship may be delayed until some further stage in the processing of the input — such as compound noun analysis in the current system — we ought to be guided by the principle of inertia. However, we have no a priori feeling for practical criteria to decide between those inferences which may be delayed and those which have to be attempted at once.

## 6.2. The nature of the inference specialists

Given that we wish to respond to specific inference needs with appropriate inference specialists, what in detail should these specialists look like? It seems that in many cases the particular trigger which signals the need for an inference also suggests the type of

inference specialist which has to be invoked in order to perform the required inference. Moreover in the light of such a direct relationship, the nature of the triggers also suggests the types of more fundamental system functions which can be used to implement the inference specialists.

Thus we believe that the complementary utilisation of semantic pattern matching between natural language words and domain objects on the one hand (as employed in the current system), and automatic categorisation on the other, will solve some of the coercion problems. In addition, we demonstrated (in section 5.4) how coercion inferences can be implemented within the standard retrieval operations of the MEMORY knowledge base. Furthermore, as the current database project has demonstrated, general and flexible semantic pattern matching on its own seems to be particularly well suited to performing that inferential reasoning required for inferring the conceptual role of objects referred to by the use of 'unorthodox' (or even 'unnatural') natural language words and expressions.[8] In other words the type of formal operation, pattern matching, and its application to information like that supplied by semantic primitives, can be applied more extensively and in a manner establishing indirect rather than direct relationships.

This type of rule can be similarly extended in its application to world rather than linguistic knowledge. Thus it should be possible to connect common situations in the real, and/or domain, world to support causal reasoning whose goal will be to restructure the input question as a, say, state or action implied by it, and which is answerable from the database.

Certain classes of inferences, especially DBMS related reasoning within the convertor, seem to require the incorporation of a planning component, for example for the computation of averages or generation of sorted lists (see 3.4). While we do not propose the incorporation of a full-fledged planner into our system, if, as suggested by Rosenschein (1981), planning systems and deduction systems should not be regarded as distinct, it would be in principle

[8] Current work in Cambridge by D.M. Carter reinforces this point

possible to embed some planning resources within the "ordinary" inference component.

## 6.3. Process control

As we discussed in detail in section 3, it may be quite common for a question to exhibit mixed properties calling for distinct inference specialists. Clearly, this introduces the question of how inferences are to be scheduled.

We feel that an attempt to impose a static order in which the different inferential processes ought to operate is, possibly, misguided, or at least, is too difficult to attempt (even though, obviously, partial orderings may exist — for example it makes sense to try coercion before conceptual completion).

Fortunately, it also seems that it is not necessary to attempt to impose such an order. It would be desirable, and, as long as the separate inference specialists are designed to operate completely independently from one another, we believe it would also be possible, to allow the specific context to determine the order of application of specialists. Thus the flow of processes within the whole inference component would largely depend on such issues as the structure of the question, and the different points in its processing at which uncertainty arises.

We believe, moreover, that the very difficult problem of dealing with interactions between specialists, calling for conflict resolution, can be reduced by systematically applying the principle of inertia. But of course all of these assumptions will have to be tested in practice.

## 6.4. Constraining inferences

The inference component itself faces two fundamental issues of control. First of all is the question of the 'direction' of the inferential reasoning, i.e. in what direction do we work, towards premises or towards conclusions. Next come issues of constraining allowable inferences so that the process does not become explosive.

These issues too depend on the particular type of inference, and can be handled in a variety of ways. For example, since conceptual completion simply invokes role filling, it should not be too explosive, especially as we have a well-defined indication of when the process ought to terminate. Further constraints on the process are imposed by the specifications, in the knowledge base, of rolesets for domain objects, together with preferences and default values (as discussed in 5.3.1). Coercion requires the progressive relaxation of preferences on slot-fillers: there may be certain problems with the concept (and thus implementation) of "progressive relaxation", but the constraints imposed by a taxonomically structured knowledge base should limit explosions as relaxation takes into account the directionality of the generalisation / specialisation hierarchies and the domain-imposed concept clustering.

Causal chains are more of a problem. Firstly, this is where the issue of "direction" is critical, because in reasoning about cause and effect, the inference engine has to decide which is going to be more useful for the subsequent processing: the premises for a given fact assumed to be true or, alternatively, the conclusions deduced from it. In addition, there is the complementary issue of the mode in which inference is to be performed: it can be that of *causal* (in the style of Rieger (1975) or Cater (1981)) or *diagnostic* (cf. Wilks (1975) and Sidner (1979)) reasoning. Experience with systems where long causal chains have been constructed on a "just in case" basis, and a vast number of useless inferences have been drawn in the process, suggests that *demand-driven* inference control may be a more viable alternative. Demand-driven control also fits better in a framework where we postulate a set of inference triggers, sensitive to the type of inference process relevant to the particular problem, and capable of postulating a goal for the process before actually invoking it.

Clearly, the longer the inference chains, the greater their potential explosion factor. We may be tempted to try and reduce the length of chains and hence range of inferences by enforcing some tight, and particularly domain-specific rules, but there are several

problems with this: determining what would be a tight domain-specific rule, its acquisition, and coping with the consequences of having to interleave tight (i.e. specific) inference rules with more relaxed, general ones (the latter indeed presents control problems on its own). However, as we noted in the previous section (5.4), we hope to be able to impose organisational structure on the inference rules themselves (in the form of frames, or scripts — cf. the CONSUL project) and thus to both guide and control their application. Alshawi's knowledge representation mechanism would allow this; the practical problem would be the management of such rule clusters across changing domains and databases.

## 6.5. Recovery from wrong / irrelevant decisions

After an inference component has jumped to what turns out to be a wrong conclusion, it ought to be able to back up, replace the incorrect assumption with an alternative, and continue inference from there. We thus have three subproblems: recognising that an error has been committed, locating the error, and correcting it. In the database context even the first of these is not entirely trivial: for example, a failure to match the natural language description of an object referred to by the user against the conceptual schema may mean either a genuine user blunder, or simply an inadequate, i.e. incomplete, schema.

On the other hand, this problem may well turn out not to be really daunting, if we adhere to the principle of inertia and if we take extra care that inference is invoked at the right time (see 6.1.1.). This last point clearly becomes even more critical in the current context, because there is no other way of analysing e.g. *Which 541 properties have parking lots* properly, since failure to invoke pragmatic inference during the analysis phase will simply lose the correct, intended, interpretation of the question though there will be no point in its subsequent processing where there is any reason even to suspect that an error has occurred.

In general, however, the adoption of the non-committal princi-

ple means that modules will not be forced to operate on the basis of insufficient data, thus alleviating the problem of backing out from a wrong decision made earlier in the interpretation process.

## 6.6. Inter-component communications channels and protocols

The issues discussed so far in this section concern the internal functioning of the inference component. The incorporation of this component into an already existing system presents a different type of problem which, in its entirety, would be too complex to tackle in the short term.

We hope, however, that the architecture we propose — a linear sequence of system processors, with an inference engine situated between them and a common knowledge base — may be of some advantage in attempting to restrict the ways in which different specialists could coordinate their actions. The architecture may limit the control regime possibilities for the inferencer and, also, make them individually simpler.

Ideally, and in the long term, a solution may be offered by a logical extension of the Interactive Determinism principle (Boguraev and Briscoe, 1984). In brief, this principle states that the interpretation of any utterance in any context, up to any level of understanding, may proceed deterministically, because if an ambiguity arises at any point of the processing, information required to resolve it is available at some other level. If we knew enough to define an appropriate system decomposition into modules, the points in analysis where uncertainties would arise, the data which would resolve them, and the modules which would have access to this data, we could build a system with a fairly simple architecture, pre-defined channels of communication, and no need for complex scheduling and process coordination.

As for the short term, we still have to face a general point: the integration of an inference component will require certain changes in (or, rather, extensions to) existing modules. Thus, for example,

in order to be able to benefit from the incorporation of an inference component, the natural language parser will require a more flexible control scheme, where external decisions about e.g. collocational interpretation of a lexical phrase ought to be used to bias its syntactic recognition algorithm. It is quite likely that all of the current system components will require some revision in order to allow for as smooth as possible an integration of the proposed inference capability.

# 7. Conclusion

In this report we have considered the provision of inference capabilities for the kind of database front end we have developed in Cambridge, in which database-independent processing is maximised for the sake of front end portability. We have examined the needs for inference which arise in question interpretation, and have illustrated forms of inference required to meet these needs. We have focussed in particular on the role of inference exploiting non-linguistic information, and further, on inference exploiting domain-dependent rather than domain-independent pragmatic knowledge. We concluded, after an examination of possible alternatives, that the best way of enhancing our system to provide the necessary inference capacity is through the use of a single integrated knowledge base accessed through an inference engine which is context-sensitive to the calling processor module. The engine itself will have specialists for the various forms of inference required.

Our specific proposal for the knowledge base is to use Alshawi's network formalism, which lends itself to a very flexible and hospitable characterisation of the many different kinds of knowledge required to support the inference needs of the various processing modules. The connectivity of the knowledge base allows us to propose strategies for the conduct of inference based on principles of inertia and least commitment, implying a constraint-based rather than representational model of computation; thus the content of a question interpretation underspecified at one stage of processing can be related to the knowledge for further specification by knowledge base-supported linking. However we have recognised that there are many detailed problems of implementation which will require further work.

The most obvious comparison between the scheme proposed and others is with the use of KL-ONE for CONSUL (Mark, 1981). As noted, the MEMORY formalism we propose to use is less exigent, and therefore in some sense more powerful, than KL-ONE (see, further, Alshawi 1983). Whether it will prove more effective

in practice, and specifically more effective for our prime purpose, that of handling both domain-independent and domain-dependent knowledge, remains to be shown.

It is obvious that, in general, our identification of inference needs is not particularly original, and that, for example, the idea of causal reasoning is no novelty in automated language processing. Our treatment of inference is constrained by the form of front end architecture that we have adopted, and specifically by the emphasis on domain-independent processing in the analyser and extractor modules. The issue underlying the report is therefore whether anything at least useful, and preferably interesting or even novel, stems from our attempt to handle inference problems within our front end architecture.

Given the other claimed advantage of our front end, namely portability, it would be at least helpful to know that established kinds of knowledge representation and inference technique could be exploited within our framework. However we hope that, though our approach has been constrained by our architecture, something of more general interest for those dealing with inference needs, and of relevance to other research, will be learnt from the strategy we propose. But this will clearly have to be established by a successful implementation and test. The many pious beliefs expressed in the report have to be endorsed by practical evidence.

What we have not considered here are questions about the provision of more adequate data models, and especially conceptual schemas; about the provision of general dialogue capabilities; and about knowledge acquisition. However what is needed as a first priority is trying the proposed scheme out for one application, and then evaluating our front end for real portability by trying it on others.

# References


Allen,James(1982) 'Recognising Intentions from Natural Language Utterances' in R.Berwick and M.Brady (eds.), *Computational Models of Discourse,* MIT Press, Cambridge, Mass, pp.107–165

Alshawi,Hiyan(1983) *Memory and Context Mechanisms for Automatic Text Processing,* Ph.D. Dissertation, Computer Laboratory, University of Cambridge

Azmoodeh,M., Lavington,S. and Standring,M.(1984) 'The Semantic Binary Relationship Model of Information' in C.J. van Rijsbergen (eds.), *Research and Development in Information Retrieval,* Cambridge University Press, Cambridge, pp.133–151

Berry-Rogghe,Genevieve(1984) 'Some Issues in the Semantics and Pragmatics of Definite Reference in the Context of Natural Language Database Access', *Computers and Artificial Intelligence, vol.3,* 49–56

Berry-Rogghe,Genevieve(1985) *Interpreting Singular Definite Descriptions in Database Queries,* Submitted to the ACL European Chapter Conference

Boguraev,Branimir and Sparck Jones,Karen(1983) 'How to Drive a Database Front End Using General Semantic Information', *Proceedings of the Conference on Applied Natural Language Processing,* Santa Monica, CA, pp.81–88

Boguraev,Branimir and Briscoe,Edward(1984) 'Control Structures and Theories of Interaction in Speech Understanding Systems', *Proceedings of the 10th International Conference on Computational Linguistics,* Stanford, CA, pp.259–266

Boguraev,Branimir and Sparck Jones,Karen(1984) 'A Natural Language Front End to Databases with Evaluative Feedback' in G.Gardarin and E.Gelenbe (eds.), *New Applications of Databases,* Academic Press, New York, pp.159–183

Boguraev,Branimir(1985,forthcoming) 'User Modelling in Cooperative Natural Language Front Ends' in G.Nigel Gilbert and Christian Heath (eds.), *Social Action and Artificial Intelligence,* Gower Press, Aldershot

Brachman,Ronald(1978) *A Structural Paradigm for Representing Knowledge,* Technical Report No. 3605, Bolt, Beranek and Newman Inc., Cambridge, Mass

Brachman,Ronald(1979) 'On the Epistemological Status of Semantic Networks' in N.Findler (eds.), *Associative Networks: Representation and Use of Knowledge by Computer,* Academic Press, New York, pp.3–50

Brachman,Ronald and Schmolze,James(1982) *Proceedings of the 1981 KL-ONE Workshop,* Fairchild Technical Report No. 618, Palo Alto, CA

Carberry,Sandra(1983) 'Tracking User Goals in an Information Seeking Environment', *Proceedings of the National Conference on Artificial Intelligence,* Washington, D.C., pp.59–64

Cater,Arthur(1981) *Analysis and Inference for English,* Ph.D. Dissertation, Computer Laboratory, University of Cambridge

Clark,Keith(1978) 'Negation as Failure' in H.Gallaire and J.Minker (eds.), *Logic and Data Bases,* Plenum Press, New York, pp.293–322

Cohen,Philip(1978) *On Knowing What to Say: Planning Speech Acts,* Technical Report No. 118, Department of Computer Science, University of Toronto

Cullingford,Richard(1978) *Script Application: Computer Understanding of Newspaper Stories,* Research Report No.116, Yale University Department of Computer Science

Damerau,Fred(1980) *The Transformational Question Answering (TQA) System: Description, Operating Experience and Implications,* Report RC8287, IBM Thomas J.Watson Research Center, Yorktown Heights, NY

Damerau,Fred(1981) 'Operating Statistics for the Transformational Question Answering System', *American Journal of Computational Linguistics, vol.7,* 30–42

Damerau,Fred(1981) 'A Note on the Utility of Computing Inferences in a Real Data Base Query Environment', *American Journal of Computational Linguistics, vol.7,* 43–45

Dietterich,Tom(1984) *Constraint Propagation Techniques for Theory-Driven Data Interpretation,* Ph.D. Dissertation, Department of Computer Science, Stanford University, CA

Fahlman,Scott(1979) *NETL: A System for Representing and Using Real-World Knowledge,* MIT Press, Cambridge, Mass

Gallaire,Herve and Minker,Jack (Eds.)(1978) *Logic for Databases,* Plenum Press, New York

Gershman,Anatole(1981) 'Figuring Out What the User Wants — Steps Toward an Automatic Yellow Pages Assistant', *Proceedings of the 7th IJCAI,* Vancouver, BC, pp.423–425

Ginsparg,Jerrold(1983) 'A Robust Portable Natural Language Database Interface', *Proceedings of the Conference on Applied Natural Language Processing,* Santa Monica, CA, pp.25–31

Gray,Peter(1984) *Logic, Algebra and Databases,* Ellis Horwood, Chichester

Grosz,Barbara(1977) *The Representation and Use of Focus in Dialogue Understanding,* SRI Technical note 151, Menlo Park, California

Grosz,Barbara(1982) 'Transportable Natural Language Interfaces: Problems and Techniques', *Proceedings of the 20th Annual Meeting of the ACL,* Toronto, Ontario, pp.46–50

Grosz,Barbara(1983) 'TEAM, a Transportable Natural Language Interface System', *Proceedings of the Conference on Applied Natural Language Processing,* Santa Monica, CA, pp.39–46

van Griethuysen,Joost(1981) *Concepts and erminology for the Conceptual Schema,* Preliminary Report ISO TC97/SC5/WG3, American National Standards Institute, New York

Harris,Larry(1982) *INTELLECT User's Guide,* Artificial Intelligence Corporation, Waltham, Mass

Hendrix,Gary, Sacerdoti,Earl, Sagalowicz,Daniel and Slocum,Jonathan(1978) 'Developing a Natural Language Interface to Complex Data', *ACM Transactions on Database Systems, vol.3,* 105–147

Hobbs,Jerry(1983) 'An Improper Treatment of Quantification in Ordinary English', *Proceedings of the 21st Annual Meeting of the ACL,* Cambridge, Mass, pp.57–64

Joshi,Aravind and Weinstein,Scott(1981) 'Control of Inference: Role of Some Aspects of Discourse Structure — Centering', *Proceedings of the 7th IJCAI,* Vancouver, BC, pp.385–387

Kaplan,S.Jerrold(1981) 'Appropriate responses to inappropriate questions' in A.Joshi, B.Webber and I.Sag (eds.), *Elements of discourse understanding,* CUP Press, Cambridge, UK, pp.127-144

King,Tim and Moody,J.K.M.(1983) 'The Design and Implementation of CODD', *Software: Practice and Experience, vol.13,* 67–79

de Kleer,Johan, Doyle,Jon, Steele,Guy and Sussman,Gerald(1979) 'Explicit Control of Reasoning' in P.Winston and R.Brown (eds.), *Artificial Intelligence: an MIT Perspective (vol.1),* The MIT Press, Cambridge, Mass, pp.93–116

Konolige,Kurt(1983) 'A Metalanguage Representation of Relational Databases for Deductive Question-Answering Systems', *Proceedings of the 7th IJCAI,* Vancouver, BC, pp.496–503

Kowalski,Robert(1982) *Logic as a Database Language,* DOC Report 82/25, Department of Computing, Imperial College, London

McKeown,Kathleen(1982) *Generating Natural Language Text in Response to Questions about Database Structure,* Technical Report MS-CIS-82-5, Department of Computer Science, University of Pennsylvania, PA

Marcus,Mitchell, Hindle,Donald and Fleck,Margaret(1983) 'D-Theory: Talking about Talking about Trees', *Proceedings of the 21st Annual Meeting of the ACL,* Cambridge, Mass, pp.129–137

Mark,William(1981) 'Representation and Inference in the Consul System', *Proceedings of the 7th IJCAI,* Vancouver, BC, pp.375–381

Moore,Robert(1979) *Handling Complex Queries in a Distributed Data Base,* Technical Note 170, SRI International, Menlo Park, CA

Moore,Robert(1982) 'Natural Language Access to Databases — Theoretical / Technical Issues', *Proceedings of the 20th An-*

*nual Meeting of the Association for Computational Linguistics,* Toronto, Ontario, pp.44–45

Palmer,Martha(1983) 'Inference-Driven Semantic Analysis', *Proceedings of the National Conference on Artificial Intelligence,* Washington, DC, pp.310–313

Pazzani,Michael and Engelman,Carl(1983) 'Knowledge Based Question Answering', *Proceedings of the Conference of Applied Natural Language Processing,* Santa Monica, CA, pp.73–81

Perrault,C.Raymond and Allen,James(1980) 'A Plan-Based Analysis of Indirect Speech Acts', *American Journal of Computational Linguistics, vol.6,* 167–182

Rieger,Charles(1975) 'Conceptual Memory and Inference' in Roger Schank (eds.), *Conceptual Information Processing,* North-Holland, Amsterdam, pp.157–289

Robson,Michael(1984) *Constraint Enforcement in a Relational Database Management System,* Ph.D. Dissertation, Computer Laboratory, University of Cambridge

Rosenschein,Stanley(1981) 'Plan Synthesis: a Logical Perspective', *Proceedings of the 7th IJCAI,* Vancouver, BC, pp.331–337

Schmolze,James and Lipkis,Thomas(1983) 'Classification in the KL-ONE Knowledge Representation System', *Proceedings of the 8th IJCAI,* Karlsruhe, Germany, pp.330–332

Sidner,Candace(1979) *Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse,* Technical Report AI-TR-537, Artificial Intelligence Laboratory, MIT

Sparck Jones,Karen(1984) *Basic Semantics Information,* Internal Memo, Computer Laboratory, University of Cambridge

Sparck Jones,Karen(1985,forthcoming) 'Compound Noun Interpretation Problems' in F.Fallside and W.Woods (eds.), *Computer Speech Processing,* Prentice-Hall, Englewood Cliffs, NJ

Sparck Jones,Karen(1983) 'Shifting Meaning Representations', *Proceedings of the 8th IJCAI,* Karlsruhe, Germany, pp.621–623

Stonebraker,Michael et al.(1976) 'The Design and Implementation of INGRES', *ACM Transactions on Database Systems, vol.1,* 189–223

Storrs,Graham, du Boulay,Benedict and Gray,Peter(1984) *A Metadata Advisor: Some Sample Queries,* Internal Report, Department of Computing Science, University of Aberdeen

Tait,John and Sparck Jones,Karen(1983) *Automatic Search Term Variant Generation for Document Retrieval,* British Library Research and Development Report 5793

Thompson,Frederick and Thompson,Bozena(1975) 'Practical Natural Language Processing: the REL System as Prototype' in M.Rubinoff and M.Yovits (eds.), *Advances in Computers,* Academic Press, New York, pp.110–169

Warren,David and Pereira,Fernando(1981) *An Efficient Easily Adaptable System for Interpreting Natural Language Queries,* Research Paper 155, Department of Artificial Intelligence, University of Edinburgh

Webber,Bonnie and Joshi,Aravind(1982) 'Taking the Initiative in Natural Language Data Base Interactions: Justifying Why', *Proceedings of the 9th International Congress on Computational Linguistics,* Prague, Chechoslovakia, pp.413–418

Weischedel,Ralph(1975) *Computation of a Unique Class of Inferences: Presupposition and Entailment,* Ph.D. Thesis, Department of Computer and Information Science, University of Pennsylvania, PA

Wilks,Yorick(1975) 'A Preferential, Pattern-Matching Semantics for Natural Language Understanding', *Artificial Intelligence, vol.6,* 53–74

Woods,William(1972) *The Lunar Sciences Natural Language Information System,* Final Report, Bolt, Beranek and Newman, Cambridge, Mass

Woods,William(1978) 'Semantics and Quantification in Natural Language Question Answering' in M.Yovits (eds.), *Advances in Computers,* Academic Press, New York, pp.1–87

Zarri,Gian-Piero(1981) 'Building the Inference Component of an Historical Information Retrieval System', *Proceedings of the 7th IJCAI,* Vancouver, BC, pp.401–408