Technical Report

UCAM-CL-TR-60
ISSN 1476-2986

Number 60

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Memory and context mechanisms for automatic text processing

Hiyan Alshawi

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

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

This technical report is based on a dissertation submitted December 1983 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Trinity Hall.

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

Contents

Chapter 1

Introduction

1.1 Memory and context mechanisms in automatic language processing

It is now generally accepted by researchers in artificial intelligence that sophisticated automatic natural language processing requires a vast knowledge base, and has to take the influence of context into account. This implies that a natural language processing system needs a storage and retrieval mechanism for holding the knowledge and making it available to the interpretation processes, and a context mechanism that indicates which elements of the base are currently relevant to interpretation. However, there is little agreement on how to represent and organize knowledge in memory, or what knowledge to represent, and on how and when to use it during language processing. Similarly, there is no universal agreement on what constitutes context, and on how contextual information should be represented and applied. The lack of agreement on these issues raises questions about the extent to which practical natural language processing systems can be based on the particular techniques used in existing experimental systems.

Much of the work in this area nevertheless continues to confirm that there are many problems of language interpretation that cannot be handled without the application of knowledge and context. These problems are not limited to esoteric language phenomena, although there are plenty of these, but the roles of memory and context in the interpretation of many common phenomena have not been worked out satisfactorily. Common phenomena include reference resolution, word sense disambiguation, and the interpretation of compound nominals. A number of artificial intelligence research projects have concentrated on one or another of these phenomena and have studied specific models for the use of memory and context in dealing with them. Examples are Sidner's work [Sidner79] on the interpretation of definite anaphora, and McDonald's work [McDonald82] on the interpretation of noun compounds. Other projects have investigated models for representing and using specific aspects of context information, for example the work by Grosz [Grosz77] on the use of focus in mechanical task dialogues and the work by DeJong [DeJong79] applying stereotyped knowledge of real-world situations to text analysis.

The work reported in this thesis does not concentrate on the use of knowledge in memory for the interpretation of a single type of construction, or on the representation and application of one aspect of context. Instead, techniques are developed that result in unified memory and context mechanisms that can be used to handle many common constructions and to take into account various different aspects of context. The techniques are relatively simple to implement, yet powerful enough to deal with a wide range of phenomena. Because of this, I believe these techniques form an appropriate basis for the design of memory and context mechanisms for (modest) natural language processing applications in the nearer future.

In order to realize practical systems it is not sufficient to design just any mechanisms that allow the interpretation of language to be made with respect to the knowledge in memory and to take context into account. The mechanisms also have to be reasonably efficient even when a large knowledge base is being utilised. The need for knowledge and context representations that would allow efficient processing was one of the main motivations for Minsky's proposal [Minsky75] for organising knowledge into frames. It was also the motivation for the design by Fahlman [Fahlman79] of a knowledge representation language for which certain operations could be performed very efficiently on a specially designed parallel machine. Efficiency considerations also had an important influence on the design of the mechanisms described in this thesis.

Before discussing the methodology followed in this project and outlining the experimental text processing system, Capture, that was built, the main apparatus and techniques for text interpretation developed in the project are listed for reference. These are:

(1) a memory representation formalism that allows the encoding of linguistic, discourse domain, and application specific knowledge, and also the encoding of the relationships between these different types of knowledge. The advantages of the representation formalism, as compared with similar formalisms are that it is simple, and that it can be given a well defined semantics.

(2) techniques for improving the efficiency of retrieval from memory. The main device is an indexing scheme that can be based on semantic clustering. This is used to increase the efficiency of memory search operations, and to allow memory searches to be restricted by the current representation of context information.

(3) a framework within which various common language interpretation problems (such as the resolution of definite references) can be solved, and the functional activities of an application task (such as database creation

from natural language texts) can be performed.

(4) the representation of different aspects of context, such as recency of mention, memory-based association, specific subject area, and preceding memory processing. The representation used also allows the effects of the various context factors to be combined.

(5) the accumulation and management of context information in a way that responds gradually to shifts of focus during text processing. The context mechanism is related to memory in a way that allows straightforward application of contextual information to language interpretation problems such as reference resolution and word sense disambiguation.

Thus the concern of this work was to tackle the problems involved in designing and implementing satisfactory mechanisms for the memory and context functions of automatic text processing systems. These functions have been shown to be vital to sophisticated text processing by much of the natural language work in artificial intelligence.

The main contributions of the techniques developed, as listed above, are the simplicity of the memory representation; the generality of the context mechanism (for flexible management, combination and application, of context factors); and the utility of the indexing scheme developed for efficient memory searching and access to entities in focus.

1.2 Aims and methodology

The research reported here has a technological rather than scientific bias. This is reflected in the long term aims of the research, in the aspects emphasized in the work that was carried out, and in the approach adopted for testing and evaluation.

The long term aim is the realization of practical automatic natural language processing systems, rather than the formulation of a well-developed theory of human language processing. It would therefore be inappropriate to evaluate this work primarily from the point of view of linguistics, cognitive science, or the philosophy of language. Of course a technological applications-oriented approach cannot be fruitfully followed without a great deal of attention being paid to the realities of language use. A system built without taking these into account would almost certainly be unusable. Moreover, a genuinely practical system has to be extensible, and this seems to imply a well motivated or principled foundation for the system. A deeper understanding of human language behaviour may also provide better

characterizations of linguistic phenomena, and bring to light constraints on language use, that could indicate technologically feasible solutions to the problems of building practical automatic language processing systems. But linguistic and cognitive considerations are only inputs to the system design, rather than its justification.

The context and memory mechanisms developed in this project do provide a framework for the formulation of relatively precise abstract theories for the role of memory and of different aspects of context in the interpretation of various language constructs. The emphasis in the work carried out was in developing the necessary computational techniques, those listed earlier, for providing such a framework. The experimental system, Capture, implemented using this framework, included a number of language interpretation operations (such as definite reference resolution) that are performed with respect to the contents of memory and the current context, and the representation and handling of various types of factors contributing to context specification (such as recency of mention). Much of the effort involved in the project was spent on these memory operations and types of context factor, and some novel solutions to the text processing problems related to them were tested by the project. However, the particular interpretation operations, and the particular context factors used and their handling, were not viewed as constituting fully fledged theories about these operations and factors; they were viewed rather as showing that the memory and context mechanisms implemented provide an appropriate apparatus for handling these phenomena and one that is compatible with a range of possible theoretical accounts of them.

The application task of creating a structured database from a collection of texts was chosen for testing the memory and context mechanisms because it combines a fair challenge in text understanding with the production of concrete output that can be independently evaluated. This task also has the advantage for testing the mechanisms developed in the project of realistically restricting the domain of discourse. The implemented system, Capture, was not intended to be a production text processing system. The database creation (or "capture") application was designed to check on the validity and utility of the underlying language processing mechanisms and was not regarded as of primary interest in its own right. The adoption of an application task is important because it sets a standard for judging the success, or failure, of a system in interpreting a text. Although, for any particular task, this standard tends to be idiosyncratic to the task, it still permits better evaluation of the interpretation process than can be achieved by inspecting the products of interpretation as new structures in a formalism that is internal to the text processing system. Judging internal representations is not an adequate way of evaluating a language understanding system. The database capture task, in contrast, involves

producing objects which have to be acceptable to an external system, the database management system. External evaluation was regarded as particularly important, in the context of this research project, because of the long term aim of realizing text processing systems for practical applications.

Ideally, a text processing system should be tested with much care being devoted to collecting appropriate language test data. Unfortunately, the example texts that were processed by the system had to be written specifically to test the system; they could not be taken from a corpus of texts written independently for some other purpose. The reasons for this included the lack of availability of a suitable corpus of texts that could be processed given the limitations on the coverage of language constructions handled, and on the interpretation possibilities that are considered, by the implemented system. When writing example texts, an attempt was made to distinguish between producing texts for testing the system and ones that were bound to work, but it is not possible to guarantee that the artificially produced examples are not biased to the system. As will be illustrated in detail by the thesis, the example texts that are handled by the system were sufficiently rich to demonstrate the use of the memory and context mechanisms in solving several pervasive language interpretation problems and performing a non-trivial text processing task.

1.3 The Capture system and what it does

The "Capture" system implemented in this project is designed to perform the task of creating a database, with a specified structure, from the relevant factual content of a body of short English texts. Two example domains were used for testing purposes, and the memory knowledge base used by the current version of the system contains knowledge relevant to both domains. For one of these domains the texts used were records about a hypothetical museum's artifacts and their collectors; for the other the texts were thought of as retailers' records of data processing machines, suppliers and manufacturers. The target databases (the "Artifacts Database" and the "Machines Database") are composed of fixed format tables, of the standard kind conforming to the so-called Relational Data Model. The relations and columns of these target relational databases are specified in advance.

The database Capture task requires the implementation of solutions to common language interpretation problems in order that explicit database statements can be generated. For example, the processing, in context, of a sentence like "It is an arrow" requires determining the referent of the pronoun "it" and selecting the correct sense of "arrow" so that the following

output (a "database creation statement") might be produced.

(ARTIFACTS/RELATION (ARTF/NUMB P670) (ARTF/TYPE weapon1))

Below are two texts from the example domains which were processed by the system, followed by the "database creation statements" that were produced for them. These statements can be used for incrementing the target relational databases, but their detailed format is not important for understanding the memory and context mechanisms and here the relevant point is that they correspond to explicit statements, or propositions, derived from the texts.

Example text from the artifacts domain:

> Jones who was a trader collected P350 from Daui. He collected P370 from Woodlark. P350 is a necklace. P370 is an armlet. P391 is a necklace that comes from Woodlark. The condition of these ornaments is good.
>
> Armstrong and Haddon were British. They were academics. Haddon collected P597 and P598 from Daui. The artifacts are necklaces. The condition of these Daui necklaces is poor.
>
> P392 and P393 are armlets that were collected by Smith. This collector was a trader. The artifacts are fair.

Although this example text uses only simple English constructions, it demonstrates the resolution of plural definite references by the system. For instance, the interpretation of the noun phrase "the artifacts" in the last sentence ultimately results in the production of the last two lines of the output given below.

Database creation statements for the artifacts text:

```
((COLLECTORS/RELATION ((COLL/OCCP trader1) (COLL/NAME Jones)))
  (ORIGIN/RELATION
    ((ORIG/ARTN P350) (ORIG/COLL Jones) (ORIG/PLAC Daui)))
  (ORIGIN/RELATION
    ((ORIG/ARTN P370) (ORIG/COLL Jones) (ORIG/PLAC Woodlark)))
  (ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P350)))
  (ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P370)))
  (ORIGIN/RELATION ((ORIG/ARTN P391) (ORIG/PLAC Woodlark)))
  (ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P391)))
  (ARTIFACTS/RELATION ((ARTF/COND good1) (ARTF/NUMB P391)))
  (ARTIFACTS/RELATION ((ARTF/COND good1) (ARTF/NUMB P350)))
  (ARTIFACTS/RELATION ((ARTF/COND good1) (ARTF/NUMB P370)))
```

```
(COLLECTORS/RELATION
  ((COLL/NATN British1) (COLL/NAME Haddon1)))
(COLLECTORS/RELATION
  ((COLL/NATN British1) (COLL/NAME Armstrong1)))
(COLLECTORS/RELATION
  ((COLL/OCCP academic1) (COLL/NAME Haddon1)))
(COLLECTORS/RELATION
  ((COLL/OCCP academic1) (COLL/NAME Armstrong1)))
(ORIGIN/RELATION
  ((ORIG/PLAC Daui) (ORIG/ARTN P598) (ORIG/COLL Haddon1)))
(ORIGIN/RELATION
  ((ORIG/PLAC Daui) (ORIG/ARTN P597) (ORIG/COLL Haddon1)))
(ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P598)))
(ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P597)))
(ARTIFACTS/RELATION ((ARTF/COND poor2) (ARTF/NUMB P598)))
(ARTIFACTS/RELATION ((ARTF/COND poor2) (ARTF/NUMB P597)))
(ORIGIN/RELATION ((ORIG/COLL Smith1) (ORIG/ARTN P393)))
(ORIGIN/RELATION ((ORIG/COLL Smith1) (ORIG/ARTN P392)))
(ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P393)))
(ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P392)))
(COLLECTORS/RELATION ((COLL/OCCP trader1) (COLL/NAME Smith1)))
(ARTIFACTS/RELATION ((ARTF/COND fair1) (ARTF/NUMB P393)))
(ARTIFACTS/RELATION ((ARTF/COND fair1) (ARTF/NUMB P392))))
```

Example text from the machines domain:

> Plexir manufactures P9999 which is a computer. It is supplied by Smith. P1010 is a terminal that is supplied by Clark. This one is made by Mikota. These machines are red.
>
> P9000 is a green printer. It is made by Plexir. P4444 is a blue computer. The cost of the machine is 7850. The peripheral is supplied by the P9999 supplier. The terminal manufacturer makes the blue machine. The cost of Mikota's peripheral is 235.

Again, although the language used in this example is fairly simple, it illustrates cases of word sense disambiguation performed by the system, such as choosing the correct senses for "green", "make", and "terminal". It also illustrates some cases of definite reference resolution.

Database creation statements for the machines text:

```
((MACHINES/RELATION ((MC/TYPE computer) (MC/MCNUM P9999)))
  (MANUFACTURES/RELATION ((M/MNAME Plexir1) (M/MCNUM P9999)))
  (SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P9999)))
  (SUPPLIES/RELATION ((SMC/SID Clark1) (SMC/MCNUM P1010)))
  (MACHINES/RELATION ((MC/TYPE terminal1) (MC/MCNUM P1010)))
  (MANUFACTURES/RELATION ((M/MNAME Mikota1) (M/MCNUM P1010)))
  (MACHINES/RELATION ((MC/COLOR red1) (MC/MCNUM P9999)))
  (MACHINES/RELATION ((MC/COLOR red1) (MC/MCNUM P1010)))
  (MACHINES/RELATION ((MC/MCNUM P9000) (MC/COLOR green1)))
```

(MACHINES/RELATION ((MC/TYPE printer2) (MC/MCNUM P9000)))
(MANUFACTURES/RELATION ((M/MNAME Plexir1) (M/MCNUM P9000)))
(MACHINES/RELATION ((MC/MCNUM P4444) (MC/COLOR blue1)))
(MACHINES/RELATION ((MC/TYPE computer) (MC/MCNUM P4444)))
(MACHINES/RELATION ((MC/MCNUM P4444) (MC/COST !7850)))
(SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P9000)))
(MANUFACTURES/RELATION ((M/MNAME Mikota1) (M/MCNUM P4444)))
(MACHINES/RELATION ((MC/MCNUM P1010) (MC/COST !235))))

Thus, apart from demonstrating that the system can carry out the database capture task, these example texts also illustrate some of the capabilities of the system for language interpretation in context. The correctness of this interpretation can (usually) be determined by examining the output produced by the system. Although the language constructions used in these examples are of a very simple nature, the correct interpretation of these constructions frequently depends on the context in which they occur in the input texts.

1.4 General description of how Capture works

The Capture system has four major components:

1) a sentence analyser

2) a memory component

3) an interpretation component

4) a task-specific (database capture) component.

The sentence analyser, or parser, is one developed independently by Boguraev [Boguraev79]. The analyser passes information to the interpretation component; it is effectively a front-end to the core of the Capture system which is made up of the interpretation, memory, and task-specific components. Each of these components communicates, in both directions, with the other two. The system has no context component as such, because, as will become apparent later, the representation and application of context is handled by the memory component, while a feature of the context mechanism is that any component can take part in the management of context information.

The analyser processes each sentence in the input text independently, producing a list of alternative analyses (i.e. parses) for the sentence. It uses an ATN grammar formalism and applies semantic wellformedness checks on the structures that it builds, mainly by applying semantic category

restrictions derived from Wilksian semantic formulae. The output analyses are dependency trees that use an extended set of case-labels to link verbs to their arguments and to exhibit the structure of noun phrases. By applying semantic checks, the analyser performs some word sense and structural disambiguation, but ambiguities that cannot be resolved using the essentially linguistic information that the analyser has at its disposal, and those that represent genuine alternative readings at the level of a single sentence, are presented as alternative structures in the analyser output, and must be resolved by the interpretation component.

The interpretation component uses the memory and context mechanisms to interpret the analyser output with respect to the contents of memory and the current representation of context information, i.e. with respect to information that might include specifying that a particular artifact is an arrow, and that this artifact is foregrounded in the current context. This component is implemented as "interpretation operations", the main ones being verb clause interpretation, state clause interpretation, predicate and argument derivation, be-clause interpretation, have-clause interpretation, noun phrase reference resolution, structural disambiguation, word sense disambiguation, and relationship interpretation for compound nouns, possessives, and function words. Collectively these operations perform the function of incorporating new structures in memory that encode the propositional content of the input text.

Interpretation operations are procedures that, typically, are given fragments of analyser structures, and use the memory and context mechanisms to perform a particular type of disambiguation, reference resolution, or memory structure creation. These operations can also have the important effect of adding new context information to the current representation of context. They also can invoke other interpretation operations, and in this way embody the gross flow of control in the system. For example, the evaluation of a clause interpretation operation may cause the evaluation of operations for noun phrase reference interpretation and for interpreting embedded clauses.

The task-specific component, or database capture component, is implemented by another set of procedures, the task-specific operations. These perform functions such as locating predicates in the database descriptions in memory, classifying entities with respect to the target database, and locating database names. Like the interpretation operations, the task-specific operations can also create new memory structures and generate context information. The task-specific component uses results of these operations to produce the database creation statements. These should then be used by a database management system to increment the target database, although, in fact, in this research the output of the system was

not fed into a database management system because this last step was not considered to be sufficiently interesting, nor does it appear to involve any difficulties given certain assumptions (discussed later) about the database management system.

The main task-specific operations, which invoke subsidiary ones, are evaluated at fixed points during the interpretation process. Interpretation and task processing is therefore interleaved, but this happens in a rather fixed manner. The typical flow of control for a single sentence is thus roughly as follows. The sentence is analysed by the parser, and alternative analyses are passed to the interpretation operations. These operations call on the memory component to perform retrieval from the knowledge base and create new structures, and they alter the context information represented. They also invoke the task-specific component which uses the results of task-specific operations that it invokes to produce database creation statements corresponding to the sentence.

In order to perform their functions, both the interpretation and task-specific operations make extensive use of the memory knowledge base. This knowledge is both about types and individuals. The knowledge base contains language-related entities such as noun and verb senses and case-labels; (world) entities in the domain of discourse, such as individuals and domain specific predicates; and task-related entities such as those taking part in the description of a particular relation in the target database. Just as important, the relationships between all these objects are also encoded in memory.

The memory component uses a simple, but powerful, representational formalism that has a well-defined semantics. This formalism was derived from, but is simpler than, previous "semantic network" formalisms, in particular NETL [Fahlman79] and KL-ONE [Brachman78], themselves based on the representation of concepts with frame-like structures and the inheritance of properties between concepts.

Capture memory contains entities which can be thought of as concepts, and loosely speaking, stand for word senses, predicates, individual objects and actions, etc. The representational formalism uses two basic types of assertions. These are specialization assertions and correspondence assertions. They have the form

(Specialization: A of B) and
(Corresponds: C1 to D1 as C2 to D2)

where A,B,C1,D1,C2 and D2 are memory entities. Roughly, the specialization assertion states that A can be described as a B and the correspondence assertion states that the relationship between C1 and D1 can be described

by the relationship between C2 and D2. There are subtypes of these two basic assertion types, but, except for the representation of context information, the knowledge base is specified only by memory entities and specialization and correspondence assertions. (*)

Retrieval from memory is based on marker processing of the type used by Fahlman [Fahlman79]. This can be thought of as a way of performing certain set operations, on the entities in memory, which are effectively limited deductions that retrieve information represented implicitly in memory. The sets involved depend on memory assertions and are represented by marking memory entities, i.e. using markers to assign marks to entities. The results of memory operations are recovered by searching for entities satisfying marking conditions and "threshold" conditions, the latter being determined by the current context information.

Software techniques were implemented in Capture for enhancing the efficiency of these searches and for reducing the effort involved in marking. For the purpose of searching, the marking and context information is indexed automatically during processing. Further, the indexing scheme makes effective use of "semantic" criteria for clustering the entities in memory.

The context mechanism depending on these components of the Capture system is as follows. A basic notion of this mechanism is the context factor: a context factor indicates that a certain set of memory entities should, for some reason or other, be foregrounded (i.e. given some precedence over other entities) in the current context. At any given time during processing, context consists of a number of context factors each being of a certain factor type. The types of factors include those derived from information about previous memory processing, associations in memory, recency of mention, and text subject area. A context factor, i.e. an instance of a factor type, is represented by a marked set of memory entities, the scope of the factor, and a significance weight, which is a numerical value indicating the current importance of the context factor.

Context factors are created by the interpretation and task specific operations that are evaluated during the processing of a text. In principle, any other components could create context factors and manage them by altering their significance weights. What is important is that context information is not thought of as being derived from one source, e.g. subject area, but from many different sources, linguistic and non-linguistic at once.

(*) In the terms that are used to describe the semantic network formalisms mentioned above, entities correspond to "nodes" and assertions correspond to "organizational relationships" that translate into "links".

When context is applied, the information conveyed by the various factors is combined by deriving context activation values for the relevant memory entities. The context activation of a memory entity is simply the sum of the significance weights of the context factors within the scope of which the entity lies. There are two basic ways of applying context; these are choice applications and threshold applications. In choice applications, context activation is used to choose between memory entities (e.g. candidate results for reference resolution) or between sets of memory entities (e.g. word sense combinations). In threshold applications memory searches are restricted to entities with context activations that are higher than some specified threshold. This is used, for example, to limit the memory searches that implement reference resolution operations. The indexing scheme mentioned earlier allows efficient access to entities satisfying a threshold condition derived by combining context factors even though individual context factors can be managed independently. Overall, the context mechanism offers a convenient way of gradually accumulating, and altering the weights of, context factors of different types, while allowing their effects to be combined efficiently for disambiguation and limiting memory search.

1.5 Organization of thesis

Chapter 2 describes and discusses the memory representation formalism. Chapter 3 is on memory retrieval processing and search, and explains the indexing scheme mentioned above. Chapter 4 describes the interpretation operations implemented in the Capture system. The context mechanism is explained in Chapter 5, which describes the types of context factors implemented for the Capture system and the problems to which context is applied in the system. Chapter 6 describes the database Capture task and the task-specific component of the Capture system. Finally, Chapter 7 makes some concluding remarks about the project and suggestions for future research.

Previous research relevant to the topic addressed in a chapter will be discussed in that chapter, so that there is no single chapter devoted entirely to the work of other AI researchers. In Chapter 2 it will be argued that the memory representation formalism is simpler than those developed by Brachman [Brachman78] and Fahlman [Fahlman79], but still retains their most important features. The relevance of Fahlman's parallel network hardware design is discussed in Chapter 3 on memory processing. When describing language interpretation operations in Chapter 4, some of the work by others on the problems handled by these operations is noted. In Chapter 5 it is argued that the context mechanism can be thought of as an extension of, and improvement on, Grosz's [Grosz77] focus mechanism. The

application-specific processing is regarded in Chapter 6 as being more similar to knowledge based approaches (such as Consul [Mark81]) to text processing applications, than to surface based approaches (such as [Sager81]) to database creation from natural language texts.

The various chapters also include their own illustrative examples and some discussion of limitations and possible extensions. Appendix A contains the output produced for a number of example texts by the current version of the system. The order of the chapters reflects an attempt to minimise forward references. However, some of these were unavoidable, the most notable being the references to context application in the chapter on interpretation operations.

Chapter 2

Memory Representation

This chapter describes the memory representation formalism and gives examples of how information relevant to the text processing performed by the system is represented in terms of this formalism. A simple formal semantics for the representation system is presented. Capture memory is also compared with similar AI knowledge representation systems.

2.1 Memory and text processing

The term "memory" is used, in AI, by many authors researching into natural language processing (e.g. [Schank82a], [Small80], [Lebowitz83], [Rieger75]) to refer to the component of the system that stores knowledge about the domain of discourse that is used during the understanding process. This memory knowledge base is usually modified and in particular incremented during processing so that memory also stores the "results", whatever form these may take, of understanding the language input to the system. The above characterization of memory applies equally well to the knowledge storing component of so called knowledge based language processing systems (such as [Walker78], [Bobrow80], [Norton83], [Mark81]). In the work described in this thesis, no distinction between these uses of "memory" and "knowledge base component" will be made. In particular, the use of the term "memory" does not imply any claims about modelling human use of knowledge as it does for some authors including Schank [Schank82a].

Capture memory will be returned to in the next section of this chapter, but before that some general remarks are made about the use of knowledge in AI language processing systems. The knowledge used by these systems has been characterized as belonging to various types including linguistic knowledge, world knowledge (alias domain knowledge) and knowledge related to a text processing application task. It is probably not possible to maintain strong distinctions between these different categories. For example, although linguistic knowledge of a syntactic nature is often stored separately as the grammar, linguistic knowledge of a more semantic nature (e.g. relating to word senses) is frequently not distinguished in the knowledge base from knowledge that is specific to the domain of discourse (e.g. knowledge about domain specific predicates).

Domain and semantic knowledge has been applied in different ways by the various experimental systems that make use of it. For example, the knowledge has been used to solve understanding problems when they are encountered, i.e. in a demand driven way (e.g. [Wilks75b] [Sidner79]). It has also been used to continuously generate new facts (forward inference) that aid the processing of the rest of the text (e.g. [Rieger75], [Cater81]), i.e. in a supply driven way.

There are various types of processing problem that the domain and semantic knowledge has been applied to. These include the following.

a) Common language interpretation problems such as reference resolution, word sense disambiguation, structural disambiguation, and the derivation of implicit relationships from certain language expressions.

b) Dealing with extended uses of language such as metaphor [Hobbs81] and ellipsis [Grosz77]. (This is in some sense an extension of (a).)

c) Deciding how to incorporate new entities and relationships derived from sentences into representations of a text [Wilks73], or how to fit them into representations of typical situations known to the system (e.g. [DeJong79], [Tait82]).

A text processing system that performs a linguistic application task such as summarizing may only require linguistic and domain knowledge. However, a system that processes text in order to perform a non-linguistic application-task also needs to access knowledge about this task. Such a text processing system will not necessarily carry out the task itself, but only generate commands, queries, or whatever, in an appropriate format so that the task itself can be carried out by some other system such as a graphics system, a robot, or a database management system. Examples of task-specific knowledge are knowledge of a graphic display driver [Zdybel81], knowledge about manipulating toy blocks [Winograd72], and knowledge about a relational database (the system described here).

How knowledge is represented and what exactly is represented depends, in part, on the requirements of all of the different processes that make use of this knowledge for performing the text processing. This means, of course, that in order to build a system that achieves a certain level of performance, it is not necessarily the case that the memory representation formalism should be completely general in the sense that any fragment of knowledge is representable, or that it should support completely general inference processes so that any information derivable from the stored knowledge can be inferred by the system. Moreover, limiting the power of the memory system can enhance the clarity of the representation and the efficiency of

access to the knowledge represented. These last considerations were taken into account when designing the memory component of the Capture system.

2.2 Memory representation in Capture

In the Capture system the knowledge stored in memory is knowledge about the domain world, linguistic knowledge of a semantic nature, and knowledge related to the database creation text processing task, although no strong distinction between these types is maintained. Thus task knowledge is regarded in this system simply as specialized domain knowledge that is used by a task-specific component of the system, and semantic knowledge is regarded as very generic domain knowledge. Representing these different types of knowledge in memory is aimed at allowing the relationships between them to be expressed explicitly in memory.

However, the most important processing operations using memory are the language interpretation ones. Therefore the attitude taken to memory representation is that it should, in the first instance, be geared towards the implementation of these interpretation operations. These operations are in themselves domain and task independent but exploit the domain and task information in memory. They make use of the knowledge stored in memory in order to solve text processing problems that were placed in category (a) in the last section, i.e. common interpretation problems. (*) Another requirement, taken to be of secondary importance, is that domain and task specific knowledge should be representable to allow performing the operations that are necessary for the database capture task, or some other similar task, to be performed.

A further constraint on the memory system is that it should perform the required retrieval operations, and especially those necessary for the language interpretation operations, reasonably efficiently, taking into account the current context. The methods employed to achieve this will be described in the chapter on memory processing and the chapter on context.

The knowledge stored in memory is essentially concerned with generic objects, predicates, and relationships, as well as specific instances of all of these. The way the implemented system is set up at the moment means that specific knowledge derived from a text processed by the system is discarded after the processing of that text is complete; but there is no reason why this knowledge could not be retained. The structures created in memory during

---

(*) Perhaps the representation scheme is in principle inadequate for problems of type (b), metaphor especially, but this is unclear.

the interpretation of a text reflect its "propositional content" and do not constitute a kind of "text representation" in the sense of text linguistics, although some of the information related to the structure of the text is retained and used by the context mechanism.

It is important to emphasize that the memory representation in Capture is thought of as a minimal knowledge representation for solving common text interpretation problems and performing simple tasks such as the database capture task. Thus the effort in design was not spent on trying to handle difficult representation problems, but rather on achieving a convenient and simple representation that nevertheless allowed the system to achieve interesting language processing behaviour. Because of this, a number of initial "features" of the representation system that were not being used for this purpose were removed as the overall system developed and the requirements on the memory system became clearer. The resulting formalism satisfies the broad requirements that some workers in natural language (in particular Small [Small80], and Sidner [Sidner79]) have considered necessary for supporting certain aspects of language interpretation, including, for example, a classification hierarchy for concepts. Because of this, and because of the experience gained in using the representation for the processing performed by the Capture system, it does not seem to be the case that the reduced nature of the representation formalism has made it too limited for fairly sophisticated text processing (but some limitations are reviewed in Section 2.9).

The representation formalism used in the Capture system was influenced, in particular, by Fahlman's NETL system [Fahlman79]. NETL had the advantage of being relatively well specified, and was conceived with a detailed processing scheme based on a "hardware network" design in mind. This is consistent with the view (discussed later) that what distinguishes "semantic network" formalisms from other knowledge representation formalisms are processing considerations. Brachman's Structured Inheritance Network formalism [Brachman78] also influenced initial versions of the representation formalism. The relationship between the representation formalism used in Capture and other formalisms like these will be discussed at the end of this chapter. Roughly speaking, and ignoring processing issues, memory representation in Capture uses a very simple formalism which can be thought of as a tangled hierarchy of small overlapping frames [Minsky75].

2.3 Entities and basic assertions

The memory contains entities, or concepts, which include generic and individual objects, predicates and relationships. However, the entities are not categorized as belonging to one of these types, but instead, the type of an entity, and the way that it takes part in memory processing, is taken to be dependent on how it is related to other entities in memory, and in particular on the organizational relationships that are being used to access it.

There are two kinds of basic assertions that give rise to these organizational relationships, specializations and correspondences. Specialization assertions have the form

(Specialization:A of B)

where A and B are memory entities. For example one specialization assertion is

(Specialization: computer of machine),

'computer' can now be viewed as a "specialization" of 'machine', and also 'machine' as a "generalization" of 'computer'; i.e. things that can be described as computers can also be described as machines. (*) Assertions of this type form the specialization, or "is-a", hierarchy of entities in memory. (Subtypes of the specialization assertion are used in order to distinguish whether generics or individuals are being related, as described later.) The hierarchy is "tangled" in that an entity can be the specialization of more than one generic entity. Thus we could have

(Specialization: woman of human), and
(Specialization: woman of female).

There is a most generic entity 'thing' that has no generalizations. (**)

A correspondence assertion has the form

(Corresponds: C1 to D1 as C2 to D2),

where C1,D1,C2 and D2 are entities in memory. C1 is viewed as a "role",

---

(*) Single quotes always denote memory entities. This is a convention that applies throughout this thesis. Double quotes indicate terms used in some special sense in this report or by others, and also text fragments; underlining is used for emphasis and when terms are being defined.
(**) I will use the term "generic" loosely for entities to contrast them with entities "lower" in the specialization hierarchy or correspondence hierarchy (see below) and also to indicate type-token distinctions.

"argument" or "slot" of "owner" D1; and as a "role specialization" of C2, where C2 is viewed as a (generic) argument of D2. The pairs C1 D1, and C2 D2, are role-owner pairs. Examples of correspondence assertions are

(Corresponds: data/processing to computer as
machine/activity to machine)
(Corresponds: supplies/agent to supply as
agent to verb/statement).

The memory's correspondence assertions form a hierarchy of ordered role-owner pairs, the most generic of which is (role, thing). This hierarchy complements the specialization hierarchy; while the specialization hierarchy forms a classification of the kinds of entities in memory, the correspondence hierarchy forms a classification of the relationships between them. A correspondence assertion states that there is a relationship between one (role-owner) pair of entities that is a refinement of a relationship that exists between another pair of entities in memory. The correspondence assertion allows the definition of arguments for concepts, i.e. what are called "roles" or "slots" in several representation languages. The correspondence assertion can thus be thought of as a role specialization assertion.

Normally, in these other systems, role specialization is taken to be an assertion between an owner, a generic role and a specialized role; and the equivalent to the first correspondence assertion can be paraphrased as "data/processing fills the machine/activity role for computer" (because generic fillers can be thought of as slots in Capture memory). However, this approach often leads to what is, in my view, an artificial distinction and asymmetry between "concepts" and "roles" (this point is discussed further in Section 2.9). In contrast, any memory entity can take part as either "role" or "owner" in many correspondence assertions, but this does not lead to confusion because the fourth argument of the assertion, The "generic-owner" is present.

This description of the specialization and correspondence assertions, since it is given mainly in terms of examples, is not intended to be a full characterization of the meaning of these assertion types. These types of assertions, when viewed as part of the text processing system, can be said to take their "meaning" from the operations that are used to process them, for example from the way that specialization assertions take part in the resolution of referents for definite noun phrases such as "the machine", and the way that correspondence assertions take part in the interpretation of compound nouns such as "computer processing". The description, in this report, of how memory is used thus constitutes an informal operational semantics for the memory formalism. The coherence of such an operational semantics for the memory system depends, in the end, on how well the text processing system works. Another way of providing a semantics for memory

assertions is discussed in the following section.

2.4 Semantics of memory assertions

As just indicated, structures in memory can be thought of as being given their meaning by the operations that make use of them in the text processing system. This sort of semantics can be specified by describing the operation of the system at some level. An alternative approach is possible. Thus another sort of semantics relates symbols to referents in the world (or possible worlds) that they describe, in principle in a rigorous and formally satisfactory way. Some advantages of having such a semantics for representation systems are discussed by Patrick Hayes [Hayes77a]. There has also been some recent interest in providing denotational semantics for frame-like representations (see [Reimer83] for example). Hayes argues that the contribution of logic, as a representation system, is its model-theoretic semantic theory. In this type of semantics for logic conjoining predicate logic assertions restricts the set of possible interpretations for the symbols occurring in them. In an analogous way, we would like the addition of new memory assertions to restrict the interpretation of memory entities. (*)

This section will attempt to indicate how such a semantics could be devised for the symbols and structures used in the Capture memory representation formalism. This could then be used to validate deductions, with respect to an interpretation imposed by the semantics, or characterize other inferences as approximate, in that they are not necessarily valid given such an interpretation.

An interpretation is taken here to be a pair of functions (called "ref" and "rel") that model the relationship between memory entities and entities in a particular world described by memory. The semantics of memory assertions can then be given as constraints on these functions, i.e. as constraints on possible valid interpretations. The function ref maps a memory entity to the set of objects in the world that the memory entity refers to, i.e. the set of of world entities that the memory entity may describe. The function rel maps a role-owner pair of memory entities to the set of pairs of objects in the world, i.e. to a relation, that the pair in memory can describe.

(*) "Interpretation" is used, in this section, in a different sense from its use in the rest of this report.

The basic constraints are as follows ($\subseteq$ stands for subset).

1. For any specialization assertion
   (Specialization: A of B)
it must be true that ref(A) $\subseteq$ ref(B).

2. For any correspondence assertion
   (Corresponds: C1 to D1 as C2 to D2)
it must be true that
   rel(C1,D1) $\subseteq$ rel(C2,D2)
and additionally that
   rel(C2,D2) $\subseteq$ ref(D2) X ref(C2), and
   rel(C1,D1) $\subseteq$ ref(D1) X ref(C1).

There are subtypes of the specialization and correspondence assertions, and these have further conditions associated with them that will be indicated in Section 2.5. One such condition that can be shown to be relevant to the inheritance of properties is that rel should yield a function when applied to certain role-owner pairs.

On the basis of these conditions it is possible to make a number of deductions that are consistent with a valid interpretation and that can be assumed to hold by the memory processing operations. I am here taking a deduction to be the derivation of a new memory assertion from a set of previous memory assertions in such a way that the conditions imposed by the new assertion are consistent with valid interpretations of the original assertions. These deductions include the transitivity of specialization and the inheritance of properties, as used by similar knowledge base systems.

For example if we have the assertions

(Specialization: computer of machine), and
(Specialization: machine of inanimate)

then the validity of the interpretation will not change if

(Specialization: computer of inanimate)

were added to memory. The transitivity of correspondence assertions can also be justified, because it preserves condition (2) above. Thus from the following memory assertions

(Corresponds: manufacture/object to manufacture as
        make1/object to make1)
(Corresponds: make1/object to make1 as
        object to verb/statement)

we can deduce that

(Corresponds: manufacture/object to manufacture as
object to verb/statement).

The implemented Capture system does not perform deductions in the manner described above, but instead performs inferential memory operations whose results are sets of memory entities. (This processing model is detailed in Chapter 3). Such an operation can be thought of as being based on "valid" inferences if the results of evaluating the operation do not vary when any deduced assertions (in the sense explained above) are included in memory. If an operation is not of this type, then it might be violating the conditions on valid interpretations imposed by the semantics of assertions and hence can be thought of as being based on "approximate" inferences.

These "approximate" inferences are used when performing memory operations on role-owner pairs for which rel may not yield a function; or on states of memory in which the conditions for a valid correspondence have been over-ridden in order to handle exceptions. However the formal semantic apparatus is still useful because we can, in principle, use it to define a clean distinction between valid and approximate inferences, and as a tool for determining whether individual inferences are valid or only approximate.

Since the Capture system as a whole makes use of inferences that are approximate as just defined, as well as a preference-like mechanism for the application of context, it cannot be regarded as a rigorous system formally supported by the semantics for memory assertions just given above. But this is not considered to be a deficiency for reasons similar to those noted by Wilks [Wilks77]; i.e. that it is not necessary to require that all the representations and processes involved in a natural language understanding system are rigorously formalized in some consistent fashion.

2.5 Subtypes of assertions

The specialization and correspondence assertions can carry further information about the relationships between their arguments. This takes the form of a list of flags given as an additional argument to the assertions. A number of these flags and the additional semantic conditions associated with them are given below. In general, the subtypes provide a refinement of the basic structure imposed on memory by the two main assertion types.

If a specialization assertion is marked by the flag "instance", then the specialized entity can refer to only one world entity in an interpretation. (*) Thus the assertion

(Specialization: A of B (instance))

imposes the additional condition

$|ref(A)| = 1$.

The flag "member" is stronger than "instance" and is used to explicitly specify all the individual entities described by the generic entity of the specialization. The additional conditions for a set of specialization assertions of the form

(Specialization: A1 of B (member))
⋮ ⋮
(Specialization: An of B (member))

are as follows ($i,j \in \{1, ..., n\}$)

$|ref(Ai)| = 1$

$\bigcup_{1}^{n} ref(Ai) = ref(B)$

$ref(Ai) \neq ref(Aj)$ if $i \neq j$.

The flag "distinct" can be attached to specialization assertions to indicate exclusive subcategorization of the entity being specialized. The additional condition for a set of specialization assertions of the form

(Specialization: A1 of B (distinct))
⋮ ⋮
(Specialization: An of B (distinct))

being as follows ($i,j \in \{1, ..., n\}$)

$ref(Ai) \cap ref(Aj) = \emptyset$ if $i \neq j$.

The approach taken to co-reference of memory entities in the system is very simple. The semantics of the specialization assertion allows co-reference to be represented as mutual specialization, and the following assertions state that the memory entities A and B co-refer (so that ref(A) equals ref(B)).

(Specialization: A of B)
(Specialization: B of A)

A co-reference mechanism is needed to "conflate" two entities which are discovered to co-refer after they have been created separately. It can also be used for asserting multiple distinct subcategorizations, and for modelling multiple role relationships between the sets of objects that two entities refer to.

(*) The notion of being an individual is taken, for the purpose of this work, to be a comment on how memory, and the language being processed, view and describe a (real or imaginary) world rather than some more fundamental comment about what exists in that world.

The import of the flags that can be associated with correspondence assertions can most easily explained in terms of "rel".

(Corresponds: C1 to D1 as C2 to D2 (uni))

means that

rel(C1,D1): ref(D1) -> ref(C1)

is a function (total and single valued).

(Corresponds: C1 to D1 as C2 to D2 (rev))

means that

rel(C1,D1): ref(D1) -> ref(C1)

is a bijection (one-one and onto).

Example of these assertions are

```
(Corresponds: mother to mothers/child as
        parent to child (uni))
(Corresponds: supplier/number to supplier/dbentity as
        dbentity/number to dbentity (rev))
```

Thus the first assertion says that children have unique mothers, and the second one says that there is a one-to-one mapping between supplier numbers and supplier entities (in some database world).

All the subtyping flags described in the present section are included in the implemented system. However, the import of the "distinct" flag and the difference between the "rev" and "uni" flags are not always taken into account by the memory operations used for the current application. Thus the "distinct" flag is ignored by some operations, and others treat "rev" simply as "uni". Similarly, many of the implemented operations do not take specialization cycles into account. These limitations of the implementation do not cause the system to fail completely (e.g. to loop indefinitely), but do mean that some possible inferences are missed by the system.

The flags attached to assertions are usually omitted in the examples given in the thesis since they are usually not relevant to the points under discussion. Overall, the subtyping of assertions as presented in this section increases the expressive power of the representation while the simplicity of memory structuring in terms of entities, specializations, and correspondences is maintained.

2.6 Semantic network implementation

As is the case for several other knowledge representation systems used in natural language understanding, the computer representation of structures built in memory may be described as a "semantic network". A comparison

between some of these representation formalisms and Capture memory is given in Section 2.9. This section simply considers the characterization of Capture memory as a "semantic network". This term is now typically associated with representations that make use of specialization hierarchies, roles, and a number of other representational tools. However, surveys of semantic networks (e.g. [Brachman79b] and [Ritchie83]) do not indicate any representation devices that are common to all systems that have been characterized as semantic networks. This suggests that the "network" characterization of these systems does not have a substantive content with regard to knowledge representation notwithstanding that many of these systems make use of similar representation devices.

A possible alternative characterization is simply the purely formal one, namely that all semantic network systems use a graphical notation to display what is being represented. This is a weak characterization since all graphs can be expressed in non-graphical notations, and it has been often pointed out that there is nothing inherently semantic or knowledge representational about graphs.

Nonetheless, the implementation of the memory in this system is still thought of as a semantic network for two somewhat different reasons. Firstly, the representational system described above is derived to a large extent from representational schemes that have been termed semantic networks. These all make use of taxonomies and property inheritance, this being one of the more common characteristics of semantic network formalisms [Ritchie83]. Since these features are substantive, this reason for my use of "semantic network" is a substantive claim about the status of Capture memory as a means of knowledge representation.

Secondly, a characterization of semantic networks that has to do with processing rather than with static description is appropriate for this system. Thus it is widely held (see e.g. [Schubert79]) that the graphical aspect of semantic networks is a way of stating which parts of the representation are to be implemented as pointers so that certain processing operations can be performed on the knowledge base in a way that makes these operations computationally tractable.

Network processing, and efficiency considerations, will be described in the following chapter. However, since a graphical notation for memory is not being used in this report, a description of the network implementing the formalism is now given, in order to give some idea of low level representation. Entities are represented by "nodes", specialization assertions by link records connecting two nodes, and correspondence assertions by link records connecting four nodes. The flags indicating subtypes of assertions are represented by indicators that are attached to

the link records encoding the assertions. All pointers implementing the links are two way pointers. A node includes the name of the entity that it represents, and pointers to the link records for the assertions that it takes part in. As with other network implementations, this implementation has some useful access properties. For example we can access, without searching, all the entities immediately below a particular entity in the specialization hierarchy, or all the generic roles for all the correspondences in which the entity is the role argument. These access properties are used to perform the marking operations with which memory processing is implemented.

2.7 Language and domain entities

The memory is exploited to process the output of the analyser (semantic parser) of the system. Hence we need to have entities for language related semantic objects that appear in the structures that the analyser produces. These include noun senses, verb senses and case relations. We also need entities to represent objects and predicates specific to the domain of discourse, and instances of these that occur in a domain. The relationship between language related entities and domain related entities is basically that of the generic to the more specific along the specialization and correspondence hierarchies. Memory structures for the data processing machines suppliers and manufactures domain will be used as examples. Examples of language related entities are given first.

There are no entities representing words, but only entities representing word senses; these originate in the lexicon used by the parser. For example there are two entities with names 'printer1' and 'printer2' for the two senses of the noun "printer" that occur in the lexicon; and we have

(Specialization: printer1 of human)
(Specialization: printer2 of peripheral).

(*)

There are also entities for verb senses, for example both 'manufacture' and 'make1' are found in the lexicon and the memory contains the assertion

(Specialization: manufacture of make1).

Entities for words as opposed to word senses could be created by assertions

---

(*) For irrelevant historical reasons, some word sense names in the lexicon do not have sense numbers.

such as

```
(Corresponds: PRINTER to printer2 as
        lex/item to word/sense)
(Corresponds: MANUFACTURE to manufacture as
        lex/item to word/sense).
```

These would be necessary for a task such as translation or paraphrase.

There are also entities for case relations like 'agent', 'object', and 'recipient', etc. These correspond to (and have the same names as) the labels that mark cases in the dependency structures output as "meaning representations" by the parser; and there are assertions like

```
(Corresponds: agent to verb/statement as
        tag/role to tagged/statement)
(Corresponds: recipient to verb/statement as
        tag/role to tagged/statement).
```

These cases may be specialized to particular verb senses

```
(Corresponds: make1/agent to make1 as
        agent to verb/statement),
```

and the specialized cases can, of course, be specialized further as in

```
(Corresponds: manufacture/agent to manufacture as
        make1/agent to make1).
```

Domain related entities include individual objects and facts, and types of objects and predicates that are relevant to a domain of discourse. For individual objects and facts we have memory assertions like

```
(Specialization: P7000 of disc-drive1 (instance))
(Specialization: Plexir1 of Paris/manufacturer (instance))
(Specialization: E1 of manufacture (instance))
(Corresponds: P7000 to E1 as
        manufacture/obj to manufacture)
(Corresponds: Plexir1 to E1 as
        manufacture/agent to manufacture)
```

State predicates derived from properties such as weight and colour are represented by entities such as 'weight/of' and 'colour/of', for example

```
(Specialization: weight/of of measure/of)
(Corresponds: weight/of/possessor to weight/of as
        measure/of/possessor to measure/of)
(Corresponds: weight/of/weight to weight/of as
        measure/of/number to measure/of).
```

Domain based constraints on the arguments of predicate entities can be expressed by specialization assertions such as

(Specialization: manufacture/agent of manufacturer)
(Specialization: weight/of/possessor of phys/obj),

or by correspondence assertions, for example

(Corresponds: manufacture/obj to manufacture/agent as
manufacturer/goods to manufacturer).

An example of a domain specific entity is 'computer/manufacturer', for which we would have the assertions

(Specialization: computer/manufacturer of manufacturer)
(Corresponds: computer to computer/manufacturer as
manufacturer/goods to manufacturer).

Correspondence assertions can be used to express relationships which also encode domain knowledge, for example

(Corresponds: component to machine as part to whole), and
(Corresponds: data/storage to disc-drive1 as
data/handling to peripheral).

2.8 Task related entities

In order to express the relationships between the domain entities and database administrative entities like the names of tables and columns in a particular target database, the database objects and predicates that are implied by record types in the database must be made explicit. These objects and predicates are in fact taken to be specializations of domain objects and predicates. The memory knowledge base used by the current version of Capture systems contains the necessary task-related knowledge for both the Machines Database and the Artifacts Database.

The target databases used are relational databases (see e.g. [Date81]) i.e. consist of relations (tables) in which the columns range over values of a certain type and each row indicates that certain statements (the underlying statements for the relation) hold between the values in that row. The domain-database mappings for target databases are defined by creating generic entities for the description of a typical relation in a relational database and then specializing this description to descriptions of the relations in the target database. The description of the generic relation for a relational database is represented in memory by the entity 'db/relp'. This

has associated with it a number of generic database objects ('relp/dbentity') and statements ('relp/statement') and we have

```
(Specialization: db/relp of db/schema)
(Corresponds: relp/dbentity to db/relp as
        schema/role to db/schema)
(Corresponds: relp/statement to db/relp as
        schema/role to db/schema)
(Specialization: relp/statement of statement).
```

Note that the structures associated with 'db/relp' specify what constitutes the description, in memory, of a relation in a relational database, rather than the description of some particular relation. 'db/relp' also includes entities for the name of the relation being described, and for generic entries ('relp/entry') in the rows of the relation:

```
(Corresponds: relp/relation to db/relp as
        schema/role to db/schema)
(Specialization: relp/relation of relation), and
(Corresponds: relp/entry to db/relp as
        schema/role to db/schema)
(Specialization: relp/entry of db/entry)
(Corresponds: relp/entry/value to relp/entry as
        entry/value to db/entry)
(Corresponds: relp/entry/column to relp/entry as
        entry/column to db/entry).
```

There are also other objects associated with the descriptions of relations, and these will be described in the chapter on task specific operations.

An example of specializing 'db/relp' to the description ('manufactures/relp') of the 'MANUFACTURES' relation in the Machines Database includes entities for the relation and column names:

```
(Specialization: manufactures/relp of db/relp)
(Specialization: MANUFACTURES/RELATION of relation)
(Corresponds: MANUFACTURES/RELATION to manufactures/relp as
        relp/relation to db/relp)
(Specialization: M/MNAME of column)
(Specialization: M/MCNUM of column)
(Specialization: M/CITY of column),
```

and entities describing a generic entry in a row in this relation, and how this relates, for example, to the name of a database entity ('dbentity'):

```
(Corresponds: manufac/name/entry to manufactures/relp as
        relp/entry to db/relp)
(Corresponds: manufac/name/value to manufac/name/entry as
        relp/entry/value to relp/entry)
(Corresponds: M/MNAME to manufac/name/entry as
        relp/entry/column to relp/entry)
```

(Corresponds: manufac/name/value to manufacturer/dbentity as
dbentity/name to named/dbentity).

There is simply one underlying predicate for this relation. It is represented by the entity 'relp/manufactures':

(Corresponds: relp/manufactures to manufactures/relp as
relp/statement to db/relp)
(Specialization: relp/manufactures of manufacture).

This is related to the database entities taking part in the db/relp. Thus we have, for example,

(Corresponds: manufacturer/dbentity to relp/manufactures as
manufacture/agent to manufacture).

Not all "underlying predicates" follow this simple pattern; they can also be classifications of objects, e.g. for indicating machine types in the Machines Database, or they can be specializations of state derived predicates.

The examples given in the last two sections should indicate how the relationships between linguistic entities such as word senses, and database entities such as column names, are expressed in memory. The way the task related memory entities and assertions are used will be discussed later, when the database capture task is described in detail.

2.9 Comparison with other formalisms and discussion

This section will first briefly outline the relationship between the design of Capture memory and a number of issues that AI work on knowledge representation (and in particular semantic networks) has been concerned with. Although these issues have been considered central by many researchers, they only cover a small part of the field of knowledge representation. The later part of the section will compare Capture memory in more detail with two similar representation systems (NETL and KL-ONE), that influenced its design.

Correspondence assertions are used to build structures in memory that resemble frames [Minsky75]. Thus the "slots" S1, S2,... for a frame F can be defined by correspondence assertions of the form

(Corresponds: S1 to F as ...)
(Corresponds: S2 to F as ...) etc.

Restricting the "fillers" of slots can be done by specialization assertions such as

(Specialization: S1 of E1).

The slot S1 itself may be thought of as a frame that inherits slots from E1, and similarly F may be a specialization of another frame E2 and inherit its slots. Frame F may also be a specialization of another frame E3, in addition to E2, allowing some notion of "multiple description" (see e.g. KRL [Bobrow77]).

The representation formalism used in Capture thus has much in common with knowledge representation languages such as KRL that are based on the organization of knowledge into frames, and structures built in Capture memory, such as the database descriptions given above, are often frame-like. KRL was intended to be as complete a knowledge representation system as possible, and included, among other things, capabilities for procedural attachment. However, in the design of Capture, it was considered that although procedures might be valid memory entities, the code for implementing them and invoking them should be handled outside the memory component.

Much of the research on semantic network formalisms has been concerned with combining semantic networks with predicate logic based formalisms. Examples of this are [Schubert79], [Shapiro79] and [Hendrix78]. One motivation for this kind of work has been the goal of maximising the expressive power of network based formalisms. However, maximising expressive power was not a design goal of Capture memory, and there was no

attempt at allowing predicate logic expressions to be represented in this formalism. A different motivation for combining semantic networks with logic is exemplified by the work by Deliyani and Kowalski [Deliyani79] in which semantic networks are thought of as providing an indexing scheme and a potentially useful strategy for guiding the search for a proof. This is consistent with the attitude taken (Section 2.6) in the design of Capture memory that the importance of semantic networks is largely a processing rather than representational issue.

Combining semantic networks with logic has been considered by others to be a non-issue. Thus it has been pointed out that it is possible to translate the syntax of predicate logic into a semantic network notation and vice-versa [Bundy79]. However, this does not show that these are equivalent systems since it does not compare semantic theories for these systems or the operations (inferences) that are performed. (See [Ritchie83] for a discussion of this point). In fact, some of the operations performed on logic-based semantic networks are similar to deductive inference operations applicable to predicate logic (see [Brachman79b], [Ritchie83]). However, something analogous to the specialization hierarchy is often used by such systems as a convenient means for the classification of objects and storing their properties in a non-redundant fashion.

Hendrix's partitioned networks [Hendrix78] are a good example of this. Incidentally, the arcs used for the purpose of classification in this system are similar to the specialization assertions used in Capture memory. In particular, the distinction between the use of "e" and "de" (element and distinct element) is reflected, in the Capture system, by the "instance" and "member" flags that can be attached to specialization assertions.

Hendrix's system brings up another issue, that of clumping together the nodes and links of a network into larger structures. Thus Hendrix relies heavily on the use of a technique called "partitioning". This allows portions of the network to be placed in "spaces". Many of the uses of partitioning during processing are handled in the Capture system by the use of marked sets of entities. In order to simulate some of the static representational uses of partitioning correspondence assertions could be used as follows.

(Specialization: S1 of space (instance))
(Corresponds: E1 to S1 as
space/node to space)
(Corresponds: E2 to S1 as
space/node to space) etc.

S1 could then act as the "supernode" for the space, i.e. the node that can be used, in the representation, to refer to the portion of network that forms the space. This would not deal with cases in which the inclusion of arcs in

spaces is significant. The latter would require an extension to the representation formalism in which assertions could be annotated by the name of a "handle-node" that could be used to refer to them (see [Fahlman79] for handle-nodes).

This extension may not, however, be necessary anyway. Thus Martin, [Martin80], asserts that the use of role-based structures gives a more natural modelling of the phenomena which partitioning was designed to model; and these are of course handled by correspondence assertions in Capture memory. The phenomena include, in particular, the representation of certain types of quantification. Thus the suggestions made by Martin (in [Martin80]) on how to formally represent various readings of quantified English expressions point the way to how certain difficult cases, not dealt with hitherto, could be represented in Capture memory. These include the representations of sentence readings exhibiting the referential/attributive distinction, and the collective/distributive distinction. Martin also claims that an advantage of his proposed representation is that it allows for the representation of partial interpretations (which can be refined later) when the quantification structure is unclear. This work by Martin discusses issues that are in fact not dealt with by the language interpretation mechanisms in Capture. Martin's work does suggest, however, that the use of structures based on correspondence assertions is appropriate for handling some difficult problems in the formal representation of English expressions.

The approaches to knowledge representation mentioned so far share some properties with Capture memory. The two formalisms discussed below had a much more significant, direct influence on the design of the memory representation used by the Capture system. These are Fahlman's NETL system [Fahlman79], and Brachman's Structured Inheritance Networks formalism on which the KL-ONE system is based [Brachman78]. Representational issues arising in the context of these approaches will be discussed with respect to the design of Capture memory in the rest of this section. For convenience the knowledge representation formalism used in the Capture system will be referred to as Memory in the rest of this section.

The NETL design was intended to be as complete a knowledge representation language as possible that was compatible with a marker processing scheme (discussed in Chapter 3). The design of KL-ONE was more concerned with producing a "clean" but general semantic network representation for knowledge representation. Nevertheless, NETL and KL-ONE are in fact very similar as knowledge representation systems. Some of the differences in the details of these representations will become apparent in the comparison with Memory. NETL and KL-ONE are semantic network representations in which there are a limited number of node types and link types. Both representations are structured around a classification hierarchy for nodes

representing concepts. The concept nodes in these hierarchies have associated with them descriptions that are built out of "role" nodes of various types and link types associated with these role descriptions. Concept nodes inherit the role-based descriptions of other concepts that are above them in the classification hierarchy.

Both NETL and KL-ONE are more complex representational formalisms than Memory. A simpler representational formalism was adequate for the purposes of the Capture system for two reasons. Firstly, as already noted, Memory was never intended to be a complete knowledge representation formalism, whatever that might be, whereas the designers of NETL and KL-ONE take completeness as one of their design aims. A number of issues that were not immediately relevant to the Capture system's test application were ignored. Secondly, as will be shown below, loosening NETL's and KL-ONE's built in ideas about what constitutes a "concept" allows some of the explicit primitives of these systems to be encoded, as needed, using Memory assertions.

The simpler ontology of Memory comes mainly from abandoning the strict classification of entities into concepts and roles. Fahlman's concluding remarks about the NETL representation include the following ([Fahlman79] p.231) "There is too much difference between the base-node of a description and a role; role reversal should be a smoother process than it currently is." In Memory, depending on which correspondence assertions are being considered, it is possible to view an entity as a concept with its own roles or as a role in descriptions associated with other entities. This leads to a natural way of having "multiple views" of entities because of the lack of a single structure imposed by the representation formalism itself.

In fact the Memory knowledge base used in the experimental system still makes use of correspondence assertions to construct a predominantly "concepts and roles" view of the knowledge represented. There are two reasons for this. Firstly, the knowledge base was originally constructed when there was a distinction between concepts and roles in the system and this way of thinking of the entities persisted after the translation into the later version of the representation formalism. Secondly, the concept/role distinction probably gives a pragmatically useful way of representing some fragments of knowledge, although it is not considered to be one that should be made into an absolute that forces artificial distinctions, but rather a derived notion dependent on view.

As mentioned earlier, a number of issues that were dealt with in NETL or KL-ONE were ignored completely. Three of the more important issues in this category (negation, inheritance-cancellation, and defined/natural terms) are discussed below.

The Capture system does not deal with negation at the memory representation level or at the level of language interpretation operations. Clearly, this is an important direction in which to extend the representation system. The easiest way to achieve this for Memory is probably to adapt Fahlman's **NOT flags which can be attached to certain nodes and links; but this was not tried.

There is no explicit provision in Memory for cancellation of inherited properties. In NETL *CANCEL links can be used to cancel inherited properties in order to allow for exceptions, for instance in the case of a three legged elephant. But many of the operations using memory will work if properties are simply explicitly over-ridden at a lower level of the specialization hierarchy. This is an imprecise solution to the problem, but it is not considered to be a major drawback compared to the use of cancellation links because these themselves are not problem free. The inheritance algorithm that takes cancellation links into account that was reported in [Fahlman81] was later shown to be inadequate. In KL-ONE (at least in some descriptions of the language) the taxonomy is taken as definitional (see below) and non-definitional information, like exceptions and cancellations, are considered to be best handled outside the representation formalism.

The defined class/natural class distinction was one that was dropped from Memory representation. This was partly because the distinction was not being used by the operations implemented for Capture. But another reason is that Capture does not include mechanisms for maintaining the consistency of the knowledge represented in Memory, which would involve checking that entities of a certain type satisfied definitional properties; this being the main consequence for processing of the distinction between natural types and defined types in NETL. KL-ONE generic concepts are interpreted as defined terms and this is considered to be important for the classification of new objects and their incorporation into the knowledge base. (*) Thus the natural/defined term distinction is closely related to the issues of consistency and the "digestion" of new information. Although these issues are important considerations for knowledge base systems, they were regarded as being beyond the scope of the Capture project.

The way in which a number of important NETL and KL-ONE link types are handled in Memory is now described. It is not possible to say, formally, that a particular fragment of these representations is equivalent to a set of

---

(*) In some versions of KL-ONE, however, natural kind concepts are distinguished from defined concepts and then the classification algorithm must be told which natural kind concepts can be used to describe a concept being classified, because this information cannot be deduced - see [Schmolze83].

Memory assertions. This is because of the lack of a uniform semantics to compare them with, or of any systematic indication of how the knowledge conveyed by certain English sentences in known contexts would be represented in the various formalisms. Comparisons have therefore to be based on suggesting how the analogues of certain types of expressions in the other formalisms would be represented in a Memory knowledge base.

In NETL every *TYPE node T (representing a typical concept) has associated with it an individual node that represents the set, S, of entities that can be described by T. Properties of this set, such as its cardinality being equal to 4, can be associated with the individual node. This can be represented directly in Memory as follows.

```
(Specialization: S of set (instance))
(Corresponds: S to T as
        set to set/member (uni))
(Corresponds: four to S as
        cardinality to set)
```

The semantics of these assertions permits the interpretation that the cardinality of ref(T) is 4, as required.

KL-ONE uses "ParaIndividual Concepts" (parametrized versions of concepts) in the descriptions of other concepts, for example a ParaIndividual of the SUPPORT concept is used in the description of the ARCH concept. In Memory there is no type distinction between an entity and the other entities (related to it using Memory assertions) that help to describe it. Thus although the SUPPORT statement in the arch description in KL-ONE is represented by a special node type (Paraindividual), linked to SUPPORT by a PARA link, it would just be an entity specialization, 'arch/support' say, of the entity 'support' in Memory as follows.

```
(Specialization: arch/support of support)
(Corresponds: arch/lintel to arch/support as
        supportee to support)
(Corresponds: arch/upright to arch/support as
        supporter to support)
```

Similarly the counterpart of NETL *IST (individual statement) nodes are (instance) specializations of entities that are specializations of the entity 'statement'. An example of an individual statement is the entity 'E20' below, representing the statement "Rockefeller owns Standard Oil".

(Specialization: owns/statement of statement)
(Specialization: E20 of owns/statement (instance))
(Corresponds: Rockefeller to E20 as
owns/owner to owns/statement)
(Corresponds: Standard/Oil to E20 as
owns/property to owns/statement)

In NETL, these individual statements can be given a time-area scope. In Memory this scoping is representable by a correspondence assertion as follows.

(Corresponds: 1890's to E20 as
statement/time/scope to statement)

The 'statement/time/scope' associated with statements scoped in this way can now be used by operations that consider the truth of statements at various times. (*)

In NETL there is a flag **PART which "Marks the PART *TYPE-role and all of its *TMAPS. (this is) Used to make PART-OF hierarchy operations more efficient." ([Fahlman79] p.269). In Memory this flag is equivalent to a recorded marker propagation that starts at the entity 'part' (recorded propagations are described in Section 3.5).

There is a distinction in NETL between two types of role ownership which are represented by *EXFOR and *EXIN links. Fahlman says that this distinction is reflected in English by the use of the prepositions "of" and "in". However, it is not clear whether this distinction has any real value since NETL processing does not exploit it.

The KL-ONE language distinguishes between four different types of inter-role relationship. The assertions given below illustrate how these can be represented in Memory, or rather one of the ways Memory assertions can be used to do this.

(Corresponds: building/block to block/object as
physical/part to physical/object)
(Specialization: building/block of block)

(Specialization: arch of block/object)
(Corresponds: arch/block to arch as
building/block to block/object)

(*) The extensions proposed in this section (e.g. the use of time scoping, and the representation of cardinality) were not implemented, but are included for the sake of comparison. However, except for cases where the contrary is stated, and for the sections in later chapters devoted to comparisons and possible extensions, all the mechanisms described in the thesis were implemented in the final version of the Capture system.

(Corresponds: lintel to arch as
arch/block to arch (uni))
(Corresponds: arch/upright to arch as
arch/block to arch)

(Specialization: arch/1 of arch (instance))
(Corresponds: lintel/1 to arch/1 as
lintel to arch)
(Corresponds: arch/1/upright to arch/1 as
arch/upright to arch)
(Specialization: arch/1/upright/1 of arch/1/upright (instance))

The four different types of KL-ONE inter-role relationship are illustrated by this example as follows.

a) The KL-ONE inter-role relationship "restriction", represented by a "modifies" link is exemplified by the relationship between the entities 'building/block' and 'physical/part'.

b) Differentiation, represented by a "differentiates" link in KL-ONE is exemplified by the relationship between 'lintel' and 'arch/block', and also between 'arch/upright' and 'arch/block'.

c) An example of "particularization" is the relationship between the entities 'arch/upright', 'arch/1/upright' and 'arch/1'. Here 'arch/1' is an individual arch and 'arch/1/upright' refers to its upright-blocks.

d) The relationship between 'lintel/1' and 'lintel' is "satisfaction", since 'lintel' refers to the individual lintel for 'arch/1'. (It is unique because rel (arch, lintel) is a function.) The relationship between 'arch/1/upright/1' and 'arch/1/upright' would also be represented by a "sat" (satisfies) link in KL-ONE.

The relationship between Memory and the KL-ONE and NETL formalisms can be summarized as follows. All three formalisms make heavy use of a classification hierarchy for concepts, that is also the basis of property inheritance between concepts. Correspondence assertions are used in Memory to provide a flexible way of modelling the relationships represented by roles in the other two formalisms. There are a number of issues, such as those related to consistency, that were considered to be out of the scope of the Capture system. However, the most important representational features of NETL and KL-ONE that depend on the various node and link types in these systems can be handled using the simpler Memory formalism.

No more will be said about the Capture memory representation formalism itself, as opposed to how it is processed and used in the Capture system. As

noted by Wilks [Wilks77] (in the context of using semantic primitives for representation), it is not necessary to assume that the deeper problems raised by a representation system have been solved before it can be used profitably for natural language processing. The memory representation formalism can be loosely thought of as providing data structures for classifying concepts (via the specialization hierarchy) and classifying the kinds of associations between them (via the correspondence hierarchy).

Chapter 3

Memory Processing

This chapter is concerned with the way information is retrieved from memory and the indexing scheme that is exploited for performing memory searches. The model for memory retrieval is an adaptation of Fahlman's marker propagation model. The indexing scheme is a new technique developed in this project for increasing the efficiency of searches that the retrieval process requires and, perhaps more importantly, for allowing efficient access to the most salient memory entities as determined by all the context information represented at a given time.

3.1 Marker processing

Much of the knowledge represented in memory is implicit, in the sense that it is not present explicitly as stand-alone statements. This is largely a consequence of the hierarchical organization of memory in which information is inherited from generic to more specific entities in memory. The process of extracting this information requires a limited sort of deduction, and will be referred to as "deductive retrieval" or "memory retrieval".

The implementation of memory retrieval is based on marker processing, i.e. on performing set operations on the nodes of a network in which a set is represented by marking its elements using a marker. This choice of processing model was strongly influenced by Fahlman's work [Fahlman79]. Thus the memory processing model used in Capture is an adaptation of Fahlman's processing model to Capture memory which is applied to text processing. Fahlman's model resembles, at first sight, the early semantic network processing model used by Quillian [Quillian68], but is in fact much more strictly controlled. Fahlman proposes the use of specialized hardware for implementing marker propagation processing on a semantic network in order to solve the problem of efficient deductive retrieval from very large knowledge bases. ([Fahlman80] sketches a possible design for this hardware, and [Hillis81] discusses a different design that is capable of supporting Fahlman's processing model.) This efficiency problem is considered (by Fahlman and others) to be central to building realistic natural language understanding systems that make use of extensive knowledge of the type represented in memory in the present system.

Furthermore the implementation of memory processing by the marker propagation hardware means that only certain types of inference operations, and in particular certain deductive retrieval operations, can take advantage of the parallel processing that the hardware is capable of. That is, the parallel processing performed by this hardware is specialized, i.e. does not aim at general concurrency, but only at performing certain particular types of retrieval operation very efficiently on a large knowledge base. Fahlman suggests, however, that simulating human-like intelligence requires only these restricted types of operation to be performed on memory. It is not clear how far this argument can be taken; however the implementation of Capture represents a test of the idea with regard to performing a certain text processing application. Thus the implementation for the system was partly motivated by the desire to build a system for text processing which operated under the restrictions that are the consequence of choosing a marker processing model that could be implemented by parallel hardware.

The simplicity of the marker processing that allows it to be performed on the specialized hardware also allowed the development, within a software implementation, of a scheme for increasing the efficiency of marker processing simulation on a serial machine. This software scheme can be based on semantic clustering, as described in detail later. This makes the processing of the knowledge base represented in memory tolerably efficient. This efficiency is useful for the development of systems making use of this processing model, and perhaps sufficient for systems concerned with limited discourse domains.

The fact that marker processing is used for performing the necessary memory retrieval operations is not in fact important for the operation of much of the text processing system. This is because memory retrieval processing is to a large extent (except e.g. for the convenience of using traces of this processing as a kind of context factor) used by language processes in a way that is independent from the actual implementation of the retrieval operations. The Capture design would thus allow for the exploitation of specialized parallel hardware if this turns out to be worthwhile, and if such hardware becomes available. At the same time much of the rest of the system can be evaluated without regard to how memory retrieval operations are performed.

The following three sections will describe memory retrieval operations and the marking operations and searches that are used to implement them. The sections after this describe the scheme used for indexing marked sets and its use for making memory searches more efficient.

3.2 Retrieval operations

Retrieval operations are performed, in much the same way as proposed by Fahlman [Fahlman79], using a marker processing model. Marker processing is really just a way of performing certain set operations on the entities in memory. The entities in each set are marked using a symbol (a "marker"), occurrences of which ("marks") are attached to the nodes that stand for the entities. Marker processing is performed by sequences of marking operations and searches. Marking operations specify the sets that are being operated on and these are described in Section 3.3. Searches (Section 3.4) extract the results, and intermediate results, of retrieval operations; they locate entities that satisfy marking conditions. ("Threshold conditions", related to the context mechanism can also be used, but these are not part of Fahlman's model and will be discussed in later sections.) Fahlman describes many examples of this type of processing in detail, and since the flavour of marker processing performed on memory is much the same as that of Fahlman's work only a simple example of a retrieval operation will be given.

An example of performing a simple operation involving an "intersection search" is retrieving any memory entities that are specializations of the entity 'make1' and that are also role-specializations of 'relp/statement'. (i.e. retrieving predicates that can be described by the "manufacture" sense of "make" and that are also underlying predicates for some database relation. The results would include 'relp/manufacture', see Section 2.8.) A marker, "M1" is used to mark all entities below 'make1' in the specialization hierarchy. A second marker "M2" is used to mark role specializations of 'relp/statement' in the correspondence hierarchy. The results of the operation can then be extracted by performing a search for entities marked by both "M1" and "M2".

In practice, retrieval operations can become fairly complicated, since they often involve further marking and search operations for checking, for example, that the results include only most-specialized entities, or that properties have been inherited in a valid way (see [Fahlman79]). The interpretation and task-specific components of Capture mainly make use of standard retrieval operations (of which there are around twenty) instead of requesting marking and search operations directly. Examples of these retrieval operations are given below.

A) Find the common ancestors in the specialization hierarchy of entities E1 and E2.

B) Find a most specialized entity C that fills the role R for entity O. That is, C is the inherited version of R with respect to O.

C) Find an entity O that is the owner of R when R is viewed as a role specialization of the generic role G.

D) Find the most specialized role-owner pairs in correspondence assertions that express the relationship between two entities E1 and E2.

All the standard memory retrieval operations are implemented in terms of the marking operations and searches for sets of entities satisfying marking conditions. The range of retrieval operations that were implemented was determined by the requirements of the interpretation component and the task specific (i.e. database capture) component for the system's test application. There was no attempt to implement a "complete" set of retrieval operations and only ones that were required were implemented; it would not be difficult to provide others as needed.

As mentioned earlier, the marker processing used by Capture is an adaptation of Fahlman's model to suit Capture memory, and is thus closer in style to Fahlman's model than the algorithms described by Woods [Woods78a], which make use of marker pairs. Further, the very low level algorithms that are used would be different if retrieval was performed on a "connection machine" of the type proposed by Hillis [Hillis81], because this would involve the use of "virtual links", rather than hardware links, between the processing elements that would correspond to memory entities. However, it might be possible to hide these differences in the "microcode" of the connection machine. The adaptation of Fahlman's model to Capture memory is determined by the types of marking operation that are allowed. These are described in the next section.

3.3 Marking operations

It was mentioned in the last section that each of the sets of entities that take part in retrieval operations is marked using a marker. The operations that perform this marking are now described. The range of marking operations thus determines the kinds of sets that can take part in retrieval operations. Marking operations depend on the network of "links" (between "nodes" standing for memory entities) that are derived from memory assertions. Thus one type of marking operation, a "marker propagation", takes an initial set of nodes and traverses the network from these nodes following links of some specified type, and marking, using a specific marker, all the nodes on the paths that are followed. (In fact the term "marker propagation" will also be used to loosely refer to marking operations in general.)

The network is derived from memory specialization assertions and correspondence assertions as follows. Specialization assertions of the form

(Specialization: A of B)

can be thought of as a pair of links; a "specialization link" from B to A, and its inverse, a "generalization link" from A to B. Correspondence assertions of the form

(Corresponds: R1 to O1 as R2 to O2)

can be thought of as a collection of links, including a "role-link" from O1 to R1, an "owner-link" from R1 to O1, a "role-specialization" link from R2 to R1, etc.

The internal low level encoding of these assertions indeed takes the form of records with two way pointers between the nodes that stand for the entities taking part in assertions and the records encoding the assertions. The assertion records correspond to "link elements" in the framework of Fahlman's formalism. However, such details will be ignored when examples of marking operations are described since this will be done in terms of the argument positions of memory assertions.

Marking operations can be classified into five major types as follows.

a) Propagation marking that begins at a single entity and marks all the entities that can be reached from this entity via a specified link type. ("link type" is used here to indicate a pair of argument positions of memory assertions, as explained above).

Examples of this type of marking operation are

(a1) Mark all the entities above the entity E in the specialization hierarchy.

(a2) Mark all the entities that are roles, or roles of roles etc. of entity E, where R is a role of entity E if there is a correspondence assertion of the form (Corresponds: R to E as - to -).

(a3) Mark all entities that can be reached by role-specialization links from entity E, where R is an immediate role specialization of E if there is a memory assertion of the form (Corresponds: R to - as E to -).

b) Propagation marking, parallel to (a), but starting from the set of entities that are marked with a specified combination of markers, instead of just a single entity.

c) Marking across one step of links, examples being

(c1) Mark all entities E1 that occur in correspondence assertions of the form (Corresponds: E1 to E2 as - to -) where E2 can be specified as a particular entity, or as the set of entities marked with a particular marker combination.

(c2) as for (c1) but according to the pattern (Specialization: E1 of E2).

(c3) as for (c1) but according to the pattern (Specialization: E1 of E2 (instance)).

d) Marking across one step of links, where further arguments of the memory assertions involved can be specified to be marked with a particular combination of markers, for example

(d1) Mark all entities E1 that occur in correspondence assertions of the form (Corresponds: E1 to E2 as E3 to -) where E2 is a specific entity and E3 is specified as the entities marked by a combination of markers.

(d2) as for (d1) but according to the pattern (Corresponds: E2 to E1 as - to E3).

e) Marking the set of entities (using a new marker) according to their existing markers.

(e1) Mark all entities that are marked with a particular marker combination (i.e. the entities that are marked with at least all the markers in a specified marker set).

(e2) Mark all entities that are marked with all the markers in one specified marker set, but not marked by any of the markers in another such set. (e1) is a special case of this marking operation.

(e3) Mark all entities that satisfy a threshold condition.

For most of these marking operations a new marker is generated and used to mark all the entities specified by the operation. The exceptions to this will be referred to as "recorded propagations", and these will be discussed later when the techniques for increasing the efficiency of marker processing are described.

3.4 Search requests

As indicated already, memory processing depends on searching for entities that are marked with specified combinations of markers. Searches are also performed that are restricted by "threshold" conditions. A threshold condition asserts that the entities being searched for should have "activations" that exceed a specified value. The activation of an entity is the sum of weights that are associated with the markers that were used to mark the entity. The weights are concerned with the representation of context information (Section 5.2), the activation of an entity being a measure of its salience in the current context. The importance of threshold searches for implementing a flexible context mechanism will become clearer in Chapter 5.

A facility was implemented whereby search requests prepackaged in a certain format can be conveniently evaluated by the system. Such a format specifies a marker combination and includes an expectation of the number of results of the search, and a number of flags that specify how the system should try to satisfy this number expectation.

One of the mode flags indicates that an activation threshold should parametrize an initial search, but that this threshold should be lowered and the search repeated if the initial search should fail to locate any entities. Another flag specifies whether or not the search should be constrained by a marker that indicates that the entities should be instances (i.e. take part in memory assertions with "instance" or "member" flags). A third flag specifies that the context mechanism should be used to select between candidate results of the search so that the search satisfies the number expectation. A fourth flag specifies that if no entities are located that have all the markers in a combination, then the search should be repeated, more than once if necessary, until the set of entities is located that have the most markers in the combination, i.e. K-1 of the markers in the combination where K is the smallest integer such that no entities have K markers in the combination.

These variants of the simple intersection and threshold searches are used, in particular, during reference resolution, which is a frequent requirement. The following section on efficient retrieval will explain how searches specified by search requests (whether or not they are packaged in the format described above) are implemented efficiently by exploiting a scheme for indexing marked sets.

3.5 Techniques for efficient retrieval

This section discusses the techniques that have been employed for reducing the amount of low level processing, on a serial machine, required for implementing the memory retrieval operations that are performed by the system. Most of these techniques are independent of the details of the representational formalism (and the actual operations that are performed), in that they could apply to any system that makes use of the style of marker processing that has been described.

In fact the techniques enhance the efficiency of the following.

1) Intersection searches. Searching for sets of entities specified by marking conditions.

2) Threshold searches. Searching for sets of entities whose activation is higher than a specified threshold.

3) The effort involved in performing marker propagations.

The use of the techniques is purely motivated by efficiency because they do not affect the outcome of retrieval operations, and hence the outcome of interpretation operations that use the retrieval operations. However, the techniques, and the extensions to them described in Section 3.6, are not merely low level implementation details because they address efficiency questions that are important for the realization of text processing systems exploiting "large" knowledge bases.

The indexing scheme

Enhancing the efficiency of memory searches (i.e. (1) and (2) above) depends on indexing marked sets. For the purpose of this indexing a tree (the cluster tree) is constructed; the nodes representing entities in memory are the leaves of this tree. The tree is constructed before any memory processing is performed, as follows. The nodes are partitioned into clusters of a chosen fixed size; the criteria for determining the order in which the nodes are put into clusters will be discussed later. The nodes in each cluster are then linked to a newly created cluster-node with special clustering links. The cluster-nodes are then themselves clustered, and this clustering is repeated until there is only one cluster node to be clustered, this being the root, or "cluster-apex", of the indexing tree formed by the clusters.

Marked sets are indexed during memory processing by marking nodes in the cluster tree. When any network node is marked, by a particular marker, all the cluster nodes above it are also marked by the same marker. (There are

exceptions to this procedure that will be explained shortly.)

Searching for all network nodes with a given combination of markers (i.e. the set of nodes each of which is marked with all the markers in some specified set of markers) can be done by starting at the cluster-apex and following the combination down through those cluster nodes having it to the final selected network nodes. Similarly, searching for the nodes with activations that are higher than a specified threshold is performed by starting at the cluster-apex and only passing through cluster-nodes whose activations exceed the specified threshold. It should be noted that the markers attached to a cluster node and its corresponding activation do not imply any semantic information since the cluster-nodes are only relevant to the indexing scheme.

The two types of search can be combined so that the cluster tree can be used to find the nodes with a given combination of markers that satisfy a threshold condition. Again, this search starts at the cluster-apex and only passes through cluster-nodes after it has been checked that their markers and activations satisfy the specified conditions.

Thus in order to reach the target nodes of the searches that have been described we will only examine the network nodes in clusters for which the cluster-node satisfies the condition determining the target set. For the intersection search (i.e. searching with respect to a marker combination), these nodes are the network nodes in clusters that have a node marked with each of the markers in the combination. This means, in particular, that the number of network nodes examined will be bounded above by the cluster size times the number of nodes in the smallest marked set. On the other hand if the target set happens to be a union of the sets of nodes in clusters then only the nodes in the intersection will be examined. A more detailed analysis of the simple intersection case will be given below.

As stated so far the scheme does not allow the inclusion in the search specification of conditions excluding nodes that are marked by any of the markers in a specified combination, or threshold searches when negative weights have been assigned to some markers. The necessary extension for dealing with this is now described.

The markers used to mark cluster nodes in the scheme described so far can be called "or-markers" in that they are used to mark a cluster node if <u>any</u> of the nodes below it are marked with the marker being indexed (or with the indexing marker - these were not distinguished in the above description). We can instead use an "and-marker" to mark a cluster node if <u>all</u> the nodes below it are marked with a particular marker (or the and-marker itself). If a marker is known to take part only in searches which specify that results

should not be marked with it, we can index this marker with and-markers instead of or-markers. Then when a search involves this marker, cluster nodes that are marked with its indexing and-markers are not passed through during the search because it is known that they will not lead to any target network nodes.

And-markers can also be used, for the purpose of threshold searches, to index a marked set of network nodes that has been assigned a negative weight. This does not affect the threshold search algorithm that evaluates the sum of the weights at a cluster node before passing through it. In the final version of the system, however, negative weights were not used. For a system in which exclusions and negative weights are sufficiently important, it may be worthwhile (although this was not considered to be the case for the present system) to always index marked sets by both or-markers and and-markers.

Efficiency analysis of the intersection search

For the purpose of efficiency comparisons on memory processing, primitive operations can be defined as the operations of marking a node, testing the marks on a node against a marker combination, and calculating the activation of a node. If we assume that the time taken to perform one of these operations on a node (in the network or tree) is a small constant, then we can evaluate the efficiency of the searches by determining the number of nodes that are examined during a search.

The simple intersection case (i.e. looking for all the network nodes that are marked by all the markers in a specified combination) will be analysed only. This case can be considered a special case of the threshold search since it is equivalent to finding nodes with weights that are greater than k-1 where k is the number of the markers in the combination, all of which have been assigned a weight equal to one, other markers having been assigned a weight equal to zero.

For the worst case of the intersection search, the number of nodes (in both network and tree) that are examined is of the same order as the number of nodes with the marker that marks the smallest set of network nodes in the intersection (the clustering factor being assumed to be a constant, c, for this analysis). Let m be the cardinality of this set. The number of nodes marked by this marker is less than the sum of the nodes in paths from the apex to each of the m nodes in the network. The length of these paths is O(log N) for a network with N nodes, and so the number of nodes marked by the marker and hence the number of nodes examined is O(m log N).

We can refine this upper bound for the worst case by observing that some of the nodes in the paths mentioned above must be shared near the top of the tree. Sharing need not occur below a level in the tree where there are more than m tree nodes, i.e. after a depth of O(log m) from the apex. Thus the number of nodes marked will be bounded by the number of nodes above this level plus the (distinct) paths from this level to the network nodes. This gives

O(m) + O(m(log N - log m)), that is O(m (log (N/m))).

(*)

For the (good) case where we can assume that all tree nodes with the combination do lead to network nodes in the intersection, let the number of nodes in the intersection be i, the order of nodes examined will be the same as the order of nodes with the combination, giving O(i log (N/i)) by the above analysis. For the (best) case, in which the number of nodes marked with the combination will be the number of nodes in a full c-branching tree with i leaves plus a path from the apex to this tree, we get

O(i) + O(log N - log i), that is O(i + log (N/i)).

(**)

As noted by Fahlman, the standard algorithm for finding the intersection of sets represented as lists takes at least O(m), where m is the cardinality of the smallest set in the intersection. This algorithm steps through the items on the shortest list checking for items in the intersection; this list being identified by maintaining counters for the number of items on each list. Thus, from the above analysis, this algorithm performs better than the cluster-based algorithm in the worst case, but the cluster-based algorithm can do better if the sets correlate well with the clustering. In addition, there is no obvious extension of the standard algorithm that can deal with the threshold search, which is important for the current application. As mentioned earlier, the threshold search is a generalization of the intersection search, and so has at least the same computational complexity as the intersection search.

For the searches that have been described the gain from using the clustering comes from excluding areas of the network that need not be searched. The size of this gain depends very much on how clustering correlates with marked sets, hence the interest in clustering criteria which will be described shortly.

(*) Following the above analysis more closely, it can be seen that the constants are not unreasonable, a concrete bound being 2m + cm(log (N/m) + 1), where log a is the largest integer less than or equal to the log of a to base c.

(**) again the constants are reasonable, a concrete bound being 2i + c(log (N/i) + 1)

Recorded propagations

The implementation of marker processing in the system is complicated further by the use of "recorded propagations" which are also motivated by efficiency considerations. Recorded propagations are aimed at reducing the effort required for marking, as distinct from the effort required for searching that has been addressed so far. In particular we would like to reduce the effort involved in repeating propagations that mark large areas of the network. A record is therefore kept of marker propagations that start from a single node and are expected to mark a large number of nodes. All the recorded propagations are marking operations of type (a) in the classification given in Section 3.3.

The record of the propagation is kept at the node at which the propagation starts. When a subsequent request is made for performing the same propagation the marker symbol noted in the record is returned, without generating a new marker symbol or marking any nodes.

Recording propagations in this way requires that these propagations are "maintained" when new assertions are added to memory. This is done by extending the propagations across links created for new assertions and marking nodes as appropriate. For example assume that a propagation is recorded that marks all specializations of entity A, and this propagation has marked entity B with a marker M. A new memory assertion (Specialization: C of B) will cause the propagation to be extended from B to C, and C will be marked with M. Extending propagations in this way during memory processing ensures that their markers can still be validly used in later processing. The fact that propagations can be recorded, and the way that the representation of context depends on marked sets, means that the presence of a mark on a node can affect several memory operations, not just one.

Clustering for the indexing scheme

Returning to the scheme for indexing marked sets that is used for searching, the clustering criteria that have been tried with this are now described. Four such methods were tested which will be referred to as the "creation", "specialization", "association", and "random" methods respectively. Each of these methods determines how nodes are clustered to form the indexing tree. More precisely, a clustering method determines the order in which network nodes are selected for placing them into cluster nodes when the indexing tree is being built by the procedure described earlier in this section. The methods also determine how to incorporate new memory entities into this tree when these are created as a result of text processing.

It should be emphasised that these different methods do not affect the final results of memory retrieval and search. This is because the search procedures use the indexing tree only to avoid searching areas of the network that are guaranteed not to include any result nodes (even though these areas may be different for different clustering methods). Hence we need not worry here about the formal properties of such a clustering, only the efficiency to be gained from it. The gain in efficiency derived from the indexing scheme is, as remarked above, improved if marked sets are well correlated with (as opposed to sparsely distributed among) the clusters. The intention, except for the "random" method, is to have "semantic" criteria for clustering that improve the correlation between marked sets and clusters.

In the "creation" method the order in which the entities are chosen for inclusion into clusters is the same as the order in which the entities are created. The majority of the entities in memory are assumed to be created as a result of processing input files written by the person who constructs the knowledge base. The order of memory assertions in these files would then presumably tend to clump semantically "close" information together reflecting the constructor's model of the knowledge being represented.

Both the "specialization" (or "classification") and the "association" methods depend on specialization and correspondence assertions. The clustering order for the methods is the same as the order in which nodes are located by two different types of exhaustive traversal of the network formed by the links representing memory assertions. In the specialization method the traversal is a depth-first search of the network using specialization links and also role-specialization links (i.e. those that link the specialized-role and generic-role arguments of correspondence assertions). For the association method the traversal is a depth-first search that mainly uses the owner-role links from correspondence assertions, though it also uses specialization links because the chains of owner-role links do not span the whole network. Thus for the specialization method entities that are similar (as determined by the classification imposed by the specialization hierarchy) tend to be in the same clusters; whereas for the association method entities that are closely associated (via the owner-role relationships imposed by correspondence assertions) tend to be in the same clusters.

In addition to the three clustering methods just described random clustering was implemented, for the purpose of comparison. This method uses a pseudo-random number generator to insure that there is no systematic semantic relationship between entities that are placed in the same clusters. Informal timing tests confirmed the expected result that random clustering was the least efficient. The specialization method appears to be the most efficient followed by the association and creation methods respectively. Because of this the system was normally run using the specialization

clustering option.

Appendix C presents graphs showing the number of network and tree nodes visited, for different clustering methods and cluster sizes, during the searches performed for processing the two example texts given in Chapter 1. These results should give an idea of how the clustering methods affect searching in the Capture implementation but should not be regarded as experimental results in any strong sense. The relatively small number of entities in memory (about 450) means that the results of these test runs cannot support any strong conclusions. Note, however, that an appropriate semantic clustering should become more advantageous than random clustering as the number of entities in memory increases. But it should be added that it is not necessarily true that there is such an appropriate semantic clustering that could be used in conjunction with a very large memory knowledge base and an arbitrary pattern of processing performed on it. It is, however, more likely that a better than random clustering method could be found if processing the knowledge base follows a more restricted pattern of use, such as that required for performing certain very common interpretation operations.

It turned out to be fairly easy to determine a good initial cluster size (4), by trial and error, for minimising the number of nodes visited during searches, for the particular memory knowledge base used by Capture. There is also a fixed limit on the maximum size to which a cluster can grow as a result of new memory entities being created and incorporated in existing clusters.

When a new entity is created, the system must choose the cluster into which it will be incorporated. The choice of cluster is determined by a function that is consistent with the clustering method being applied. For example, for specialization clustering a cluster that includes a generalization of the entity is chosen, otherwise, if there is no such cluster, a cluster including a role-generalization of the new entity is chosen. When the maximum size is exceeded for a particular cluster, a new cluster is created near the old one in the cluster tree. Of course, the whole network could be reclustered after a large number of updates take place, but this was not done automatically by the system.

3.6 Other techniques for efficient retrieval

Despite the use of the indexing scheme for implementing memory searches and the use of recorded propagations, for very large networks the cost of marking large areas with temporary markers will still be high, perhaps prohibitively so. An extension of the indexing scheme for use with very large

networks is therefore considered. This is aimed at reducing the marking effort during propagate/intersect algorithms. The extension was not included in the implemented system because it was felt that, given the size of the experimental knowledge base, it would not lead to any discernible increase in processing efficiency. However, in principle, using the extension, we could perform "approximate" marker propagations on a high level of the cluster tree and then use the information gained in this way to perform restricted propagations, where necessary, on the network itself.

This would require the introduction of "cluster links" between the cluster nodes as follows. If one or more pairs of nodes in different clusters are linked by some link type, then a cluster link of the same type would be created between the respective cluster nodes. Propagations that use these cluster links are approximate in the sense that a cluster link does not imply the existence of a network link between any two particular network nodes.

For each of the network propagations that we wish to avoid performing, an approximate propagation is performed instead at the higher level using the corresponding links. A desired combination of markers on a cluster node means that the nodes in that cluster might be in the desired set. The paths leading to the cluster nodes with this combination can then be traced back to the sources of the marker propagations. The network nodes included in these paths would then be marked with special "pass" markers. Restricted propagations could then be performed on the main network by going through nodes that have pass markers. This would lead to marking all the nodes in the desired intersection, and should restrict marker spreading in the main network considerably.

Apart from this extension other obvious efficiency enhancements are possible, such as the replacement of the lists used for indicating the set of markers on a node by bit vectors or AVL trees. This kind of very low level efficiency enhancement was not considered to be interesting enough for inclusion in an experimental system. It would be interesting, however, to try to record some of the propagations that start at a set of nodes, rather than restricting recorded propagations to those that start at a single node. However, maintaining these recorded propagations would be more complicated because the sets that they start from can change during memory processing.

As far as clustering is concerned, a possibility that is worth investigating is using the hierarchy derived from memory assertions directly as the search tree. (*) Indexing markers would have to be distinguished from ordinary

---

(*) This was suggested to me by Graeme Ritchie, personal communication.

markers in this case. The scheme would have the advantages of not requiring additional storage space for the indexing tree, and good correlation between the "clustering" and some important marked sets. (Control over the cluster size, however, is not possible. There is also the complication that the memory hierarchies can branch upwards, which entails lower efficiency, or the need to restrict indexing to a minimum spanning tree.)

All the implemented techniques and the suggestions in this section for improving the efficiency of memory retrieval have, as stated earlier, the property that they do not affect the outcome of retrieval operations. A different class of techniques that may be useful are those that depend on further information for determining, in a possibly error-prone way, sets of entities that can be ignored during the retrieval process. This point is returned to when possible applications of the context mechanism are discussed.

3.7 Comments on Capture memory processing

This section makes some general remarks about the Capture memory processing model and summarizes the main points about it that were discussed in detail in previous sections.

The processing model is designed to enable retrieving semi-explicit information stored in memory. "Threshold" conditions on searches allow these retrieval operations to be restricted, if necessary, to entities that are highly relevant to the current context, an issue that will be discussed in detail in Chapter 5. Thus the model is not a general deductive retrieval system in the sense that it could be used for any particular AI application, for example one involving planning. Instead, the emphasis has been on having as simple a basic model as possible, and trying to build a text processing system relying only on this limited form of access to a knowledge base. This is consistent with the simple, perhaps minimal, representation formalism used by the system.

One advantage of having such a simple scheme for memory representation and processing is that it is easier to specify what operations can be performed on the memory knowledge base. There have been calls (see e.g. [Ritchie83]) for more formal specifications of the operations that can be performed on knowledge bases that have been termed semantic networks. I have not attempted to specify formally the operations that can be performed by the processing model; however, the first steps towards this have been taken by specifying the types of marking operation and memory search that can be used to implement retrieval operations. Retrieval

operations were not, however, constrained in the way they combined the available types of marking and search operation.

It is also the simplicity of the marker processing model and of the representation formalism that makes it possible, in principle, to implement memory retrieval on parallel hardware, or alternatively to take advantage of software techniques for improving the efficiency of retrieval operations, as was done for the implemented system. But if arbitrary functions were defined that could access memory representation structures directly, it would not be possible, without general purpose parallel hardware, to have a massively parallel hardware scheme, or to have a uniform indexing scheme that could be used for all software retrieval operations.

Although the advantage gained from a particular clustering method depends, as already mentioned, on the pattern of use, once this method has been chosen, the programmer exploiting the memory processing model described in this chapter is to some extent freed from investing further effort in making individual search operations more efficient. Freeing the programmer using the knowledge base from efficiency considerations is cited by Fahlman [Fahlman79] as a motivation for his parallel hardware design.

As stated earlier, it is difficult to judge how successful the implemented and suggested techniques for improving the efficiency of memory retrieval will be for processing large knowledge bases. This will depend on the particular knowledge base, the clustering method, and, as with McDermott's scheme for indexing Planner Databases [McDermott75], on the pattern of use. With the small knowledge base used in the project, retrieval operations were performed efficiently enough, so the size of the memory knowledge base was not seen as a limiting factor. However, given that the techniques for improving efficiency are expected to be relatively more effective for larger knowledge bases represented in memory, it can be expected that these techniques would be quite adequate if the memory knowledge base was an order of magnitude larger than it is at present. If this turns out to be the case then the use of software techniques should be adequate for applications in the near future, until specialized hardware becomes available. For natural language processing systems, the difficulty of constructing all the other components required for extensive coverage means that the lack of specialized hardware for memory processing is not the current limiting factor.

In any case techniques for efficient simulation of specialized memory processing hardware are useful for experimenting with hardware designs. For example, experience gained in the present project suggests that the number of markers that can be stored at a node in Fahlman's design may be too small. (This could be due to the way context information was represented

with markers in this project). The use of software simulators will be more important for evaluating more complex hardware designs such as Hillis's Connection Machine [Hillis81]. The use of "virtual connections" in his design, instead of direct connections between processing elements, complicates the implementation of retrieval operations, and makes it more difficult to evaluate what the efficiency gains are for a particular application.

As will be discussed in Chapter 5, the use of the indexing scheme to perform threshold searches provides the means for implementing a very flexible context mechanism that is nevertheless computationally acceptable. It will also be explained how the processing model allows traces of memory processing to be used as a type of factor contributing to context.

The low-level implementation of retrieval operations in terms of the processing model described in this chapter will not be discussed when it is explained how these operations are used by the text interpretation and the database capture task components of the system. The design of these components is largely independent of the marker processing model used to implement them. Nevertheless, the efficiency considerations discussed in this chapter are important to the feasibility of using these components in conjunction with larger scale knowledge bases.

Chapter 4

Interpretation

4.1 Interpretation operations in Capture

This chapter describes the operations that interpret the output of the sentence analyser with respect to the current context and the contents of memory. These operations will be referred to as language interpretation memory operations (or simply interpretation operations). The operations described are aimed at solving common language understanding problems such as reference resolution, word sense and structural disambiguation, and inferring implicit relationships between nouns and modifiers. All these problems are considered to be basic to natural language understanding in general; the interpretation operations used to solve them are themselves thus assumed to be domain and task independent although they may exploit domain and task knowledge.

"Interpretation" here is taken (intuitively) to be concerned with making explicit the propositional content of fragments of language and incorporating this content into memory. It is therefore dependent on context and the existing knowledge in memory. The process of interpreting a text does not result in some form of text representation that can stand alone from memory, but rather results in the incorporation of new information into an existing knowledge base, and in this way fits what Schank often refers to as an "understanding" process. (The interpretation process in Capture also results in the creation of new context information - see Chapter 5.)

Since the analyser does resolve certain ambiguities, and its output does indicate some semantic relationships between sentence constituents (as illustrated in the following section), the basis for the interpretation process is initiated by the analyser. However, any analyser that handles the sentences of a text independently from one another cannot be expected to resolve all ambiguities and references presented by each sentence in the text. The interpretation component has the responsibility of handling remaining ambiguities and references, and completing the interpretation process by the incorporation of new knowledge into memory. Thus, in Capture, the basically linguistic processing performed by the analyser is separated from the interpretation process. This separation had the

advantage of allowing these processes to be studied in a focused way, and is similar to the current approach taken at SRI (see [Grosz82]). The analyser is not forced by this approach to make decisions that should really take memory and context into account because it can pass alternative structures to the interpretation component, as described later in this chapter, which can apply context using the mechanism described in the next chapter.

The operations that were implemented are not considered to be an exhaustive list but they are sufficient for processing the example descriptive texts handled by the system. These examples, although much simpler than completely unedited texts, were not trivial and did provide instances of the requirements to be met by the interpretation component in sufficiently challenging forms. The interpretation operations for handling certain constructions are based on algorithms that are similar to the algorithms used by other AI researchers for handling these constructions, and the most relevant parallels of this sort will be indicated when the particular interpretation operations are described. However, Capture integrates these various procedures into an overall processing framework in a coherent way by the common exploitation of the memory and context mechanisms.

The language interpretation operations embody the flow of control within the system as a whole. Thus an operation that interprets a certain type of clause can lead to the evaluation of other interpretation operations such as reference resolution, case specialization, compound noun interpretation, and the interpretation of subordinate clauses. All of these imply memory processing. The way that task specific processing fits into this control structure is explained later.

Interpretation operations are implemented by procedures (in fact LISP functions) and as such are rather unrestricted. However, interpretation operations are intended to have, and do in fact have, many characteristics in common, the most important being as follows. Interpretation operations take as arguments structures produced by the analyser. The operations may evaluate other interpretation operations for processing substructures of their arguments, and then use the results of these subsidiary evaluations. The operations perform memory retrieval using the marker processing model described in the previous chapter. They can create new memory entities and add new specialization and correspondence assertions to memory. New context factors are also created by interpretation operations as described in the chapter on the context mechanism.

At first sight it might appear that the interpretation of the analyser output

is compositional in nature (*) because of the way that the interpretation of structures uses the results of the interpretation of substructures. An example of a possible control sequence is one in which "interpret clause" calls "interpret verb case" which in turn calls "interpret noun phrase". But the interpretation of analyser structures is not in fact compositional because evaluating interpretation operations has side-effects creating structures in memory and altering the current context information. These side-effects can, in principle, influence the interpretation of other structures that are at the same level as, or lower than, the original structure in the analysis tree.

In order to provide necessary background the description of the various interpretation operations is prefaced by a brief account of Boguraev's sentence analyser and the structures it produces.

4.2 The analyser and its output

The analyser is described in [Boguraev79] and [Boguraev82]. It parses the sentences of the input text independently of one another. The analyser interleaves the use of syntactic information and semantic information (based mainly on semantic category restrictions) during the processing of a sentence. The control structure and grammar follow Woods' Augmented Transition Network (ATN) formalism [Woods70].

The entry for a word in the system lexicon contains a list of word senses and the syntactic category information associated with them. Each word sense has a semantic definition which is a Wilksian semantic formula [Wilks75a]. Such a formula is a binary tree of semantic primitives, although the rules for constructing a valid formula in Boguraev's system differ slightly from Wilks' original specification. The most important primitive in a formula is the "head" and this plays a direct role in applying semantic category restrictions. Also important are category restrictions on the fillers of verb sense cases and the "arguments" of prepositions. The restrictions are used to check the semantic well-formedness of structures being built by the ATN interpreter. This is done at points when major constituents are constructed, the most important of these being building the structure for a clause and building the structure for a noun phrase.

The output of the analyser for a sentence is a structure for each reading of the sentence, given that individual sentences may be ambiguous on their

---

(*) i.e. that the interpretation of a structure is only a function of the interpretation of its substructures, see e.g. [Hirst83]

own (whether absolutely or because the semantic power of the analyser is insufficient to resolve the ambiguities is irrelevant). The analyser output structures are case-labelled dependency structures for clauses, and are centred around the main verb sense. The case labels indicate a generic relationship between the verb and its arguments (or sometimes, a noun and its arguments). The labels are taken from an extended set of cases that include the more conventional ones such as "agent", "object" and "recipient", and others such as "mental-object". The case slot fillers are either representations of noun phrases ("noun-args"), representations of states ("state-args") or subordinate clauses with pointers to the actual case fillers. Other information, about e.g. tense and aspect, is also included in the representation of a clause. (The detailed structure of noun-args and state-args will become clear later when the interpretation operations relating to them are discussed.)

An example of a complex analysis structure is the one below produced for "Smith who was a german trader collected P316 which is an arrow". This structure indicates that the sentence was analysed as a top level clause ("Smith collected P316") with verb sense "collect1" and two cases "agent" and "object". The noun group structures for "Smith" and "P316" each fill the "agent" case of an embedded "be1" clause ("Smith was a trader" and "P316 is an arrow"). The adjective "german" led to building an embedded "state-clause" (which is the same as the structure for "the nationality of Smith is german"). The semantic category primitives (e.g. "move" and "man") that follow noun and verb senses in these structures are ignored by the interpretation component because it has access to the more specific memory descriptions of the noun and verb senses.

The analysis structures can be regarded as "meaning representations" (see e.g. [Sparck Jones83b]), but for the purpose of the Capture project they are regarded primarily as data structures encoding linguistic analyses that need to be further interpreted with respect to the current context and the contents of memory. The structure of the Capture system allows it, in principle, to be able to process the output of a more purely linguistic parser (such as the structures used by Bresnan and Kaplan [Kaplan82] in conjunction with "lexical functional grammar"). In this case some of Capture's interpretation operations would have to be made more complex and do some of the work performed in the present system by Boguraev's analyser.

```
(clause
  (type dcl)
  (tns past)
  (v
    (collect1
      move
      (@@
        agent
        ((trace (clause v agent))
          (clause
            (type relative)
            (tns past)
            (v
              (be1
                be
                (@@ agent (n (Smith1 man)))
                (@@
                  recipient
                  ((trace (clause v agent))
                    (clause
                      (v
                        (be2
                          be
                          (@@
                            agent
                            (n (trader1 man (@@ det (a1 one))))  )
                          (@@
                            state
                            (st
                              (n (nationality NIL))
                              (val (German1 kind)))) ))) ))) ))) )
      (@@
        object
        ((trace (clause v agent))
          (clause
            (type relative)
            (tns present)
            (v
              (be1
                be
                (@@ agent (n (P316 thing)))
                (@@
                  object
                  (n
                    (arrow1
                      thing
                      (@@
                        det
                        (an1
                        one)))) ))) ))) )))
```

The interpretation operations implemented in Capture are described below with examples. They include various clause interpretation operations, operations for predicate and argument derivation, noun phrase reference interpretation, relationship interpretation, and for word sense and structural disambiguation.

4.3 Sentence interpretation

"Interpret-sentence" is the highest level interpretation operation below the "interpret-paragraph" operation. This latter operation only alters context information as described in Chapter 5 when context management is discussed. (The creation of context information by this and other interpretation operations will be left to Chapter 5.) For the present purpose it is sufficient to say that, interpret-paragraph having been applied, each of the sentences in the paragraph is interpreted using interpret-sentence.

The interpret-sentence operation performs two functions. The first of these is calling appropriate disambiguation operations for selecting between alternative structures produced by the analyser for the sentence if more than one analysis is produced. The second function performed by interpret-sentence is calling the appropriate clause interpretation operations for the clauses in the selected analysis and handling the relationship between embedded and higher level clauses (the various types of clause interpretation operation are described in later sections of this chapter). Usually, when an embedded clause is encountered, interpret-sentence invokes the appropriate type of interpretation operation for this lower clause, and hands back a memory entity which is used by the operation interpreting the higher level clause. The memory entity that is passed back is the result of interpreting the noun phrase structure that is shared between the higher and lower clauses. This procedure is applied recursively to handle any depth of embedding.

An example is the interpretation of "Smith supplies the machine that is manufactured by Plexir", for which the analysis structure is

```
(clause
  (type dcl)
  (tns present)
  (v
    (supply1
      give
      (@@ agent (n (Smith1 man)))
      (@@
        object
        ((trace (clause v object))
```

```
(clause
  (type relative)
  (tns present)
  (aspect (passive))
  (v
    (manufacture
      make
      (@@ agent (n (Plexir1 *org)))
      (@@
        object
        (n
          (machine
            thing
            (@@
              det
              (the1
              one)))) ))) ))) ))).
```

The result of interpreting the shared structure (for "the machine") would be a memory entity, such as 'P9999', so that the interpretation of the higher clause proceeds as though it were "Smith supplies P9999". The procedure for handling embedded clauses is different for certain "be-clauses", e.g. "P9999 is a computer that is manufactured by Plexir". For this example, the entity 'P9999' is passed down to the embedded clause, which is interpreted as though it were "P9999 is manufactured by Plexir".

The operations for selecting between alternative analyses are described in Section 4.10, after the various types of clause interpretation operations have been described.

4.4 Predicate clause interpretation

There are two predicate clause interpretation operations, which correspond to the two types of predicate clauses that are handled by the system. The types of predicates, and the corresponding clauses, will be referred to as "verbal" and "state" respectively. Each of the two clause interpretation operations leads to the construction of structures in memory that encode the propositional content of the clause. In both cases this involves the creation of a new predicate instance entity in memory.

The two operations will involve subsidiary operations for predicate and argument derivation. These will be described in Section 4.5. Clauses that serve a referential function, such as restrictive relative clauses, are handled by reference resolution operations (Section 4.7), and do not result in structure creation. The interpretation of two different types of clauses, "be" and "have" clauses, is described in Section 4.6. The description which follows therefore applies only to the top level processing of the verbal and state

types of clause.

a) Verbal clause interpretation

Interpreting verbal clauses can be illustrated by the processing applied to the analyser's structure for "Plexir manufactures P777", which is

```
(clause
  (type dcl)
  (tns present)
  (v
    (manufacture
      make
      (@@ agent (n (Plexir1 *org)))
      (@@ object (n (P777 thing)))) )).
```

The first step of processing is creating an instance of a predicate corresponding to the verb. The entity created is an instance of the entity corresponding to the verb sense (i.e. an instance of the entity 'manufacture'), or an instance of a specialization of the entity corresponding to the verb sense. The second alternative applies to the present example, the specialized predicate having been found by a task specific operation (described elsewhere). In both cases, by the associativity of specialization, the newly created entity, E1 say, is a specialization of the verb sense taken from the analyser clause representation.

Each case of the clause is now processed, usually in the order in which the filler of the case appears in the surface text. (*) The case labels are specialized to the newly created predicate instance, E1, (see below). Thus 'agent' is specialized to 'supplier/dbentity' in this case, and the entity 'Plexir' is asserted to fill the 'supplier/dbentity' role for E1 by adding a correspondence assertion to memory. Similarly, 'P777' is asserted to correspond to the 'part/dbentity' of E1.

In the example 'Plexir' and 'P777' are explicit referents for which entities were assumed to exist in memory, and the subsidiary noun-arg interpretation operation that is given, for instance, the "noun-arg" (n (Plexir *org)), does not do any work. However, the situation appears the same from the point of view of the clause interpretation operation when the filler is more complex. Thus for the other types of noun-arg fillers a noun-arg interpretation operation is evaluated, parametrized by information such as the specialized role that the noun-arg fills, returning a unique memory

---

(*) This order is recovered by the system, if possible, by comparing the output of the analyser against the original surface sentence.

entity that the clause interpretation operation can use as a filler. This is done, as described in Section 4.7 on noun-phrase reference interpretation, for definite noun phrases, pronouns, and noun phrases referring to newly mentioned entities and sets of entities. The interpretation of embedded clauses also results in a memory entity so that the processing can continue as in the simple case.

b) State clause interpretation

State clause structures are produced by the parser after analysing adjectival phrases and "be" phrases predicating properties (as opposed to existence). The analyser structures constructed for these cases are the same. For example, the structure given below is constructed both for "P777 is red" and "The colour of P777 is red", (and also for "the red P777" in a sentence containing this noun phrase).

```
(clause
  (type dcl)
  (tns present)
  (v
    (be2
      be
      (@@ agent (n (P777 thing)))
      (@@
        state
        (st (n (colour NIL)) (val (red1 kind)))) ))).
```

The processing of anaphora, embedded clauses, etc. is handled in a manner that is transparent to the clause interpretation operation, in the style described for verbal clause interpretation above. So to explain the state-clause interpretation operation only a simple example is needed, say "P777 is red". In the operation the first step is to create a predicate instance entity, E2 say, in memory. This entity is derived from the "state-arg"

```
(st (n (colour NIL)) (val (red1 kind)))
```

which fills the state case of the clause, using a predicate specialization operation. The value of the state is also taken from the state-arg and is asserted to fill the specialization of the role 'be/state/value' for E2. In the context of the Machines Database the specialization for the example is 'machine/colour/value' and the assertion

```
(Corresponds: red1 to E2 as
        machine/colour/value to relp/machine/colour)
```

is created in memory.

Similarly, the filler of the 'agent' case for the clause is asserted to fill the specialization of the role 'be/state/agent' for 'E2', and the following assertion is created

(Corresponds: P777 to E2 as
machine/dbentity to relp/machine/colour).

4.5 Predicate and argument derivation

In the section on predicate clause interpretation above, it was stated that predicates are derived during the interpretation of verb and state clause structures. In fact, the predicate entities in memory that correspond to the predicates for these clauses can be specialized by a task-specific operation, as is the case when the system is working in the database capture mode. This happens in a way that is transparent to the interpretation component, so that different task-specific components could return different predicate specializations, or a "null" task could simply always return the predicate entity unspecialized. However, ignoring this, locating generic predicate entities for these clause types is straightforward. Thus for verb clauses, the required predicate entity has the same name, e.g. supply1, as the verb sense in the clause, and the derivation of the predicate entity is immediate. The generic predicate entity for a state-clause is derived from the "state-arg" which fills the "state" case of the clause. Thus the state-arg for "These machines are blue" is

(st (n (colour NIL)) (val (blue1 kind)))

and the corresponding generic predicate entity is 'colour/of'. 'colour/of' is found by evaluating a memory retrieval operation that searches for a specialization of 'be/state' that has 'colour' filling its 'be/state/generic' role. The case labels in the analyser structures for the clauses must be specialized, if possible, to the arguments of the predicate entities, or to task specializations of these entities.

In both state and verb clauses, a memory retrieval operation, which can be described as "role-specialization", is used. This specializes role entities in memory to the most specific entities corresponding to them, i.e. the specialized arguments that are owned by the corresponding predicates. This depends on an assumption that Charniak calls the case-slot identity theory [Charniak81]. This amounts to saying that the cases which can be identified from a linguistic analysis of a sentence can be identified with generic slots in a conceptual memory organized around frames.

The names of the generic role entities are derived from the case labels of the analyser dependency structures. For the verb cases of verb clauses the generic role entities have the same names as the case labels. For state clauses the case label 'agent' is mapped onto the role entity 'be/state/agent' and the case label 'value' is mapped onto 'be/state/value'. The generic (i.e. less specialized) role entities though less informative, can still be used to continue processing when they cannot be specialized by the role specialization operation.

As examples of specialized argument derivation, the case label "agent" in the analyser representation for "Plexir manufactures the machine" is mapped onto the entity 'agent' which is specialized, via 'make1/agent' to 'manufacture/agent', and the same label "agent" in the representation for "This artifact is fragile" is mapped onto the entity 'be/state/agent' and then specialized to the entity 'condition/of/possessor'.

4.6 Be and Have clause interpretation

This section describes the interpretation operations used for handling two special types of clauses, "be-clauses" and "have-clauses". These interpretation operations are different in type from the predicate clause interpretation operations described earlier because the clauses are mapped directly onto memory assertions, rather than memory entities representing a predicate instance. This is a consequence of the way memory is structured, i.e. specialization and correspondence assertions are closely connected with the classification and relationship senses of "be" and "have" respectively.

a) Be-clause interpretation

This operation handles clauses like "P333 is a computer", in which "is" asserts that an object can be described as fitting a certain type, i.e. generic concept. This sense of the verb "be" is output as "be1" by the analyser, instead of "be2", which is used for state assertions, (these reflect the use of nominal and adjectival subject complements respectively). The structure for "P333 is a computer" output by the analyser is given below.

```
(clause
  (type dcl)
  (tns present)
  (v
    (be1
      be
      (@@ agent (n (P333 thing)))
      (@@
        object
        (n (computer thing (@@ det (a1 one)))) ))) ).
```

The memory assertion created by the be-clause interpretation is a specialization assertion. The operation involves finding, or creating, the entities for the arguments of the specialization assertion corresponding to the be-clause and then adding this assertion to memory. This is straightforward for the example clause, and the new memory assertion is

(Specialization: P333 of computer (instance)).

The specific entity, 'P333', is taken from the 'agent' case of the clause, and the generic entity, 'computer' in this example, is derived from the 'object' case. If there is no entity 'P333' in memory, a new entity is created.

More complex examples require further processing in order to determine the generic and specific entities. Examples are:

1) This machine is a computer.
2) Wintron is a computer manufacturer.
3) P7780 is a disc-drive that is made by Plexir.

In (1) the noun phrase "This machine" is handled by reference resolution and an entity is returned and used, in the same way as for definite references in verb-clause and state-clause interpretation. In (2) the compound "computer manufacturer" is analysed by the compound noun interpretation mechanism run in "creation mode" (described later). The compound noun interpretation creates, or locates, a memory entity that corresponds to "computer manufacturer", E17 say, and the new memory assertion for the clause is

(Specialization: Wintron of E17 (instance)).

For sentence (3) the interpretation of the main clause is straightforward, but the embedded structure (for "disc-drive made by Plexir") cannot be handled in the normal way because this would lose the information that the referent of "disc-drive" in this clause is known to be 'P7780'. Thus before the embedded clause is handled by evaluating a verb-clause interpretation operation, 'P7780' is attached to the noun-arg structure for "disc-drive", so

precluding normal definite reference resolution, and the embedded structure is then interpreted as "P7780 is made by Plexir".

b) Have-clause interpretation

As with be-clause interpretation, the interpretation of "have" clauses leads to the creation of memory assertions, to represent their propositional content, which in this case are correspondence assertions. An example is the clause "it has a bolt", for which the dependency structure is

```
(clause
  (type dcl)
  (tns present)
  (v
    (have8
      have
      (@@ agent (n (it dummy)))
      (@@
        object
        (n (bolt1 thing (@@ det (a1 one)))) ))) ).
```

The correspondence assertion that is created for this would be (assuming 'P9980' is the referent for "it")

```
(Corresponds: E18 to P9980 as
        machine/component to machine)
```

where E18 is a newly created entity that is asserted to be a specialization of 'bolt1'. A pair of entities is derived from the 'agent' and 'object' cases of the clause. A memory operation (described in Section 4.8 below) for finding the relationship between this pair of memory entities is evaluated in order to find as specific a relationship as possible that can be used in the correspondence assertion. Thus this interpretation operation tries to make the relationship described by the have-clause more explicit before its propositional content is incorporated in memory. Again, this interpretation operation can lead to the evaluation of reference and compound noun operations. An example of this is "The machine supplied by Smith has an adjustment bolt."

4.7 Noun phrase reference interpretation

As explained in the previous sections, reference resolution is performed as a subsidiary operation to the clause interpretation operations. The resolution of definite references made by noun phrases is essential to any text processing system that seeks to exploit the information conveyed by texts. This is certainly true of the database capture task, tasks like story understanding and summarizing, and even tasks that might be able to do without "in-depth" understanding, for example translation.

The interpretation operation described in this section resolves the definite references made by various types of noun phrase by locating a memory entity, or set of entities, that corresponds to the noun phrase. The noun phrase types dealt with by the interpretation operations have included pronouns, definite noun phrases, compound nominals, possessives, and restrictive adjectival and relative clauses. Examples of these are:

It is made by Marconi.
P7720 comes from there.
These machines are red.
The terminal manufacturer makes P9920.
The cost of Mikota's peripheral is 235.
Smith supplies the blue machine.
Jones supplies the machine that is manufactured by Plexir.

The context mechanism is used to disambiguate possible candidate referents when the constraints derived from the analyser representation are not strong enough to identify a referent uniquely. The application of context to reference resolution occurs after the constraints have been identified by the reference interpretation operation, in a manner described in detail in Section 5.3. This section is concerned with how the constraints on referents are derived and how the output of this process is formulated as a memory search request.

The main constraints on referents are encoded in the request as markers. Each of these markers will have been used to mark a set of memory entities satisfying a constraint, the derivation of constraints will be described later in this section. The marking involved in this process is either performed specifically for evaluating a reference interpretation operation, or will have been performed previously in the case of recorded propagation markers (Section 3.5).

The memory search request is a request for an intersection search to be performed using the constraint markers, so that entities satisfying the constraints can be located. The information in the request other than the constraint markers is used to control the way the search is performed. In

the requests built by reference resolution operations this information specifies that the initial search should be parametrized by a context threshold, that the constraints on the search can be weakened if they are too strong and no memory entities satisfy all of them, and that context should be used to limit the number of referents to the "number expectation" (explained below) that is encoded in the request. The request is evaluated, in the manner described in the section on memory search requests, so that the result of the request is a memory entity, or set of entities, that satisfies as many of the constraints as possible, agrees with the number expectation, and is the best choice with respect to the current context information.

Mellish [Mellish80] uses a reference resolution mechanism which monitors the set of entities satisfying constraints that are derived from the text, some of them by inference. These constraints are accumulated until they are strong enough to identify the referent uniquely. Reference resolution in the Capture system is similar to Mellish's mechanism as far as the accumulation of constraints is concerned. It would also be possible to allow the reference resolution operation to take advantage of further constraints derived from inferences that are valid in the domain of discourse.

The constraints that can be derived from a sentence are applied immediately by the reference resolution operation used in Capture, the context mechanism being used to select between alternative candidates that satisfy the constraints. For texts in general, and for the example texts processed by Capture, it is not possible to wait until the accumulated constraints identify a unique referent because this does not always happen. (This does tend to happen, however, in the mechanics problem texts processed by Mellish's system, presumably because in such texts the problem needs to be described without any possibility of ambiguity.) The focus mechanisms that are used by Grosz [Grosz77] and Sidner [Sidner79] for definite reference interpretation are discussed in Chapter 5.

The way the constraints are derived from analyser structures is as follows. If the head noun sense in the noun phrase is a generic entity in memory, then a constraint marker is generated from a marker propagation that marks all specializations (including role specializations) of this entity. For example, for the phrase "the machine" for which the noun-arg representation produced by the analyser is

(n (machine thing (@@ det (the1 one))))

all entities below the entity 'machine' in the specialization and correspondence hierarchies will be marked by the marker for this constraint. The constraint marker for pronouns is derived by marking the specializations of a generic entity in memory that subsumes the entities to

which the pronoun can refer. For example, if the head of the noun-arg is "it" then the marker used for this constraint will be the one that marks all entities below the entity 'inanimate' in the hierarchies. Other special anaphoric words such as "there" and "then" are treated similarly. Thus from the noun-arg (n (there1 spread)) in the analysis of "P7720 comes from there" a constraint marker marking specializations of 'where/ent' is included in the search request.

The specialized role entities derived from the case labels of verb and state clauses (as described in the section on predicate and argument specialization) also generate a constraint marker. For example, the 'agent' case in the analysis of "He supplies the disc-drive" is used, in the context of the Machines Database, to derive the entity 'supplier/dbentity'. Any entities that can be described as a 'supplier/dbentity' are marked by the marker for this constraint.

Nominal modifiers, such as "terminal" in "The terminal manufacturer" and possessors such as "Mikota" in "Mikota's peripheral" are used to derive another constraint marker. This marker is generated by a relationship interpretation operation (evaluated in reference mode as described in Section 4.8). Relative have-clauses, for instance the relative clause in "The machine that has the bolt", and also noun phrases such as "The machine with the bolt", are treated in the same way as possessives, so that relationship interpretation can be used to generate a constraint based on a more specific relationship than that suggested by "has" and "with".

Predicative restrictive relative clauses are used to generate a constraint marker. This is the case for the relative clauses in "The machine that is manufactured by Plexir", and "The machine that is blue". In the first of these examples the constraint marker marks the arguments of specializations of the entity 'manufacture' that have the 'manufacture/agent' role filled with 'Plexir'. For the second example, the entities marked are arguments of specializations of the derived predicate 'colour/of' that have the 'colour/of/colour' role filled with 'blue1'. Since the analyser representation for "the blue machine" is the same as for the second example "that is blue", the predicate constraint marker generated is the same for both the adjectival noun phrase and the relative clause examples.

Finally the "number expectation" for the referent (set) is derived from the noun phrase representation and included in the search request. This can be either an integer, or "many" i.e. an unknown number greater than one. The number expectation is "1" for singular noun phrases and "many" for plural noun phrases, for instance "the IBM machines", that do not indicate the exact number of referents to be expected. This additional information is

available when the determiner is "both" as in "both machines" in which case the number expectation is "2"; or from number words, which are treated as adjectives, as in "the three machines", giving "3".

Details of the application of context information for reference resolution and the algorithm for plural reference are given in the chapter on context.

4.8 Relationship interpretation

Of the types of interpretation operation that are subsidiary to clause interpretation, it remains to describe "relationship interpretation". Relationship interpretation operations are used to derive explicit relationships from language constructs in which relationships are implicit or vague. These types of language construct include compound noun phrases, possessive noun phrases, prepositional phrases using "with", and "have" clauses. The interpretation of relationships is done in two different modes, which will be called "reference" mode and "creation" mode respectively. These two modes, roughly speaking, reflect the given/new distinction. In "creation" mode evaluation can lead to the creation of new memory structures, whereas the results of evaluation in "reference" mode can be used for memory searches e.g. for reference resolution.

There has been less work on relationship interpretation as compared with, say, reference resolution. However, the problem of determining implicit or vague relationships was addressed in natural language work at SRI (where it is classified under "basic pragmatic functions" [Grosz81]), and more recently by Steinacker and Trost [Steinacker83]. Compound noun analysis (see e.g. [Sparck Jones83a] for a discussion of the difficulties involved), is treated in the Capture system as a special case of relationship interpretation. Processes for the interpretation of compound nouns have been investigated by McDonald [McDonald82]. The approaches adopted in these projects all depend on the use of a conceptual memory, or knowledge base, the relationships identified being chosen from among those represented in memory. In particular, McDonald uses the NETL knowledge representation language and marker processing for implementing his compound noun interpreter. The algorithm used in Capture for relationship interpretation is similar to the basic algorithm used by McDonald. McDonald augments his basic algorithm with heuristics for selecting between possible interpretations, a function which can be performed, in the interpretation framework used by Capture, by the context mechanism (see Section 5.3).

The results of relationship interpretation operations are memory correspondence assertions or memory entities taking part in inherited

correspondences. Relationship interpretation makes use of a number of memory retrieval operations for locating the relevant correspondence assertions. There are three basic memory retrieval operations for relationship interpretation:

Op-1. Find a specialization of entity A that is the owner in a correspondence assertion (or inherited correspondence) in which entity B is (or can fill) the role.

Op-2. Find a specialization of entity A that is the role in a correspondence assertion (or inherited correspondence) in which entity B is (or can be) the owner.

Op-3. Find a specialization of entity A that is the role in a correspondence assertion in which entity C is the owner, where C is also the owner in a correspondence assertion in which the role is a specialization of B.

Each of these three retrieval operations will in general return more than one result. Variants on these operations reflect how context is used, whether only most specialized entities are to be accepted, and whether the results from the three different types are to be combined (or compared).

Because of the use of these operations in compound noun analysis, the arguments that were called "A" and "B" above will be referred to as the "head" and the "modifier" respectively. These head-modifier pairs are recognised in analyser representations as follows.

a) Possessives. The analysis for a possessive noun phrase such as "P777's manufacturer" is

```
(n (manufacturer man (@@ poss-by (n (P777 thing)))) )
```

where 'manufacturer' is the head and 'P777' is the modifier.

b) Attribute modifier. The analysis of a prepositional phrase using "with", for instance "the machine with the bolt" is

```
(n
  (machine
    thing
    (@@
      attribute
      (n (bolt1 thing (@@ det (the1 one)))) )
    (@@ det (the1 one)))).
```

Here 'machine' is the head and 'bolt1' is the modifier.

c) Nominal modifier. Simple noun-noun modification is represented in analyser structures which do not postulate any relationship that might hold between them. For instance the structure for "the computer manufacturer" is

```
(n
  (manufacturer
    man
    (@@ det (the1 one))
    (## nmod (((n (computer thing)))) ))).
```

d) Ambiguous nominal modifier. Since the analyser does not do any (non-syntactic) analysis of noun-noun compounds, regarding nominal modifiers simply as a list attached to the head, it is not possible for it to select among possible noun senses corresponding to the surface noun modifiers. The analyser representation contains a list of the noun senses for the ambiguous modifier. For example the structure for "The terminal manufacturer" is

```
(n
  (manufacturer
    man
    (@@ det (the1 one))
    (##
      nmod
      (((n (terminal1 thing))
        (n (terminal2 spread)))) ))).
```

One of 'terminal1' and 'terminal2' is selected as the modifier (Section 4.9), 'manufacturer1' being the head.

When interpreting all these structures in "creation" mode, the result of the relationship interpretation operation is a single (newly created) memory entity. The result of relationship interpretation for a modifier-head pair in "reference" mode is a set of specializations of the head.

A procedure is needed for dealing, in "creation" mode, with situations where there is more than one modifier on a noun. These may be different types of modifier (in the general sense of "modifier" adopted here) as well as multiple nominal modifiers. The algorithm that was implemented repeatedly applied the operation for creating a new entity from a modifier-head pair so that the result of each application was used as the "head" in the following application. However, the problem of determining the order in which this should be done was not solved satisfactorily, even though a first

approximation to this was tried. (*) This problem also arises in the case of multiple nominal modifiers in "reference" mode. But interpretation of "possessive", "attribute", and "nominal" modifiers happens independently in "reference" mode. Each of these modifiers, if present, is combined directly with the head of the noun phrase to produce a set of specializations of the head that are then used for reference resolution.

The rest of this section will give some examples of relationship interpretation for simple modifier-head pairs, first in "creation" mode and then in "reference" mode.

a) creation mode

The interpretation in "creation" mode of "computer manufacturer", taken from the sentence "Plexir is a computer manufacturer", say, proceeds as follows. A retrieval operation, Op-1 above, when used in relationship interpretation, locates the relationship represented by the correspondence assertion

(Corresponds: manufacturer/goods to manufacturer as
organisation/obj to organisation)

because it is possible for 'computer' to fill the 'manufacturer/goods' role of 'manufacturer'. Two new memory entities, E1 and E2 say, are created and then the following memory assertions are made

(Specialization: E1 of manufacturer)
(Specialization: E2 of computer)
(Corresponds: E2 to E1 as
manufacturer/goods to manufacturer).

The entity E1 representing the concept "computer manufacturer" is returned as the result of the interpretation operation.

Other actions might have occurred if the contents of memory had been different. For example, if the original correspondence assertion was not present, but instead, the following correspondence assertion was present and located by retrieval operation Op-2

(Corresponds: machine/manufacturer to machine as
goods/manufacturer to manufactured/goods),

(*) The algorithm proposed by Marcus [Marcus80] for determining the structure of compounds is promising for solving this problem since it is well suited to the approach to relationship interpretation being described here, which would play the role of his assumed semantic function.

then a new memory entity, E3 say, and the following memory assertions would be created.

(Corresponds: E3 to computer as
machine/manufacturer to machine)

E3 would be the result of the interpretation operation in this case. The retrieval operation Op-3 is not used in "creation" mode, and so if neither Op-1 or Op-2 had suggested any possibilities then the result of the operation would be just the entity 'manufacturer' i.e. the head. If Op-1 suggests one or more entities that could be owned by the head and at the same time Op-2 suggests one or more entities that could be filled by the head, then all of these entities are compared by the context mechanism, which selects one of them. The actions taken after this depend on whether Op-1 or Op-2 suggested the entity, and follow the pattern illustrated by the "computer manufacturer" example.

Finally, we consider an example of relationship interpretation in "creation" mode that is used to handle a "with" prepositional phrase. When interpreting the noun-arg representing "a pot with a lid", 'pot' and 'lid' are the head and modifier respectively. Op-1 locates the memory correspondence assertion

(Corresponds: cover1 to container1 as part2 to whole).

This is located because 'lid1' is a specialization of 'cover1' and 'pot1' is a specialization of 'container1'. Two new memory entities, E4 and E5 say, are then created together with the memory assertions

(Specialization: E4 of pot1)
(Specialization: E5 of lid1)
(Corresponds: E5 to E4 as cover1 to container1).

E4 is the memory entity that is the result of the interpretation operation i.e. stands for the concept "a pot with a lid".

b) reference mode

Relationship interpretation in "reference" mode is a search operation that does not involve the creation of new structures in memory. The three basic operations Op-1, Op-2, and Op-3 are applied, each of which returns a (possibly empty) set of specializations of the head of the pair. The union of these sets is returned as the result of the operation, unless all the sets are empty in which case the head itself is returned as the result. If the set of candidate specializations of the head is non-empty, then the reference resolution operation that initiated the relationship interpretation will mark the entities in this set with a marker that is used along with the other

constraint markers in the reference resolution process. In other words, the relationship interpretation operation can be thought of as a request to search for specializations of the head that embody the constraint indicated by the modifier.

An example is the interpretation of "the Mikota peripheral" given that the following sentences have already occurred in the text being processed.

Plexir makes P7770 which is a disc-drive.
P9900 is a disc-drive that is made by Mikota.
P9000 is a computer that is manufactured by Mikota.
Mikota manufactures P9200.
P9200 is a printer.

For "the Mikota peripheral", the basic operation Op-3 searches for specializations of 'peripheral' that are the role in a correspondence assertion in which C is the owner, where C is any entity that is also the owner in a correspondence assertion in which 'Mikota' is the role. The correspondence assertions located by Op-3 in this case would have predicate instances that are specialization of 'make1' or 'manufacture' as their owners. The specializations of the head returned by Op-3 in this case are 'P9900' and 'P9200'. Similarly, the entities returned after interpreting "the peripheral manufacturers" would be 'Plexir' and 'Mikota'.

The interpretation of the possessive structure "Mikota's peripheral" would proceed in exactly the same way as that of its noun-noun compound equivalent illustrated above, again returning 'P9900' and 'P9200' as the possible specializations for the head.

An example that uses Op-1 is the relationship interpretation for "the machine with the bolt" in the context of a previous sentence such as "Plexir manufactures P7200 which has a bolt". The sentence providing the context would have led to the creation of an instance of 'bolt1', 'E19' say, and the correspondence assertion

(Corresponds: E19 to P7200 as machine/component to machine).

Op-1 searches for specializations of 'machine' which are in the owner position in which specializations of 'bolt1' are in the role position. This locates the correspondence assertion just mentioned and returns 'P7200' as a candidate specialization for 'machine'.

4.9 Disambiguation operations

The interpretation operations described so far assume that a single analyser structure has been selected for the sentence being processed. As explained

earlier in this chapter the analyser outputs alternative structures reflecting ambiguities that it cannot deal with. The disambiguation operations of the interpretation component that choose between these alternatives are described in this section.

The alternative structures produced by the analyser are usually whole sentence readings. An exception to this, in which alternative substructures are indicated within a sentence representation, are alternative modifier structures for compound nouns. The analyser produces such alternatives either because they correspond to valid alternative analyses at the sentence level, or because the analyser does not have access to the (domain) knowledge that could be used for disambiguation. The operations described in this section handle three types of ambiguity exhibited in the output of the analyser; first, choosing between alternative analyses of a sentence that reflect word sense ambiguity; second, choosing between alternative senses for modifiers in nominal compounds; and third, choosing between alternative analyses of a sentence with different case relationship structures. These disambiguation operations depend on the context mechanism, and their main action is to derive the alternatives they are meant to handle explicitly and present them in a form that allows the context mechanism to choose between them.

If two or more of the three types of ambiguity just mentioned occur in the analyser output for a sentence, then these are handled in the following order. The analyses are partitioned into groups that have the same word senses (ignoring for the moment multiple senses of modifiers in compound nouns). One of the groups is then selected using the sense disambiguation operation. The case structure disambiguation operation is then applied to this group of analyses to select one of them. The selected analysis is then interpreted by one of the clause interpretation operations, and this leads to noun modifier sense selection when ambiguous modifiers are encountered during the interpretation of the analysis. Descriptions of the three disambiguation operations and some examples are now given.

The word sense selection operation described below subsumes the association based technique described by Philip Hayes [Hayes77b]. This is because the context information on which disambiguation depends in Capture combines the type of memory association information used by Hayes with other types of context information as described in Chapter 5.

1) Analysis selection based on word senses

The combinations of word senses present in each of the candidate analyses are extracted. For example, the combinations for the two analyses of "The

printer is green" are (be2 green2 printer1) and (be2 green1 printer2), which may be paraphrased as "The man who prints is a novice" and "The colour of the printing machine is green" respectively. A score is calculated for each of the combinations. This score comes from the context activation for the senses (as provided by the context mechanism) with an additional bias weight for senses that do have memory entities. The bias is introduced so that senses with memory entities associated with them are preferred over those that do not, even when there is no context information attached to the memory entities. The result of the operation is simply the analysis (or set of analyses) for which the sense combination has the highest score. If more than one of the combinations share the highest score, then one of these would be chosen at random.

2) Selection of noun modifier senses

As mentioned above, the analyser does not produce alternative sentence analyses for readings with different possibilities for the senses of noun modifiers in compound noun phrases. Instead, the alternatives are presented as a substructure of a single sentence representation (see [Tait83]). This substructure only indicates the order in which nominal modifiers occurred in the surface sentence and a list of noun senses for each of them. This is because the analyser does not have available to it the knowledge needed to determine the structure and meaning of noun-noun compounds. An example is listing the two senses of "terminal" in the structure for "The terminal manufacturer" (which was given earlier in the previous section). The modifier sense disambiguation operation simply selects, from the list of senses for each modifier, the sense that has the highest context activation as provided by the context mechanism. The selected sense is then used by a relationship interpretation operation. An alternative strategy would be to use relationship interpretation to propose possible interpretations starting with the different senses and then select among these on the basis of context activation, but it is not clear whether this more complex strategy would work better.

3) Selection of analyses on the basis of case structures

An example of a sentence for which the analyser produces more than one case structure is "P4740 is manufactured by P5050's manufacturer in London". In one of these structures the prepositional phrase "in London" is attached to the verb by the case label 'location'. In the other analysis the prepositional phrase is attached to the noun "manufacturer" again using the label 'location'. The disambiguation operation works by deriving from each analysis structure a set of entities representing specializations of the case

labels present in the structure. This set of entities will consist of 'manufacture/agent' 'manufacture/obj', and 'manufacture/loc', for the analysis in which the prepositional phrase is attached to the verb and the set of entities derived from the other analysis consists of 'manufacture/agent', 'manufacture/obj' and 'location'. As with the word sense disambiguation operation, a score is calculated for each of the entity sets and the analysis for which the entity set has the highest score is chosen as the result of the disambiguation operation. The score is based on the sum of the context activations of the entities in the set, together with a bias towards more specialized entities over less specialized ones. In the example given it is this bias that leads to the choice of the structure with the prepositional phrase attached to the verb, because 'manufacture/loc' is more specialized than 'location'. However, in situations where all the case relations have been specialized, the choice of structure depends solely on the context activations of the memory entities to which the case labels were specialized.

4.10 Discussion of coverage

The implemented interpretation operations described in this chapter clearly only handle a small subset of English construction types. It is possible to extend the coverage of the system to other, less common constructions, both by making the implemented operations more sophisticated and by including new interpretation operations. Some suggestions along these lines will be made below. However this still leaves a number of quite important language phenomena, representing a wide range of constructions, that cannot be handled by the framework for parsing and interpretation used in the Capture project. Some of these phenomena will be discussed under the heading of "problems" at the end of this section.

a) extensions

The possible extensions of coverage within the framework adopted in Capture include the following.

-- The handling of some adjectival phrases should make use of relationship interpretation (Section 4.8). An example is "the lidded pot" for which relationship interpretation should proceed as it does for "the pot with the lid".

-- There are a number of constructions for which specialization assertions should be created in a way similar to their creation during the

interpretation of be-clauses. Examples are "P300 is a fine example of shell currency", and "These artifacts were used as currency in the Massim area".

-- During the interpretation of be-clauses it could be checked whether a relationship (rather than the normal type inclusion) was intended. An example is the second sentence in "The collection includes a number of heads from the Pacific. The Daui heads are human". (*)

-- Reference to objects that have not been mentioned explicitly, but which only implicitly exist in the descriptions of instance entities in memory. For example the "ornaments" in "This canoe and its ornaments were collected in Daui". The memory representation is well suited to implementing reference resolution of this type.

-- The interpretation of noun phrases could be extended to allow reference resolution to cover generic entities. An example of a noun phrase for which reference resolution should return a generic entity is "the dodo" in "The dodo is extinct". Again, the memory representation used in Capture is suitable here, but the circumstances under which reference to generic entities is applicable is still a matter of controversy among linguists and philosophers.

-- Creation of instances of entities of known type, and their identification with named objects when the name is given as new information. A simple example of this is "Haddon collected a jug from Daui. This is P259".

-- A case of compound noun phrase interpretation that could be handled is when the compound introduces a new object as well as conveying information about it. An example is "the canoe paddle P790" in the sentence "The canoe paddle P790 was collected by Nilfisk".

-- Definite noun phrases with "other" could be handled as another case of reference resolution. The choice of entity satisfying the reference constraints could be the "second best" candidate, as determined by context, or a candidate closely associated with the "best candidate". An example is "The other spear comes from Woodlark".

b) problems

(*) Such examples can be thought of as ellipsis, but the analyser would not recognise them as such, the analyser indeed does not handle ellipsis at all, and other forms of ellipsis are problems for Capture: see below.

Turning now to limitations on the coverage of constructions that could be handled by developing the interpretation operations of the Capture system, there are at least the following three categories of problem phenomena. The phenomena in category (1) are judged to be easier to deal with than those in category (2), which are in turn considered more tractable than those in category (3). The problems in category (1) are a consequence of the current state of the analyser; it is difficult to see exactly how the work involved in solving them would be partitioned between a better analyser and an extended interpretation component. (2) and (3) are clearly problems for Capture regardless of the quality of the analyser.

(1) Problems due to the incomplete coverage of grammatical constructions by the analyser. In this category we have sentences with conjunctions (except for simple noun phrase conjunctions); sentences for which punctuation determines the parse; certain types of elliptical sentences; sentences that can only be analysed with the help of a phrasal lexicon; and sentences in which relationships are determined by the surface word order. Examples are

"Plexir manufactures and supplies P2000"
"Haddon collected the spear and the jug from Keroka"
"Haddon collected the spear, and the jug from Keroka"
"So does Mikota"
"Haddon tried to pull the wool over their eyes"
"Clark and Nilfisk travelled to Daui in that order"
"P300 is a shell necklace in good condition".

(2) There are a number of phenomena that can only be handled by complex (but probably feasible) extensions to the memory representation and interpretation mechanisms. These include negation, constructions conveying temporal information, and conditional constructions. Examples are

"Smith does not supply this machine"
"The artifact must have come from Daui unless he
visited Keroka before 1890".

(3) Phenomena that would require new mechanisms to be incorporated in addition to those forming the framework on which the Capture system is based. These include modality, simile, and metaphor. For example

"The artifact probably comes from Woodlark but
could be from Keroka"
"The spear head is like a razor".

In addition to the types of construction listed in this section, there are constructions that cannot be handled by the database capture component because they present difficulties for the process of translation into database statements, given, in particular, the rather restricted data modelling

capabilities of current relational databases. These will be discussed in Chapter 6.

In conclusion of this section, and of the chapter as a whole, I should emphasize the underlying reason for the choice of the types of construction handled by the interpretation component. The types chosen cover common constructions that occur frequently. The aim was to demonstrate that a significant number of common interpretation problems can be handled in a coherent framework by the exploitation of the memory and context mechanisms provided. It appears that these mechanisms can support an interpretation component with a good complexity/performance ratio; this ratio being important for the technological methodology adopted in this project. The interpretation operations that were described often use the context mechanism as an "oracle" for selecting between memory entities; this mechanism is the subject of the next chapter.

Chapter 5

Context Mechanism

This chapter discusses the context mechanism used by the Capture system. The representation, application, and management of context by the mechanism are described in sections 5.2, 5.3, and 5.4 respectively. There are also sections discussing some implications of the mechanism and its applicability to other areas of text processing. Section 5.1 gives general motivation for the design of the context mechanism; more detailed comparisons with existing AI mechanisms for context are given in Section 5.6.

5.1 Use of context in text processing

A broad definition of the context with respect to which a fragment of text is interpreted is that it is the knowledge the system has of the state of the world, with further information saying which parts, or views, of this knowledge take precedence at a given point, or for a particular text. In the work reported here "context" is used to refer only to the additional "precedence" information, and the context mechanism is concerned with the way that this information is accumulated and used during text processing. The knowledge that the system has of the world is stored in "memory"; this knowledge includes the knowledge resulting from the interpretation of the earlier part of the text. Thus while the broad definition of context includes the contents of memory, context in the more restricted sense used here is limited to information about how, at any point in processing, the existing contents of memory take part in the interpretation of the incoming text.

Context information is needed to handle a key problem in interpretation: given the potentially very large size of the system's knowledge base, the problem is how should the system choose the right fragments of knowledge so as to restrict the possible interpretations, inferences, and searches that are part of its responses to language constructs.

The information for specifying which fragments of knowledge take precedence has been encoded in different ways in various AI approaches to context. Thus a number of systems have approached the restriction problem by organizing the knowledge base into sections, often called "frames" [Charniak78], so that when one of these sections has been activated the

retrieval and inference operations are confined to the knowledge represented in that selected section. This approach is taken further when the structure once activated, guides the interpretation of the text, the interpretation being viewed as a process of instantiating predictions made by the structure (see e.g. [DeJong79]). This active direction of text interpretation is most naturally associated with the action or event sequence structures commonly referred to as "scripts", but it should be noted that when more static (i.e. non-action) structures are used they can still exert a quite powerful influence on the interpretation process, by the tacit assumption that the input text is supplying fillers for the slots in the structure. (*)

Two problems that are associated with approaches based on frames and scripts are firstly that the correct frames or scripts are difficult to select ("the frame activation problem"), and secondly, that the rigidity of the structures, especially scripts, means that systems based on this approach can only handle texts that fit the predetermined structure. Work at Yale on Memory Organisation Packets (MOPs) is a partly successful attempt to solve the problem of script rigidity. MOPs are defined by Schank as follows ([Schank82b] p. 97): "A MOP consists of a set of scenes directed towards the achievement of a goal. A MOP always has one major scene whose goal is the essence or purpose of the events organized by the MOP." The knowledge from many MOPS can be drawn on to construct a single script for the text being processed, so that the knowledge used for interpreting the text is less rigid, and so that the problem of having the same information represented in many scripts is avoided.

As discussed in the chapter on memory processing, Fahlman [Fahlman79] has advocated the use of parallel hardware as an alternative solution to enabling deductive memory retrieval operations to be performed on a large knowledge base. Thus, instead of restricting the retrieval process to information in a frame corresponding to the current context, the whole of the knowledge base is operated on in a well specified parallel manner (see [Charniak82] for discussion).

This parallel approach shifts the main use of context from constraining the scope of knowledge structures that retrieval operations use to choosing between alternative results for these operations when they do not produce a unique result. But this second mode of context application is also likely to be needed to supplement the first if the knowledge structures used for context selection are looser than those of early script systems, as they are when

---

(*) The term "script" has been used for various types of structures that guide text processing, but I will use it in the more common action-sequence sense adopted originally at Yale, see e.g. [Schank75].

MOPS are used (see [Lehnert83]). This is because loosening the rigid expectations makes it more likely that alternative expectations are derived, from different active MOPS, say. (As far as I can tell not much attention has been devoted to the full consequences of this problem by protagonists of MOPS and related structures.)

The use of context to disambiguate between alternative results of memory operations can be thought of in terms of the current "focus". Focus is related to the notion of context used here in that while context gives information about the relevance of knowledge, focus specifies the entity (or the group of entities) that is currently most relevant. In other words focus is a derived notion that is determined by the current context information. (The relationship between focus and Capture's representation of context information will be discussed more precisely later.) Because of this, selecting memory entities on the basis of context information can often be thought of as choosing between them on the basis of how close they are to the focus of the discourse. Also, returning to the issue of the application of context for enhancing efficiency, the notion of focus can play a role because the searches for the results of memory operations can begin with the items that are in focus (see [Grosz77]).

Interpreting, for Capture, the view of context discussed in general above amounts to the following. In the Capture system, the knowledge base, including the results of earlier interpretation, are represented in memory and made use of via memory operations. For the purpose of general text processing, i.e. excluding task specific processing, the use of context information amounts to restricting the processing and results of the language interpretation memory operations that were discussed in the previous chapter. Before the details of the Capture context mechanism are given, two preliminary points should be made. First, as remarked in the chapter on interpretation, the structures created in memory as a result of processing (part of) a text do not include a representation of the structure of the text of the type imposed by "text grammarians". However, some information about the structure of the text does get encoded as context information (in particular, as "recency of mention" factors that effectively make available some information about the order of the original text). Second, another aspect of context required for general text processing is syntactic context, especially of the preceding sentence. It may be argued that this does not fit in with viewing context as "that which restricts memory operations". However, if knowledge of syntactic structures, and the building of instances of these occurs in memory, as it does for the RUS system [Bobrow80] for example, then it might be possible to subsume syntactic parallelism within this characterisation of context. No attempt has been made to model this aspect of the context required for text processing in the present version of the system, and indeed, as mentioned already, the

analyser processes each sentence independently from preceding ones.

Returning to Capture, the way in which context is applied to restrict the results and processing of memory operations depends on the notion of "context activation" for memory entities. At any given time during the processing of a text, a context activation for each entity in memory can be derived from the context information available at that point in processing. The context activation of an entity is a numerical value that is calculated from the current context in a manner explained in Section 5.2 below. Context activation is used in the Capture implementation to influence memory processing in the following two ways. First, the choice between possible alternative memory entities (e.g. referents), or sets of memory entities (e.g. word sense combinations), is done on the basis of comparing numerical context activation values. Second, context activation can be used to restrict memory processing if the searches implementing memory operations are restricted to entities with context activations higher than a specified threshold. The context mechanism is concerned with providing the ability to accumulate, represent, and manage context information of various sorts, derived from various sources, so that it can be applied in terms of context activation.

Context information is accumulated and modified gradually using the mechanism in a way which leads to gradual change of the context activations of memory entities. This gradual change of the context information represented results in a gradual shift of focus during the processing of a text because we can think of the focus-space [Grosz77] within this framework to be the set of memory entities, at any given time during processing, that have context activations that are higher than some predefined threshold. Further remarks comparing the context mechanism with Grosz's use of focus will be given in Section 5.6. However, the context mechanism does not boil down to a mechanism for maintaining a focus space during reading since context activation is also used as the basis for discriminating between entities that lie outside the focus-space.

The next section, on context representation, will describe how various factors contributing to context are represented so that they can be combined to derive the context activation of memory entities. The section after that explains how context is applied for solving text processing problems. The section on management of context factors describes how context information is accumulated gradually during the reading of the text.

5.2 Representation of context information

Context information is represented, at any given time during processing, by a collection of context factors. Each context factor contributes to the context activation of a particular set of memory entities. This set of entities is the scope of the context factor.

There are various types of context factor, each of the context factors present being an instance of one of these types. The type of a context factor determines the way that the factor is managed, as discussed later. The different types of context factor that have been used and the specification of their scopes are given later in this section.

Apart from its scope, a context factor (i.e. an instance of a context factor type) has associated with it a significance weight. When a factor is created, it is given an initial significance weight that depends on its type. Subsequently, as the processing of a text continues, the weight of a factor is degraded, often by gradual decay, in a way that also depends on the type of the factor. The details of the management of the significance weights of the implemented types of factor are given in Section 5.4.

We can now give a precise definition to the notion of context activation that was mentioned in the previous section. The context activation of a memory entity is the sum of the current significance weights of the context factors within the scope of which the entity lies. Thus at any point in processing, the relative importance of each entity is determined by the context factors that contribute to its activation score.

At the implementation level, for each context factor there is a marker that marks all the entities in its scope. The marker for a context factor is indexed, for search purposes, in the manner described in the chapter on memory processing (Section 3.5). The numerical significance weight of the context factor is attached (as a LISP property) to its marker symbol. This means that the significance weight of a context factor can be altered without accessing the entities in its scope. The context activation of a memory entity is calculated, at any given time, by examining the marks that are attached to it and summing the significance weights attached to the marker symbols.

The types of context factor currently implemented for the Capture system fall naturally into seven major types, some of which include more than one (sub)type. The classification into major types groups together types of factor that are similarly motivated, but is not part of the context mechanism as such. The seven major types and the ways that the scopes of the various types are determined is as follows.

1) Recency

There are three types of recency of mention context factor. These are the sentence factor type, the paragraph factor type and the text factor type. The scope of a sentence factor includes any entities that are mentioned explicitly in a sentence, or implicitly referred to by anaphoric expressions in that sentence. The scope also includes any other memory entities that are created as a result of interpreting the sentence. For example, assuming that the referent for "the machine" is 'P7700' in the sentence "Marconi manufactures the machine that is supplied by Smith",then the scope of the sentence factor would be 'Marconi', 'P7700', 'Smith', and 'E1'; where 'E1' is a specialization of the predicate 'manufacture' that is created during the interpretation of the sentence. These entities will also be included in the scope of a paragraph context factor for the paragraph including the particular sentence. The scope of the paragraph factor is all the entities referred to in the paragraph or created as a result of interpreting the paragraph. Finally, for each text processed, there is a single text factor whose scope consists of the entities mentioned in or created by the whole of the text.

2) Emphasis

The scope of an emphasis context factor is a single memory entity. Such entities are referents for noun phrases in sentences with a structure widely regarded as foregrounding the referent. (Compare e.g. with Sidner's syntactic marking [Sidner79]). Two types of emphasis factor were implemented in Capture. "Syntactic-topic emphasis" factors foreground topics of sentences in the passive voice, for example the referent of "machine" in "The machine is supplied by Smith". "Be-clause emphasis" factors foreground the agents of certain be-clauses, for example "Plexir" in "Plexir is a manufacturer".

3) Processing-history

These context factors increase the context activation of entities that take part in memory processing. In other words a trace of memory processing is used as a context factor because it is considered that a side-effect of a memory entity's involvement in processing should be that the entity is foregrounded. Fortunately, in the framework of marker processing, and given the way that context is represented in Capture, encoding traces of memory processing as context information is straightforward. In fact, a processing history factor has as its scope all of the memory entities that were marked by a marker propagation in memory. No new marker symbol is

generated to specify the scope of this factor since the marker used in the propagation can serve this purpose. For example, if during memory processing a propagation for marking the entities above 'P6000' in the specialization hierarchy is performed, then the scope of the corresponding context factor might include, for example, 'disc-drive', 'peripheral', 'machine/dbentity', 'machine', and 'inanimate'.

4) Deixis

The scope of a deixis context factor is the set of memory entities for which the sum of significance weights from recency of mention context factors is higher than a preset system constant. Thus the entities in the scope of such a factor will have their context activations increased if they have been mentioned frequently and recently enough in the preceding text. (*) The behaviour of a deixis factor depends, in part, on that of other factors; this is also true of association and subject-area factors (see below).

5) Subject-area

The subject area (or topic) type of context factor is designed to increase the context activation of entities in memory that are considered to be related to a particular subject area (in the sense of discourse topic). In fact, the scope of such a context factor is the set of entities in memory that are related to (i.e. take part in some of the same memory assertions as) a specified set of entities that are taken a priori to represent concepts that are central to the topic. For example, this set of "core" concepts for the "data processing manufacturers and suppliers" topic (corresponding to the enterprise that the Machines Database is concerned with) is taken to include the entities 'machine', 'supply1', and 'manufacturer'. The entities in the scope of a context factor for this topic might include 'machine/component', 'supplies/agent', and 'Mikota1'. The information stating that certain entities are central to a topic is itself represented by memory assertions.

6) Association

The purpose of association context factors is to increase the context activation of entities in memory that are closely associated with entities that are currently in focus (cf. Grosz's use of "implicit focus" [Grosz77] and Sidner [Sidner79]). For this purpose an entity is closely associated with an

---

(*) This type of textual deixis is different from spatial and temporal deixis factor types which are not currently handled by the system.

entity in focus if it is above the foregrounded entity in the specialization or correspondence hierarchies, or if they both take part in a correspondence assertion in memory. There are two types of association factor, primary association factors and secondary association factors. The scope of a primary association context factor is the set of all entities that are close, in the above sense, to any entities that have context activations that are higher than a certain preset constant. The scope of a secondary association factor is all the entities associated with the entities in a particular primary association factor.

7) Task

There is only one type of task-specific context factor in the current system. The scope of such a factor is the set of memory entities describing a particular database relation (Section 2.8). This type of context factor is primarily relevant to the evaluation of some task-specific operations such as the extraction of the names of relational columns, rather than to language interpretation operations, although it can affect these latter operations indirectly. An example of a task specific factor would have as its scope the roles of the entity 'supplies/relp', and the roles of these roles etc. The entities 'SMC/MCNUM', 'SMC/SID', and 'SUPPLIES/RELATION' would be included in the scope of this factor.

Apart from the task-specific factor type, I believe that the types of factor listed above are all necessary for processing "static" descriptive texts, e.g. texts describing objects. Some other types, that could be similarly encoded, will be mentioned later when the utility of the context mechanism for processing different kinds of texts is discussed. Intuitively, however, it would seem that the types of factor used by the system could play a role in processing most other types of connected texts, e.g. stories, as well as simple descriptive ones.

The "context factor", encoded as a marked set and a significance weight, is used as a uniform representation for the different aspects of context listed above. The advantages of this uniform representation for the application and management of context information will become apparent in the discussions of application and management in Sections 5.3 and 5.4 which follow.

5.3 Context application

There are two kinds of ways in which context information is applied in the system. These are "choice applications" and "threshold applications". In choice applications context activation is used to select between memory entities or sets of memory entities. This is required (or at least has been used in Capture for) reference resolution, word sense selection, database name retrieval, and also, in a limited way, for some other operations. In threshold applications a context activation threshold is used to define a focus space. This is used to restrict reference resolution searches, to decide on the referents of plural noun phrases, and to create association context factors.

In choice applications, context information is applied at points at which the system does not have enough specific information to select between competing entities (or lacks more powerful mechanisms than Capture's, e.g. for inference, for extracting more from the available information). In threshold applications, context information is used to locate the memory entities that are currently most relevant, for such purposes as restricting search processing. The way in which entities satisfying a threshold condition are accessed was described in Section 3.5.

The application of context to specific problems will now be described. These applications may involve either or both of choice and threshold applications of context activation derived from any combination of factors, and the descriptions will be given in terms of these derived activation values. Appendix B gives some examples of the effect of particular types of context factor when context is applied. The way the various types of context factor are created during processing is explained in Section 5.4 below. The descriptions of the various applications of context are followed by examples showing how the problems occur in texts processed by Capture. (*)

1) Application to singular definite reference

Searches for referents of singular definite noun phrases are done using constraint markers, i.e. markers that mark all entities satisfying a constraint on possible referents, as described in Section 4.7. An initial search request that is parametrized by the focus threshold is evaluated. Thus the search ignores entities that satisfy the constraints if they have context activations that are lower than the threshold. It also ignores entities that satisfy the threshold condition but not the constraints (Section 3.4). If

---

(*) The numbers given for these texts are from Appendix A, which includes the output generated by the system for the texts.

this initial search fails to locate any candidate referents, then a second search is made without the threshold condition. If more than one entity is located by either search, the context activations of the entities are compared and the one with the highest context activation is chosen. When the context activations of the two best candidates for reference are equal, a derived association factor is created (Section 5.4 below). If this in turn has no selective effect then the choice between the best candidates is made arbitrarily. Context activation is also used to select between the results of searches for referents if the constraints on reference are weakened (Section 4.7).

The following example contains instances of singular definite reference.

Text no. A18:

> Wintron manufactures P5050 which is a disc-drive. P1010 is a computer which is made by this manufacturer. It has a bolt. P8770 is a printer that is made by Plexir. Both peripherals are supplied by Clark. Smith supplies the machine with the bolt.
>
> P4740 is manufactured by P5050's manufacturer in London. It is a micro-computer that is supplied by Jones. He supplies P8800 which is a terminal. The cost of the computer is 25. The computers are red. The three peripherals are green.

Examples:

-- The referent for "it" in "It has a bolt" is 'P1010' which is preferred over the other candidate 'P5050' on the basis of context activation.

-- In the interpretation of "It is a micro-computer that is supplied by Jones" 'P4740' is selected as the referent for "it" because its context activation was higher than that of 'P8770', 'P1010', and 'P5050'.

-- 'Jones' is chosen as the referent for "he" in "He supplies P8800 which is a terminal", the context activations for 'Smith' and 'Clark' being lower than for 'Jones'.

-- The micro-computer 'P4740' is selected as the referent for "computer" in "The cost of the computer is 25" because 'P1010', the only other computer mentioned in the text, has a lower context activation than 'P4740'.

Text no. A20:

> Plexir manufactures P9000. It is a micro-computer. Wintron manufactures P7000 which is a disc-drive. P9000 is supplied by Smith.

P8000 is a computer. It is supplied by Jones. The status of this supplier is 10. The status of P9000's supplier is 20. The micro-computer is red. The manufacturer manufactures P9090.

Example:

-- The referent for "manufacturer" in the last sentence of the text is taken to be 'Plexir', even though 'Wintron' was the last mentioned manufacturer.

2) Application to plural definite reference

The application of context to the resolution of plural definite noun phrases depends on whether the number of entities in the set being referred to is known. When this is the case the correct number of referents is chosen from the entities satisfying the reference constraints by selecting those with highest context activations.

Text no. A16:

P8080 is supplied by Peters. The status of the supplier is 20.

Clark supplies P7780 and P7790. P7720 is supplied by Robinson. These three machines are manufactured by Plexir.

Example:

-- A new entity is created as the referent for "machines" in "These three machines are manufactured by Plexir". 'P7780', 'P7790', and 'P7720' are specializations of this entity. In the process of choosing three machines, 'P8080' was dropped because its context activation was lower than that for the other machines that were chosen.

Text no. A21:

Haddon collected P33 which is an armlet. He collected P37 from Woodlark. It is a necklace.

Bevan donated P571 and P352. P571 is a skirt. P352 is a necklace. Bevan collected both artifacts at Mount-Hagen.

Example:

-- The referent for "artifacts" in "Bevan collected both artifacts at Mount-Hagen" is taken to be 'P571' and 'P352'. The other artifacts mentioned, 'P33' and 'P37' have lower context activations.

When the number of entities in the set being referred to is not known the following algorithm is used. An initial search is made for a memory entity which satisfies both the constraints and a focus threshold condition and has already been created to describe the elements of a set. Such an entity may have been the referent of another plural noun phrase or it may have been created as a result of interpreting a simple conjunction. If the search locates many such entities then the one with the highest context activation is chosen as the referent. If, on the other hand, no such entity satisfying the threshold condition is found, then a search is made for all (individual) entities that satisfy the reference constraints. The set of entities located by this search is taken to be the referent of the plural noun phrase, and a new entity describing its elements is created.

Text no. A30:

> Jones who was a trader collected P350 from Daui. He collected P370 from Woodlark. P350 is a necklace. P370 is an armlet. P391 is a necklace that comes from Woodlark. The condition of these ornaments is good.
>
> Armstrong and Haddon were British. They were academics. Haddon collected P597 and P598 from Daui. The artifacts are necklaces. The condition of these Daui necklaces is poor.
>
> P392 and P393 are armlets that were collected by Smith. This collector was a trader. The artifacts are fair.

Examples:

-- The referent for "they" in "They were academics" is an entity which describes 'Haddon' and 'Armstrong'. This entity was created during the interpretation of the sentence "Armstrong and Haddon were British", and it is chosen as the referent because it satisfies the threshold condition during the interpretation of "They were academics".

-- The referent for "artifacts" in the sentence "The artifacts are necklaces" refers to an entity describing 'P597' and 'P598'. Although there is another entity (describing the artifacts 'P370', 'P350', and 'P391') which was created as a result of interpreting "The condition of these ornaments is good", this entity did not satisfy the threshold constraint. The entity describing 'P597' and 'P598' has a higher context activation and satisfies the threshold constraint when the reference resolution is performed.

-- The referent for "artifacts" in "The artifacts are fair" is an entity that describes 'P392' and 'P393'. This satisfied the threshold constraint, and its

context activation was higher than the entities describing the other groups of artifacts.

3) Application to word sense disambiguation

It has already been mentioned in Section 4.9 that the context mechanism is used in selecting from alternative word sense combinations presented by the output of the analyser. The two cases that will be considered here are disambiguation of words in general and disambiguation of compound noun phrase modifiers. Structural disambiguation will be considered later.

All that the context mechanism does in the first case, word sense selection, is compute the sum of the context activations of the set of word senses present in each of the analyses. The analysis with the highest sum is then selected (but see Section 4.9 for the handling of more complex cases). The choice between alternative nominal modifier senses in compound noun phrases is done simply by selecting the sense that has the highest context activation.

Text no. A33:

> P900 is a spear. P700 is an armlet. This artifact was collected in Daui. It is common. The weapon was collected from there. P940 and P950 are arrows. P200 is a spear blade.

Examples:

-- Two analyses are produced for "P940 and P950 are arrows" with different senses for "arrow". The analysis containing the sense 'arrow1' (a weapon) is preferred over the analysis containing 'arrow2' (a sign) because of the higher context activation of 'arrow1'.

-- An analysis of "P200 is a spear blade" which contains 'blade1' (part of an instrument) is preferred over one containing 'blade2' (loud jovial man), because the context activation of 'blade1' was higher than that of 'blade2'.

Text no. A14:

> Plexir manufactures P9999 which is a computer. It is supplied by Smith. P1010 is a terminal that is supplied by Clark. This one is made by Mikota. These machines are red.
>
> P9000 is a green printer. It is made by Plexir. P4444 is a blue computer. The cost of the machine is 7850. The peripheral is supplied by the P9999 supplier. The terminal manufacturer makes the blue

machine. The cost of Mikota's peripheral is 235.

Examples:

-- The analyser produces two analyses for the sentence "P1010 is a terminal that is supplied by Clark", one of which contains the sense 'terminal1' (a computer peripheral), and the other 'terminal2' (a place, as in hovercraft-terminal). The analysis containing 'terminal1' is selected because it had a higher context activation than 'terminal2'.

-- The analysis of "This one is made by Mikota" that contains 'make1' (which corresponds to manufacturing) is preferred over an analysis containing another sense of "make", because of the higher context activation of 'make1'.

-- Two analyses are produced for "P9000 is a green printer". One of these contains the senses 'green2' and 'printer1', and can be paraphrased as "The colour of the printing machine P9000 is green". The other contains the senses 'green1' and 'printer2', a possible paraphrase being "P9000 is a novice at printing". The analysis containing the first combination of senses (colour and machine) is chosen because the activation sum for this combination was higher than for the other combination.

-- During the interpretation of "The terminal manufacturer makes the blue machine", the analyser representation of the compound "terminal manufacturer" presents 'terminal1' (peripheral), and 'terminal2' (place), as alternative senses for the modifier. The higher context activation of 'terminal1' means that it is chosen before the compound noun interpretation (see Section 4.8) proceeds.

4) Selecting database names

As well as applying context to linguistic interpretation problems it is also used in connection with the specific needs of the database capture component. The context activation of the entities forming descriptions of database relations allows the correct column names in the database implementation to be selected. The task-specific memory operation for selecting generic database entries (i.e. entities representing column name-value slot pairs, as explained later in Section 6.4) chooses among alternative entities on the basis of their context activation. In fact the context factors that allow the selection of the correct database-related entities are explicitly controlled (Section 5.4) by task-specific operations. Selecting database names is a local (clause level) application of context information. Explicit control is possible for task-specific operations because they are

better defined, and hence easier to control, than language interpretation operations. An example of this application of context is as follows.

P9000 is supplied by Smith. The cost of the machine is 200.

The column name 'SMC/MCNUM' (the machine number column in the 'SUPPLIES/RELATION') is chosen during the processing of the first sentence whereas 'MC/MCNUM' (the machine column in the 'MACHINES/RELATION') is chosen for the second sentence.

A more interesting application of context for the database capture task is selecting between alternative underlying predicates, as will be indicated shortly.

5) Generating context factors

Another, rather different, way in which context activation is used during processing is in the generation of further context factors. Context activation thus plays a role in "bootstrapping" context information. Thus the generation of subject-area factors is conditional on whether the entities in a predefined set concerned with that subject domain have, on average, sufficiently high context activations. What counts as a "sufficiently high" context activation for this purpose is determined by the size of the set concerning the domain, and a preset system constant.

Further, the scope of association context factors, as opposed to their creation, depends on existing context information. A context activation threshold search is performed to locate a set of "highly active" memory entities at the time the association factor is created, and the scope of the primary association context factor is the set of entities associated with this threshold set (Section 5.2).

6) Other applications of context activation

The five forms of context application just discussed are those which have been most fully tested; but the implemented system is also capable of applying context information for choosing between the results of other memory operations. However, these additional uses of context activation have only been tested in a few cases, so it cannot be said, with any degree of confidence, that the context mechanism can be usefully applied to the problems concerned. The intention of the remarks which follow is therefore primarily to suggest that it is relatively straightforward to apply context information in the style already described to a wide range of problems.

(6a) Relationship interpretation

The interpretation of implicit relationships for compound nouns, possessives, and have-clauses, in creation mode (Section 4.8), allows the application of context for choosing between alternative interpretations. Context activation can be used for choosing between alternative memory entities that capture the possible relationships. For example (assuming there is only one sense for "distributor"), the relationship implicit in the sentence "The car has a distributor" may be captured by 'machine/component' or 'merchandise/dealer', and the context activations of these two entities could be used to choose between them. (Strictly, the relationships are captured by the pairs 'machine/component' to 'machine', and 'merchandise/dealer' to 'merchandise'.) Another example is determining the relationship implicit in the nominal compound "computer maintenance" in the sentence "The success of the company depended on computer maintenance". The memory entities that would have to be chosen from, on the basis of context activation, in this example might be 'computer/application', or 'maintenance/of/machine'. (*)

(6b) Structural disambiguation

Choosing between alternative analyses of a sentence that are different in structure but have the same word senses is performed using a score based in part on context activation (Section 4.9). The score includes the sum of the context activations of entities which are specializations of the case relationships present in a particular structure. The score also includes a bias for specialized case relationships over unspecialized ones. Most of the examples of structural disambiguation that have been processed by the system depend on this bias towards most specialized relationships rather than context activation. It is therefore not clear, at present, whether this way of applying context activation to structural disambiguation is likely to be fruitful in general.

However, structural disambiguation using context activation alone was specifically tested with the following example. Prior memory assertions were created stating that a 'supply1' statement has the 'location' case specialized to 'supplies/loc' and that this case can be filled by a 'city'. The text given below was then processed.

(*) The interpretation in reference-mode of compound noun, and other relationships, does not apply context directly. This is because relationship interpretation in this mode simply generates another constraint for the reference resolution process (see Section 4.7). Context activation is then used for selection at the end of the reference resolution operation.

P9999 is a disc-drive that is supplied by Smith. This peripheral is manufactured by Mikota. He supplies P7777 which is a terminal. It is manufactured in London by Plexir. Clark supplies P9000 which is manufactured by Marconi in Paris.

The context mechanism is used for disambiguation to choose between two alternative structures produced by the analyser for the final sentence in the paragraph. In one of the analyses the prepositional phrase "in Paris" is attached to the embedded clause and the specialized case entities derived from the sentence were 'manufacture/obj', 'manufacture/agent', 'manufacture/loc', 'supplies/obje', and 'supplies/agent'. In the other structure the prepositional phrase is attached to the main clause, and the specialized case entities were 'manufacture/obj', 'manufacture/agent', 'supplies/obje', 'supplies/loc', and 'supplies/agent'. The first analysis was chosen because of the higher context activation sum for the specialized case entities derived from it.

(6c) Underlying database predicates

The database capture task uses an operation for mapping the predicates of verb-clauses and state-clauses onto the predicates underlying database relations (Section 6.4). The implemented system allows for choosing between alternative underlying predicates on the basis of context activation.

An example of a situation in which context was applied for determining underlying predicates is as follows. A column for the colour of an artifact was added to the Artifacts Database; there is a colour column already in the Machines Database. The underlying predicates referring to colour in the descriptions in memory of the two databases were the entities 'relp/artifact/colour' and 'relp/machine/colour' respectively. A single "confusing" text was created to try out the application of context to sorting out the appropriate underlying predicates. The text processed using this modified description of the Artifacts Database is given below.

P500 is an armlet. It was collected by Haddon. P550 is red.

P9000 is a disc-drive that is supplied by Smith. The cost of the machine is 200. P9900 is red.

The underlying predicate chosen during the processing of "P550 is red" was 'relp/artifact/colour'. The other underlying predicate that was a specialization of 'colour/of', 'relp/machine/colour', was preferred during the interpretation of "P9900 is red". These choices were made according to context activation; and as a result the appropriate database creation

statements were generated.

It has not been possible in the present project to do any very thorough tests on the application of context activation to relationship interpretation (in creation-mode), structural disambiguation, and identifying underlying database predicates. This is primarily because context dependent ambiguities arising from these cases seem to occur less frequently than in the other cases discussed earlier. Nevertheless, it appears that using the context mechanism in these situations does not complicate the control structure, and this is an important consideration for natural language processing systems.

This section described how context activation is applied in Capture for various disambiguation operations (choice applications) and for accessing the memory entities that are most relevant to the current context (threshold applications). The interpretation operations applying context need not be aware of how the various factors that contribute to the context activation values at any given time were created or how their weights were managed. Some implications of the way context is applied in Capture are described in Section 5.5 below.

5.4 Management of context factors

Having talked about the types of context factor used in the Capture system, and having illustrated the sort of problem context application is intended to handle, I will discuss the way that the various types of factor are managed during the processing of a text.

In the model for context used by the system, the management of context information amounts to creating context factors and adjusting the significance weights associated with them. The creation of all context factors, except processing history factors, is in fact done by the interpretation operations and the task-specific operations. The specification of interpretation and task-specific operations thus includes appropriate calls to routines that create new context factors. (Processing-history factors are directly created by the memory component even though they are ultimately the result of evaluating interpretation and task operations.)

For some types of context factor, e.g. emphasis factors, the text processing operation creating a factor also determines its scope. The scope of the other context factors, e.g. association, is determined by the state of memory and the context information present at the time the new factor is created. The ability to create context factors when evaluating the text processing operations that make use of memory allows us to view interaction with the memory as resulting in the accumulation of context information. This is indeed a significant feature of the Capture philosophy.

The significance weights associated with factors are initially assigned as standard values, and are then subsequently degraded as processing continues. The detailed way in which the significance weight of a context factor is managed depends on its type. In particular, the initial weight associated with a factor is determined by its type. To manage decay, the system maintains lists of all the instances of factors of the different types which are consulted when the significance weights of factors of any specific type are degraded. This is because the significance weights of factors are not degraded individually but rather all the factors of a given type are degraded together. Factors are removed from the system when their significance weights fall below a certain threshold.

The set of initial significance weights for the types was determined by trial and error as the system developed and new test paragraphs were processed. When a new type of factor was introduced, an initial weight for the type was chosen intuitively, which was then increased or decreased according to the behaviour of the system for the example texts. This was admittedly a fairly crude approach and indeed was not even based on any serious conduct of experiments. But in its defence, it should be said that attempts at more

controlled experiments would have been inappropriate in any case because the example texts were written specifically for testing the system, and were not taken from an independent corpus of texts. As a result, no claims can be made about the real importance of the types of context information represented by the Capture factor types and about the way they are managed in the implemented system. Thus although the long term aim of the research is to develop computational systems adequately handling linguistic reality, it has not been possible in this project to demonstrate conclusively that this is possible with the type of context mechanism developed.

A brief description is now given of the management of the various types of context factor whose scopes were defined in the section on the representation of context information. The significance weights of many of the factors used in Capture are degraded using a standard degrading procedure. This applies to sentence-recency, emphasis, deixis, subject-area and association factors, while paragraph-recency factors, processing-history factors and task-specific factors are not degraded using this standard degrading procedure. This procedure is as follows. At those points during processing when instances of certain context factor types (in fact recency and emphasis factors) are created the significance weights of all existing factors of the types mentioned above are divided by a system constant. This standard degrading ratio was 2 in the final version of the system, and integer arithmetic was used for convenience. After degrading factors in this way those with zero weights are removed from the system's records. It needs to be emphasized that degrading factors of a given type together, and degrading in a standard way are just details of how context management was carried out in the test Capture implementation, they are not regarded as theoretically motivated elements of the context mechanism as such.

1) Recency

Sentence recency factors and paragraph recency factors are managed differently. Sentence recency factors are created by the "interpret-sentence" operation and given an initial weight of 100. The creation of a new sentence-recency factor causes a standard degrade to occur. A paragraph recency factor, with initial weight 50, is created by the "interpret-paragraph" operation. The weights of paragraph recency factors are degraded explicitly to 0 when a new factor of this type is created. The creation of a paragraph recency factor also causes a standard degrade on other factors.

2) Emphasis

Be-clause emphasis factors are created by the "interpret be-clause" operation, and given an initial weight of 90. The creation of a be-clause emphasis factor causes a standard degrade. Emphasis factors relating to the syntactic topics of passive sentences are created by the operation for interpreting case fillers, and given an initial weight of 20; this does not, however, involve a standard degrade.

3) Processing-history

Since any memory operations can use memory retrieval that is implemented by marker processing, processing-history factors can be created by any memory operation. Such factors are given an initial weight of 20. The significance weights of processing-history factors are degraded by dividing them by 3. This happens at the end of processing a sentence. The weight of a recorded factor is incremented (also by 20) when a request is made for the propagation to be repeated; however, this is not a true incrementation; it is really creating a new factor.

4) Deixis

Reference evaluation of noun phrases with deictic determiners generates a deixis context factor that is given an initial weight of 90.

5) Subject-area

At most one subject-area factor can be generated by the "interpret-paragraph" operation. It is given an initial significance weight of 60.

6) Association

Association context factors are created by the operations for word sense and structure disambiguation. They are also created as a result of evaluating reference resolution operations when the best candidates for reference have equal context activations. The two factors (primary and secondary associations) generated by any one of these operations are created before the selection done by the operation is performed so that the factors can affect the result of the operation itself. The initial weight given to each of the two association context factors is 50.

7) Task

Database capture task factors are created by the operation that builds instances of underlying predicates for relations. The initial significance weight given to such a factor is 180. The weight of the factor is degraded explicitly (to the initial weight of a processing history factor) by the task specific operation that generates database creation statements.

Thus, overall, the management of context factors in the Capture test implementation is as follows. The evaluation of text processing operations that interact with the memory component cause the creation of factors which, depending on their type, are either degraded uniformly at specified points during subsequent processing, or more directly by text processing operations managing them.

One of the considerations that had to be taken into account with respect to the management of context information was the overall activation of the entities in memory. In the implemented system the initial weights and degrading ratios ensure that after the first few sentences,the number of highly activated memory entities does not keep increasing. That is, the number of salient entities at a given time during processing is stable. Because of this, the various constant activation thresholds used by the system lead to roughly the same behaviour as the processing of the text is performed. However, the introduction of new types of context factor in an extended system might require a more flexible approach that would allow system operation with very different overall activations of the entities in memory. One way of doing this, which was not investigated, might be to periodically reset the system thresholds to some fixed ratio of the sum of the significance weights of all current context factors.

In the process of trying to determine first approximations for managing the weights of the context factors, association factors seemed to cause instability, whereas exact management of the weights given to other factors did not seem to be necessary for stability. This may be because association is a rather loose and unstructured factor type. It might therefore be necessary, for systems using association as a context factor, to control the circumstances under which different degrees of association can be used profitably.

Very different schemes for the management of context information were experimented with. For example, a mechanism was implemented in which memory operations could be explicitly parametrized by only a specified set of context factors. A separate mechanism was also implemented which allowed memory operations to be parametrized by the set of factors that were currently at the top of a "context factor stack". These mechanisms

were finally abandoned because they greatly complicated the flow of control in the system by requiring that multiple contextual environments should be tracked so they could be applied as necessary. In contrast, application of context in the final version of the system happens automatically with respect to (all) the context information present at the time of application. One of the issues that the project came increasingly to be concerned with was how far this constraining but simple control structure is adequate for text processing. The more complex control structures for managing context information that were abandoned were too unconstrained, making both programming and evaluation of the system more difficult. The fact that the facilities originally provided by these control structures could be implemented in terms of the simpler model for context management suggests, but this is only a conjecture, that they may not be necessary anyway.

5.5 Implications of the context mechanism

This section discusses some implications of the use of a Capture-like context mechanism in other language processing systems. The implications are for the application of context information in such systems, the class of texts that they can (in principle) handle, and more generally, for the management of context information in text processing.

In Capture's context mechanism it is not necessary to specify explicitly how the different types of context information interact during processing. Basically, it is only necessary to specify the memory operations that cause the creation of the various factor types, and to specify how the significance weights associated with these types should be managed. After this at any given point in the processing the context information present can be applied without regard to the types of the factors that have been created. This simplifies the design of the system in that new types of context factors can be included without changing those parts of the system that apply the context information to interpretation problems.

The use of significance weights gives a simple way of comparing the relative importance of different types and instances of factors when context is applied. The use of numerical values in AI systems in this way may seem ad hoc, but it does have important advantages in certain cases, like the context one presented here, because of the convenience of numerical values for comparing, degrading, and combining the effects of different context factors. For a discussion of the merits of the use of numerical values for a different purpose see [Cater81].

A separate issue is that the approach taken for applying context information makes the interpretation process more deterministic. Thus the context mechanism acts like an "oracle" that determines the path to be taken at choice points faced by the interpretation process. This leads to a more easily manageable control structure, since it is not necessary to keep a record of the alternative paths that were not followed, or to be able to decide when to backtrack and try other alternatives.

Further, building a context mechanism for deterministic language interpretation may result in a system which handles constraints on text structure naturally, for example constraints on the occurrence of referring expressions. Such a system would probably fail to process certain (misleading) texts correctly. (*)

The Capture implementation is biased towards a deterministic interpretation control structure. For example, some of the factor types used by the Capture system make it more awkward to allow the whole system to backtrack cleanly. In particular, "processing history" factors are (by their very nature) unavoidable side-effects of processing. Thus a backtracking mechanism for Capture would have to be able to save and reinstate information about the context factors present at a given point during processing.

A system that makes use of such a context mechanism for deterministic interpretation clearly cannot handle texts that do not satisfy the following condition. The "best" interpretation of each sentence with respect to the context provided by the preceding sentences is in fact the correct interpretation in the context of the complete text. Whether the context mechanism can always determine the "best" choice depends on how well the range of context factors and significance weights capture all the elements affecting the interpretation of text, which may be evaluated by agreement with human interpretation. It is not difficult to construct texts that do not satisfy the condition just given, i.e. that fool the interpretation process into making decisions that need to be revised because they are inconsistent with information supplied much later by the text. However, such misleading texts are rare and a technological approach to practical text processing, such as the approach taken in the design of Capture, can, I believe, afford to ignore them.

(*) Cf. the claim made by Marcus [Marcus80] that a basically deterministic mechanism for syntactic analysis can result in a grammar that elegantly captures generalizations reflecting constraints on sentence structure; and the expectation that a deterministic parser would fail to process "garden path" sentences.

A separate implication of the context mechanism is that it allows different components of the language processing system to contribute to the current representation of context by creating context factors. This information is then potentially available to any other component that makes use of the knowledge stored in memory. Components creating factors can also manage the factors belonging to them by altering their significance weights.

This view of context, whereby loosely speaking it cannot be placed in a single box of a language processing system, possibly explains to some extent why linguists have found it so difficult to arrive at a foolproof characterization of what context is.

The components of a natural language processing system often correspond to different levels of language analysis (syntax, semantics, pragmatics, etc.). Because of this, and because of the relationship between context information and the current focus, the above view of the management of context in a language processing system is consistent with the view of focus taken recently by Grosz, Joshi and Weinstein [Grosz83], i.e. that it does not "yield comfortably to any account that is strictly a syntactic or semantic or pragmatic one."

The context mechanism provides the computational means for realizing this view of context in automatic language processing systems. Furthermore, the use of the indexing scheme for implementing "threshold searches" gives a computationally efficient way of accessing a constantly changing focus space even when the factors that contribute to context are managed independently by different components.

5.6 Comparison with other models for context and focus

In the first section of this chapter various context mechanisms were viewed as means for restricting the use of a knowledge base for the purpose of taking context into account when performing natural language understanding. The following sections showed how this view of context is realised in the Capture system as the representation and use of information for constraining searches of operations using a particular memory model, and for selecting from the results of these operations. In this section the context mechanism developed for Capture is compared with other models for context and focus. In order to avoid confusion the context mechanism used in the Capture system will be referred to in this section as the Context Mechanism.

In what follows I shall first briefly discuss the use of context mechanisms based on scripts and related knowledge structures ([Schank75], [Tait82], [Schank82a]); and also a "marker passing" theory of contextual influence [Charniak83]. Context mechanisms concerned with focus will then be discussed ([Sidner79], [Wilensky82], and [Grosz77]). This work on focus is the most important for the evaluation of the Context Mechanism.

Scripts, taken to mean knowledge structures that encode events in typical situations, (see e.g. [Schank75], [Tait82]), have been used to provide a strongly predictive context for processing texts by following the stereotypes encoded in the scripts. Thus a script is typically used to encode sequences of actions that occur in a certain type of situation, so that the understanding process can interpret a text taken as being about this type of situation by matching objects and events mentioned in the text against those expected by the script. The Context Mechanism is an attempt at producing a more flexible system that can deal with texts that do not fit any predetermined standard situation exactly. Trying to apply scripts to interpreting such texts can lead to errors because the predictions made by the script tend to take over the interpretation process; scripts can lead to disaster if the script chosen does not in fact fit the text at all (e.g. interpreting a news article in which "Pope's death shakes world" occurs as news about an earthquake that caused the death of one person).

It is not being argued that the use of knowledge structures concerning typical situations cannot play a role in the interpretation of a certain type of text (nor that, more generally, large memory structures should not be used by a context mechanism for text processing). The problem is that systems relying primarily on scripts tend naturally to rigidity as the penalty for predictive control of interpretation. Thus early script systems were clearly far too rigid; later script based approaches are either still too rigid or are sufficiently underdetermined to raise the problems the original script

idea was intended to overcome (cf. the remarks on MOPS below). In contrast the less rigid use of context information by the Context Mechanism means that no single context factor can dominate the interpretation process to the point of completely distorting the propositional content of the text. The Context Mechanism is, roughly speaking, used to solve interpretation problems when they arise, rather than to predict the message being conveyed by the text.

There is no context factor type in the current Capture implementation for taking advantage of stereotyped sequences of events; such a factor type did not seem to be necessary for the system's current text application. However, the Context Mechanism does not rule out the use of script-like information to implement a type of context factor which would be used to increase the context activation of memory entities in memory structures representing generic sequences of events. This information would still not be used in a strongly predictive fashion, but would have the advantage of being combined with information represented by other context factors, and hence would not dominate the interpretation. Another advantage of bringing scripts into this framework is that context information derived from other factors could be used to implement a reliable method of script activation (rather than e.g. keyword triggering, or pattern matching against sentence representations). This could be done by monitoring the context activation totals associated with the sets of memory entities representing the various scripts, and activating a script (i.e. increasing the activation of event entities in it) when the total associated with it exceeds a certain threshold, or, perhaps, when this is much higher than the totals associated with the other scripts represented in memory.

More recent language understanding work [Lehnert83] makes use of MOPS to overcome some of the problems associated with the use of scripts for understanding narratives. MOPS (for a definition see Section 5.1) are smaller and more generic structures than scripts, which can be combined in a flexible way with one another to form individual scripts ad hoc, as suggested by links ("strands") between different MOPS. Schank [Schank82a] claims that generalizing and merging the information present in many scripts and representing it in a non-redundant way as MOPS is a more realistic model of human memory organization. This is consistent with the approach for memory representation in the present system where memory entities are not thought of as belonging strictly to large rigid structures but instead derive their meaning from the whole of memory.

The use of MOPS does seem to represent an improvement over scripts in terms of memory organization for natural language processing. However, it is not clear whether the systems in which MOPS are used have an adequate mechanism for deciding which MOPS to activate during processing. This is

particularly true if the number of MOPS present in memory, and the number of "strands" between them, is realistically large. This is because in such a situation the MOPS activated lexically, or by following MOP strands, need not all be relevant to the current context. This suggests that it might be possible to activate MOPS by monitoring the context activation of the memory entities associated with them, in a scheme similar to that proposed above for the use of the Context Mechanism for script activation.

Such a scheme may also provide a solution for the "frame activation problem" as stated by Charniak [Charniak82]. In fact implementation of the subject area context factor can by thought of as an attempt at such a solution. Furthermore, this solution seems to be capable of handling Charniak's "baseball" example, which he concedes cannot be handled by his own proposal for indexing on slots. Thus the context activations of (the appropriate senses of) "ball, "bat", "pitcher", and "diamond", could be used to judge whether to create a context factor for the "baseball" subject area.

Judging from more recent publications [Charniak83], it appears that Charniak has moved from the model based on indexing the roles of frames towards a model based on marker passing for contextual influence. This model has some properties in common with the Context Mechanism in that they both attempt to make use of a NETL-like model for memory representation, and marking to implement a context mechanism; however there are fundamental differences.

Charniak's Marker Passing theory for context is one that is applied during sentence analysis (see [Hirst82]), to assist the parser with which it acts in parallel to choose word senses and case-labels. It is thus a mechanism for disambiguating word senses and case-labels only within the context of the single sentence in which they occur. The Context Mechanism used by Capture, on the other hand, acts on sentences taken together, and deals with interpretation problems that are unresolved at the sentence level using extra-sentential context information.

In the Charniak and Hirst model, sense combinations are preferred if some connection between them is found in the memory network by passing markers in a way that is similar to their use by Quillian [Quillian68]. Charniak refers to this as "dumb marker passing", and suggests that it should be constrained somehow, but exactly how is unclear because the theory described in [Charniak83] had not yet been implemented. "Dumb marker passing" uses information that is analogous to the information used by Capture association factors and (to a lesser extent) processing-history factors, but otherwise the relationship between marker passing in Charniak's theory and the use of marking in the Capture Context Mechanism (i.e. as part of a general representation for different types of context

information) is only in terms of low level implementation mechanics. Thus overall, the Charniak and Hirst mechanism operates at a different level of language analysis than the Context Mechanism; further, the fact that both mechanisms use "marking" is only a superficial similarity.

The Context Mechanism is more closely related, in its basic philosophy, to focus based mechanisms for interpretation in context. The rest of this section discusses mechanisms of this sort.

Sidner [Sidner79] has developed a theory of definite anaphora interpretation (for English) that is based on determining the focus of a discourse; this is the entity on which the speaker centers attention, and which typically changes as the discourse progresses. This discourse focus is generally the preferred candidate referent for an anaphoric expression, and hence the knowledge-base element that is examined first during anaphor interpretation. More specifically, Sidner's algorithm for focussing keeps track of the current focus, an alternative focus list, and a focus stack. Focus can shift to an alternative focus if an anaphor cannot refer to the current focus. The focus stack records such rejected items, so elements on the stack can be considered after the list of alternate foci, if they prove unsatisfactory. Elements that are associated with all these candidate entities are also considered as possible referents for anaphoric expressions. Association is defined with respect to information in a knowledge base that is assumed to be similar to Fahlman's NETL, and is similar to the information used for the Capture association context factor type.

Sidner provides detailed rules for the interpretation of various classes of definite anaphora, which indicate how the (discourse) focus, alternative focus list, focus stack, and associations, can be used to interpret the anaphora. There are exceptions to the basic use of discourse focus which include a recency rule for processing certain pronouns, the use of an "actor focus" as well as the discourse focus, and co-present foci for the interpretation of anaphora such as "the one ....the other".

These exceptions, and the need to keep track of alternative, associated, and stacked foci, dilute the notion of a single discourse focus of attention. This suggests that the use of context activation may be more appropriate since it does not make assumptions about a single (if shifting) focus of attention, but instead, relies simply on relative context activation. Additionally, taking new context factors into account would require explicit detailed changes in Sidner's algorithms, whereas the algorithms used in the Context Mechanism remain the same.

However, Sidner's work on definite anaphora comprehension is better motivated linguistically, and much more sophisticated in its treatment of the

various types of anaphora, than the reference interpretation operations implemented for Capture. It is therefore worth considering how the insights of Sidner's work could be used effectively in a system with an organization like that of Capture. In fact it appears that some features of Sidner's model (such as the effect of linguistic form on emphasis and foregrounding) could probably be incorporated into a more sophisticated version of Capture's interpretation operations and context factor management; others would fall out naturally from the information encoded as context activation (such as the preference ordering for various categories of reference candidate).

"Activation" is also being used, to indicate foregrounding, in a context mechanism that is under development for the UC (Unix Consultant) system at Berkeley. The mechanism is an extension to PHRAN [Wilensky81] and is used by UC to support natural language dialogue with a user who is learning to use the Unix operating system. The mechanism (described in [Wilensky82]) maintains a "Context Model" which consists of a number of entries (entities, broadly defined) with associated levels of activation. Activation is used for what were termed "choice applications" in Section 5.3, in particular for some aspects of reference and word sense selection. Entries in the Context Model include assertions, objects, and representations that will be sent to other components (e.g. a discourse planner). When the PHRAN parser has analysed a sentence, the results of the analysis are matched against entries in the Context Model, and new entries are added to the model. The entries are grouped into "clusters" representing associated fragments of knowledge, and increasing the activation of an entry causes an increase in the activations of entries in the clusters to which it belongs (cf. association factors). Activation decays over time, and entries with low activations are removed. There is also an indexed database of clusters from which clusters can be brought into the current Model when new entries indexing them are added to the model. In the reverse direction, a new cluster can be created from entries in the Model and placed in the database, where it is indexed by those entries which are most highly activated.

In UC, the current Context Model and the database of clusters have a combined function much like memory in the Capture system; the entries in the Context Model correspond, roughly, to memory entities with context activations that are higher than some threshold. In the Context Mechanism it is not necessary to pull highly relevant items out of memory and keep them in a current context model because it is always possible, using the indexing scheme, to access these items efficiently and to restrict memory searches to them. This additionally means that memory assertions themselves can serve as associations (for the purpose of "spreading activation") so that it is not necessary to have a separate index from entries in a current context model to clusters in a database. The need in UC to create and index new clusters (which does not arise in the Capture Context

Mechanism) causes some problems (pointed out in [Wilensky82]). The UC system should be able to determine, on its own, when to create a new cluster (an operation which can be regarded as storing the current context). At present the UC system must be instructed explicitly by the user to do this. A related problem is that the system cannot compare clusters and may therefore create more than one cluster each of which describes the same situation.

In the UC context mechanism there is no analogue of the "context factors" used in Capture's Context Mechanism for independent management of components of context activation. Again, the Capture indexing scheme allows this generalization to be accommodated with efficient access to the most salient entities. However, the use in UC of an activation-based model for dialogue understanding indicates that this somewhat intuitive notion is useful in language processing settings different from those involving descriptive texts to which the Context Mechanism has been applied in the Capture project. Further development of both the UC context mechanism and the one described here may provide support for this claim.

Grosz's work on focus [Grosz77] directly influenced the design of the Capture Context Mechanism; thus the Mechanism can be thought of as a generalization of Grosz's "global focus" mechanism that is more widely applicable, and that tries to address certain problems brought to light by Grosz's work.

In Grosz's work focus information is used for resolving definite references made in task-oriented dialogues. The dialogue fragments studied were taken from conversations between an expert and a novice being instructed at assembling an air compressor. Focus is represented as a highlighted set of nodes in a knowledge base. This set is encoded as a space in the partitioned network formalism developed by Hendrix [Hendrix78]. The focus space is taken from spaces in a partitioning of the knowledge base that mirrors the mechanical task of assembling an air compressor. Focus is used ([Grosz77] p.5) to "differentiate among the items in the knowledge base on the basis of relevance", i.e. to select one space as salient so nodes in this focus space are considered first as candidates for definite reference.

In what follows, I shall first consider the similarities between Grosz's focus mechanism and the Context Mechanism, and then the way the Context Mechanism extends and improves on Grosz's model. Grosz's immediate focus mechanism for ellipsis interpretation is a separate matter not of immediate relevance to Capture, so I shall consider only her mechanism for global focus.

It has already been mentioned that the set of memory entities with context activations that are higher than a specified threshold can be thought of as the counterpart, in Capture, of Grosz's focus space. In fact, initial candidates for reference resolution are looked for in such a set of entities using the threshold search (Section 5.3). If the search for referents in this set fails, the search is widened to memory entities outside it, just as the search for candidate nodes is widened to nodes outside the focus space in Grosz's case. In Grosz's model the focus space is augmented by giving precedence to nodes that are implicitly in focus, e.g. to subparts of those objects or participants in those events that are in focus. In the Context Mechanism information on which implicit focus depends is utilised either explicitly in the form of association context factors or implicitly in the form of processing-history context factors.

Turning now to the developments made in Capture, it should be emphasized that in the Capture system the function of the Context Mechanism extends beyond the resolution of references made by definite noun phrases. It is used, for example, for choosing word sense combinations (Section 5.3). It is not clear exactly how Grosz's mechanism could be used effectively for performing this function because of the difficulty of guaranteeing that nodes corresponding to the correct word senses would be in focus (or implicit focus), since only nodes concerned with a stage of the assembly task would be in focus. Thus the form of the knowledge exploited for focus in Grosz's model limits the function of the model.

An advantage of the Context Mechanism's use of context activation is that the activations of arbitrary sets of entities, and in particular word sense combinations, can be added in order to choose between the sets. This can itself be used to trigger other factors (Section 5.4) to further sharpen context. The analogue, in Grosz's focus framework, might be to count the number of nodes belonging to each set that are included in the focus space, but this would be less discriminating than summing context activations. Context activation can, of course, be used to choose between any two memory entities even when they are both included (or both not included) in the analogue of the focus space that is defined by a context activation threshold.

Another way in which the Context Mechanism extends the use of focus is that it is designed to allow the combination of very different types of context information. The type of discourse structure information that is based on the mechanical assembly task in Grosz's work could, in principle, be used as a context factor type. This would imply a representation of the discourse structure information in memory in Grosz's style, with the subsequent creation of context factors increasing the context activations of subsets of the entities in the structures corresponding to the partitions used by the

focus mechanism. The information represented by context factors generated in this way could then be combined with that from other factors in the usual way, contributing to the choice of referents for definite noun phrases.

The Context Mechanism also addresses a problem noted by Grosz which is associated with the important issue of how focus is shifted during the discourse. In her model shifting is heavily dependent on the structure imposed by the assembly task, and in fact Grosz points out ([Grosz77] p.158) that "The major problem to adapting the focus representation to kinds of discourse other than task oriented dialogues is to augment the mechanisms for shifting focus... For such discourses, shifts in focus are often more gradual than in the task dialogues, and structural indications of shifts (segmentation) occur less often." The Context Mechanism provides a means for the gradual modification, through acquisition and degrading, of context information. New context factors are created as a side-effect of interaction with memory, and the significance weights associated with these factors are degraded as processing progresses. This results in the presence of many factors with different scopes, and a relatively smooth and gradual shift of the focus space defined in terms of a context activation threshold.

The Capture Context Mechanism provides a less rigid alternative to strongly predictive application of discourse knowledge structures. However, context factors derived from knowledge structures such as scripts or the task-oriented structures used by Grosz could contribute to the context information represented and hence to the definition of focus. Context information from other types of context factor can be used to "activate" such structures. The Context Mechanism was claimed to have advantages over a number of focus based mechanisms for interpretation in context. Thus it is not necessary to keep salient entities separately from memory because the indexing scheme allows efficient implementation of activation-threshold searches. The Context Mechanism generalizes and improves Grosz's mechanism for global focus by allowing for the following: the application of context to a wider range of interpretation problems; discrimination between sets of entities and between entities outside the focus space; a gradual shift of focus; and the combination of heterogeneous context information in a uniform mechanism.

5.7 Using the context mechanism in other frameworks

The issue considered in this section is whether the Capture context mechanism, viewed as a general AI technique for language processing independently of the type of task and system structure tested here, could be used for processing other types of text or within other system architectures.

With regard to processing a wider range of texts it would seem to be possible to implement some additional context factor types that were not considered to be necessary for, or testable in, the database capture task (or the particular choice of test database domains). For example, for texts in which spatial and temporal relationships are prominent, that is where temporal and spatial proximity would affect foregrounding, it would seem appropriate to provide context factor types for temporal proximity to events being described, and for spatial proximity. The interpretation of texts describing sequences of instructions for robot tasks would probably benefit from the introduction of these factor types. Similarly, for modelling conversational discourse, a type of context factor could be provided that would increase the context activation of memory entities that are closely related to the participants' goals; such a factor type would in particular be needed in human-machine dialogue to handle information relevant to the user's goals.

As suggested earlier, further types of context factor for processing texts that follow scripts or other predictive discourse structures could be used to "activate" such structures by increasing the context activations of the entities that are part of the memory representations of such structures. It may also be appropriate for processing certain types of narratives to have a type of context factor that would be used to periodically foreground sets of memory entities closely associated with the principal characters of the narratives. This assumes that these characters could be identified in the first place, and a similar assumption also applies to the activation of the predictive discourse structures that were just mentioned. Lower level context factors, such as the ones implemented in the present system, could be used to "bootstrap" the creation of these higher level types of context factor, by using context activation to choose the relevant discourse structures and story characters.

These remarks about additional types of context factor for dealing with more complex texts than the simple descriptive ones handled by the system are, of course, only speculative. Thus I have not indicated how such additional factor types would be managed, i.e. which text processing operations would create instances of them and how their significance weights would be degraded. However, the inclusion of new factor types does not complicate the control structure for applying context information, and because of this it is easy to envisage, if only at the general level, natural ways of extending the range of data that might be dealt with within the structure of the implemented system.

Moving to system structure, the context mechanism is largely independent of many of the design decisions about system component relationships that were made during the project. Some of the context factor types, in particular history of processing and association, depend, respectively, on

the processing and representation models adopted in the system. It is probably straightforward to design the counterpart of the association function for a different underlying memory mechanism, although making use of some trace of memory processing as a context factor type could prove difficult for different memory processing models. (This may be considered to be an advantage of the marker processing scheme.) In general, however, the use of context factors and context activation should be applicable to systems in which memory representation and processing were of a different type from those used for Capture.

The context mechanism might also be useful in variations on the text processing strategy used in the project. For example, it could be used in a text processing system in which an inferencer would generate all possible inferences by combining the results of language interpretation, with, and only with, statements in a focus space determined by context activation. This could constrain the inferences made by the system (e.g. by Cater's system [Cater81]) without missing those necessary for understanding. This type of context application is different from the application of context in the present system which can be characterized roughly as constraining an inference process which is primarily a matter of search, as opposed to one constraining the generation of new statements within a less restricted framework.

Another example of the use of the context mechanism in a different text processing system environment would be its use for earlier interaction with the (sentence) analyser component of the system. Thus the context mechanism could be used to constrain, during parsing, the choice of word senses and semantic features to be associated with the referents of anaphora. Such a scheme would have disadvantages; for example, it is not clear to what extent it is safe to make use of constraints predicated by context before all the syntactic information in the sentence has been utilised (see e.g. [Ritchie76]). In addition, for most cases, this use of memory in conjunction with context to constrain analyser processing is simply an issue concerning efficiency (given, of course, that cognitive modelling is not being considered). Thus the incorrect possibilities considered by the analyser, if it was not constrained, could be filtered out at a later stage by the other components [Woods73]. Early application of context information would therefore only seem worthwhile if the gains in efficiency were important enough to offset the disadvantages, such as the one mentioned above, associated with such an approach. If the gains in efficiency are worthwhile, and this is unclear at present, then it seems likely that the context mechanism developed for the present system would prove useful to this approach.

Perhaps a more promising application for the context mechanism in aiding the analysis process would be in the design of a robust analyser that would cope with violations which would cause the type of analyser used by the present system to fail. These violations include the lack of agreement between semantic category restrictions on verb cases and their syntactic fillers (see e.g. [Wilks78]), and various forms of ungrammaticality. In such cases it might be possible for the context mechanism to choose the appropriate senses for all the words occurring in the sentence, and to choose referents for anaphora, to enable the analyser to produce at least a partial structure for the sentence. Under such an arrangement, the context mechanism would be operating in a mode that would be prone to error. Despite this, the system could still be more useful than one that failed completely.

Chapter 6

Database Capture Task

6.1 Language processing tasks

It is not easy to formulate a general definition for "understanding" natural language texts. In particular what an understanding system does in terms of drawing inferences and assimilating new information as a result of processing a text (or any other form of discourse) probably depends largely on the task for which the text was read. It is therefore appropriate, especially within the context of automatic language processing, to evaluate understanding with respect to a task to be performed as the result of processing a text.

This chapter describes how Capture carries out a specific task, that of creating a relational database, as the result of the text processing that it performs. The present section attempts to delineate the type of task processing performed by the Capture system as compared with that of other experimental natural language processing systems. The description of the task specific component of Capture, and the remarks made about other systems, are only concerned with what happens (in the text processing system) before an external system (e.g. a robot, a database management system (DBMS), or an electronic mail system) takes over to actually perform the task itself. This "one-way" assumption is a limiting one, i.e. it excludes interaction between the language processor and the external task system during the latter's operations; but it is one that is commonly made. Further, Capture is not, at present, concerned with response to a user in natural language, e.g. to seek clarification of input texts.

The style of task specific processing performed by Capture is very different from a number of systems that have a similar application, i.e. generating a database from a body of texts. These are the systems that perform this task without the use of a knowledge base relating to the domain with which the texts are concerned (i.e. the equivalent of memory in the Capture system). Two examples of such systems are those described by Sager [Sager81] and Cowie [Cowie83].

Sager's system uses sophisticated syntactic processing to produce tables from the processed texts, one application being the automatic processing of

medical records. Each sentence is processed by a parsing component, a transformation component (which "regularizes" the parse), and then formatted by a third component into tables called "information formats" (in later work the tables are derived by flattening a "format tree" produced for the sentence). The last component uses rules that are specific to the domain that is being handled. These rules use syntactic information and a domain specific categorization of words in order to place each word from the sentence in one of the columns of the information format. For example, in medical radiology reports, there is a subcategory "TEST" of nouns that includes "film" and "X-rays", and there is a TEST column in the information format so that the processing of "X-rays indicate metastasis" places "X-rays" into this column. The information format for a subject area is not an existing target database format, but is defined after analysing syntactic and lexical regularities in a sample of the documents to be processed. Sager's system is thus not a database generator in the most commonly accepted sense of database (e.g. a strict relational model database). The formatted material generated can be described as a semi-formal database with a mixture of structural and textual elements.

Cowie's system processes descriptive texts about single objects. It uses keywords to identify parts of the object description (e.g. keywords "flower" and "petal" for the flower-part of a plant description). It chooses fillers for attributes of these parts (e.g. "red" would fill the colour attribute of the flower-part of the plant description) by checking property names attached to words in the dictionary (e.g. "vision-colour" for "red"). Syntactic information is ignored; the text being split into segments at "pivotal points" such as conjunctions and punctuation marks.

Apart from its straightforward limitations, e.g. failure to pick up relevant information if it is somewhat indirectly expressed, the approach taken by Cowie cannot handle text concerning more than one object or situation, since this would confuse, or simply merge, the information associated with the objects or situations described. A related problem is that both these systems do not have satisfactory mechanisms for reference resolution, and so would not be able to process the examples handled by the Capture system, even though Sager's system does replace some of the words in the format tables with antecedents (also appearing in the format tables), and is therefore capable of handling some cases. Both systems make use of simple domain specific lexical categorization. This is not adequate for database classification of objects (described in Section 6.4) because one level of lexical categorization does not capture, for example, the (specialization) relationship between "machine", "peripheral", and "disc-drive". In general, both Cowie's and Sager's systems can be viewed as template-filler processors, where the templates constitute a very minimal knowledge base, and with rather weak control of the detailed character of the fillers, which

are primarily text strings.

However, if other tasks are considered, then we can find many examples of language processing systems that make use of more interesting knowledge bases. One category of such tasks are those that are essentially linguistic in nature, for example translation, summarization, and question answering. The systems described by Dejong [DeJong79] and Tait [Tait82] use knowledge encoded in script-like formalisms in order to produce summaries of texts. The BORIS system [Lehnert83] uses a conceptual memory knowledge base organized in terms of MOPS, in order to achieve a depth of understanding which allows sophisticated question answering behaviour after processing a short story.

The knowledge based processing in Capture is better characterized as not only making use of the knowledge base to perform language processing, but also as using additional knowledge in order to perform an extra-linguistic application task. Examples of other systems that fit this characterization are the Consul system being developed at USC/ISI (see e.g. [Mark81]) and the natural language interface to a graphics system developed at BBN (see e.g. [Brachman79a]). Both these systems use the KLONE formalism for knowledge representation.

In the BBN system the PSI/KLONE interface (i.e. the interface with "Parsing and Semantic Interpretation using KLONE") is used to drive a sophisticated graphics display using natural language commands. In addition to encoding general conceptual knowledge about the objects being displayed (and referred to by the user) the task requires encoding knowledge about how to display these objects. A "display expert" component uses both the general conceptual knowledge and the display specific knowledge in order to drive the actual display. Attached procedures are used to generate "display form" concepts from object descriptions or from representations of commands to alter the display. The attached procedure mechanism is then used again to invoke "how to draw" procedures attached to the "display form" concepts.

The Consul system is designed to provide a co-operative natural language interface to interactive services such as an electronic mail service. The knowledge base includes concepts in terms of which users phrase their requests ("user knowledge"); knowledge about basic operations, such as "transfer", that can be used to describe services ("systems knowledge"); and specific knowledge about a particular service ("service knowledge"), such as concepts for transfer operations for sending different kinds of messages. The system uses rules, themselves represented in KLONE, for mapping user knowledge requests (created by the parser) into service knowledge structures that correspond to actually executable service operations. (If this is not possible, then the knowledge structures are used for explaining

why the user's request cannot be carried out.)

The framework chosen for application task processing in the Capture system is very similar to that of these two knowledge-based approaches. Thus the memory knowledge base used by Capture encodes language related, domain related, and task-specific knowledge, and also encodes the relationships that hold between them. In particular, the task specific knowledge is independent from the language and domain related knowledge in the sense that memory entities and assertions representing the task specific information (including further restrictions on the domain knowledge that apply in the context of the task) can simply be added to a memory knowledge base already containing assertions encoding the language and domain related knowledge. But the way that control is organized in the task-specific component in Capture is different from the use of attached procedures, or the application of mapping rules for operating on knowledge structures. In Capture, the task-specific component uses the knowledge in memory via task-specific operations. These have the same properties as the interpretation operations used in the system, being procedures that interact with memory by the use of retrieval operations and the creation of new memory assertions. These task specific operations are called at a number of points during text processing as described in the following sections.

It is not argued that Capture's approach to application task processing is to be preferred as such to the approaches taken in the BBN and USC/ISI systems, the motivation for the task component was rather different: the implementation of the database capture component was mainly aimed at verifying that a non-trivial application task could be performed (demonstrating, as suggested earlier, a certain level of language understanding) within the overall framework of the Capture system as a whole, based on the representation and processing techniques developed in the project.

6.2 The database capture task

The "database capture task" is the processing of natural language texts to extract that part of their propositional content that can be included in a structured database with a pre-specified format, the "target database". There are many possible applications for a system performing this task such as the generation of relational databases from e.g. personnel records, computer maintenance records, etc.

A text processing system for database capture can also be used, of course, by a human operator for the purpose of creating a database when no

independent text already exists. The question of whether or not writing texts for such a system is superior to some other user interface (e.g. an interactive form-filling system) was not addressed in this project.

It should be noted at this point that this task is distinct from that performed by the kind of natural language interface for database update discussed by [Kaplan81] and [Maier82]. Their type of database update system would handle DBMS-oriented natural language commands such as "Change the status value for Smith from 15 to 30". These systems concentrate on the interpretation of command verbs with respect to a given state of a database. The database capture task is concerned, instead, with processing paragraphs of text that need not have been written with a database in mind, and that do not make assumptions about the state of the database. However, it is easy to see that there is an overlap between systems performing the update and creation tasks: some updates could be formulated in a more natural way detached from the database management system. Most of the effort, by far, in building natural language interfaces to databases has gone into the design of natural language query systems (e.g. [Woods78b], [Walker78], [Boguraev83] and many others). Much of this work has been concerned with interpreting the quantificational structure of natural language questions in order to generate corresponding database queries, an issue which was not tackled in this project. Clearly however, natural language query systems are less suitable as an application task for investigating mechanisms for interpreting connected prose as compared with the database capture task (see below).

Database capture, in the sense defined, is considered to be a possible practical application area for the general text processing techniques that were developed in this project. Thus the work reported here on database creation from natural language text is to a limited extent a study of the feasibility of such an application of automatic text processing. However, the main motive for choosing this task is its utility for testing the natural language processing techniques used by the system. The advantages of the database capture task in this respect are the following. Working with limited discourse domains in this application does not mean that we have made a simplifying assumption that is too unrealistic. The task requires the interpretation of language constructs in context in order to generate explicit database statements. This means that, in particular, there must be mechanisms for anaphoric reference resolution, and the resolution of sentence level word sense and structural ambiguity. These problems associated with sentence level ambiguity are crucial to all text processing tasks.

Another advantage of this task is that we can make simplifying assumptions about the texts to be dealt with that cannot be made when performing other

types of processing. For example, in-depth comprehension of narratives requires paying much attention to the affects and goals of the characters in stories. Having a test text type which does not require handling such complexities means that it is possible to concentrate on the more straightforward linguistic aspects of descriptive text, such as reference resolution and the derivation of the propositional content of descriptive sentences. Techniques for dealing with these latter issues are, of course, still relevant to processing narratives. Thus studying techniques for processing simple descriptive texts, such as the texts used for database creation, which are not concerned with human social affairs is considered to be worthwhile, and probably a necessary first step, before general text processing can be tackled.

The present system is not able to handle real, unedited, texts. This is because the coverage of structures handled by Boguraev's analyser is not wide enough for this to be feasible (for example it cannot handle conjoined clauses), and also because of the small size, at present, of the system lexicon. In addition, the number and complexity of the language interpretation operations would have to be increased since the ones that were implemented cannot, and were not expected to, handle the full range of interpretation possibilities exhibited by descriptive texts (see Section 4.10). However, I believe that doing the work necessary to overcome these shortcomings within the framework, for memory and context mechanisms, of the present system could lead to a realistic system for database input for some very restricted applications, though it is certainly the case that the amount of work involved would be very considerable.

6.3 The target databases

The target databases used to illustrate the processing that can be performed by the system are relational databases. The choice of the relational model for this purpose was made because the model is well known and has a reasonably well defined and simple organizational structure. Attributing semantics to a relational database is not a well defined process, although there have been attempts to incorporate semantic specification into variations on the relational model (e.g. Borkin's Semantic Relation Model [Borkin80]). The information in memory about a database can be thought of as providing a kind of "conceptual schema", or at least as providing a semantic definition, for the database that is adequate for

performing the database capture task for simple texts. (*)

If a different database model had been used, then the form of the database descriptions in memory would have to be changed (to a lesser or greater extent depending on the kind of data model involved), and so would the task specific operations, described later, that make use of these. However, the language interpretation operations and the context mechanism used with these operations would not be affected by the choice of a different database model. Examples indicating how the semantics of a relation in a relational database are expressed in memory were given in Chapter 2 on memory representation. Further details of this are included in the section below on task specific operations.

As mentioned earlier, the two test databases were the Machines Database, dealing with data processing machines, and the Artifacts Database, about museum artifacts. The "enterprise", in database systems jargon, for the Machines Database could be a retail organization that wished to keep track of descriptions of the data processing machines (models) that it marketed, their suppliers and manufacturers. In this database MACHINES/RELATION gives the model number, its cost, categorization into machine type (e.g. computer or printer), the weight of the model, and (somewhat unrealistically) its colour. The MANUFACTURES/RELATION specifies the model number, the manufacturer name, and the city of manufacture. The SUPPLIERS/RELATION gives the names of suppliers and their status (which can be interpreted as credit status, or an indication of reliability). The SUPPLIES/RELATION records the suppliers that (regularly) supply particular models.

For the Artifacts Database the enterprise is envisaged to be an anthropology museum, the database being created from natural language records about artifacts and collectors. In this database ARTIFACTS/RELATION gives the number of an artifact, its condition, and a categorization of its type (e.g. ornament or weapon). The ORIGIN/RELATION specifies the collector of an artifact and its place of collection. The COLLECTORS/RELATION gives a collector's name, occupation (e.g. missionary), and nationality. The test databases thus have a modest level of complexity, and certainly cover data broad enough to challenge the interpretation mechanism if described in running text. There is of course no requirement for test purposes that the number of database objects handled is large, i.e. that the database itself should be large.

(*) It may be possible to provide more rigorous semantics for relational databases via their memory descriptions and the semantics of memory assertions discussed in Chapter 2, but investigating this was considered to be out of the scope of the present project.

The specific column names of the relations in the two databases are listed below. Underlined columns indicate the attributes forming a key for each relation.

A) Machines Database

MACHINES/RELATION
  MC/MCNUM MC/COST MC/TYPE MC/COLOR MC/WEIGHT
SUPPLIERS/RELATION
  S/SID S/STATUS
SUPPLIES/RELATION
  SMC/SID SMC/MCNUM
MANUFACTURES/RELATION
  M/MNAME M/MCNUM M/CITY

B) Artifacts Database

ARTIFACTS/RELATION
  ARTF/NUMB ARTF/TYPE ARTF/COND
ORIGIN/RELATION
  ORIG/ARTN ORIG/COLL ORIG/PLAC
COLLECTORS/RELATION
  COLL/NAME COLL/OCCP COLL/NATN

The keys specified above are the only candidate keys for the relations. The functional dependencies that are assumed to hold for the two enterprises can be summarized by saying that all the relations are in "third normal form" (they are, in fact, in fourth normal form). Thus there is no functional dependency between, say, the colour of machines and their cost, or between the occupation and nationality of collectors. (*)

One reason for handling two domains (and corresponding target databases) was to increase the range of example texts. A second reason was to verify that a change in the domain and target database would involve only additions to the contents of memory (unless it turned out in practice that the different domains naturally led to input texts of substantially different complexity). This limitation of changes to memory only does hold for Capture proper (i.e. excluding the analyser and its dictionary) as far as the tests went, although processing examples from a second domain did bring to light a number of "bugs" in the implementation.

(*) Explanations of functional dependency and third normal form are given in many database system textbooks such as [Date81].

A number of assumptions were made about the capabilities of the database management system to which the information extracted from the texts would be passed, but care was taken to make these sensible ones, matching the current state of DBMS research (though they were perhaps in advance of current commercial DBMS). The first assumption is that the DBMS is thought of as operating under the "open world assumption". That is, assertions corresponding to tuples that are not included in relations are interpreted as being unknown, rather than false. This is because the database capture task being performed is thought of as a process of accumulating information rather than updating an existing complete model of the world described by the database.

A second and necessary assumption, related to the open world assumption, refers to the treatment of null values in the database. It is assumed that the DBMS allows "null" or unknown values for entries in non-key columns. For instance, a tuple in the 'MACHINES/RELATION' with key 'P9000' can have the 'MC/COLOR' entry unknown. This is a necessary assumption for many of the applications envisaged for database capture from text, because the texts used need not have been written with a particular data model in mind, and only known or observed information, which may not be complete, will be recorded in the texts. Further, it is not always possible to specify default values for particular columns of relations. A mechanism for default values is in any case often considered to be the responsibility of the DBMS. (*)

A third assumption is that the DBMS can enforce constraints about a particular database in such a way that some of the information passed to it by the text processing system could be ignored, or could lead to other side effects in the database such as the deletion of a tuple. As noted earlier, it is assumed that the DBMS has no way of communicating about these events to the text processing system and therefore they cannot affect the interpretation of the text. Thus, for example, if the sentence "P9000 is green" is encountered, and then "P9000 is red", it is for the DBMS to perform an update, or to ignore the information generated by the text processor for the second sentence because it is considered to violate a constraint.

In summary, the text processing system assumes that it is sending information to a sophisticated DBMS that has its own means for dealing with incomplete or even inconsistent information. The database descriptions represented in memory are thus not intended to help the DBMS carry out its functions, but only to allow the text processing system to generate

---

(*) Nevertheless the framework of the Capture system can, I believe, straightforwardly accommodate task-specific operations for some handling of default database values, which would be specified in the descriptions of database relations in memory, but this was not tried out.

information from text that may be used by a DBMS. The techniques used for describing databases in memory could perhaps be useful for a DBMS since the descriptions can be regarded as high level conceptual schema. Similar techniques have indeed been tried for this purpose, but the problems involved are considered to be database theory problems (as yet largely unresolved), and only the text processing issues that are relevant to the database capture task have been considered in the present work reported here.

One consequence of the assumptions just listed is that the interface between the text processing system and the DBMS is very simple. The output of the task processor component of the text processing system is a list of "database creation statements". Each database creation statement has the form

(<relation name> (<list of (<column-name>, <value>) pairs>)).

One, or more, of the column names in a creation statement must form the "key", or "joint key", for the relation in the statement. For example, processing the sentence "Plexir supplies P7720 which is a computer" results in the following two database creation statements.

(SUPPLIES/RELATION ((SMC/SID Plexir) (SMC/MCNUM P7720)))
(MACHINES/RELATION ((MC/MCNUM P7720) (MC/TYPE computer))).

'MC/MCNUM' is the key for the 'MACHINES/RELATION', and 'SMC/SID' and 'SMC/MCNUM' form the key for the 'SUPPLIES/RELATION'.

Database creation statements can be translated by a post-processor into a request in the data manipulation language (DML), for example SEQUEL [Chamberlin74], of the DBMS to include the new information into the target database. The post processor would do this using a syntactic transformation, and (depending on the DML) might also have to evaluate the rather trivial memory operation of locating all the column names for a relation. (*) For the first database creation statement, a SEQUEL-like statement might be

INSERT INTO SUPPLIES/RELATION:
<SMC/SID=Plexir, SMC/MCNUM=P7720>.

The assumption that the DBMS can handle unknown values is relevant here because it allows database creation statements which do not contain all column names to be translated into a DML statement for inserting a tuple into the database. As remarked by Mike Gray ([Gray81] p.148) "When there are unknown values in a database, it is not always appropriate to reject a

---

(*) Note that whether creation statements can be inserted efficiently is a question for the DBMS.

request for the insertion of a new tuple whose key matches that of an existing tuple. It may be possible to merge the information in them." For the second database creation statement above, the SEQUEL-like statement might therefore be

```
INSERT INTO MACHINES/RELATION:
 <MC/MCNUM=P7720,
  MC/TYPE=computer,
  MC/COLOR=*unknown*,
  MC/COST=*unknown*,
  MC/WEIGHT=*unknown*>.
```

A further database creation statement generated from a later sentence might be

```
(MACHINES/RELATION ((MC/MCNUM P7720) (MC/COST 900)))
```

with the corresponding DML statement

```
INSERT INTO MACHINES/RELATION:
 <MC/MCNUM=P7720,
  MC/TYPE=*unknown*,
  MC/COLOR=*unknown*,
  MC/COST=900,
  MC/WEIGHT=*unknown*>.
```

It is assumed that the DBMS would merge the information (i.e. cost is 900) from the second INSERT statement into the tuple created by the first one.

In a production system it would also be necessary to replace the word senses that currently appear in database creation statements with surface lexical stems, or with database specific codes (e.g. for colours). The information required for doing this can easily be represented in memory if necessary.

The following sections will describe how the database creation statements are generated by the task processor of the text processing system.

6.4 Task specific operations

This section describes memory operations that are specific to the database capture task. More precisely, these operations are concerned with locating the relational database description entities represented in memory that correspond to language related entities in memory. The memory operations know about the form in which these database descriptions are represented in memory. Memory representation examples concerned with the database descriptions were given in Section 2.8. As explained in that section, a relational database is described in terms of collections of memory entities for each of the relations in it. These collections are structured by correspondence assertions (centred around an entity such as 'manufactures/relp') in order to encode the relationships between language related entities (e.g. 'manufacture'), predicates underlying relations (e.g. 'relp/manufactures') and database name entities (e.g. 'MANUFACTURES/RELATION'). The task-specific operations are implemented in terms of memory retrieval operations, which in turn are implemented in terms of the marker processing model described in Chapter 3.

Task specific operations can be classified into operations that either (a) locate entities describing the predicates underlying relations that correspond to language-related entities, or (b) extract (and check) actual DBMS implementation names (i.e. the names of relations and columns) from the entities describing the underlying predicates (e.g. 'relp/entry' entities standing for typical entries in the column of a relation, see Section 2.8). Some of these operations are now described; the first three are of type (a), while the following three operations are of type (b).

1) Locating underlying predicates

The predicates sought are specializations of language related and domain related predicate entities, 'manufacture' and 'colour/of', for example. The operation simply finds a specialization of the language-derived predicate that is also asserted to be the 'relp/statement' of a relation (Section 2.8). For example, the following memory assertions

(Specialization: manufacture of make1)
(Specialization: relp/manufactures of manufacture)
(Corresponds: relp/manufactures to manufactures/relp as
relp/statement to db/relp)

are used to locate the underlying predicate entity 'relp/manufactures' from the language predicate entity 'make1'.

2) Classification of database entities

In both the test databases one or more of the underlying predicates classified individual entities in the database domain. These classifications appear in the target databases as the columns for machine type, artifact type and collector occupation. The task-specific operation is given a memory entity, 'P997' say, that is an instance of a type of object in the database domain and, if possible, returns an entity that is a valid classification of the object for that database. For example, the artifact type for 'P790', known to be an armlet, would be 'ornament1'. Similarly 'P9000', a minicomputer, would yield 'computer'. The memory assertions used for the first example are

```
(Specialization: P790 of armlet1 (instance))
(Specialization: P790 of artifact/dbentity)
(Specialization: armlet1 of ornament1)
(Corresponds: db/artifact/types to artifact/dbentity as
        dbentity/classif to dbentity)
(Specialization: ornament1 of db/artifacts/types).
```

The last two memory assertions encode task specific knowledge which asserts that 'ornament1' is a valid database classification in the Artifacts Database. In another database, for instance one about jewelry prices, the entity 'armlet1' might itself be a valid classification, though it is not in the Artifacts Database. The database classification operation finds the generic database entity, e.g. 'artifact/dbentity', for the object that it is given, and then uses the 'dbentity/classif' associated with this to choose an entity that is a valid classification and that is above the entity being classified in one of the hierarchies.

3) Finding database description entities for domain-related entities

An operation of this type would, for instance, be given the domain-related entity 'armlet1' and return 'artifact/dbentity', and similarly 'disc-drive1' would return 'machine/dbentity'. The following memory assertions are used by the operation in the second example.

```
(Specialization: disc-drive1 of peripheral)
(Specialization: peripheral1 of machine)
(Specialization: machine/dbentity of machine)
(Corresponds: machine/dbentity to supplies/relp as
        relp/dbentity to db/relp)
```

The entity returned by the operation corresponds to a 'relp/dbentity' in a database description and shares an ancestor, in the specialization hierarchy, with the domain-related entity given to the operation.

4) Database names for arguments of underlying predicates

This operation locates the names, in the target database, of the column and relation appropriate to a database value, where this value has been asserted to be the argument of an instance of an underlying predicate for the relation. For example, suppose we are given the following memory assertions

(Specialization: E22 of relp/machine/weight (instance))
(Corresponds: 290 to E22 as
machine/weight/value to relp/machine/weight)

and have a part of the database description as follows.

(Specialization: MACHINES/RELATION of relation (instance))
(Corresponds: MACHINES/RELATION to machine/relp as
relp/relation to db/relp)
(Specialization: MC/WEIGHT of column (instance))
(Corresponds: machine/weight/entry to machine/relp as
relp/entry to db/relp)
(Corresponds: machine/weight/value to machine/weight/entry as
relp/entry/value to relp/entry)
(Corresponds: MC/WEIGHT to machine/weight/entry as
relp/entry/column to relp/entry)
(Corresponds: relp/machine/weight to machine/relp as
relp/statement to db/relp)
(Corresponds: machine/weight/value to relp/machine/weight as
weight/of/weight to weight/of).

The relevant underlying predicate is the entity 'relp/machine/weight' and the relevant generic argument to it is 'machine/weight/value'. Retrieval operations for finding fillers of (inherited) roles in correspondence assertions make use of the assertions listed above to locate 'machine/weight/entry', and then the column name 'MC/WEIGHT'. The same types of retrieval operations make use of the correspondence assertions to locate the entity 'machine/relp', and from this the relation 'MACHINES/RELATION'.

In general, however, locating the correct relation and column names is not simply a matter of following a prescribed chain of role-owner relationships in correspondence assertions. Thus the 'relp/entry', with which the entities 'relp/entry/column' and 'relp/entry/value' are associated, may have its 'relp/entry/value' filled by the name, or number, of a database entity rather than a simple value (such as the weight value above). For example, the other argument of the predicate instance 'E22' may have been filled by inclusion of the assertion

(Corresponds: P9999 to E22 as
machine/dbentity to relp/weight/of)

where the following assertions, in addition to those given above, appear in

the database description.

(Specialization: MC/MCNUM of column (instance))
(Corresponds: machine/mcnum/entry to machine/relp as
relp/entry to db/relp)
(Corresponds: mcnum/value to machine/mcnum/entry as
relp/entry/value to relp/entry)
(Corresponds: mcnum/value to machine/dbentity as
dbentity/number to numbered/dbentity)
(Corresponds: machine/dbentity to relp/machine/weight as
weight/of/possessor to weight/of)
(Specialization: machine/dbentity of numbered/dbentity).

The entity 'mcnum/value' is used, instead of the generic argument of the underlying predicate (i.e. 'machine/dbentity'), by the operation for locating the column 'MC/MCNUM'.

Thus the task specific operation for locating column names from predicate arguments needs to check whether the predicate argument is described as a numbered or named database entity, and then uses an appropriate "role", 'dbentity/name' or 'dbentity/number', of the database entity to locate the column name in the database description.

Another complication in this operation is that the generic underlying predicate argument, for instance 'machine/dbentity' in the last example, may take part in the description of more than one relation. The appropriate names are chosen due to their higher context activation, which is determined by a task specific factor whose scope is the set of entities in the description of a database relation. This factor is created after the creation of an instance of an underlying predicate for that relation. In the above example this ensures that that 'MACHINES/RELATION' and 'MC/MCNUM' are chosen as the results of the operation, but not 'MANUFACTURES/RELATION' and 'MMC/MCNUM', which would have been chosen if the predicate instance 'E22' had been created as a specialization of 'relp/manufactures'.

5) Database names for classification

It has been explained how a database entity, such as 'P790', because it is an 'armlet1' and an 'artifact/dbentity', can be given a classification, 'ornament1', with respect to a target database. The part of the database description that allows this to be done also indicates the database names that are used to encode this classification in the target database. Simple role extraction retrieval operations, specified by this task specific operation, are used to obtain this information once the 'dbentity/classif' (see (2) above) for the database entity has been identified. For the armlet example this entity is 'db/artifact/types', the relation name is

'ARTIFACTS/RELATION', the column for 'P970' is 'ARTF/NUMB', and the column for 'ornament1' is 'ARTF/TYPE'. The relevant memory assertions, in addition to the ones given in the section on classification of database entities, are as follows.

(Corresponds: db/artifact/types to artifact/dbentity as
dbentity/classif to dbentity)
(Corresponds: ARTIFACTS/RELATION to db/artifact/types as
classif/relation to dbentity/classif)
(Corresponds: ARTF/NUMB to db/artifact/types as
classif/id/col to dbentity/classif)
(Corresponds: ARTF/TYPE to db/artifact/types as
classif/type/col to dbentity/classif)

6) Checking for keys

This operation simply checks that a list of entities representing columns includes all the columns in the key for a relation. For example, the operation succeeds if it is given the columns 'SMC/SID' and 'SMC/MCNUM', and the relation 'SUPPLIES/RELATION'. The relevant memory assertions in this case are

(Corresponds: SMC/MCNUM to SUPPLIES/RELATION as
rel/key/col to relation)
(Corresponds: SMC/SID to SUPPLIES/RELATION as
rel/key/col to relation).

The operations that were described in this section provide the task-specific component of the Capture system with the necessary tools for exploiting the database-related knowledge represented in memory. The way these operations are used in performing the database capture task is described in the following section.

6.5 Interleaving interpretation and task operations

This section explains when the task processor evaluates task-specific memory operations so that it can produce the database creation statements corresponding to the text. In fact the evaluation of task-specific and language interpretation operations is interleaved in a simple manner, as explained in this section. An alternative to this mode of processing would be to perform the necessary interpretation operations on all of the output of the analyser for the text before evaluating the task-specific operations required to produce the database creation statements. But this is a less satisfactory option.

Thus there are two reasons why interpretation and task operations should be interleaved. First, interleaving allows the interpretation process to make use of certain assumptions about entities in the database domain. For example, the interpretation process might be able to make use of the assumption that in the database world the 'object' case of manufacturing actions is normally filled by a 'machine'. The intention is to be able to make use of such a restriction without its persisting when the database specific part of memory is removed. An example of how this works is given later in this section. The second reason for interleaving has to do with accumulating context information. The evaluation of task-specific operations leads to the creation of context factors. This can, in principle, aid subsequent interpretation operations if, for example, the context activations of entities concerned with the database domain are increased directly as a result of the creation of these context factors, or indirectly if association or subject area context factors are created.

The sequences of task specific operations that are evaluated are different for handling predicate clauses (verb and state clauses) from those for handling "be-clauses". These two cases will now be described with some examples of the resulting database creation statements. The handling of predicate clauses is described first.

1) Predicate clauses

After the generic predicate entity for the clause, e.g. 'manufacture' or 'weight/of', has been identified (Sections 4.5), a task-specific operation for specializing this predicate entity is evaluated. In the database capture task this is the operation that was referred to as locating underlying predicates (Section 6.4), and it would return 'relp/manufactures' for 'manufacture' and 'relp/machine/weight' for 'weight/of'. The predicate instance entity created for the clause is asserted to be a specialization of the underlying predicate. The evaluation of the task-specific operation has now been completed and control is returned to the clause interpretation operation, which need not know whether the predicate instance was specialized to an underlying predicate for a relation.

However, if the newly created predicate instance was asserted to be an underlying predicate, this can affect the evaluation of argument specialization and reference resolution operations that are initiated by the clause interpretation operation. For example, during the interpretation of the sentence "It is manufactured by Plexir", the predicate entity 'manufacture' is specialized to 'relp/manufactures' so that the argument entity derived from the case label 'object' is not 'manufacture/obj' but 'machine/dbentity'. The fact that the object case may be described as a

'machine/dbentity' can be used as a constraint by the reference resolution operation that is evaluated for the pronoun "it".

The clause interpretation operation evaluates reference resolution operations for each of the noun phrases of the clause, the referents in the current example being 'Plexir1' and a particular 'machine/dbentity', 'P9000' say. Control then reverts to the task processor. This evaluates a task-specific operation for finding the database relation and column names for each of the arguments of the instance of the underlying predicate created for the clause. The task processor then merges the information that results from evaluating these task-specific operations into one or more database creation statements. For instance, the following database creation statement is output for "It is manufactured by Plexir":

(MANUFACTURES/RELATION ((M/MCNUM P9000) (M/MNAME Plexir1))).

An example in which the underlying predicate is derived from a state-clause is "The weight of the machine is 220". The interleaving of interpretation and task-specific operations is the same as for the verb-clause case, so the database creation statement output by the task processor might be

(MACHINES/RELATION ((MC/MCNUM P8000) (MC/WEIGHT 220))).

Finally, an operation is evaluated to check that the set of column names in the database creation statement(s) includes the columns in the key of the target relation as specified in memory.

2) Be-clauses

The output produced by the task processor for the sentence "P2000 is an armlet" is

(ARTIFACTS/RELATION ((ARTF/NUMB P2000) (ARTF/TYPE ornament1))).

A be-clause interpretation operation (Section 4.6) begins processing the analyser representation of the clause by identifying, or creating, the generic entity for the clause, which is simply 'armlet1' in this case. It also evaluates a reference resolution operation for identifying, or creating, the referent of the subject of the clause (i.e. the entity whose description is being refined by the clause) which in this case is the entity 'P2000'. 'P2000' is then asserted to be a specialization of 'armlet1'. At this point the be-clause interpretation operation activates the task processor which handles the consequences, for the task, of the new specialization assertion.

The task processor evaluates a memory operation (Section 6.4) to find a generic database entity for the entity 'P2000'. This operation returns 'artifact/dbentity', and the following assertion is included in memory

(Specialization: P2000 of artifact/dbentity).

The task processor then evaluates an operation to find a classification of 'P2000' with respect to the Artifacts Database. This operation succeeds for 'P2000', and 'ornament1' is returned as a valid classification (Section 6.4). Since the classification operation has succeeded the task processor evaluates an operation that enables it to construct the database creation statement that encodes this classification for the Artifacts Database. This operation, which was referred to as finding database names for classification, extracts the relation and column names from the description of the Artifacts Database, and the database creation statement given above, encoding the classification, is output by the task processor.

The database capture task processing for an embedded clause occurs, in the manner described above for predicate-clauses and be-clauses, at the time that the embedded clause is interpreted, i.e. before the interpretation of the higher level clause has been completed. Thus the task output for embedded clauses is produced first; for example, the output produced for the sentence "Plexir manufactures P9000 which is supplied by Smith" is

(SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P9000)))
(MANUFACTURES/RELATION ((M/MNAME Plexir1) (M/MCNUM P9000)))

Similarly, the sentence "P9000 is a computer that is manufactured by Mikota" results in the following database creation statements

(MANUFACTURES/RELATION ((M/MNAME Mikota1) (M/MCNUM P9000)))
(MACHINES/RELATION ((MC/TYPE computer) (MC/MCNUM P9000))).

Some embedded clauses do not convey new information, and these do not generate any database creation statements. Restrictive relative clauses, which are interpreted by reference resolution operations, are of this type. For instance the sentence "Plexir manufactures the computer that is supplied by Smith", might produce the output

(MANUFACTURES/RELATION ((M/MNAME Plexir1) (M/MCNUM P9000)))

Verb and state clauses for which the clause predicates do not correspond, i.e. cannot be specialized to, the underlying predicate for a relation in the target database, do not generate any database creation statements. An example of such a clause is "Haddon donated P1234". Nonetheless, the memory interpretation of such clauses is still important for the database

capture task because the information conveyed by them might be necessary for later processing, for instance reference resolution in the interpretation of the sentence "The artifact that was donated by Haddon comes from Woodlark".

The task processing for clauses with plural noun phrases can result in the generation of more than one database creation statement. The memory interpretation of such clauses involves retrieving, or creating, a memory entity that represents the set of objects referred to by the plural noun phrase. For example during the interpretation of the sentence "Smith supplies P9000 and P9090" an entity, 'E1' say, will have been created, and memory will contain the following assertions

(Specialization: P9000 of E1 (member))
(Specialization: P9090 of E1 (member))
(Corresponds: E1 to E2 as machine/dbentity to relp/supplies).

The task specific operations initiated by the task processor will return 'E1' and 'Smith' as well as the database names. This is because it is 'E1' that fills the 'machine/dbentity' argument of the underlying predicate instance 'E2'. The task processor evaluates a retrieval operation for locating the elements of the set represented by 'E1' and then produces the following database creation statements

(SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P9090)))
(SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P9000))).

Similarly, the output generated for the sentence "They supply the printers" might be

(SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P1000)))
(SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P2000)))
(SUPPLIES/RELATION ((SMC/SID Clark1) (SMC/MCNUM P1000)))
(SUPPLIES/RELATION ((SMC/SID Clark1) (SMC/MCNUM P2000))).

6.6 Handling other constructions

There are different categories of constructions that cannot be handled by the system, so database creation statements are not generated for them. There are various reasons for these limitations. The main categories of excluded constructions are

(1) Constructions that cannot be handled by the analyser or by the interpretation component.

(2) Constructions that cannot be mapped in a straightforward way into statements in a relational database.

(3) Cases for which the mapping into database creation statements depends on inference mechanisms not supplied in this project.

(4) Cases that could be handled if the task specific operations were more sophisticated.

(5) Constructions that could be handled if additional task specific operations were defined.

Examples of categories (1), (2), and (3) will be given below under the heading "limitations" and examples of (4) and (5) under the heading "extensions". I will indicate the categories to which, in my view, these examples belong. Needless to say, these examples represent only a sample and there are many other constructions that would not be handled by the system.

Limitations on coverage

-- A number of constructions that cannot be handled by the analyser were described in Section 4.8. These include, for example, "P200 and P700 were collected by Clark and Nifilsk respectively" (parallelism with surface order); "The condition of P700 is not good" (negation); "So does Plexir" (ellipsis); and "The spear head is razor sharp" (metaphor). (category 1)

-- Comparatives cause problems for database input. Examples are "P700 is older than P900", and "The condition of P700 is better than the condition of P200". The first example would require allowing the use of ranges of values in the database. It might be possible to handle the second example using a specification of allowable state descriptors (see below). (category 2)

-- Uncertain facts. For example "This artifact might have been collected in Daui". (category 2)

-- The database capture component always assumes a distributive interpretation of statements such as "The cost of the machines is 200". Even though the interpretation component does not commit itself to choosing a distributive or collective interpretation, translation into database creation statements must do this. (category 3)

-- Another type of quantification that cannot be handled by the database capture task is exemplified by "Two of them are from Woodlark". (category 2)

-- There are many cases that require rule-based inferences to manipulate structures created by the interpretation component. In order to deal with these a mechanism for applying rules represented in memory would have to be implemented (cf. the mapping rules in Consul, [Mark81]). Examples are "P700 arrived with P200" (implying that they were collected at the same place); "P300 was bought from a Woodlark trader" (possibly implying that the origin of the artifact is Woodlark); and "P400 should be handled with care" (implying that the artifact should be classified as "fragile"). (category 3)

Extensions of coverage

-- The database descriptions could include a specification of allowable state value descriptions so that "The condition of this artifact is excellent" would lead to 'good' via the relationship in memory between the entities 'good' and 'excellent'. A similar extension is possible for place names and would make use of knowledge of spatial inclusion for regions. (category 4)

-- Examples like "Bevan supplied us with a canoe-prow from Daui" show that the operation identifying underlying database predicates should check the restrictions on the arguments of the underlying predicates. Currently these restrictions are only used to aid the interpretation process. (category 4)

-- Relationship interpretation currently affects the generation of database creation statements indirectly, via reference resolution. An additional task-specific operation could be implemented in order to generate database creation statements on the basis of relationship interpretation. For example, if there was a relation in the Artifacts Database listing the parts of artifacts, then a suitable database creation statement would be generated after interpreting the relationship in "The pot has a lid". (category 5)

-- For the purpose of generating database creation statements, a task specific operation could be defined that would allow the inheritance of specified properties from roles to owners in correspondence assertions. For example, given the sentence "The pot's lid is in poor condition" we may wish to state in the database that the condition of the whole assembly, pot plus lid, is poor. (category 5)

-- Another necessary extension for a more complete system is the implementation of a task specific operation for taking predicate names as values for specified database columns. For example, the Artifacts Database might have had a column indicating how an artifact was acquired by the museum. The values generated would then include "bought" and "bequeathed". (category 5)

6.7 Applicability to similar tasks

As the discussion of restrictions on texts handled for database capture (Section 6.2) suggests, there are some restrictions on the other types of text processing application task that might be undertaken using the techniques explored in this project. The restrictions include handling only texts with limited domains of discourse, because we do not know, and are unlikely to know in the near future, how to represent in memory all of the knowledge required for text processing without a domain limitation. An example of a task requiring this knowledge might be summarizing all the articles in a news magazine. However, it has to be accepted that this is not a limitation specific to the Capture techniques. It applies to all knowledge-based approaches to language processing investigated to date. Another restriction is that the interpretation of the texts for the purpose of the task cannot depend on a deep understanding of human social affairs and goals (see e.g. [Lehnert83]). This rules out such tasks as answering questions about the motivation of characters after reading passages from a novel. Research in this area is still at a very preliminary stage, and so there is not much to take over for inclusion in a Capture-like framework with wider applicability. Related to this restriction is another one also concerned with the representation of plans and goals. This is the need for "user modelling" required for sophisticated processing of texts conveying, for example, a sequence of instructions to be carried out by an automated office assistant. Another restriction is that the techniques used in the project do not cover the type of inference required for detailed analysis of causality and enablement for physical events (see e.g. Waltz [Waltz81]).

These restrictions would still permit a number of application tasks to be considered. One is gathering statistics from natural language reports. Example applications are gathering statistics from medical reports (see [Sager81]), weather reports and financial market reports. Another possible application task is checking for the mention of specific facts or events when analysing historical records. A related application is intelligent retrieval from collections of legal documents such as patent records (see e.g. [Lebowitz83]) or case reports. Another task associated with information retrieval is the automatic classification of documents by processing their abstracts. I believe the Capture techniques could be effectively applied to such types of task, and would improve on the use of more superficial analysis techniques (for instance those described in [King79] for historical records).

Given a well developed system based on the techniques used in this project, the alterations to the system necessary for tailoring it to one of the above applications are as follows. It would be necessary to extend the dictionary to enable the analyser to parse sentences in the new domain. The design of the

analyser as a general semantic parser [Boguraev83] means that, in principle, the changes to the dictionary would only involve the inclusion of new words and the addition of new word senses to existing entries. It would then be necessary to create memory entities for the new word senses and relate them using memory assertions to existing entities in memory.

The above alterations would still be task independent. Further memory entities and assertions might have to be included in order to represent further restrictions and specializations applicable in the context of the task. Additional entities and assertions would also need to be included to represent knowledge specific to the application task itself. Finally it would be necessary to implement a task-specific component that would be invoked after the creation of memory assertions, or (roughly in the manner used in this project) at specific points during the processing of clauses, by the language interpretation component. The realization of the task specific component would include the implementation of task specific operations that made use of memory retrieval. The complexity of the task component and its task specific memory operations would very much be a function of the complexity of the application task.

Chapter 7

Summary and Conclusions

The work reported in this thesis was concerned with the development of practical computational mechanisms for the representation and use of knowledge and context for automatic text processing. The overall design approach was that the basic mechanisms provided should be simple, well defined, and efficient, but at the same time flexible enough for dealing with a wide range of common language interpretation problems and types of factor contributing to context.

In this chapter a brief summary is first given of the techniques developed in this project. This is followed by some remarks about the performance of the techniques in the Capture system, and a more general comparative evaluation of the memory and context mechanisms involved. Finally, some of the more promising directions for future research along the lines taken in this project are indicated.

7.1 Techniques developed in this project

A representation formalism was developed for encoding a memory knowledge base for a text processing system. Memory contains entities and two basic types of assertions. Specialization assertions define a (tangled) hierarchy that gives a classification of the entities in memory, while correspondence assertions define a hierarchy that classifies the associations between them. Subtypes of these assertions can give more information about classifications and indicate the functionality implied by particular associations.

The representation of contextual information (other than the context provided by the current contents of memory) is done uniformly in terms of instances of various types of context factors, the types reflecting different kinds of information that contribute to context. Each factor indicates that the memory entities in its scope should be foregrounded, or regarded as having an increased relevance, by an amount specified by the current significance weight of the factor. Factors can be created by the operations that are evaluated for performing the processing of a text. The application of context for disambiguation and focusing makes use of the context activation values of memory entities, these being derived from the context information present by summing the significance weights of the relevant

context factors.

Language interpretation and application task components can exploit the various types of knowledge stored in memory and the relationships between these types that are also recorded in memory. The results of memory retrieval operations for this purpose can be chosen, if necessary, on the basis of the context activations of the relevant memory entities. The retrieval operations are set operations implemented by marker processing performed on the network structure derived from memory assertions. The scopes of context factors are also implemented as marked sets (which incidentally gives a convenient way of using traces of memory processing as a factor contributing to context).

The main technique developed for increasing the efficiency of the memory and context mechanisms was the automatic indexing of marked sets, and hence processing and context information. This interacts with the technique of recording certain marker propagations (used to decrease the effort involved in marking) because the marked sets corresponding to the recorded propagations need to be "maintained" when new memory assertions are created. The indexing scheme can be used for efficient access to memory entities that satisfy specified marking and/or activation threshold conditions. In particular, this includes intersection searches that are restricted to a "focus-space" determined by a specified context activation threshold. The mechanism allows such searches to be done even though the various context factors present can be managed independently by altering their significance weights.

7.2 Observations on the evaluation of the techniques by Capture

Overall, the use of the techniques in the implemented Capture system was a success in that a number of simple texts were correctly processed, demonstrating non-trivial language interpretation and application task processing. This involved the use of a memory knowledge base consisting of around 450 entities and over 750 memory assertions, and the use of an interesting range of context factor types.

Thirty or so different short example texts were successfully processed by the system (as well as a similar number of non-interesting variants on these). The example texts in Appendix A (which includes the final output generated by these examples) pretty much exhaust the kinds of possibilities that the present version of the Capture system can handle with respect to the two target databases that were used. It should be noted that exactly the same version of the Capture system (i.e. with the same initial contents of

memory, and initial significance weights for context factor types, etc.) was used to produce all the output in Appendix A.

Unfortunately, many of these artificial examples read somewhat awkwardly. This is partly due to the limitations on coverage at both the analysis and interpretation levels. Another reason is that devising texts in order to test particular constructions meant that the most natural choice of linguistic expression to maintain cohesiveness was often sacrificed. For example, in some cases where pronouns would have been appropriate other types of definite reference were used instead (so it is just as well that this distinction was not being used to generate context information, as it probably should in a more complete system). Given the overall aims of this project, the deficiencies of the example texts are not as serious as they would have been if, for example, the main objective was to formulate a precise theory for the interaction and relative importance of particular context factor types.

The implementation of the interpretation component concentrated on operations for solving common language interpretation problems. These operations seem to work reasonably robustly for the kind of simple descriptive texts that were processed by the system. In particular, memory is used during the interpretation of implicit relationships and various types of clauses, and the context mechanism is vital for the operations implemented for definite reference resolution, including plural references, and for word sense disambiguation (of the type that cannot be handled by the analyser).

The implementation of the database capture component illustrated the use of knowledge specific to an application task, including the relationships in memory between this knowledge and the language and domain related knowledge. The task required solutions to text interpretation problems such as reference resolution so explicit database creation statements could be generated, and also knowledge of the domain so that, for example, the classification of entities with respect to the target database could be performed.

The implemented system for database capture is, however, still a long way from being an ideal production text processing system. Building such a system awaits solutions to a number of presently poorly understood problems such as the processing of metaphors and robust parsing. Even a more modest system would require a large amount of work, using the techniques adopted in the implementation of Capture, for extending the coverage of the system as much as possible. A number of extensions of coverage were indicated in the chapters on interpretation and the database capture task.

7.3 General conclusions about the memory and context mechanisms

The memory and context mechanisms provided by the techniques developed in this project do have a number of identifiable advantages over previous ones proposed by workers in AI. The techniques were not all tested (or could not be tested) to the limit in the Capture system, but the mechanisms were used to implement a working system capable of performing non-trivial text processing; and on the basis of this experience some general claims about the memory and context mechanisms can be made.

The memory representation system has the advantage of being simpler than the semantic network formalisms on which it was based (the NETL and KL-ONE systems) while still retaining their most important features. This is important for ease of implementation and for the construction and maintenance of knowledge bases. The formalism still has a number of deficiencies in common with those on which it was based, such as the lack of facilities for a sophisticated treatment of temporal information. Thus although the representation has proved useful for solving some pervasive language interpretation problems it was not intended to be (and is not claimed as) a complete knowledge representation formalism adequate for all possible text processing needs. The other main advantage of the representation formalism, and one related to its simplicity, is that it can be given a well specified semantics. This allows a greater degree of confidence in using the representation, and gives a way of deciding the circumstances under which operations can be performed on it.

The context mechanism is flexible enough to handle a wide range of contextual phenomena. The mechanism provides a less rigid framework for context management and application than that provided by script-like approaches. The mechanism developed can be thought of as a generalization of and improvement on Grosz's focus mechanism. Thus two important advantages over Grosz's mechanism are that various different factors contributing to context can be combined when context is applied, and that context can be accumulated and altered gradually, leading to a gradual shift of focus during the processing of a text. Another advantage is that context factors can be managed independently, and, if necessary, by different components of the system.

The management of context factors by different components was not fully tested in the implemented system. This was because, in Capture, some factors (e.g. certain association factors) are created by the interpretation component where, looking back, the memory component seems like a better choice. The structure of the system also meant that contributions of surface analysis to context were handled indirectly by the interpretation component rather than directly by the analyser. It should be possible, however, to

design a system that overcomes these problems of Capture and so make full use of the advantages offered by the context mechanism.

The scheme for indexing marked sets is used profitably by both the memory and context mechanisms and is a new technique developed in this project. For memory processing it allows certain searches, including intersection searches, to be performed efficiently; while the effort involved in marking sets of entities is reduced by keeping records of certain marker propagations. For the context mechanism, the indexing scheme allows ready access to entities that are in focus where the degree of being in focus can be specified. The indexing scheme offers a computationally efficient way of doing this even though the various context factors can be managed independently by frequent changes to their significance weights.

Various different semantic criteria can easily be used to make the indexing more effective, and the experience gained from the Capture system indicates that the semantic clustering criteria that were tried are preferable to "random" clustering. However, it needs to be emphasised that the size of the memory knowledge base used by the implemented system was unfortunately too small for serious investigation of the effectiveness of different clustering criteria, or indeed the overall effectiveness of the use of the indexing scheme as compared with other possible algorithms. This would require observing the behaviour of the system with a very large knowledge base, most of which would be irrelevant to any particular short text. Nevertheless, since the efficiency of memory processing (let alone access to the focus set) has been considered important enough for hardware designs to be investigated, investigations of the possibilities offered by software methods are surely worthwhile.

Much of the work in designing the memory and context mechanisms was spent on trying to achieve a good "performance/complexity" ratio. In testing how far the simplicity of the mechanisms could be pushed, it was discovered that a number of more esoteric features of memory representation and the application of context did not appear to contribute to performance and could be abandoned. Apart from being interesting in itself, the fact that the final simpler mechanisms can handle a wide range of phenomena means that they are well suited for incorporation into experimental systems that are investigating other aspects of language use.

7.4 Directions for further research

There are a number of promising directions for research that can take advantage of and build on the memory and context mechanisms developed in

this project. Possibilities for extending the coverage of constructions and the range of factor types, and for using the Capture techniques for different system designs and application tasks, have already been indicated in previous chapters. The mechanisms also provide a framework for carrying out more linguistically motivated investigations than were attempted here. Thus there is the possibility of developing a successful theory for tricky problems in reference resolution, such as plural references and references to generics. The resolution of plural references is certainly a problem in which memory and context interact in a non-trivial way. Various possible algorithms could be devised that made use of memory and context in order to determine, for example, whether an existing set, a new set of entities, or a generic description is intended as the referent.

Another area for which the mechanisms provide a good framework is the interaction of different factors contributing to context, their relative importance, the way they degrade over time, and the extent to which this varies with different sorts of texts. The testing of various algorithms for reference and of theories about the interactions of context factors really needs to be done with respect to some corpus of collected texts. Care should be taken when editing such texts, for the purpose of overcoming difficulties not being investigated, that the amount of information lost should be kept to a minimum.

An issue concerning the structure of natural language processing systems that may be worth investigating is one raised by the following principle which plays a part in the way reference resolution and the way that context application are handled in the Capture system. The principle is that information relevant to interpretation (e.g. constraints on reference and context factors) should be accumulated and represented in a way that allows it to be all applied at once. The extent to which this principle can be realized affects robustness and control issues. Observing it may help overcome problems that current designers face in deciding when to apply certain bits of information, since there always seem to be counterexamples to theories that state that certain information should be applied first.

Systems for language generation, as opposed to language interpretation, that rely on the memory and context mechanisms described are worth investigation. The role of these mechanisms would be especially relevant to the choice of descriptions for objects and relationships. The specialization and correspondence hierarchies present a range of possible descriptions with varying degrees of specificity or generality, (cf. the work by D.D. McDonald [McDonald81]). The context mechanism would help in choosing the appropriate degree of specificity, and in determining whether to use modifiers, implicit or explicit relationships, pronouns or other definite noun phrases. For this purpose the context activations of the entities being

described could be used both absolutely by comparison with predetermined thresholds, and relatively by comparison with other entities in memory that candidate descriptions might refer to.

The design of the Capture system could serve as the basis for practical semi-automated systems for database creation from collections of texts. Such a system would depend on the intervention of a human operator for resolving difficulties that it could not handle, and in this way would be similar to some proposed designs (see e.g. [Kay80]) for translation aids. The approach of finding simple solutions to common language interpretation problems taken in the design of Capture should make it a good starting point for building such a system. The quantifiable nature of context activation may also turn out to be an advantage. Decisions made by the system on the basis of context activation could be regarded as safe or not with respect to specified safety margins. For example, if the difference between the context activations of the best candidates for resolving a reference was less than such a margin then the human operator could be asked to confirm the decision taken by the system.

Appendix A

Example texts and output

This appendix contains the final output generated by the implemented system for a number of example texts for the Machines Database and for the Artifacts Database. The order in which these examples are presented roughly reflects the order in which they were used for testing as the system developed, so that the earlier examples in the appendix are simpler than the later ones which exhibit a wider range of interpretation problems. However, all the output given in this appendix was produced by exactly the same version of the system (using the same initial memory knowledge base) in September 1983. Typical times taken for the processing of each sentence in these texts are around 0.4 seconds for the analysis phase and a further 0.8 seconds for interleaved interpretation and task-specific processing; this includes Lisp garbage collection with the system running in about 1M bytes of store on an IBM 3081.

TEXT A01

JONES SUPPLIES P1234.
THE WEIGHT OF THE MACHINE IS 25.
THE STATUS OF THE SUPPLIER IS 10.
THE WEIGHT OF P6666 IS 25.
IT IS BLUE.

```
((SUPPLIES/RELATION ((SMC/MCNUM P1234) (SMC/SID Jones)))
  (MACHINES/RELATION ((MC/MCNUM P1234) (MC/WEIGHT 25)))
  (SUPPLIERS/RELATION ((S/SID Jones) (S/STATUS 10)))
  (MACHINES/RELATION ((MC/MCNUM P6666) (MC/WEIGHT 25)))
  (MACHINES/RELATION ((MC/MCNUM P6666) (MC/COLOR blue1))))
```

TEXT A02

P888 IS GREEN.
THE WEIGHT OF THE MACHINE IS 40.
SMITH SUPPLIES IT.
JONES SUPPLIES IT.
HE SUPPLIES P789.
THE STATUS OF THE SUPPLIER IS 15.
THE STATUS OF SMITH IS 20.
HE SUPPLIES P999.

```
((MACHINES/RELATION ((MC/MCNUM P888) (MC/COLOR green1)))
  (MACHINES/RELATION ((MC/MCNUM P888) (MC/WEIGHT 40)))
  (SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P888)))
  (SUPPLIES/RELATION ((SMC/SID Jones) (SMC/MCNUM P888)))
  (SUPPLIES/RELATION ((SMC/MCNUM P789) (SMC/SID Jones)))
  (SUPPLIERS/RELATION ((S/SID Jones) (S/STATUS 15)))
  (SUPPLIERS/RELATION ((S/SID Smith1) (S/STATUS 20)))
  (SUPPLIES/RELATION ((SMC/MCNUM P999) (SMC/SID Smith1))))
```

TEXT A03

THE WEIGHT OF P4444 IS 24.
SMITH SUPPLIES IT.
THE STATUS OF THE SUPPLIER IS 10.
THE STATUS OF JONES IS 20.
HE SUPPLIES P9999.
THE MACHINE IS RED.
CLARK SUPPLIES IT.
ROBINSON SUPPLIES THE MACHINE WHICH IS SUPPLIED BY SMITH.
THE WEIGHT OF THE MACHINE THAT JONES SUPPLIES IS 29.

```
((MACHINES/RELATION ((MC/MCNUM P4444) (MC/WEIGHT 24)))
  (SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P4444)))
  (SUPPLIERS/RELATION ((S/SID Smith1) (S/STATUS 10)))
  (SUPPLIERS/RELATION ((S/SID Jones) (S/STATUS 20)))
  (SUPPLIES/RELATION ((SMC/MCNUM P9999) (SMC/SID Jones)))
  (MACHINES/RELATION ((MC/MCNUM P9999) (MC/COLOR red1)))
  (SUPPLIES/RELATION ((SMC/SID Clark1) (SMC/MCNUM P9999)))
  (SUPPLIES/RELATION ((SMC/SID Robinson1) (SMC/MCNUM P4444)))
  (MACHINES/RELATION ((MC/MCNUM P9999) (MC/WEIGHT 29))))
```

TEXT A04

SMITH SUPPLIES P9999.
JONES SUPPLIES P4444.
THEY SUPPLY P7777.
THE MACHINES ARE RED.

```
((SUPPLIES/RELATION ((SMC/MCNUM P9999) (SMC/SID Smith1)))
  (SUPPLIES/RELATION ((SMC/MCNUM P4444) (SMC/SID Jones)))
  (SUPPLIES/RELATION ((SMC/MCNUM P7777) (SMC/SID Smith1)))
  (SUPPLIES/RELATION ((SMC/MCNUM P7777) (SMC/SID Jones)))
  (MACHINES/RELATION ((MC/COLOR red1) (MC/MCNUM P9999)))
  (MACHINES/RELATION ((MC/COLOR red1) (MC/MCNUM P4444)))
  (MACHINES/RELATION ((MC/COLOR red1) (MC/MCNUM P7777))))
```

TEXT A05

PHILIPS MANUFACTURES P4444.
IT IS RED.
THE COLOUR OF P4040 IS BLUE.
THE MACHINES ARE SUPPLIED BY SMITH.
JONES SUPPLIES P9999.
IT IS MADE BY MARCONI.
THE STATUS OF THE SUPPLIER IS 90.
THIS MACHINE IS GREEN.
CLARK SUPPLIES THE MACHINE WHICH IS MADE BY PHILIPS.
THE WEIGHT OF THE GREEN MACHINE IS 29.

```
((MANUFACTURES/RELATION ((M/MCNUM P4444) (M/MNAME Philips)))
  (MACHINES/RELATION ((MC/MCNUM P4444) (MC/COLOR red1)))
  (MACHINES/RELATION ((MC/MCNUM P4040) (MC/COLOR blue1)))
  (SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P4444)))
  (SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P4040)))
  (SUPPLIES/RELATION ((SMC/MCNUM P9999) (SMC/SID Jones)))
  (MANUFACTURES/RELATION ((M/MNAME Marconi) (M/MCNUM P9999)))
  (SUPPLIERS/RELATION ((S/SID Jones) (S/STATUS 90)))
  (MACHINES/RELATION ((MC/MCNUM P9999) (MC/COLOR green1)))
```

```
  (SUPPLIES/RELATION ((SMC/SID Clark1) (SMC/MCNUM P4444)))
  (MACHINES/RELATION ((MC/MCNUM P9999) (MC/WEIGHT 29))))
```

TEXT A06

```
MARCONI MANUFACTURES P9999.
PHILIPS MAKES P4444.
THE STATUS OF SMITH IS 50.
THE STATUS OF CLARK IS 80.
THEY SUPPLY THE MACHINES.
```

```
((MANUFACTURES/RELATION ((M/MCNUM P9999) (M/MNAME Marconi)))
  (MANUFACTURES/RELATION ((M/MCNUM P4444) (M/MNAME Philips)))
  (SUPPLIERS/RELATION ((S/SID Smith1) (S/STATUS 50)))
  (SUPPLIERS/RELATION ((S/SID Clark1) (S/STATUS 80)))
  (SUPPLIES/RELATION ((SMC/MCNUM P9999) (SMC/SID Smith1)))
  (SUPPLIES/RELATION ((SMC/MCNUM P9999) (SMC/SID Clark1)))
  (SUPPLIES/RELATION ((SMC/MCNUM P4444) (SMC/SID Smith1)))
  (SUPPLIES/RELATION ((SMC/MCNUM P4444) (SMC/SID Clark1))))
```

TEXT A07

```
P9999 IS GREEN.
P7777 IS RED.
JONES SUPPLIES THE MACHINES.
P8888 IS RED.
MARCONI MANUFACTURES THE MACHINES THAT ARE SUPPLIED BY JONES.
PHILIPS MANUFACTURES THE RED MACHINES.
```

```
((MACHINES/RELATION ((MC/MCNUM P9999) (MC/COLOR green1)))
  (MACHINES/RELATION ((MC/MCNUM P7777) (MC/COLOR red1)))
  (SUPPLIES/RELATION ((SMC/SID Jones) (SMC/MCNUM P9999)))
  (SUPPLIES/RELATION ((SMC/SID Jones) (SMC/MCNUM P7777)))
  (MACHINES/RELATION ((MC/MCNUM P8888) (MC/COLOR red1)))
  (MANUFACTURES/RELATION ((M/MNAME Marconi) (M/MCNUM P9999)))
  (MANUFACTURES/RELATION ((M/MNAME Marconi) (M/MCNUM P7777)))
  (MANUFACTURES/RELATION ((M/MNAME Philips) (M/MCNUM P7777)))
  (MANUFACTURES/RELATION ((M/MNAME Philips) (M/MCNUM P8888))))
```

TEXT A08

MARCONI MANUFACTURES P9999.
IT IS SUPPLIED BY SMITH.
THE STATUS OF THE SUPPLIER IS 30.
THE STATUS OF JONES IS 40.
THEY SUPPLY P9900.
THIS MACHINE IS A COMPUTER.
IT IS MADE BY PHILIPS.
THE MACHINES ARE RED.
THE COLOUR OF P4444 IS GREEN.
THE COMPUTER MANUFACTURER MAKES THE GREEN MACHINE.
P9000 IS BLUE.
IT IS A DISC-DRIVE.
IT IS SUPPLIED BY THE STATUS 30 SUPPLIER.
HE GAVE P8888 TO THE P9999 MANUFACTURER.
THE DISC-DRIVE IS MADE BY THIS MANUFACTURER.
THE WEIGHT OF THE MACHINES THAT ARE MADE BY MARCONI IS 35.

```
((MANUFACTURES/RELATION ((M/MCNUM P9999) (M/MNAME Marconi)))
  (SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P9999)))
  (SUPPLIERS/RELATION ((S/SID Smith1) (S/STATUS 30)))
  (SUPPLIERS/RELATION ((S/SID Jones) (S/STATUS 40)))
  (SUPPLIES/RELATION ((SMC/MCNUM P9900) (SMC/SID Smith1)))
  (SUPPLIES/RELATION ((SMC/MCNUM P9900) (SMC/SID Jones)))
  (MACHINES/RELATION ((MC/TYPE computer) (MC/MCNUM P9900)))
  (MANUFACTURES/RELATION ((M/MNAME Philips) (M/MCNUM P9900)))
  (MACHINES/RELATION ((MC/COLOR red1) (MC/MCNUM P9999)))
  (MACHINES/RELATION ((MC/COLOR red1) (MC/MCNUM P9900)))
  (MACHINES/RELATION ((MC/MCNUM P4444) (MC/COLOR green1)))
  (MANUFACTURES/RELATION ((M/MCNUM P4444) (M/MNAME Philips)))
  (MACHINES/RELATION ((MC/MCNUM P9000) (MC/COLOR blue1)))
  (MACHINES/RELATION ((MC/TYPE disc-drive1) (MC/MCNUM P9000)))
  (SUPPLIES/RELATION ((SMC/MCNUM P9000) (SMC/SID Smith1)))
  (MANUFACTURES/RELATION ((M/MCNUM P9000) (M/MNAME Marconi)))
  (MACHINES/RELATION ((MC/WEIGHT 35) (MC/MCNUM P9999)))
  (MACHINES/RELATION ((MC/WEIGHT 35) (MC/MCNUM P9000))))
```

TEXT A09

MARCONI MANUFACTURES P9999.
IT IS SUPPLIED BY SMITH WHO SUPPLIES P7777 WHICH IS RED.
THE RED MACHINE IS MADE BY PHILIPS.
THE ONE THAT IS MADE BY MARCONI IS BLUE.

```
((MANUFACTURES/RELATION ((M/MCNUM P9999) (M/MNAME Marconi)))
  (MACHINES/RELATION ((MC/MCNUM P7777) (MC/COLOR red1)))
  (SUPPLIES/RELATION ((SMC/MCNUM P7777) (SMC/SID Smith1)))
  (SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P9999)))
```

```
(MANUFACTURES/RELATION ((M/MNAME Philips) (M/MCNUM P7777)))
(MACHINES/RELATION ((MC/MCNUM P9999) (MC/COLOR blue1))))
```

TEXT A10

```
P7777 IS A COMPUTER THAT IS MANUFACTURED BY PHILIPS.
THE MANUFACTURER MAKES P7700 WHICH IS SUPPLIED BY SMITH.
JONES AND CLARK SUPPLY THE COMPUTER.
THE STATUS OF THE COMPUTER SUPPLIERS IS 20.
```

```
((MANUFACTURES/RELATION ((M/MNAME Philips) (M/MCNUM P7777)))
  (MACHINES/RELATION ((MC/TYPE computer) (MC/MCNUM P7777)))
  (SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P7700)))
  (MANUFACTURES/RELATION ((M/MCNUM P7700) (M/MNAME Philips)))
  (SUPPLIES/RELATION ((SMC/MCNUM P7777) (SMC/SID Clark1)))
  (SUPPLIES/RELATION ((SMC/MCNUM P7777) (SMC/SID Jones)))
  (SUPPLIERS/RELATION ((S/STATUS 20) (S/SID Clark1)))
  (SUPPLIERS/RELATION ((S/STATUS 20) (S/SID Jones))))
```

TEXT A11

```
MARCONI MANUFACTURES P9999 WHICH IS A DISC-DRIVE.
IT IS SUPPLIED BY SMITH.
THE STATUS OF THE SUPPLIER IS 30.
THE STATUS OF JONES IS 40.
THEY SUPPLY P7777 WHICH IS MANUFACTURED BY PHILIPS.
IT IS A TERMINAL.
THESE MACHINES ARE RED.

P9000 IS MANUFACTURED BY MARCONI.
IT IS A BLUE PRINTER.
P4444 IS GREEN.
THE BLUE MACHINE IS SUPPLIED BY THE STATUS 30 SUPPLIER.
THE TERMINAL MANUFACTURER MAKES THE GREEN MACHINE.
THE WEIGHT OF THE MACHINES THAT ARE MADE BY MARCONI IS 35.

SMITH SUPPLIES P1010 WHICH IS A COMPUTER.
THIS ONE IS MANUFACTURED BY IBM.
THE SUPPLIER SUPPLIES P6200 WHICH IS MADE BY PHILIPS.
THE WEIGHT OF THE COMPUTER IS 75.
```

```
((MACHINES/RELATION ((MC/TYPE disc-drive1) (MC/MCNUM P9999)))
  (MANUFACTURES/RELATION ((M/MNAME Marconi) (M/MCNUM P9999)))
  (SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P9999)))
  (SUPPLIERS/RELATION ((S/SID Smith1) (S/STATUS 30)))
  (SUPPLIERS/RELATION ((S/SID Jones) (S/STATUS 40)))
  (MANUFACTURES/RELATION ((M/MNAME Philips) (M/MCNUM P7777)))
```

```
(SUPPLIES/RELATION ((SMC/MCNUM P7777) (SMC/SID Smith1)))
(SUPPLIES/RELATION ((SMC/MCNUM P7777) (SMC/SID Jones)))
(MACHINES/RELATION ((MC/TYPE terminal1) (MC/MCNUM P7777)))
(MACHINES/RELATION ((MC/COLOR red1) (MC/MCNUM P9999)))
(MACHINES/RELATION ((MC/COLOR red1) (MC/MCNUM P7777)))
(MANUFACTURES/RELATION ((M/MCNUM P9000) (M/MNAME Marconi)))
(MACHINES/RELATION ((MC/MCNUM P9000) (MC/COLOR blue1)))
(MACHINES/RELATION ((MC/TYPE printer2) (MC/MCNUM P9000)))
(MACHINES/RELATION ((MC/MCNUM P4444) (MC/COLOR green1)))
(SUPPLIES/RELATION ((SMC/MCNUM P9000) (SMC/SID Smith1)))
(MANUFACTURES/RELATION ((M/MCNUM P4444) (M/MNAME Philips)))
(MACHINES/RELATION ((MC/WEIGHT 35) (MC/MCNUM P9999)))
(MACHINES/RELATION ((MC/WEIGHT 35) (MC/MCNUM P9000)))
(MACHINES/RELATION ((MC/TYPE computer) (MC/MCNUM P1010)))
(SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P1010)))
(MANUFACTURES/RELATION ((M/MNAME IBM) (M/MCNUM P1010)))
(MANUFACTURES/RELATION ((M/MCNUM P6200) (M/MNAME Philips)))
(SUPPLIES/RELATION ((SMC/MCNUM P6200) (SMC/SID Smith1)))
(MACHINES/RELATION ((MC/MCNUM P1010) (MC/WEIGHT 75))))
```

TEXT A12

P7777 IS A COMPUTER.
IT IS MANUFACTURED BY PHILIPS.
P9920 IS A DISC-DRIVE THAT IS MANUFACTURED BY MARCONI.
THIS MANUFACTURER MAKES P9999 WHICH IS SUPPLIED BY JONES.
SMITH SUPPLIES THE COMPUTER.
THE STATUS OF P7777'S SUPPLIER IS 20.

```
((MACHINES/RELATION ((MC/TYPE computer) (MC/MCNUM P7777)))
  (MANUFACTURES/RELATION ((M/MNAME Philips) (M/MCNUM P7777)))
  (MANUFACTURES/RELATION ((M/MNAME Marconi) (M/MCNUM P9920)))
  (MACHINES/RELATION ((MC/TYPE disc-drive1) (MC/MCNUM P9920)))
  (SUPPLIES/RELATION ((SMC/SID Jones) (SMC/MCNUM P9999)))
  (MANUFACTURES/RELATION ((M/MCNUM P9999) (M/MNAME Marconi)))
  (SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P7777)))
  (SUPPLIERS/RELATION ((S/SID Smith1) (S/STATUS 20))))
```

TEXT A13

P9999 IS A MAIN-FRAME THAT IS SUPPLIED BY SMITH.
HE SUPPLIES P7777 WHICH IS A TERMINAL.
JONES SUPPLIES P7700.
IT IS A DISC-DRIVE.
MARCONI MANUFACTURES THE COMPUTER IN PARIS.
THE PERIPHERALS ARE MADE BY PHILIPS.

```
((SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P9999)))
  (MACHINES/RELATION ((MC/TYPE computer) (MC/MCNUM P9999)))
  (MACHINES/RELATION ((MC/TYPE terminal1) (MC/MCNUM P7777)))
  (SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P7777)))
  (SUPPLIES/RELATION ((SMC/MCNUM P7700) (SMC/SID Jones)))
  (MACHINES/RELATION ((MC/TYPE disc-drive1) (MC/MCNUM P7700)))
  (MANUFACTURES/RELATION
    ((M/MNAME Marconi) (M/MCNUM P9999) (M/CITY Paris)))
  (MANUFACTURES/RELATION ((M/MNAME Philips) (M/MCNUM P7777)))
  (MANUFACTURES/RELATION ((M/MNAME Philips) (M/MCNUM P7700))))
```

TEXT A14

PLEXIR MANUFACTURES P9999 WHICH IS A COMPUTER.
IT IS SUPPLIED BY SMITH.
P1010 IS A TERMINAL THAT IS SUPPLIED BY CLARK.
THIS ONE IS MADE BY MIKOTA.
THESE MACHINES ARE RED.

P9000 IS A GREEN PRINTER.
IT IS MADE BY PLEXIR.
P4444 IS A BLUE COMPUTER.
THE COST OF THE MACHINE IS 7850.
THE PERIPHERAL IS SUPPLIED BY THE P9999 SUPPLIER.
THE TERMINAL MANUFACTURER MAKES THE BLUE MACHINE.
THE COST OF MIKOTA'S PERIPHERAL IS 235.

```
((MACHINES/RELATION ((MC/TYPE computer) (MC/MCNUM P9999)))
  (MANUFACTURES/RELATION ((M/MNAME Plexir1) (M/MCNUM P9999)))
  (SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P9999)))
  (SUPPLIES/RELATION ((SMC/SID Clark1) (SMC/MCNUM P1010)))
  (MACHINES/RELATION ((MC/TYPE terminal1) (MC/MCNUM P1010)))
  (MANUFACTURES/RELATION ((M/MNAME Mikota1) (M/MCNUM P1010)))
  (MACHINES/RELATION ((MC/COLOR red1) (MC/MCNUM P9999)))
  (MACHINES/RELATION ((MC/COLOR red1) (MC/MCNUM P1010)))
  (MACHINES/RELATION ((MC/MCNUM P9000) (MC/COLOR green1)))
  (MACHINES/RELATION ((MC/TYPE printer2) (MC/MCNUM P9000)))
  (MANUFACTURES/RELATION ((M/MNAME Plexir1) (M/MCNUM P9000)))
  (MACHINES/RELATION ((MC/MCNUM P4444) (MC/COLOR blue1)))
  (MACHINES/RELATION ((MC/TYPE computer) (MC/MCNUM P4444)))
  (MACHINES/RELATION ((MC/MCNUM P4444) (MC/COST 7850)))
  (SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P9000)))
  (MANUFACTURES/RELATION ((M/MNAME Mikota1) (M/MCNUM P4444)))
  (MACHINES/RELATION ((MC/MCNUM P1010) (MC/COST 235))))
```

TEXT A15

P8880 IS A COMPUTER THAT IS MANUFACTURED BY MIKOTA.
THE COST OF THE MACHINE IS 2595.

P7770 IS MANUFACTURED BY PLEXIR.
MARCONI MAKES P7200.
THE COST OF BOTH MACHINES IS 4000.

```
((MANUFACTURES/RELATION ((M/MNAME Mikota1) (M/MCNUM P8880)))
  (MACHINES/RELATION ((MC/TYPE computer) (MC/MCNUM P8880)))
  (MACHINES/RELATION ((MC/MCNUM P8880) (MC/COST 2595)))
  (MANUFACTURES/RELATION ((M/MNAME Plexir1) (M/MCNUM P7770)))
  (MANUFACTURES/RELATION ((M/MCNUM P7200) (M/MNAME Marconi)))
  (MACHINES/RELATION ((MC/COST 4000) (MC/MCNUM P7200)))
  (MACHINES/RELATION ((MC/COST 4000) (MC/MCNUM P7770))))
```

TEXT A16

P8080 IS SUPPLIED BY PETERS.
THE STATUS OF THE SUPPLIER IS 20.

CLARK SUPPLIES P7780 AND P7790.
P7720 IS SUPPLIED BY ROBINSON.
THESE THREE MACHINES ARE MANUFACTURED BY PLEXIR.

```
((SUPPLIES/RELATION ((SMC/SID Peters1) (SMC/MCNUM P8080)))
  (SUPPLIERS/RELATION ((S/SID Peters1) (S/STATUS 20)))
  (SUPPLIES/RELATION ((SMC/SID Clark1) (SMC/MCNUM P7790)))
  (SUPPLIES/RELATION ((SMC/SID Clark1) (SMC/MCNUM P7780)))
  (SUPPLIES/RELATION ((SMC/SID Robinson1) (SMC/MCNUM P7720)))
  (MANUFACTURES/RELATION ((M/MNAME Plexir1) (M/MCNUM P7780)))
  (MANUFACTURES/RELATION ((M/MNAME Plexir1) (M/MCNUM P7790)))
  (MANUFACTURES/RELATION ((M/MNAME Plexir1) (M/MCNUM P7720))))
```

TEXT A17

P9999 IS A DISC-DRIVE THAT IS SUPPLIED BY SMITH.
THIS PERIPHERAL IS MANUFACTURED BY MIKOTA.
HE SUPPLIES P7777 WHICH IS A TERMINAL.
IT IS MANUFACTURED IN LONDON BY PLEXIR.
CLARK SUPPLIES P9000 WHICH IS MANUFACTURED BY MARCONI IN PARIS.

```
((SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P9999)))
  (MACHINES/RELATION ((MC/TYPE disc-drive1) (MC/MCNUM P9999)))
```

```
(MANUFACTURES/RELATION ((M/MNAME Mikota1) (M/MCNUM P9999)))
(MACHINES/RELATION ((MC/TYPE terminal1) (MC/MCNUM P7777)))
(SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P7777)))
(MANUFACTURES/RELATION
  ((M/MNAME Plexir1) (M/MCNUM P7777) (M/CITY London)))
(MANUFACTURES/RELATION
  ((M/MNAME Marconi) (M/MCNUM P9000) (M/CITY Paris)))
(SUPPLIES/RELATION ((SMC/MCNUM P9000) (SMC/SID Clark1))))
```

TEXT A18

WINTRON MANUFACTURES P5050 WHICH IS A DISC-DRIVE.
P1010 IS A COMPUTER WHICH IS MADE BY THIS MANUFACTURER.
IT HAS A BOLT.
P8770 IS A PRINTER THAT IS MADE BY PLEXIR.
BOTH PERIPHERALS ARE SUPPLIED BY CLARK.
SMITH SUPPLIES THE MACHINE WITH THE BOLT.

P4740 IS MANUFACTURED BY P5050'S MANUFACTURER IN LONDON.
IT IS A MICRO-COMPUTER THAT IS SUPPLIED BY JONES.
HE SUPPLIES P8800 WHICH IS A TERMINAL.
THE COST OF THE COMPUTER IS 25.
THE COMPUTERS ARE RED.
THE THREE PERIPHERALS ARE GREEN.

```
((MACHINES/RELATION ((MC/TYPE disc-drive1) (MC/MCNUM P5050)))
  (MANUFACTURES/RELATION ((M/MNAME Wintron1) (M/MCNUM P5050)))
  (MANUFACTURES/RELATION ((M/MNAME Wintron1) (M/MCNUM P1010)))
  (MACHINES/RELATION ((MC/TYPE computer) (MC/MCNUM P1010)))
  (MANUFACTURES/RELATION ((M/MNAME Plexir1) (M/MCNUM P8770)))
  (MACHINES/RELATION ((MC/TYPE printer2) (MC/MCNUM P8770)))
  (SUPPLIES/RELATION ((SMC/SID Clark1) (SMC/MCNUM P5050)))
  (SUPPLIES/RELATION ((SMC/SID Clark1) (SMC/MCNUM P8770)))
  (SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P1010)))
  (MANUFACTURES/RELATION
    ((M/MCNUM P4740) (M/MNAME Wintron1) (M/CITY London)))
  (SUPPLIES/RELATION ((SMC/SID Jones) (SMC/MCNUM P4740)))
  (MACHINES/RELATION ((MC/TYPE computer) (MC/MCNUM P4740)))
  (MACHINES/RELATION ((MC/TYPE terminal1) (MC/MCNUM P8800)))
  (SUPPLIES/RELATION ((SMC/SID Jones) (SMC/MCNUM P8800)))
  (MACHINES/RELATION ((MC/MCNUM P4740) (MC/COST 25)))
  (MACHINES/RELATION ((MC/COLOR red1) (MC/MCNUM P1010)))
  (MACHINES/RELATION ((MC/COLOR red1) (MC/MCNUM P4740)))
  (MACHINES/RELATION ((MC/COLOR green1) (MC/MCNUM P5050)))
  (MACHINES/RELATION ((MC/COLOR green1) (MC/MCNUM P8770)))
  (MACHINES/RELATION ((MC/COLOR green1) (MC/MCNUM P8800))))
```

TEXT A19

PLEXIR MANUFACTURES P7777 WHICH IS COMPUTER.
IT HAS A SCREW.
WINTRON MAKES P9999 WHICH HAS A BOLT.
THE COST OF THE MACHINE WITH THE SCREW IS 9000.
THE MACHINE THAT HAS THE BOLT IS SUPPLIED BY CLARK.

```
((MACHINES/RELATION ((MC/TYPE computer) (MC/MCNUM P7777)))
  (MANUFACTURES/RELATION ((M/MNAME Plexir1) (M/MCNUM P7777)))
  (MANUFACTURES/RELATION ((M/MNAME Wintron1) (M/MCNUM P9999)))
  (MACHINES/RELATION ((MC/MCNUM P7777) (MC/COST 9000)))
  (SUPPLIES/RELATION ((SMC/SID Clark1) (SMC/MCNUM P9999))))
```

TEXT A20

PLEXIR MANUFACTURES P9000.
IT IS A MICRO-COMPUTER.
WINTRON MANUFACTURES P7000 WHICH IS A DISC-DRIVE.
P9000 IS SUPPLIED BY SMITH.

P8000 IS A COMPUTER.
IT IS SUPPLIED BY JONES.
THE STATUS OF THIS SUPPLIER IS 10.
THE STATUS OF P9000'S SUPPLIER IS 20.
THE MICRO-COMPUTER IS RED.
THE MANUFACTURER MANUFACTURES P9090.

```
((MANUFACTURES/RELATION ((M/MCNUM P9000) (M/MNAME Plexir1)))
  (MACHINES/RELATION ((MC/TYPE computer) (MC/MCNUM P9000)))
  (MACHINES/RELATION ((MC/TYPE disc-drive1) (MC/MCNUM P7000)))
  (MANUFACTURES/RELATION ((M/MNAME Wintron1) (M/MCNUM P7000)))
  (SUPPLIES/RELATION ((SMC/SID Smith1) (SMC/MCNUM P9000)))
  (MACHINES/RELATION ((MC/TYPE computer) (MC/MCNUM P8000)))
  (SUPPLIES/RELATION ((SMC/SID Jones) (SMC/MCNUM P8000)))
  (SUPPLIERS/RELATION ((S/SID Jones) (S/STATUS 10)))
  (SUPPLIERS/RELATION ((S/SID Smith1) (S/STATUS 20)))
  (MACHINES/RELATION ((MC/MCNUM P9000) (MC/COLOR red1)))
  (MANUFACTURES/RELATION ((M/MCNUM P9090) (M/MNAME Plexir1))))
```

TEXT A21

HADDON COLLECTED P33 WHICH IS AN ARMLET.
HE COLLECTED P37 FROM WOODLARK.
IT IS A NECKLACE.

BEVAN DONATED P571 AND P352.
P571 IS A SKIRT.
P352 IS A NECKLACE.
BEVAN COLLECTED BOTH ARTIFACTS AT MOUNT-HAGEN.

P576 IS FRAGILE.
HADDON DONATED IT TO LAMBERTS.
P220 IS A HARD BOX.

```
((ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P33)))
  (ORIGIN/RELATION ((ORIG/COLL Haddon1) (ORIG/ARTN P33)))
  (ORIGIN/RELATION
    ((ORIG/ARTN P37) (ORIG/COLL Haddon1) (ORIG/PLAC Woodlark)))
  (ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P37)))
  (ARTIFACTS/RELATION ((ARTF/TYPE clothing1) (ARTF/NUMB P571)))
  (ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P352)))
  (ORIGIN/RELATION
    ((ORIG/PLAC Mount-Hagen) (ORIG/COLL Bevan1) (ORIG/ARTN P571)))
  (ORIGIN/RELATION
    ((ORIG/PLAC Mount-Hagen) (ORIG/COLL Bevan1) (ORIG/ARTN P352)))
  (ARTIFACTS/RELATION ((ARTF/NUMB P576) (ARTF/COND fragile1)))
  (ARTIFACTS/RELATION ((ARTF/NUMB P220) (ARTF/COND hard1)))
  (ARTIFACTS/RELATION ((ARTF/TYPE container1) (ARTF/NUMB P220))))
```

TEXT A22

P4830 IS A CANOE-PROW.
IT WAS DONATED BY HADDON.
HE COLLECTED IT IN DAUI.
BEVAN COLLECTED P981 WHICH IS A CANOE MODEL FROM WOODLARK.
BOTH ARTIFACTS ARE FRAGILE.

```
((ARTIFACTS/RELATION
    ((ARTF/TYPE navigation/artifact) (ARTF/NUMB P4830)))
  (ORIGIN/RELATION
    ((ORIG/ARTN P4830) (ORIG/PLAC Daui) (ORIG/COLL Haddon1)))
  (ARTIFACTS/RELATION ((ARTF/TYPE model1) (ARTF/NUMB P981)))
  (ORIGIN/RELATION
    ((ORIG/COLL Bevan1) (ORIG/ARTN P981) (ORIG/PLAC Woodlark)))
  (ARTIFACTS/RELATION ((ARTF/COND fragile1) (ARTF/NUMB P981)))
  (ARTIFACTS/RELATION ((ARTF/COND fragile1) (ARTF/NUMB P4830))))
```

TEXT A23

P562 IS A POT THAT WAS DONATED BY HADDON.
THE CONDITION OF THE POT IS GOOD.
HE COLLECTED IT FROM WOODLARK.
P371 WAS COLLECTED BY BEVAN FROM THERE.
THIS ARTIFACT IS AN ARMLET.

```
((ARTIFACTS/RELATION ((ARTF/TYPE container1) (ARTF/NUMB P562)))
  (ARTIFACTS/RELATION ((ARTF/NUMB P562) (ARTF/COND good1)))
  (ORIGIN/RELATION
    ((ORIG/ARTN P562) (ORIG/PLAC Woodlark) (ORIG/COLL Haddon1)))
  (ORIGIN/RELATION
    ((ORIG/COLL Bevan1) (ORIG/ARTN P371) (ORIG/PLAC Woodlark)))
  (ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P371))))
```

TEXT A24

P20 IS A PADDLE THAT WAS COLLECTED FROM WOODLARK.
P70 WAS COLLECTED IN NEW-GUINEA.
THE WOODLARK ARTIFACT WAS COLLECTED BY ARMSTRONG.
HADDON COLLECTED P70.
THE CONDITION OF THESE ARTIFACTS IS GOOD.
ARMSTRONG'S PADDLE IS SMALL.

```
((ORIGIN/RELATION ((ORIG/ARTN P20) (ORIG/PLAC Woodlark)))
  (ARTIFACTS/RELATION
    ((ARTF/TYPE navigation/artifact) (ARTF/NUMB P20)))
  (ORIGIN/RELATION ((ORIG/ARTN P70) (ORIG/PLAC New-Guinea)))
  (ORIGIN/RELATION ((ORIG/COLL Armstrong1) (ORIG/ARTN P20)))
  (ORIGIN/RELATION ((ORIG/COLL Haddon1) (ORIG/ARTN P70)))
  (ARTIFACTS/RELATION ((ARTF/COND good1) (ARTF/NUMB P20)))
  (ARTIFACTS/RELATION ((ARTF/COND good1) (ARTF/NUMB P70))))
```

TEXT A25

HADDON COLLECTED P590 WHICH IS A POT.
HE COLLECTED P520.
IT IS A CANOE WITH A PADDLE.
BOTH ARTIFACTS COME FROM NEW-GUINEA.
BEVAN COLLECTED P422.
THIS ARTIFACT IS A LARGE JUG WITH A LID.
THE CONDITION OF THE HADDON CONTAINER IS GOOD.
THE ONE THAT WAS COLLECTED BY BEVAN IS FRAGILE.

```
((ARTIFACTS/RELATION ((ARTF/TYPE container1) (ARTF/NUMB P590)))
  (ORIGIN/RELATION ((ORIG/COLL Haddon1) (ORIG/ARTN P590)))
  (ORIGIN/RELATION ((ORIG/ARTN P520) (ORIG/COLL Haddon1)))
  (ARTIFACTS/RELATION
    ((ARTF/TYPE navigation/artifact) (ARTF/NUMB P520)))
  (ORIGIN/RELATION ((ORIG/PLAC New-Guinea) (ORIG/ARTN P590)))
  (ORIGIN/RELATION ((ORIG/PLAC New-Guinea) (ORIG/ARTN P520)))
  (ORIGIN/RELATION ((ORIG/ARTN P422) (ORIG/COLL Bevan1)))
  (ARTIFACTS/RELATION ((ARTF/TYPE container1) (ARTF/NUMB P422)))
  (ARTIFACTS/RELATION ((ARTF/NUMB P590) (ARTF/COND good1)))
  (ARTIFACTS/RELATION ((ARTF/NUMB P422) (ARTF/COND fragile1))))
```

TEXT A26

ARMSTRONG COLLECTED P56.
IT IS A BOX.
THIS CONTAINER IS SMALL.
HADDON COLLECTED P62 AND P63.
THESE ARTIFACTS ARE JUGS.
THE CONDITION OF THE JUGS IS POOR.
ARMSTRONG'S ARTIFACT COMES FROM MOUNT-HAGEN.
THE JUGS WERE COLLECTED IN WOODLARK.

```
((ORIGIN/RELATION ((ORIG/ARTN P56) (ORIG/COLL Armstrong1)))
  (ARTIFACTS/RELATION ((ARTF/TYPE container1) (ARTF/NUMB P56)))
  (ORIGIN/RELATION ((ORIG/COLL Haddon1) (ORIG/ARTN P63)))
  (ORIGIN/RELATION ((ORIG/COLL Haddon1) (ORIG/ARTN P62)))
  (ARTIFACTS/RELATION ((ARTF/TYPE container1) (ARTF/NUMB P63)))
  (ARTIFACTS/RELATION ((ARTF/TYPE container1) (ARTF/NUMB P62)))
  (ARTIFACTS/RELATION ((ARTF/COND poor2) (ARTF/NUMB P63)))
  (ARTIFACTS/RELATION ((ARTF/COND poor2) (ARTF/NUMB P62)))
  (ORIGIN/RELATION ((ORIG/ARTN P56) (ORIG/PLAC Mount-Hagen)))
  (ORIGIN/RELATION ((ORIG/PLAC Woodlark) (ORIG/ARTN P63)))
  (ORIGIN/RELATION ((ORIG/PLAC Woodlark) (ORIG/ARTN P62))))
```

TEXT A27

P4302 IS AN ARMLET.
P4370 IS A NECKLACE.
NILFISK COLLECTED THE ARMLET IN DAUI.
THE COLLECTOR WAS A NORWEGIAN ANTHROPOLOGIST.
HE COLLECTED THE NECKLACE IN WOODLARK.
HE COLLECTED P4360 WHICH IS AN ARMLET FROM THERE.
THE CONDITION OF THIS ONE IS GOOD.
THE CONDITION OF THE DAUI ARMLET IS POOR.

```
((ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P4302)))
```

```
(ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P4370)))
(ORIGIN/RELATION
  ((ORIG/COLL Nilfisk1) (ORIG/ARTN P4302) (ORIG/PLAC Daui)))
(COLLECTORS/RELATION ((COLL/NAME Nilfisk1) (COLL/NATN Norwegian1)))
(COLLECTORS/RELATION
  ((COLL/OCCP anthropologist1) (COLL/NAME Nilfisk1)))
(ORIGIN/RELATION
  ((ORIG/COLL Nilfisk1) (ORIG/ARTN P4370) (ORIG/PLAC Woodlark)))
(ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P4360)))
(ORIGIN/RELATION
  ((ORIG/COLL Nilfisk1) (ORIG/ARTN P4360) (ORIG/PLAC Woodlark)))
(ARTIFACTS/RELATION ((ARTF/NUMB P4360) (ARTF/COND good1)))
(ARTIFACTS/RELATION ((ARTF/NUMB P4302) (ARTF/COND poor2))))
```

TEXT A28

SMITH WHO WAS A BRITISH MISSIONARY COLLECTED P259.
P259 IS A NECKLACE WHICH COMES FROM DAUI.
P261 IS AN ARMLET.
THE CONDITION OF BOTH ARTIFACTS IS GOOD.
THE MISSIONARY COLLECTED P593 AND P594 FROM WOODLARK.
P593 IS A JUG.
P594 IS AN ARMLET.
THE CONDITION OF THIS ARTIFACT IS GOOD.
THE JUG IS POOR.

```
((COLLECTORS/RELATION ((COLL/NAME Smith1) (COLL/NATN British1)))
  (COLLECTORS/RELATION ((COLL/OCCP missionary1) (COLL/NAME Smith1)))
  (ORIGIN/RELATION ((ORIG/ARTN P259) (ORIG/COLL Smith1)))
  (ORIGIN/RELATION ((ORIG/ARTN P259) (ORIG/PLAC Daui)))
  (ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P259)))
  (ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P261)))
  (ARTIFACTS/RELATION ((ARTF/COND good1) (ARTF/NUMB P261)))
  (ARTIFACTS/RELATION ((ARTF/COND good1) (ARTF/NUMB P259)))
  (ORIGIN/RELATION
    ((ORIG/COLL Smith1) (ORIG/PLAC Woodlark) (ORIG/ARTN P594)))
  (ORIGIN/RELATION
    ((ORIG/COLL Smith1) (ORIG/PLAC Woodlark) (ORIG/ARTN P593)))
  (ARTIFACTS/RELATION ((ARTF/TYPE container1) (ARTF/NUMB P593)))
  (ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P594)))
  (ARTIFACTS/RELATION ((ARTF/COND good1) (ARTF/NUMB P594)))
  (ARTIFACTS/RELATION ((ARTF/COND poor2) (ARTF/NUMB P593))))
```

TEXT A29

P957 AND P950 WERE COLLECTED BY SMITH.
THESE ARTIFACTS ARE JUGS.
THE COLLECTOR WAS A MISSIONARY.
P921 AND P922 WERE COLLECTED BY BEVAN.
THESE ARTIFACTS ARE SPEARS THAT CAME FROM DAUI.
BEVAN WAS A BRITISH ACADEMIC.
SMITH WAS AMERICAN.

```
((ORIGIN/RELATION ((ORIG/COLL Smith1) (ORIG/ARTN P950)))
  (ORIGIN/RELATION ((ORIG/COLL Smith1) (ORIG/ARTN P957)))
  (ARTIFACTS/RELATION ((ARTF/TYPE container1) (ARTF/NUMB P950)))
  (ARTIFACTS/RELATION ((ARTF/TYPE container1) (ARTF/NUMB P957)))
  (COLLECTORS/RELATION ((COLL/OCCP missionary1) (COLL/NAME Smith1)))
  (ORIGIN/RELATION ((ORIG/COLL Bevan1) (ORIG/ARTN P922)))
  (ORIGIN/RELATION ((ORIG/COLL Bevan1) (ORIG/ARTN P921)))
  (ORIGIN/RELATION ((ORIG/PLAC Daui) (ORIG/ARTN P922)))
  (ORIGIN/RELATION ((ORIG/PLAC Daui) (ORIG/ARTN P921)))
  (ARTIFACTS/RELATION ((ARTF/TYPE weapon1) (ARTF/NUMB P922)))
  (ARTIFACTS/RELATION ((ARTF/TYPE weapon1) (ARTF/NUMB P921)))
  (COLLECTORS/RELATION ((COLL/NAME Bevan1) (COLL/NATN British1)))
  (COLLECTORS/RELATION ((COLL/OCCP academic1) (COLL/NAME Bevan1)))
  (COLLECTORS/RELATION ((COLL/NAME Smith1) (COLL/NATN American1))))
```

TEXT A30

JONES WHO WAS A TRADER COLLECTED P350 FROM DAUI.
HE COLLECTED P370 FROM WOODLARK.
P350 IS A NECKLACE.
P370 IS AN ARMLET.
P391 IS A NECKLACE THAT COMES FROM WOODLARK.
THE CONDITION OF THESE ORNAMENTS IS GOOD.

ARMSTRONG AND HADDON WERE BRITISH.
THEY WERE ACADEMICS.
HADDON COLLECTED P597 AND P598 FROM DAUI.
THE ARTIFACTS ARE NECKLACES.
THE CONDITION OF THESE DAUI NECKLACES IS POOR.

P392 AND P393 ARE ARMLETS THAT WERE COLLECTED BY SMITH.
THIS COLLECTOR WAS A TRADER.
THE ARTIFACTS ARE FAIR.

```
((COLLECTORS/RELATION ((COLL/OCCP trader1) (COLL/NAME Jones)))
  (ORIGIN/RELATION
    ((ORIG/ARTN P350) (ORIG/COLL Jones) (ORIG/PLAC Daui)))
  (ORIGIN/RELATION
    ((ORIG/ARTN P370) (ORIG/COLL Jones) (ORIG/PLAC Woodlark)))
  (ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P350)))
  (ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P370)))
```

```
(ORIGIN/RELATION ((ORIG/ARTN P391) (ORIG/PLAC Woodlark)))
(ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P391)))
(ARTIFACTS/RELATION ((ARTF/COND good1) (ARTF/NUMB P391)))
(ARTIFACTS/RELATION ((ARTF/COND good1) (ARTF/NUMB P350)))
(ARTIFACTS/RELATION ((ARTF/COND good1) (ARTF/NUMB P370)))
(COLLECTORS/RELATION ((COLL/NATN British1) (COLL/NAME Haddon1)))
(COLLECTORS/RELATION ((COLL/NATN British1) (COLL/NAME Armstrong1)))
(COLLECTORS/RELATION ((COLL/OCCP academic1) (COLL/NAME Haddon1)))
(COLLECTORS/RELATION ((COLL/OCCP academic1) (COLL/NAME Armstrong1)))
(ORIGIN/RELATION
  ((ORIG/PLAC Daui) (ORIG/ARTN P598) (ORIG/COLL Haddon1)))
(ORIGIN/RELATION
  ((ORIG/PLAC Daui) (ORIG/ARTN P597) (ORIG/COLL Haddon1)))
(ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P598)))
(ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P597)))
(ARTIFACTS/RELATION ((ARTF/COND poor2) (ARTF/NUMB P598)))
(ARTIFACTS/RELATION ((ARTF/COND poor2) (ARTF/NUMB P597)))
(ORIGIN/RELATION ((ORIG/COLL Smith1) (ORIG/ARTN P393)))
(ORIGIN/RELATION ((ORIG/COLL Smith1) (ORIG/ARTN P392)))
(ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P393)))
(ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P392)))
(COLLECTORS/RELATION ((COLL/OCCP trader1) (COLL/NAME Smith1)))
(ARTIFACTS/RELATION ((ARTF/COND fair1) (ARTF/NUMB P393)))
(ARTIFACTS/RELATION ((ARTF/COND fair1) (ARTF/NUMB P392))))
```

TEXT A31

BEVAN COLLECTED P317 AND P325.
BOTH ARTIFACTS ARE CANOE-PROWS.
P219 IS A PADDLE THAT WAS COLLECTED BY ARMSTRONG.
ALL THREE ARTIFACTS COME FROM DAUI.
THE CONDITION OF BEVAN'S ARTIFACTS IS GOOD.

P719 IS A CANOE THAT WAS COLLECTED FROM WOODLARK.
IT WAS COLLECTED BY SMITH WHO WAS A TRADER.
THIS COLLECTOR WAS AMERICAN.
BEVAN WAS A BRITISH ACADEMIC.
P219'S COLLECTOR WAS A BRITISH MUSEUM-KEEPER.
HE COLLECTED P918 WHICH IS A CANOE-PROW.

```
((ORIGIN/RELATION ((ORIG/COLL Bevan1) (ORIG/ARTN P325)))
  (ORIGIN/RELATION ((ORIG/COLL Bevan1) (ORIG/ARTN P317)))
  (ARTIFACTS/RELATION
    ((ARTF/TYPE navigation/artifact) (ARTF/NUMB P317)))
  (ARTIFACTS/RELATION
    ((ARTF/TYPE navigation/artifact) (ARTF/NUMB P325)))
  (ORIGIN/RELATION ((ORIG/COLL Armstrong1) (ORIG/ARTN P219)))
  (ARTIFACTS/RELATION
    ((ARTF/TYPE navigation/artifact) (ARTF/NUMB P219)))
  (ORIGIN/RELATION ((ORIG/PLAC Daui) (ORIG/ARTN P317)))
  (ORIGIN/RELATION ((ORIG/PLAC Daui) (ORIG/ARTN P325)))
  (ORIGIN/RELATION ((ORIG/PLAC Daui) (ORIG/ARTN P219)))
```

```
(ARTIFACTS/RELATION ((ARTF/COND good1) (ARTF/NUMB P325)))
(ARTIFACTS/RELATION ((ARTF/COND good1) (ARTF/NUMB P317)))
(ORIGIN/RELATION ((ORIG/ARTN P719) (ORIG/PLAC Woodlark)))
(ARTIFACTS/RELATION
  ((ARTF/TYPE navigation/artifact) (ARTF/NUMB P719)))
(COLLECTORS/RELATION ((COLL/OCCP trader1) (COLL/NAME Smith1)))
(ORIGIN/RELATION ((ORIG/COLL Smith1) (ORIG/ARTN P719)))
(COLLECTORS/RELATION ((COLL/NAME Smith1) (COLL/NATN American1)))
(COLLECTORS/RELATION ((COLL/NAME Bevan1) (COLL/NATN British1)))
(COLLECTORS/RELATION ((COLL/OCCP academic1) (COLL/NAME Bevan1)))
(COLLECTORS/RELATION ((COLL/NAME Armstrong1) (COLL/NATN British1)))
(COLLECTORS/RELATION
  ((COLL/OCCP museum-keeper) (COLL/NAME Armstrong1)))
(ARTIFACTS/RELATION
  ((ARTF/TYPE navigation/artifact) (ARTF/NUMB P918)))
(ORIGIN/RELATION ((ORIG/COLL Armstrong1) (ORIG/ARTN P918))))
```

TEXT A32

THE CONDITION OF P971 IS GOOD.
HADDON COLLECTED IT.
HE WAS A BRITISH ACADEMIC.
BEVAN WAS A BRITISH MUSEUM-KEEPER.
HE COLLECTED P956.
THE CONDITION OF THIS ONE IS FAIR.
BOTH ARTIFACTS ARE SPEARS.

SMITH WHO WAS A GERMAN TRADER COLLECTED P316 WHICH IS AN ARROW.
HE COLLECTED P612 FROM WOODLARK.
THIS ARTIFACT IS A NECKLACE.
THE THREE WEAPONS COME FROM DAUI.
THE CONDITION OF THE ARROW IS GOOD.
THE ORNAMENT IS POOR.

```
((ARTIFACTS/RELATION ((ARTF/NUMB P971) (ARTF/COND good1)))
  (ORIGIN/RELATION ((ORIG/COLL Haddon1) (ORIG/ARTN P971)))
  (COLLECTORS/RELATION ((COLL/NAME Haddon1) (COLL/NATN British1)))
  (COLLECTORS/RELATION ((COLL/OCCP academic1) (COLL/NAME Haddon1)))
  (COLLECTORS/RELATION ((COLL/NAME Bevan1) (COLL/NATN British1)))
  (COLLECTORS/RELATION ((COLL/OCCP museum-keeper) (COLL/NAME Bevan1)))
  (ORIGIN/RELATION ((ORIG/ARTN P956) (ORIG/COLL Bevan1)))
  (ARTIFACTS/RELATION ((ARTF/NUMB P956) (ARTF/COND fair1)))
  (ARTIFACTS/RELATION ((ARTF/TYPE weapon1) (ARTF/NUMB P971)))
  (ARTIFACTS/RELATION ((ARTF/TYPE weapon1) (ARTF/NUMB P956)))
  (COLLECTORS/RELATION ((COLL/NAME Smith1) (COLL/NATN German1)))
  (COLLECTORS/RELATION ((COLL/OCCP trader1) (COLL/NAME Smith1)))
  (ARTIFACTS/RELATION ((ARTF/TYPE weapon1) (ARTF/NUMB P316)))
  (ORIGIN/RELATION ((ORIG/COLL Smith1) (ORIG/ARTN P316)))
  (ORIGIN/RELATION
    ((ORIG/ARTN P612) (ORIG/COLL Smith1) (ORIG/PLAC Woodlark)))
  (ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P612)))
  (ORIGIN/RELATION ((ORIG/PLAC Daui) (ORIG/ARTN P316)))
```

```
(ORIGIN/RELATION ((ORIG/PLAC Daui) (ORIG/ARTN P971)))
(ORIGIN/RELATION ((ORIG/PLAC Daui) (ORIG/ARTN P956)))
(ARTIFACTS/RELATION ((ARTF/NUMB P316) (ARTF/COND good1)))
(ARTIFACTS/RELATION ((ARTF/NUMB P612) (ARTF/COND poor2))))
```

TEXT A33

P900 IS A SPEAR.
P700 IS AN ARMLET.
THIS ARTIFACT WAS COLLECTED IN DAUI.
IT IS COMMON.
THE WEAPON WAS COLLECTED FROM THERE.
P940 AND P950 ARE ARROWS.
P200 IS A SPEAR BLADE.

```
((ARTIFACTS/RELATION ((ARTF/TYPE weapon1) (ARTF/NUMB P900)))
  (ARTIFACTS/RELATION ((ARTF/TYPE ornament1) (ARTF/NUMB P700)))
  (ORIGIN/RELATION ((ORIG/ARTN P700) (ORIG/PLAC Daui)))
  (ORIGIN/RELATION ((ORIG/ARTN P900) (ORIG/PLAC Daui)))
  (ARTIFACTS/RELATION ((ARTF/TYPE weapon1) (ARTF/NUMB P950)))
  (ARTIFACTS/RELATION ((ARTF/TYPE weapon1) (ARTF/NUMB P940)))
  (ARTIFACTS/RELATION ((ARTF/TYPE weapon1) (ARTF/NUMB P200))))
```

Appendix B

Examples of the influence of context factor types

This appendix gives examples of the effects of particular types of context factor in instances of the application of context information to interpretation problems. As discussed elsewhere in this thesis, the management of weights associated with context factors in the Capture system is not regarded as a precise theory about the relative importance of various factor types, but rather as an initial test of the context mechanism developed in the project. The initial weights and degrading procedures used during the processing of all the examples in this appendix are those detailed in Section 5.4.

The numbers given are the differences in the contributions of major factor types to the activations of the best two candidates that were chosen between when context was applied. Thus only contributions that were different are indicated, and negative numbers indicate contributions that were against the choice made by the system. The example texts were introduced in Section 5.3.

Example text A18 from Section 5.3(1)

Wintron manufactures P5050 which is a disc-drive. P1010 is a computer which is made by this manufacturer. It has a bolt. P8770 is a printer that is made by Plexir. Both peripherals are supplied by Clark. Smith supplies the machine with the bolt.

P4740 is manufactured by P5050's manufacturer in London. It is a micro-computer that is supplied by Jones. He supplies P8800 which is a terminal. The cost of the computer is 25. The computers are red. The three peripherals are green.

-- "He supplies P8800 which is a terminal";
'terminal1' (machine) preferred to 'terminal2' (place).
Differences: processing 16; association 12.

-- "The cost of the computer is 25";
'P4740' preferred to 'P1010' as referent for "the computer".
Differences: recency 78; emphasis 13; processing 2; association -2.

Example text A20 from Section 5.3(1)

Plexir manufactures P9000. It is a micro-computer. Wintron manufactures P7000 which is a disc-drive. P9000 is supplied by Smith.

P8000 is a computer. It is supplied by Jones. The status of this supplier is 10. The status of P9000's supplier is 20. The micro-computer is red. The manufacturer manufactures P9090.

-- "The manufacturer manufactures P9090";
'Plexir1' preferred to 'Wintron1' as referent for manufacturer.
Differences: association 50.

Example text A14 from Section 5.3(3)

> Plexir manufactures P9999 which is a computer. It is supplied by Smith. P1010 is a terminal that is supplied by Clark. This one is made by Mikota. These machines are red.
>
> P9000 is a green printer. It is made by Plexir. P4444 is a blue computer. The cost of the machine is 7850. The peripheral is supplied by the P9999 supplier. The terminal manufacturer makes the blue machine. The cost of Mikota's peripheral is 235.

-- "This one is made by Mikota";
'make1' (manufacture) preferred to 'make8' (force).
Differences: association 136.

-- "P9000 is a green printer";
'printer2' (machine) preferred to 'printer1' (person).
Differences: subject-area 30; processing 8.

-- "It is made by Plexir";
'P9000' preferred to 'P1010' as referent for "It".
Differences: recency 116; emphasis 44; association -26;
deixis -7; subject-area -7; processing 6.

Example text A33 from Section 5.3(3)

> P900 is a spear. P700 is an armlet. This artifact was collected in Daui. It is common. The weapon was collected from there. P940 and P950 are arrows. P200 is a spear blade.

-- "This artifact was collected in Daui";
'P700' preferred to 'P900' as referent for "This artifact".
Differences: recency 75; deixis 90; emphasis 34.

-- "It is common";
'P700' preferred to 'P900' as referent for "It".
Differences: recency 138; deixis 45; processing 32; emphasis 17.

-- "P940 and P950 are arrows";
'arrow1' (weapon) preferred to 'arrow2' (sign).
Differences: processing 10.

-- "P200 is a spear blade";
'blade1' (instrument) preferred to 'blade2' (loud jovial man).
Differences: association 63; processing 3.

Example text from Section 5.3(6b)

> P9999 is a disc-drive that is supplied by Smith. This peripheral is manufactured by Mikota. He supplies P7777 which is a terminal. It is manufactured in London by Plexir. Clark supplies P9000 which is manufactured by Marconi in Paris.

-- "Clark supplies P9000 which is manufactured by Marconi in Paris";
'manufacture/loc' preferred to 'supplies/loc' as a derived structure relationship.
Differences: association 56; processing 12.

Example text from Section 5.3(6c)

> P500 is an armlet. It was collected by Haddon. P550 is red.
>
> P9000 is a disc-drive that is supplied by Smith. The cost of the machine is 200. P9900 is red.

-- "P550 is red";
'relp/artifact/colour' preferred to 'relp/machine/colour' as underlying database predicate.
Differences: processing 6.

-- "P9900 is red";
'relp/machine/colour' preferred to 'relp/artifact/colour' as underlying database predicate.
Differences: task 20; processing 8.

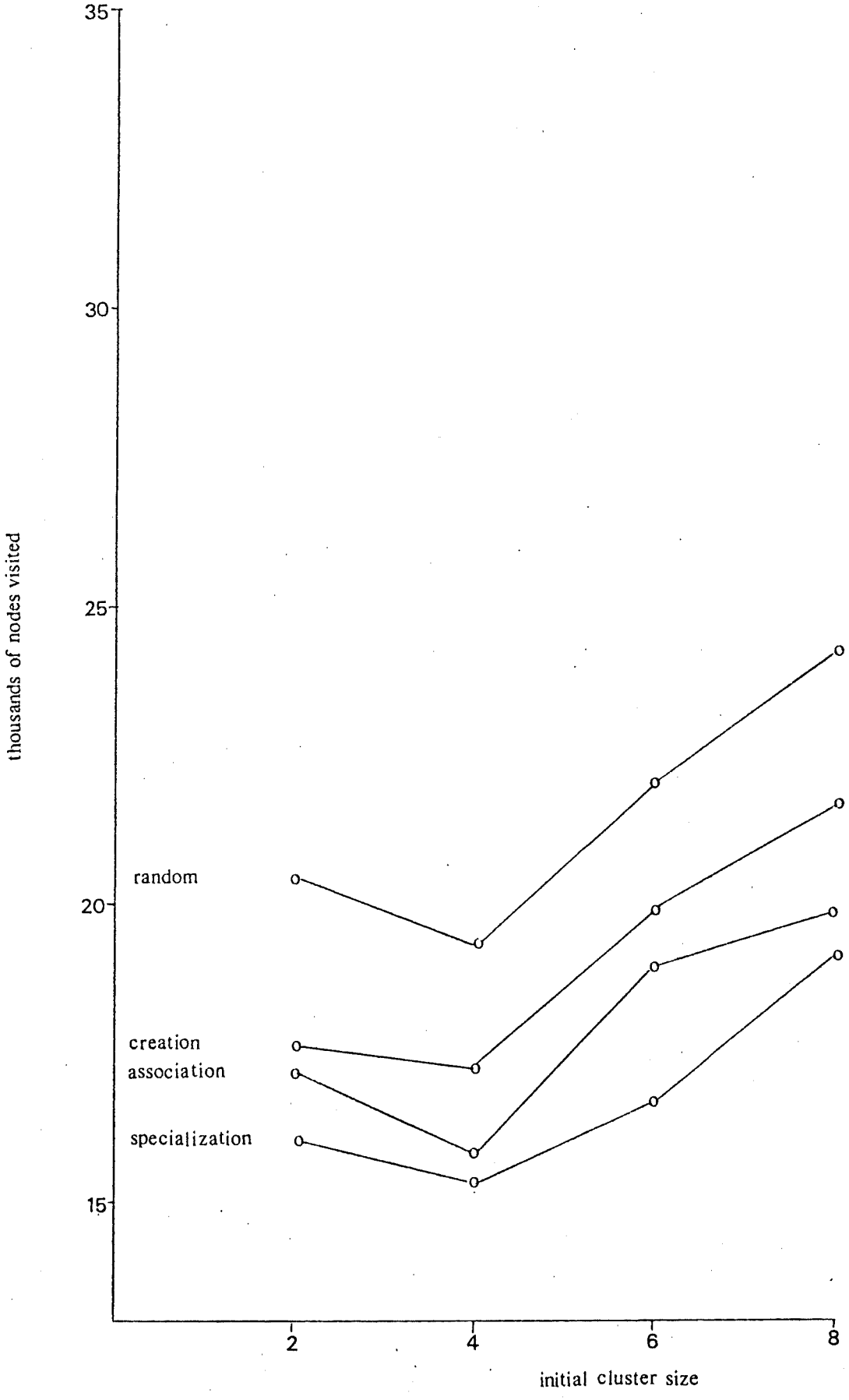
Appendix C

Effect of clustering on search efficiency

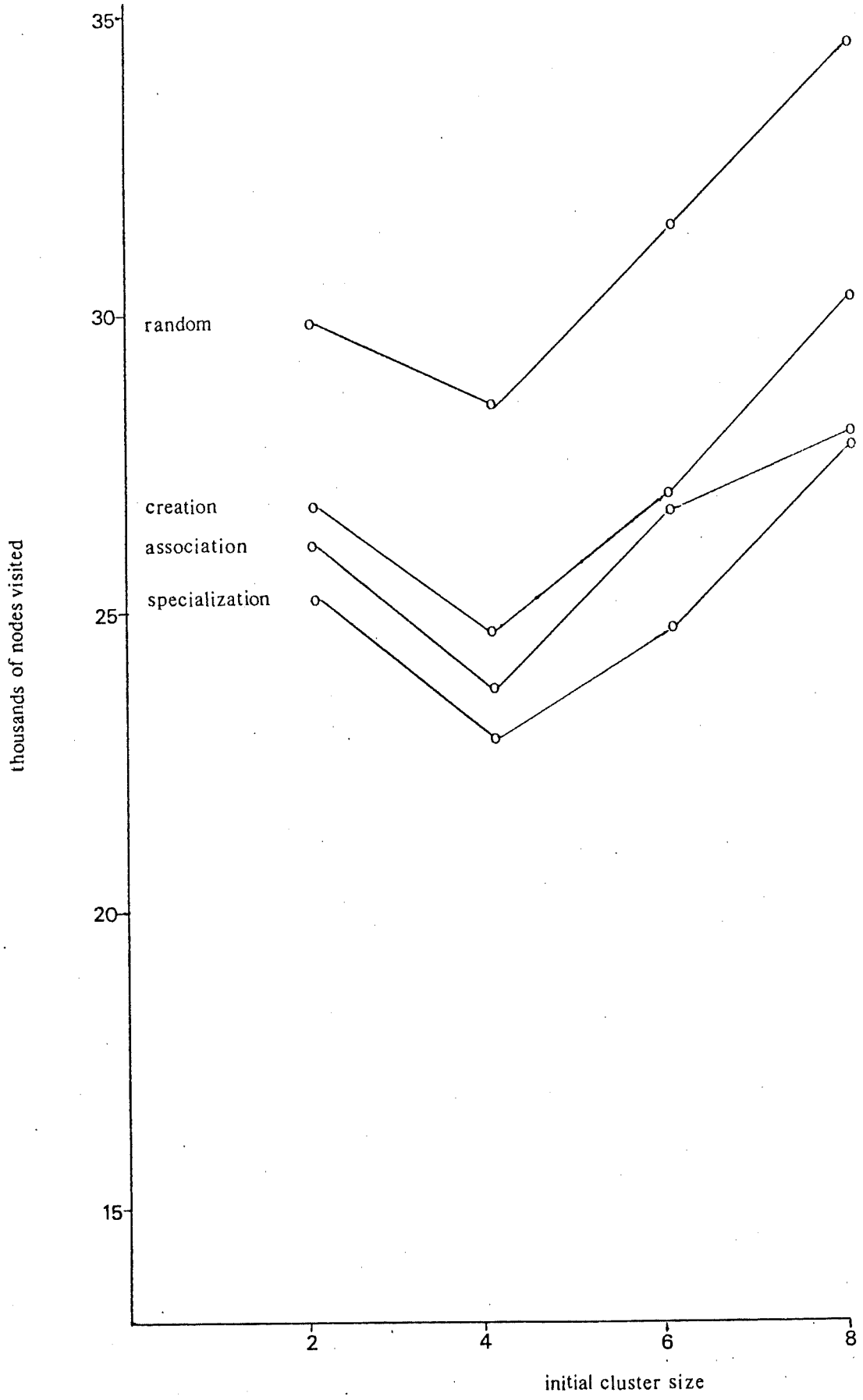
This appendix shows how the number of nodes visited during search operations for two example texts varied with the clustering method and the cluster size used. It should be emphasised that the information given in this appendix is only intended to give a general impression of how the various clustering methods affected memory searches for the relatively small memory knowledge base used by the test implementation. The size of the knowledge base (around 450 entities) means that the information given here cannot be regarded as experimental evidence, in any strong sense, about the utility of the clustering methods tried.

The nodes counted include both the network nodes and the cluster nodes in the indexing tree that were visited during searches performed for processing the two example texts. These are the example texts given in Chapter 1 (i.e. texts A30 and A14 in Appendix A). The maximum cluster size (i.e. the size to which clusters are allowed to grow to when new entities are created) is, in each case, 1.5 times the initial cluster size indicated.

Text A30


35
30
25
20
15
thousands of nodes visited
random
creation
association
specialization
2
4
6
8
initial cluster size

Text A14


35
30
25
20
15
thousands of nodes visited
random
creation
association
specialization
2
4
6
8
initial cluster size

References

[Bobrow77]
D. Bobrow and T. Winograd, "An Overview of KRL, a Knowledge Representation Language"; Cognitive Science 1, 1977, 3-46.

[Bobrow80]
R.J. Bobrow and B.L. Webber, "Knowledge Representation for Syntactic/Semantic Processing"; Proceedings, AAAI-1, Pittsburgh, August 1980, 316-323.

[Boguraev79]
B.K. Boguraev, "Automatic Resolution of Linguistic Ambiguities" (PhD thesis); Technical Report No. 11, Computer Laboratory, Cambridge University, August 1979.

[Boguraev82]
B.K. Boguraev and K. Sparck Jones, "A Natural Language Analyser for Database Access"; Information Technology: Research and Development, 1, 1982, 23-39.

[Boguraev83]
B.K. Boguraev and K. Sparck Jones, "How to Drive a Database Front End Using General Semantic Information"; Proceedings, ACL Conference on Applied Natural Language Processing, Santa Monica, 1983, 81-88.

[Borkin80]
S.A. Borkin, "Data Models: A Semantic Approach for Database Systems"; Cambridge, Mass.: MIT Press, 1980.

[Brachman78]
R.J. Brachman, "A Structural Paradigm for Representing Knowledge"; Technical Report No. 3605, Bolt Beranek and Newman Inc., May 1978.

[Brachman79a]
R.J. Brachman, "Taxonomy, Descriptions, and Individuals in Natural Language Understanding"; Proceedings, 17th Annual Meeting of the ACL, San Diego, 1979, 33-37.

[Brachman79b]
R.J. Brachman, "On the Epistemological Status of Semantic Networks". In "Associative Networks", N.V. Findler (ed.), New York: Academic Press, 1979.

[Bundy79]
A. Bundy, "What's The Difference? Predicate Calculus and Semantic Nets (Again)"; AISB Quarterly, October 1979, 8-9.

[Cater81]
A.W.S. Cater, "Analysis and Inference for English" (PhD thesis); Technical Report No. 19, Computer Laboratory, Cambridge University, September 1981.

[Chamberlin74]
D.D. Chamberlin and R.F. Boyce, "SEQUEL: A Structured English Query Language"; Proceedings, ACM SIGMOD Workshop on Data Description, Access and Control, 1974, 249-264.

[Charniak78]
E. Charniak, "On the Use of Framed Knowledge in Language Comprehension"; Artificial Intelligence, 11, 1978, 225-265.

[Charniak81]
E. Charniak, "The Case-Slot Identity Theory"; Cognitive Science, 5, 1981, 285-289.

[Charniak82]
E. Charniak, "Context Recognition in Language Comprehension". In "Strategies for Natural Language Processing", W.G. Lehnert and M.H. Ringle (eds.), Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1982.

[Charniak83]
E. Charniak, "Passing Markers: A Theory of Contextual Influence in Language Comprehension"; Cognitive Science, 7, 1983, 171-190.

[Cowie83]
J.R. Cowie, "Automatic Analysis of Descriptive Texts"; Proceedings, ACL Conference on Applied Natural Language Processing, Santa Monica, 1983, 117-123.

[Date81]
C.J. Date, "An Introduction to Database Systems"; 3rd Edition, Reading, Mass.: Addison-Wesley, 1981.

[DeJong79]
G. DeJong, "Prediction and Substantiation: A New Approach to Natural Language Processing"; Cognitive Science, 3, 1979, 251-273.

[Deliyani79]
A. Deliyani and P.A. Kowalski, "Logic and Semantic Networks"; Communications of the ACM, 22, 1979, 184-192.

[Fahlman79]
S.E. Fahlman, "NETL: A System for Representing and Using Real-World Knowledge"; Cambridge, Mass.: MIT Press, 1979.

[Fahlman80]
S.E. Fahlman, "Preliminary Design for a Million-Element NETL Machine"; Proceedings, AAAI-1 Conference, 1980, 249-252.

[Fahlman81]
S.E. Fahlman, D.S. Touretzky, W. van Roggen, "Cancellation in a Parallel Semantic Network"; Proceedings, IJCAI-81, Vancouver, August 1981, 257-264.

[Gray81]
M.A. Gray, "Implementing Unknown and Imprecise Values in Databases". In "Databases (Proceedings of the First British National Conference on Databases)", S.M. Deen and P. Hammersley (eds.), London: Pentech Press, 1981.

[Grosz77]
B.J. Grosz, "The Representation and Use of Focus in Dialogue Understanding"; SRI Technical Note No. 151, SRI International, July 1977.

[Grosz81]
B.J. Grosz (ed.), "Research on Natural Language Processing at SRI"; SRI Technical Note No. 257, SRI International, November 1981.

[Grosz82]
B.J. Grosz, N. Haas, G. Hendrix, J. Hobbs, P. Martin, R. Moore, J. Robinson, and S. Rosenschein, "DIALOGIC: A Core Natural-language Processing System"; Proceedings of the Ninth International Conference on Computational Linguistics, Prague, July 1982, 95-100.

[Grosz83]
B.J. Grosz, A.K. Joshi and S. Weinstein, "Providing a Unified Account of Definite Noun Phrases in Discourse"; Proceedings, 21st Annual Meeting of the ACL, 1983, 44-50.

[Hayes77a]
Patrick J. Hayes, "In Defence of Logic"; Proceedings, IJCAI 5, Cambridge, Mass., 1977, 559-565.

[Hayes77b]
Philip J. Hayes, "On Semantic Nets, Frames and Associations"; Proceedings, IJCAI 5, Cambridge, Mass., 1977, 99-107.

[Hendrix78]
G.G. Hendrix, "Encoding Knowledge in Partitioned Networks"; SRI Technical Note No. 164, SRI International, June 1978.

[Hillis81]
W.D. Hillis, "The Connection Machine"; AI Memo 646, Artificial Intelligence Laboratory, MIT, September 1981.

[Hirst82]
G. Hirst and E. Charniak, "Word Sense and Case Slot Disambiguation"; Proceedings, National Conference on Artificial Intelligence, AAAI-82, August 1982, 95-98.

[Hirst83]
G. Hirst, "A Foundation for Semantic Interpretation"; Technical Report CS-83-03, Department of Computer Science, Brown University, January 1983.

[Hobbs81]
J.R. Hobbs, "Metaphor Interpretation as Selective Inferencing"; Proceedings, IJCAI-81, Vancouver, August 1981, 85-91.

[Kaplan82]
M.K. Kaplan and J. Bresnan, "Lexical-Functional Grammar: A Formal system for Grammatical Representation". In "The Mental Representation of Grammatical Relations", J. Bresnan (ed.), Cambridge, Mass.: MIT Press, 1982.

[Kaplan81]
S.J. Kaplan and J. Davidson, "Interpreting Natural Language Database Updates"; Proceedings, 19th Annual Meeting of the ACL, 1981, 139-141.

[Kay80]
M. Kay, "The Proper Place of Men and Machines in Language Translation"; Xerox Palo Alto Research Center, October 1980.

[King79]
T.J. King, "The Design of a Relational Database Management System for Historical Records"; Ph.D. thesis, University of Cambridge, 1979.

[Lebowitz81]
M. Lebowitz, "The Nature of Generalization in Understanding"; Proceedings, IJCAI-81, Vancouver, August 1981, 348-353.

[Lebowitz83]
M. Lebowitz, "Intelligent Information Systems"; Proceedings, 6th ACM SIGIR Conference, 1983, 25-29.

[Lehnert83]
W.G. Lehnert, M.G. Dyer, P.N. Johnson, C.J. Yang and S. Harley, "BORIS - An Experiment in In-Depth Understanding of Narratives"; Artificial Intelligence, 20, 1983, 15-62.

[Maier82]
D. Maier and S.C. Salveter, "Supporting Natural Language Updates in Database Systems"; Proceedings, European Conference on Artificial Intelligence, 1982, 244-249.

[Marcus80]
M.P. Marcus, "A Theory of Syntactic Recognition for Natural Language"; Cambridge, Mass.: MIT Press, 1980.

[Mark81]
W. Mark, "Representation and Inference in the Consul System"; Proceedings, IJCAI-81, Vancouver, August 1981, 375-381.

[Martin80]
W.A. Martin, "Roles, Co-Descriptors, and the Formal Representation of Quantified English Expressions"; Technical Manual 139, Laboratory for Computer Science, MIT, May 1980.

[McDermott75]
D.V. McDermott, "Very Large Planner Type Data Bases"; AI Memo 339, Artificial Intelligence Laboratory, MIT, 1975.

[McDonald82]
D.B. McDonald, "Understanding Noun Compounds" (PhD thesis); Report CMU-CS-82-102, Department of Computer Science, Carnegie-Mellon University, January 1982.

[McDonald81]
D.D. McDonald, "Language Production: the Source of the Dictionary"; Proceedings, 19th Annual Meeting of the ACL, Stanford, 1981, 57-62.

[Mellish80]
C.S. Mellish, "Some Problems in Early Noun Phrase Interpretation"; Proceedings, AISB Conference, Amsterdam, July 1980.

[Minsky75]
M.L. Minsky, "A Framework for Representing Knowledge". In "The Psychology of Computer Vision", Winston (ed.), New York: McGraw-Hill, 1975.

[Norton83]
L.M. Norton, "Automated Analysis of Instructional Text"; Artificial Intelligence, 20, 1983, 307-344.

[Quillian68]
M.R. Quillian, "Semantic Memory". In "Semantic Information Processing", M. Minsky (ed.), Cambridge, Mass.: MIT Press, 1968.

[Reimer83]
U. Reimer, and U. Hahn, "A Formal Approach to the Semantics of a Frame Data Model"; Proceedings, IJCAI-83, Karlsruhe, 1983, 337-339.

[Rieger75]
C.J. Rieger, "Conceptual Memory and Inference"; In "Conceptual Information Processing", R.C. Schank, N.M. Goldman, C.J. Rieger, and C.K. Riesbeck (eds.), Amsterdam: North-Holland, 1975.

[Ritchie76]
G.D. Ritchie, "Problems in Local Semantic Processing"; Proceedings, AISB Conference, Edinburgh, 1976, 234-241.

[Ritchie77]
G.D. Ritchie, "Computer Modelling of English Grammar"; Thesis CST-1-77, Department of Computer Science, University of Edinburgh, 1977.

[Ritchie83]
G.D. Ritchie and F.K. Hanna, "Semantic Networks - A General Definition and A Survey"; Information Technology: Research and Development, 2, 1983, 187-231.

[Sager81]
N. Sager, "Natural Language Information Processing"; Reading, Mass.: Addison-Wesley, 1981.

[Schank75]
R.C. Schank and the Yale A.I. Project, "SAM - A Story Understander"; Research Report No. 43, Department of Computer Science, Yale University, 1975.

[Schank82a]
R.C. Schank, "Reminding and Memory Organization: An Introduction to MOPs"; In "Strategies for Natural Language Processing", W.G. Lehnert and M.H. Ringle (eds.), Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1982.

[Schank82b]
R.C. Schank, "Dynamic Memory"; Cambridge: Cambridge University Press, 1982.

[Schubert79]
L.K. Schubert, R.G. Goebel and N.J. Cercone, "The Structure and Organization of a Semantic Net for Comprehension and Inference". In "Associative Networks", N.V. Findler (ed.), New York: Academic Press, 1979.

[Schmolze83]
J.G. Schmolze and T.A. Lipkis, "Classification in the KL-ONE Knowledge Representation System"; Proceedings, IJCAI-83, Karlsruhe, 1983, 330-332.

[Shapiro79]
S.C. Shapiro, "The SNePS Semantic Network Processing System"; In "Associative Networks", N.V. Findler (ed.), New York: Academic Press, 1979.

[Sidner79]
C.L. Sidner, "Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse"; Technical Report AI-TR-537, Artificial Intelligence Laboratory, MIT, June 1979.

[Small80]
S. Small, "Word Expert Parsing. A Theory of Distributed Word-based Natural Language Understanding"; Technical Report No. 954, Department of Computer Science, University of Maryland, 1980.

[Sparck Jones83a]
K. Sparck Jones, "Compound Noun Interpretation Problems". In "Computer Speech Processing", Lecture Notes, SERC/CREST-ITG Advanced Course, Cambridge, July 1983.

[Sparck Jones83b]
K. Sparck Jones, "Shifting Meaning Representations"; Proceedings, IJCAI-83, Karlsruhe, 1983, 621-623.

[Steinacker83]
I. Steinacker and H. Trost, "Structural Relations - A Case Against Case"; Proceedings, IJCAI-83, Karlsruhe, 1983, 627-629.

[Tait82]
J.I. Tait, "Automatic Summarising of English Texts"; Ph.D. Thesis, University of Cambridge, 1982.

[Tait83]
J.I. Tait and K. Sparck Jones, "Automatic Search Term Variant Generation for Document Retrieval"; R&D Report No. 5793, British Library, 1983.

[Walker78]
D.E. Walker, "Understanding Spoken Language"; New York: Elsevier North-Holland, 1978.

[Waltz81]
D.L. Waltz, "Toward a Detailed Model of Processing for Language Describing the Physical World"; Proceedings, IJCAI-81, Vancouver, August 1981, 1-6.

[Wilensky81]
R. Wilensky, "A Knowledge-based Approach to Language Processing: A Progress Report"; Proceedings, IJCAI-81, Vancouver, August 1981, 25-30.

[Wilensky82]
R. Wilensky, "Talking to Unix In English: An Overview of an On-line UNIX Consultant"; Report No. UCB/CSD 82/104, Department of Computer Science, University of California, Berkeley, September 1982.

[Wilks73]
Y. Wilks, "An Artificial Intelligence Approach to Machine Translation". In "Computer Models of Thought and Language", R.C. Schank and K.M. Colby (eds.), San Francisco: W.H. Freeman and Company, 1973.

[Wilks75a]
Y. Wilks, "An Intelligent Analyser and Understander of English"; Communications of the ACM, 18, 1975, 264-274.

[Wilks75b]
Y. Wilks, "A Preferential, Pattern-matching Semantics for Natural Language Understanding"; Artificial Intelligence, 6, 1975, 53-74.

[Wilks77]
Y. Wilks, "Good and Bad Arguments about Semantics Primitives"; Communication and Cognition, 10, 1977, 181-221.

[Wilks78]
Y. Wilks, "Making Preferences More Active"; Artificial Intelligence, 11, 1978, 197-223.

[Winograd72]
T. Winograd, "Understanding Natural Language"; Edinburgh: Edinburgh University Press, 1972.

[Woods70]
W.A. Woods, "Transition Network Grammars for natural language analysis"; Communications of the ACM, 13, 1970, 591-606.

[Woods73]
W.A. Woods, "An Experimental Parsing System for Transition Network Grammars". In "Natural Language Processing", R. Rustin (ed.), New York: Algorithmics Press, 1973.

[Woods78a]
W.A. Woods, "Research in Natural Language Understanding"; Report No. 3797, Bolt Beranek and Newman Inc., April 1978.

[Woods78b]
W.A. Woods, "Semantics and Quantification in Natural Language Question Answering"; Advances in Computers, 17, 1978, 1-87.

[Zdybel81]
F. Zdybel, N.R. Greenfeld, M.D. Yonke and J. Gibbons, "An Information Presentation System"; Proceedings, IJCAI-81, Vancouver, August 1981, 978-984.