Technical Report

UCAM-CL-TR-47
ISSN 1476-2986

Number 47

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Automatic summarising of English texts

John Irving Tait

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

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

This technical report is based on a dissertation submitted December 1982 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Wolfson College.

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

http://www.cl.cam.ac.uk/TechReports/

ISSN 1476-2986

Automatic Summarising of English Texts

J. I. Tait

The thesis describes a computer program called Scrabble which can summarise short English texts. It uses large bodies of predictions about the likely contents of texts about particular topics to identify the commonplace material in an input text. Pre-specified summary templates, each associated with a different topic, are used to condense the commonplace material in the input. Filled-in summary templates are then used to form a framework into which unexpected material in the input may be fitted, allowing unexpected material to appear in output summary texts in an essentially unreduced form. The system's summaries are in English.

The program is based on technology not dissimilar to a script applier. However, Scrabble represents a significant advance over previous script-based summarising systems. It is much less likely to produce misleading summaries of an input text than some previous systems, and can operate with less information about the subject domain of the input than others.

These improvements are achieved by the use of three main novel ideas. First, the system incorporates a new method for identifying the topic or topics of an input text. Second, it allows a section of text to have more than one topic at a time, or at least a composite topic which may best be dealt with by the computer program simultaneously applying to the text predictions associated with more than one simple topic. Third, Scrabble incorporates new mechanisms for the incorporation of unexpected material in the input into its output summary texts. The incorporation of such material in the output summary is motivated by the view that it is precisely unexpected material which is likely to form the most salient matter in the input text.

The performance of the system is illustrated by means of a number of example input texts and their corresponding Scrabble summaries.

Acknowledgements

First, and foremost, I would like to thank my supervisor, Karen Sparck Jones. Karen has the knack in dealing with her students of knowing when to encourage, when to coerce, when to criticise and when to let nature take its course. My progress as a research student has required large doses of all four at various times. I would particularly like to thank her for the enormous amount of effort she has put into reading and commenting on the text of this thesis during a time when she has had a great many other commitments.

Second I would like to thank Arthur Cater for allowing me to use his in many ways excellent AD-HAC semantic parser and generator without which the work described here could not have taken anything like its eventual course. Equally I would like to thank Arthur for the invaluable hours we have spent discussing the many problems of natural language processing.

Third, Branimir Boguraev has been a most valuable colleague throughout my time in Cambridge, and I have greatly appreciated the effort he has put into producing Cambridge versions of various LISP programming tools.

I would also like to thank my many other friends and colleagues in the Computer Laboratory, the Computing Service and more generally in Cambridge for the support and assistance they have given me.

I am especially grateful to the organisations who have given me financial support: International Computers Ltd. through the ICL Research Studentship, and more recently the British Library Research and Development Department, who have given me the opportunity of working on an extremely interesting research project whilst I have been finishing up the work described here.

Last, but not least, I would like to thank Mu who has contributed so much to all I have done in the last five and half years.

For my parents.

Contents

1. Introduction

This thesis describes a computer program called Scrabble which can summarise short English texts. The program represents an exploration of new techniques for the automatic extraction of summaries from input textual material. The technique combines relatively shallow (and therefore computationally inexpensive) inferential analysis of an input text with sophisticated mechanisms for the construction of a summary text from the products of this shallow inferential analysis.

Scrabble uses large bodies of predictions about the likely contents of texts about particular topics to identify the commonplace material in an input text. Pre-specified summary templates, each associated with a different topic, are used to condense the commonplace material in the input. Filled in summary templates are then used to form a framework into which unexpected material in the input may be fitted, allowing this material to appear in output summary texts in an essentially unreduced form. The system's summaries are in English.

The program is based on three main novel ideas. First it incorporates a new method for identifying the topic or topics of an input text. Second it embodies the notion of a section of text having more than one topic at a time, or at least a composite topic which may best be dealt with by the computer program simultaneously applying to the text predictions associated with more than one simple topic. Third, Scrabble incorporates new mechanisms for the incorporation of unexpected material in the input into its output summary texts. The incorporation of such material in the output summary is motivated by the view that it is precisely the unexpected material in the input which is likely to form the most salient matter in an input text.

I think it is generally accepted that it would be desirable if there were computer programs which understood natural language. The specific motivation for the Scrabble project was the hope of overcoming the problems posed by the extreme expense of general purpose inference mechanisms like that proposed by [Cater81]. However, in the course of the project the focus has moved from such wide issues to a more specific technical reappraisal of some of the techniques which have been proposed for tightly controlled, specialised inference mechanisms. The production of summaries was chosen as a task domain for the project because the construction of a summary is widely considered to require some understanding of the summarised material, and because it would provide a good test of some particular properties of tightly controlled specialised inference mechanisms.

Before beginning the more technical matter of the thesis, two remarks about the limitations and aims of this work must be made.

First, although the system deals only with English, I believe Scrabble could operate with any language as its input and output, not necessarily the same for each. Most of the program manipulates an interlingual representation of natural language, and in fact an early version of the program generated Japanese, though this is no longer possible for low-level technical reasons. Thus the program might, at a certain level, be regarded as a prototype for a machine translation system.

Secondly, this work addresses some aspects of ***computer*** processing of natural language. I make no claim that anything in this work is relevant to understanding human use of natural language. Scrabble is intended to produce summaries which are reasonably similar to those which a human being might produce for the same initial text. It is only these summaries which are intended to serve as a basis for comparison with human linguistic behaviour, and not the processes by which they are derived. Moreover, any discussion of other computer systems is couched in the same terms: I will compare only with human beings the computer systems actual or potential outputs when presented with particular natural language inputs.

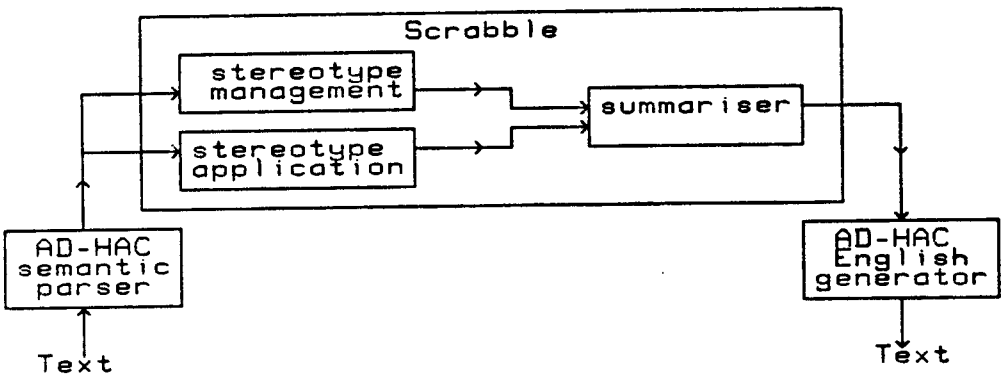
The name Scrabble is based on the ordinary English verb "to scrabble", that is to move quickly and erratically looking for something: in the computer programs case the topic or topics of incoming text. The name is not an acronym, nor has it anything to do with the word game.

1.1 The Organisation of the Scrabble system

The production of a summary by Scrabble involves the use of five different programs. Three of these programs form the Scrabble core: the ***stereotype management*** module; the ***stereotype application*** module; and the summariser. The other two programs are the AD-HAC semantic parser and the AD-HAC generator. They are the work of Cater (see [Cater82]), and form subsidiary parts of the Scrabble system. The core Scrabble programs do not manipulate English text sentences as such, but operate on deep meaning representations (in fact CD-structures, described in detail by Cater). The AD-HAC semantic parser is used to convert input English sentences into CD-representations for analysis by the Scrabble core. The AD-HAC generator is used to produce the output summary text from CD-representations formed by the Scrabble summariser.

Stereotypes are bundles of predictions about the likely contents of input texts. Each bundle is associated with a particular topic and each is associated with a ***summary template*** defining how to summarise an entirely commonplace text about that topic. Stereotypes are thus the central apparatus for summarising.

The five component programs forming the Scrabble system fit together as follows.



When processing a text, control passes between the programs as follows. For every input sentence the AD-HAC semantic parser is invoked . It hands a representation of the meaning of the input sentence to the stereotype application module, and a set of clues about possible topics of the text are handed to the stereotype management module. The application module informs the management module of the success or otherwise of each existing stereotype in predicting the input material. The management module may decide that an existing stereotype may not, in fact, be appropriate for the input text, or that it was relevant but is no longer so, or that the topic of the text has changed, so some new stereotype should be ***instantiated***. At the end of input text analysis, a representation of the entire text is passed from the stereotype management and application modules to the summariser. The summariser constructs a ***summary representation*** from the summary templates of stereotypes that predicted the contents of parts of the text together with any unexpected material in the input. This summary representation is passed to the AD-HAC English generator and turned into an English text.

The ***text representation***, which forms the input to the summariser has three parts. First is the state of any stereotype instances which the manager decided were related to the topics of the input text. These states represent the *way* in which the text satisfied the system's various predictions about its contents. Second are the CD-representations of the parts of the text which were not expected by the system. Third is a model of the way in which the elements of the first two parts of the text representation were interrelated in the input text.

The system is written entirely in Cambridge LISP.

1.2 A Sample Scrabble Run

This section presents the output from a typical Scrabble run, in order to give a flavour of the capabilities of the system.

The program output is presented in a form which will be used in various places later in the thesis. Lines output by the program are indented at the right and left hand sides and printed in a different font ("A Different Font "). Lines running the full width of the page, in the same font as the main text, are explanatory comments.

In the output square brackets ("[" and "] ") mark where the AD-HAC semantic parser was operating. Usually they enclose various forms of debugging output, but these have been removed to improve clarity. The remainder of the output is as it was produced during the Scrabble run. Hopefully the output will be self explanatory: but whether it is or not, all the processing necessary to produce the summary will be described in great detail in the rest of this thesis.

```
                    Times to this point total 0.106
 seconds + 1.241 seconds
Run of AD-HAC with Scrabble version 15.14 in 850K at
12.15 on 1 OCT 82
[]
Entering Scrabble for sentence :
 (MARY AND JOHN WENT TO THE ZOO)
Spent 163 ms in Scrabble
[]
Entering Scrabble for sentence :
 (THEY FED THE MONKEYS SOME PEANUTS)
Spent 50 ms in Scrabble
[]
Entering Scrabble for sentence :
 (AFTER THEY SAW THE ZEBRA , THEY WENT TO A RESTAURANT)
Spent 888 ms in Scrabble
[]
Entering Scrabble for sentence :
 (JOHN TOLD THE WAITER THAT HE WANTED A STEAK)
Spent 47 ms in Scrabble
[]
Entering Scrabble for sentence :
 (BECAUSE THEY DIDN'T HAVE MONEY THE WAITER WAS ANGRY)
Spent 33 ms in Scrabble
[]
Entering Scrabble for sentence : (THEY RAN AWAY)
Spent 37 ms in Scrabble
English summary is:
   JOHN AND MARY WENT TO A ZOO.
   THEY WENT TO A RESTAURANT BUT A WAITER WAS FURIOUS
BECAUSE THEY DIDN'T HAVE SOME MONEY.
END OF LISP RUN AFTER 6.96+6.42 SECS - 94.8% STORE USED
```

1.3 Plan of the Rest of the Thesis

The structure of the rest of the thesis is as follows: Briefly, Chapters 2 and 3 describe the background to Scrabble, in terms of tasks within natural language processing, previous attempts to build similar systems, and the overall design of the Scrabble system, Chapters 4, 5 and 6 describe in detail the three main subprograms which form the Scrabble program, and Chapter 7 presents conclusions.

More specifically, Chapter 2 discusses the role of inference in natural language processing, and reviews various previous attempts to build inference systems for natural language processing. In particular, Chapter 2 contains detailed critiques of work based on ***scripts*** and to a lesser extent, ***frames***.

Chapter 3 is an overview of the Scrabble system. It discusses the solutions embodied in Scrabble to various shortcomings of previous systems, describes why summarising was chosen as a test bed for more general ideas about inference, and why the AD-HAC semantic parser and English generator were chosen for the Scrabble system, and gives a more detailed overview of the structure of the Scrabble program.

Chapter 4 describes the mechanism by which stereotypes are managed. It includes descriptions of how stereotype are ***suggested*** as being appropriate for the text, of how they are ***identified*** as actually being appropriate, and of how the model of the input text which is handed to the summariser is constructed.

Chapter 5 describes the mechanism by which the stereotype's predictions are compared with the input text and the results of the comparison are recorded.

Chapter 6 describes the process by which summaries are formed.

Chapter 7 contains some conclusions, focussing on comparisons between Scrabble and other inference systems and problems with the current Scrabble system which must be dealt with before it can form the basis of a practically useful natural language processing system.

Chapters 4, 5 and 6 all contain detailed examples of the summarising process, whilst Chapter 5 contains an annotated example stereotype. More examples of the summaries produced by Scrabble may be found in Appendix A. Appendix B contains an additional example stereotype prototype.

2. The Background to Scrabble

Inference, in the context of this work, is taken as the process by which a listener or reader draws out those parts of the meaning of a piece of natural language which are not manifest in the text itself. This chapter discusses why any task to which a natural language processing system is likely to be applied will require the system to exhibit some inferential abilities. An idealised model of a natural language is then described which is intended to illustrate the boundaries between those parts of the system which provide its inferential abilities and those which provide its linguistic abilities, as these terms are used in this thesis. Next, various proposals for the implementation of computer systems possessing these capabilities are discussed. Finally some proposals for the construction of natural language processing systems capable of summarising their input are discussed.

The discussion of meaning presented here is intended only to form a minimal framework within which the Scrabble program may be discussed. Nothing in this chapter is intended to constitute a genuine theory of meaning for natural language in general.

2.1 The Necessity for Inference in Natural Language Processing

In this section I want to emphasize how all pervading is the need for inferential abilities in a system which is to perform tasks which demonstrate even quite trivial understanding of natural language.

I assume the ***manifest meanings*** of a piece of text are those parts of the meaning which are available without recourse to inference. Conversely I will call those parts of the meaning which are only made available through inference ***inferred meanings***.

I also assume that the meanings of each word in the text are accessible by reference to some kind of dictionary containing information similar to that found in a conventional English dictionary. In particular, I assume that words will be associated with a number of different senses, and that each word-sense will have associated with it a description of its meaning containing enough information to allow the use of different senses of the same word to be distinguished given sufficient context. However, at this stage I will make no assumptions about how the meanings of the words are represented in the dictionary.

I further assume that some sort of grammar of the language is available, and also a set of (semantic) rules determining how different senses of the words are likely to combine. Additionally, I assume that no part of the manifest meaning of a sentence comes from its context. Hence the ***manifest meaning*** of a sentence is that derivable from the

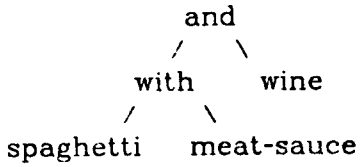
dictionary, grammar and word sense selection rules. The derivation of the ***inferred meaning*** requires additional sources of information.

A system which could only obtain the manifest meaning of a piece of text would be incapable of performing even quite simple tasks. Consider the sentence[1]:

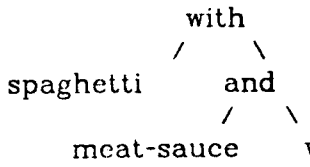
(1) I would like spaghetti with meat sauce and wine.

This sentence has two possible structures which must be distinguished for it to be properly understood. They are:

Structure A



Structure B



Structure A corresponds to a glass of wine accompanying a plate of spaghetti and meat sauce, whilst Structure B corresponds to a plate of spaghetti covered with wine and meat sauce. I think most people would expect the meal corresponding to Structure A if they said sentence (1) to a waiter. But note that the information necessary to select the appropriate sentence meaning is unlikely to be part of the manifest meaning as manifest meaning was defined earlier. In particular consider the rather similar sentence:

(2) I would like pork with mushrooms and cider.

where the (or at least my) preferred meaning is the one corresponding to Structure B above, although this is perhaps a weaker preference than that in the case of sentence (1).

Of course, one might reply that the disambiguated meaning could be made manifest, in my sense, by including in the dictionary entries for pork and spaghetti information about ingredients they are likely to be combined with in cooking, but I think this would be a rather extended notion of a dictionary.

Since, then, the most appropriate meanings for sentences (1) and (2) cannot be selected by the use of a dictionary, a grammar, and sense selection rules, the most likely sentence meanings cannot be selected on the basis of manifest meaning alone. What is required is the application of fragments of an amorphous body of information (in this case about eating habits and cooking) in a step by step manner until a most likely sentence meaning is discovered. I think such processing is only reasonably described as inference.

---

[1]This example closely follows one in [Charniak76a].

There are a variety of devices in natural language which directly signal that the text has meaning beyond that which is manifest. Amongst them are the use of pronouns and other anaphoric devices, such as some uses of definite noun phrases and the need to use textual context to select a unique meaning for a sentence. I will call such devices ***problem signals***.

The meaning of a piece of text containing a pronoun can only be fully divined if the intended referent of the pronoun can be found. For example:

(3) They threw them peanuts.

is clearly somewhat lacking, semantically, because one cannot tell who either occurrence of "they" refers to. But if it is used in context:

(4) John and Mary went to the zoo. They saw some monkeys. They threw them peanuts.

one gets a more complete picture, in which John and Mary throw some monkeys some peanuts. This is not part of the manifest meaning of the text and the association of "John and Mary" with the "they" of the second sentence is a task probably requiring inference.

The selection of a unique meaning for a sentence by means of the use of context is illustrated by the pairs[2]:

(5) John goes to the zoo often. He is very fond of one particular seal.

(6) The royal proclamation was finally complete. The king sent for his seal.

In both cases, if the second sentence is taken in isolation, there is no means of determining the intended sense of the word "seal". However, when they are taken in the context provided by the other sentence, in each only one sense seems appropriate. For example (5) the preferred sense is that of an animal; for example (6) some sort of marking or stamping instrument.

My final example is of a rather different character. In each of the above cases the presence of some problem which I have claimed requires some form of inference for its resolution has been signalled by some property of the text itself. In each of the above examples part of the manifest meaning of the text fragment is that its meaning is incomplete; pronouns signal that their referents should be determined; elliptical sentences are often syntactically incomplete, or failing that they have transparent semantic omissions; the occurrence of polysemous words indicates that one of their senses must be selected. Of course, if a natural language processing system ignored the problem signal in either case it would not be obvious that the referent of a pronoun had not been identified, or that a particular

---

[2]Examples taken from [Lehnert79].

sense of a polysemous word had not been selected, until the system was subjected to some test of its comprehension of the input. The following example is best illustrated by considering the response to a simple test of comprehension.

Consider the paragraph[3]:

(7) John went to the restaurant and ordered. The waitress brought him a hamburger and he ate quickly. He tipped the waitress and left.

Now consider the following four questions:

(8) What did John order?

(9) Who served John the hamburger?

(10) What did John eat?

(11) Did John leave the restaurant?

I do not think it is difficult to answer that John ordered a hamburger, John ate the hamburger, and John left the restaurant. None of this is manifest from the text; it can only be deduced by combining the manifest meaning of the text with information about the normal course of events in a restaurant. Again to be able to pass this trivial comprehension test the language user must be capable of applying inferential powers.

The reader might have noted that the examples of anaphora and the disambiguation of polysemous words involve the use of information from a sentence in the text, but not the one in which what might be called the problem signal occurs. This transmission of information across sentential boundaries is not an essential feature of these phenomena, but is, nevertheless, a rather thorny problem. It is, to some extent, a property of my definition of what is manifest about the meaning of a piece of text. Consider:

(12) John and Mary went to the zoo. They saw an elephant.

If one removes from my definition of manifest meaning the assumption that one considers the whole sentence in isolation several problems emerge. The computer system of [Wilks75b] would probably be able to determine the referent of "they" by using the information from the dictionary entry for "they" that it seeks a plural referent, and the only available plural object is "John and Mary". To do so it would only use what I have called word sense combination rules. But, for example, another computer system, that of [Cater82] would not deduce the referent at the stage where only the lexicon, the grammar and the the word sense combination rules (as they are represented in his system) are active. The system of [Wilks75b] does not recognise

---

[3]This example is also taken from [Lehnert79].

sentential boundaries as important (at least in comparison to clause boundaries) and it is from here that the difference springs. However the system of [Wilks75a] uses rules rules called Common Sense Inference Rules which one would undoubtedly class as inference rules, and I believe this would indicate that Wilks would broadly accept my conception of manifest meaning, but that he might argue about the size of text fragment one should consider to be the unit of manifest meaning. I think it is safe to assume that there exist examples which illustrate the same points whether one considers the largest unit of text which has manifest meaning to be the clause, paragraph or even a multi-paragraph unit.

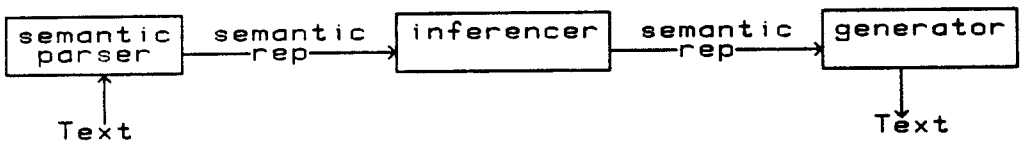
I hope this section has sufficiently emphasized that there are a number of commonly occurring phenomena of natural language which can only be dealt with satisfactorily if the language user has inferential abilities. That is, to handle them one does not only need to know the manifest meanings of the sentences but also their inferred meanings. I do not pretend that this is a complete survey; it is merely an attempt to show that some such phenomena exist, and list a few of the more commonly occurring ones.

The next section describes a model of a computer system which reflects the divisions have been made between manifest meaning and inferred meaning in this section.

2.2 An Idealised Model of a Natural Language Processing System

This section presents a simple but convenient model for the structure of a computer system which reflects the distinction I made in the previous section between manifest and inferred meaning.

The model is divided into three parts, a semantic parser, a generator and an inferencer[4]. For the moment I will ignore any objections to this model, and any alternative models, whatever their merits. Diagrammatically the three components are connected as follows:



---

[4]This is now the standard word in the literature, repulsive though some may find it.

The semantic parser maps the sentences of an incoming text into some form of semantic representation language. There are several assumptions I make about the semantic representations produced by the semantic parser. First, I assume distinct structures in the semantic representation language are produced for each meaning of an input sentence. (This, of course, also implies that some strong assumptions about the nature of meanings of sentences must be made) . Secondly, I assume the structures produced to represent different sentences with the same underlying meaning are identical. Thirdly, I assume the semantic representation language is well defined and suitable for manipulation by computer.

I assume the semantic parser employs a dictionary, a grammar and a set of word sense selection rules to produce these semantic representations. Hence the representational structures it provides will characterize the set of manifest meanings a sentence has. In particular, note that I assume that the semantic representations output by the parser will reflect any problem signals, in the sense of section 2.1, which occurred in an input sentence. The parser has too little information to resolve the problems they indicate.

The generator takes semantic representations, similar to those produced by the semantic parser, as input, and converts them into natural language. I assume that its operation is essentially the inverse of that of the semantic parser.

When the system is reading a piece of text the inferencer will take the representation produced by the parser and attempt to clarify it, for example selecting among possible readings identified by the parser and filling in elliptical and anaphoric references. In general, to do this it would require a combination of abilities to keep track of textual context, to manipulate general knowledge about how the world works, and to perform various kinds of inference. I also assume the role of the inferencer is to perform such other manipulation as is necessary to allow the system to respond appropriately to an incoming text. The appropraieness of the response would probably be determined by the success of the system at performing some given task. Such a task might be the production of other linguistic outputs, for example, answering questions about the original text, or translating it into another language, or actions, like moving toy blocks around a table (as was simulated by the program of [Winograd72]).

Before turning to some previous attempts to build or design computer programs with the abilities of such an inferencer, I will introduce two terms describing the distinction between the semantic parser and generator on the one hand and the inferencer on the other. I think it is clear that the dictionaries, grammars and sense combination rules, which determine whether a sentence is likely to be meaningful (given it is not placed in some bizarre context) are sources of information about language. Whilst one might argue that since the inferencer is an integral part of language processing the information it

uses is also information about language, it is more difficult to make this case for, say, the information required to answer questions (8)-(11) about text (7) in section 2.1. Thus a useful, but crude, working distinction may drawn between ***linguistic knowledge*** which is the sort of information manipulated by the parser and generator, and ***world knowledge***, which is the sort of information manipulated by inferencer.

2.3 Two Styles of Inference

One useful way to classify programs to do the job of the inferencer in the above model is to categorise them as ***explanatory*** or ***predictive*** in nature.

I classify as explanatory any inferencer whose fundamental mode of operation is to take a new input semantic representation and attempt to integrate it with what has gone before in the text. That is, it essentially provides an explanation of how an input sentence relates to previous parts of the text.

I classify as predictive any inferencer whose fundamental mode of operation is to take the text so far and analyse it to produce expectations about the subsequent inputs. That is, from one part of the text it attempts to predict the content of the rest.

In practice, predictive inferencers have generally used large units of information about specific text topics to set up their predictions, and explanation-based inferencers have exploited information about more general situations, organising the information into smaller units. But, of course, one can imagine the construction of a predictive inference system based on small information units, and conversely an explanatory inference system based on large information units.

Examples of inference programs which are fundamentally explanatory are [Rieger75] and [Cater81]. Programs which are fundamentally predictive include [Cullingford78] and [DeJong79]. A system which falls into both classes is that of [Wilensky78], since, whilst the program attempts to find explanations for new input, the way earlier inputs have been explained can significantly affect the way later inputs are processed, thus introducing an element of prediction. This is also true of the unimplemented system of [Charniak72].

In general explanatory inference programs seem to offer greater potential coverage whilst predictive inference programs offer at least the hope of greater efficiency. Explanatory programs are unlikely to be thrown by an unexpected turn in the text, since they tend to have available, at worst, knowledge which is always true, and this may be used to provide some sort of explanation for practically any input. On the other hand in order to achieve this explanatory programs must of necessity work very hard to explain the perfectly ordinary.

Predictive inference systems offer out the hope that perfectly ordinary material may be processed very rapidly: recognising that some utterance belongs to the set of utterances that are usual in a given context is probably more computationally efficient than trying to explain *why* they are usual.

It seems likely that future very powerful inference systems for natural language processing will combine both predictive and explanatory inference. Clearly, systems which combine both forms of inference system cannot be built until the technology necessary for the construction of both explanatory and predictive inference systems is available. The necessary technology is either simply unavailable or is, as yet, too new to have undergone a thorough explorations of its strengths and limitations. Much of the remainder of this thesis constitutes an attempt to examine and improve the technology for the construction of predictive inference systems. This concentration on predictive inference is done only in the interests of setting reasonable boundaries to the current project. It does not imply any fundamental commitment to one form of inference system rather than the other.

2.4 Scripts and Frames

This section briefly summarises two more or less contemporaneous papers. One, [Minsky74], concentrates on the proposal that information manipulated by inference systems should be organised into large units, around particular topics in the case of text. The other, [Schank75], primarily proposes techniques for the use of expectation in inference, but at the same time suggests the use of large, topic-specific, knowledge structures. The predictive inference systems which will be discussed in subsequent parts of this chapter are all, to one extent or another, descendants of these two papers.

The essence of Minsky's idea is contained in the very first paragraph of the paper:

> "It seems to me that the ingredients of most theories both in Artificial Intelligence and Psychology have been on the whole too minute, local and unstructured to account either practically or phenomenologically -- for the effectiveness of common-sense thought. The 'chunks' of reasoning, language memory and perception ought to be larger and more structured; their factual and procedural contents must be more intimately connected in order to explain the apparent power and speed of mental activities." (page 1)

He then goes on to propose a class of structures for representing stereotyped situations called which he calls ***frames***. Frames are networks of nodes and relations the highest level of which are things which are always true about the situation, the lowest level being ***slots*** which are filled by the details of a particular instance of the situation. He envisages much of the power of the theory as coming from the

inclusion of expectations and presumptions within each frame. He also proposes that the frames should be linked by an ***information retrieval network*** which is responsible for proposing new frames when an existing one appears inappropriate, and that slots should be filled by a matching process controlled by constraints from within the frame.

However, it should be emphasized that much of [Minsky74] is vague and open to many different interpretations.

Schank and Abelson in [Schank75] propose the use for computer natural language understanding of scripts representing such things as the normal course of events when eating a meal in restaurant, or when travelling by subway or by bus. Schank and Abelson justify the use of such large complexes of information by considering the exploration of the expectations set up by reading a sentence. The paper then says:

> "For our purposes a script is a predetermined, stereotyped, sequence of actions that define a well-known situation. A script is in effect a very boring little story."

Elsewhere, in [Schank78], a retrospective and rather different justification of the script approach is given. It begins with a discussion of the proposal of ***causal chains*** as a mechanism for linking together the propositions in a multi-sentence text. Causal chains are a combination of original propositions (that is those which have been derived directly from the input text) and inferred propositions linked by bonds describing their relationship in terms of:

a) one proposition being the reason for another;
b) one being the result of another;
c) one enabling another to take place;
d) a proposition stimulating another which is an idea.

Schank notes that not all texts can be linked together in this way and suggests that the addition of scripts, seen as "pre-packaged sequences of causal chains" (page 28), is the answer. His view is that scripts are appealed to by the author in situations in which he can assume that the reader has invoked via a script a pre-packaged causal chain which is accessible merely by allusion.

Neither Schank and Abelson nor Minsky are very specific about the information which should be contained in a script or frame. However in [Charniak75b] there is a fairly detailed, if informal example of what Charniak calls a supermarket frame. It should be noted that this is precisely the kind of structure called a script in the terminology of [Schank75].

```
Goal: SHOPPER owns PURCHASE-ITEMS
SHOPPER decide if to use basket,
    If So Set Up *cart-carry Fi
    SHOPPER obtain BASKET *cart-carry
    SHOPPER obtain PURCHASE-ITEMS
     |   Method-suggested
     \-> Do for all ITEM in PURCHASE-ITEMS
            SHOPPER choose
                 ITEM in PURCHASE-ITEMS - COLLECTED
            SHOPPER at ITEM
                Side-condition COLLECTED at ITEM also
                |    Method-suggested
                \->  *cart-carry (SHOPPER,
                                  BASKET,
                                  COLLECTED,
                                  ITEM)
            SHOPPER hold ITEM
            ITEM in BASKET *cart-carry
            COLLECTED := COLLECT + ITEM
        End
    SHOPPER at CHECK-OUT-COUNTER
    SHOPPER pay for PURCHASE-ITEMS
    SHOPPER leave SUPERMARKET
```

Charniak's supermarket frame

The items all in upper-case, like 'SHOPPER' are ***frame variables***, the items in all lower case, like 'at' are constants which in an actual implementation would be expressed in some suitable semantic representation language. Those items with initial capitals, like 'Do' indicate the flow of activity through the frame. 'Cart-carry' is a sub-frame, as indicated by the *, which describes the normal process of picking up goods in a supermarket in the approved manner. The frame variables are assumed to have associated with them descriptions of the appropriate kinds of entities which could fill them (SHOPPERS are adult humans, PURCHASE-ITEMS are things one might buy at a supermarket, and so on). Note also that the order of the statements in the frame indicates the temporal order in which they must occur.

If one ignores its lack of detail, this is a reasonable general description of what usually goes on in shopping in a supermarket, and is precisely the kind of information frames were designed to represent.

As indicated by this example there is a great deal of confusion in the literature about the use of the words "script" and "frame". In general those who have followed Minsky most closely have used the word "frame" to subsume "script" (see, for example, [Bobrow77a] and [Bobrow77b]). Schank and his colleagues have never used the word "frame" to refer to any of their information structures, and have generally only used the word "script" to refer to pre-packaged descriptions of stereotypical sequences of actions and states, although some of the so-called sketchy scripts described in [DeJong79] are perhaps better described as pre-packaged prototypical text representations. In the remainder of this chapter I will use the names used by the authors of the documents under discussion at any

particular point. The reader should note that this means the terminology will reflect the general inconsistency in the use of the word "script" and "frame" in the field.

In the latter part of the thesis I will, however, confine the use of the word "script" to action oriented stereotyped descriptions, and the word frame to representations of more static situations.

2.5 Some Early Attempts to Implement Script- and Frame-based Systems

Before the detailed discussion of some of the projects which influenced Scrabble, I want to draw out some distinctions between different views of the ideas in [Minsky74] and [Schank75] as reflected by subsequent work.

There are two primary lines which have been followed from these two sources. The first I will call ***structured retrieval networks***, the second I will call ***predictive processors***. By structured retrieval networks I mean those systems which emphasize the use of frames to organise atomic facts into clumps about single topics in order to ease the problems of retrieving the facts relevant to a given stereotypical situation. By ***predictive processors*** I mean those systems which emphasize the use of frame-like representations to set up expectations about what will occur after some given event has occurred. At least until 1979, within the natural language processing community predictive processing was associated exclusively with Schank and his collaborators, whilst structured retrieval networks were associated with [Fahlman79] in particular, although many whose main precursors were the workers on semantic networks such as [Quillian68], for example [Hendrix79], were undoubtedly influenced by the views expressed in [Minsky74], if only indirectly.

As was pointed out in section 2.3, the focus of this thesis is on the technology for predictive systems. Although structured retrieval networks come from the same stable, as it were, they are not predictive inference systems in any sense. They have been included here only for the sake of completeness when discussing frame and script based computer systems, and there will be no significant further discussion of them.

However, predictive processing systems require sophisticated means of retrieving suitable predictions given the context, and the natural organisation for the predictions is to clump predictions about the same topic together for retrieval purposes. Thus predictive processing systems require subsystems with similar properties to the structured information retrieval network systems.

In the discussion below I will concentrate on three major attempts to exploit frame/script technology. The first two, [Cullingford78] and [DeJong79] are are discussed in detail because they are the largest scale predictive processing systems constructed to date. The third attempt, [Charniak79], although not unambiguosly a predictive processing system is discussed in detail because it has had one or two significant influences on the Scrabble project.

Before turning to these projects two noticeable absentees should be mentioned. They are the GUS and KRL-0 projects ([Bobrow77a] and [Bobrow77b]) carried out at XEROX PARC and the PAL project of [Sidner79]. Both GUS and PAL are systems which carry out a dialogue with a human user, and this is one of the reasons for their rather scant treatment. It is often rather difficult to relate results from systems which can interject their own responses into the stream of incoming natural language to the needs of text processing systems which cannot indulge in such interaction. Secondly, both KRL-0 and the FRL system which underlies PAL are primarily information retrieval systems and thus are difficult to compare to Scrabble. The most relevant work done with the KRL-0 system was the construction of a model of SAM, and so does not call for independent discussion here.

2.6 SAM - Cullingford

Probably the first attempt to build a script- or frame-based natural language understanding system was the SAM system ([Cullingford78], [Schank75c], [Schank77]). It was a script-based system, and was built around the notion of prediction. The complete SAM system was the product of one the most ambitious natural language processing projects undertaken to date. The complete system was capable of reading English texts and summarising them, paraphrasing them, answering questions about them or translating them into a variety of other languages. Of course, it could not do any of these tasks perfectly by any means, and the texts it could handle were short and concerned a limited range of topics.

It ran as a six module system, the modules being:

1) an English to Conceptual Dependency parser called ELI (based on [Riesbeck75]),

2) the PP-MEMORY, which is probably best described as an inference system specialising in the analysis of nominals,

3) SAM itself, the script application program,

4) a question answering system, QUALM [Lehnert78],

5) a summarising and paraphrase system, and

6) a generation system capable of producing output in various different natural languages.

A distinctive feature of SAM is that the various modules all intercommunicate using a semantic primitive-based representation scheme called Conceptual Dependency [Schank75b]. Very briefly, Conceptual Dependency is a semantic primitive based representation system which combines the use of a small number of primitives, primarily oriented towards the representation of the meanings of verbs and adjectives, with a structuring scheme organised around a small set of case labels.

In outline, when processing a text control passes between SAM's six modules as follows. First, an input sentence is mapped into its Conceptual Dependency representation by ELI. Next, the Conceptual Dependency structure produced by ELI is processed by the PP-memory, which does some simple reference identification for anaphora markers in the Conceptual Dependency structure, and also assigns unique tokens for any nominal representations in the structure.

Next control passes to SAM proper, the script applier. The script applier is itself a very complicated program. Crudely, the script applier attempts to match the expanded Conceptual Dependency structure output by the PP-memory against the Conceptual Dependency structures it expects. These expected structures fall into two categories. First are those which indicate that the input sentence may require a new script for its analysis. Second are those which represent expected continuations of an input text which has already been partly analysed by some specific script. If the input Conceptual Dependency structure falls into the first category a new script will be loaded, setting up new expectations in the second category. Whichever category the input structure falls into, the precise way the input material matches the loaded script's expectations will be recorded.

Once the matching process has been completed, control is passed back to ELI, and the process is repeated for the next sentence in the input.

When the whole text is read the story representation is passed over for processing by the other the question answering system or the summarising and paraphrasing system, which use the generation system to produce their results in any one of a number of different languages. The story representation is formed by the records of the way the input material as a whole matched the expectations of the scripts used to analyse it.

The focus of this section is process by which the text is analysed, so no detailed discussion of the modules which operate on the story representation will be presented here. The summarising system will be briefly discussed in section 2.10.

One of the problems of describing the SAM system is that the authors are attempting to make some fairly strong psychological claims whilst not presenting evidence for their claims by experiment in the usual manner for psychology. Rather, they make a variety of intuitive appeals on the basis of the workings of their programs. This position tends to make accounts of the programs rather confusing, especially if one is attempting to examine the programs as programs and not as psychological models.

The discussion presented here focuses on the SAM program itself: there will be little mention of the other modules which support the script application mechanism in the complete SAM system.

SAM has a number of problem features which would present serious difficulties if any attempt were made to extend SAM: there are six problems features considered here, as follows:

1) the use of so-called causal chains,

2) the distinction between different sorts of variable items in scripts,

3) the possible interrelations between scripts,

4) the mechanism used to activate scripts,

5) the use of non-script-based inference, and

6) the representation of nominal-type objects used in the system.

Causal Chains

Causal chains link the various expectations of a script. It is not clear how they increase the efficiency of the system, or extend the possible range of language behaviour the system can handle. For example, to analyse the physical movement of an object in terms of small movements and the locations of the object between them, as seems to be demanded by the causal chain mechanism, leads, so far as I can see, most naturally to an infinite regress, and at the same time does not give any better expectations about the utterances which are likely to occur in a text for which a given script is appropriate. I do not consider:

(13) We went from Liverpool Street to Kings Cross by tube. First we were at Moorgate, then we were at the Barbican, then we were at Farringdon.

to be the most obvious adult description of a trip by underground. Indeed, if the script applier were merely to absorb the second sentence as an expected utterance about a tube trip I think it would be likely to be in error. It seems to me that the system should make

some sort of attempt to analyse this sentence in terms of why the author would write it, and to classify it as an expected utterance about a tube trip seems likely to obscure this view. Note that at the same time the events it describes are perfectly ordinary things to occur on the London underground journey from Liverpool Street to Kings Cross: it's just strange to talk about them, at least if the author is an experienced traveller. Of course, if the author was a child describing his or her first ride on the underground, such a text might be considered quite ordinary. However, the system should establish that the author *is* such a child before processing the text as an ordinary one about an underground journey. Equally I do not wish to imply that the text is in any sense illegitimate: it merely fails to follow the stereotype for such a text, even though the journey described does follow the stereotype for such journeys.

Variable Items in Scripts

Another problem feature of SAM is the separation of ***script variables*** and ***settings***. Settings are the physical locations in which what might be called parts of the action take place. Since some of the example script variables include, for instance, in the subway[5] script, such diverse things as the patron, a train and a strap, I cannot see the advantage to be gained by placing, for example, the inside of a subway car in a completely different class of objects (physical settings) associated with subways.

Interrelations between Scripts

Expectations in SAM are hierarchically organised. At the top-level there are ***situations*** associated with each knowledge domain with which SAM deals. Situations are, in effect, very high level scripts. They are divided into ***segments*** and each segment has a number of scripts which can fill it. For example, the situation $TRIP[6], organises all the reasons SAM knows for going somewhere, and all the ways of getting there and back. The hierarchy formed looks like this:

$TRIP			situation
$GOTRIP	$GOALTRIP	$RETURNTRIP	segments
$BUS	$RESTAURANT	$BUS	scripts
$TRAIN	$MUSEUM	$TRAIN	
$DRIVE	$SUPERMARKET	$DRIVE	
$SUBWAY		$SUBWAY	

---

[5]In the American sense.

[6]The objects prefixed by "$" are the names of scripts and script-like objects.

Additionally, scripts are divided into ***scenes***, which are elements of the action which occur in a particular place, and ***tracks***, which are variants on the same scriptal theme. For example, "self service" and "table service" are two of the tracks in the restaurant script. It seems to me that there are no practical functional distinctions between the various levels of this hierarchy, except perhaps in determining whether a script or an ordinary Conceptual Dependency structure can fulfil a particular expectation.

This hierarchy effectively defines the possible interrelations between scripts: scripts may only be related to one another by going up the hierarchy of scripts and then down again. Since the hierarchy becomes more general the further up one goes, it is impossible to make the detailed expectations of one script interact with and support the detailed expectations of another. In particular, the possibility of two low-level detailed scripts being simultaneously active and used to analyse the same piece of text is not considered. This makes it impossible to analyse text like:

(14) John and Mary went to the zoo. After they had seen the monkeys they felt hungry, so they went to the restaurant. They could see the lions and tigers from their table.

correctly, that is without recognising "They could see the lions and tigers from their table" as being unexpected. It can only be expected if the zoo and restaurant scripts are allowed to interact directly, which Cullingford's scheme does not allow for.

Script Activation

A further problem feature of SAM is the technique used to activate scripts to see whether they apply to a particular text segment. It appears from [Cullingford78] that the essential mode of script activation is as follows: each script[7] has a set of patterns, called ***headers***, which describe the Conceptual Dependency representations of the utterances one might expect to begin a text segment for which the script is appropriate; all scripts (and therefore, I infer, all script headers) are arranged in a particular order, with the most widely applicable (that is to say, situations) first. When a Conceptual Dependency structure enters the system it is matched against each header in turn. If a match is found the full set of expectations associated with the script is added to the list of script header patterns, as are those associated with scripts higher in the hierarchy. The list is then ordered so that the expectations associated with a script one or more of whose headers have been matched are searched before any script at the same level of the hierarchy, but

---

[7]In the rest of this section the word "script" should be taken as subsuming script, situation or situation segment.

after any further up the hierarchy.

There is a very serious problem with this algorithm. Consider an utterance whose Conceptual Dependency structure neither matches the expectations of any active script, nor the headers of any other. Before it is determined that this structure is unexpected, in the sense that the SAM system has no means of dealing with it, all the expectations of the currently active scripts, and more to the point, all the headers of *all* the scripts in the system must be examined. Quite a simple script, for subway journeys, has four headers, so one is talking in terms of examining at least four times as many patterns as the system possesses scripts to determine that the system cannot deal with an utterance. The problem is further exacerbated by the use of various inference rules (described in more detail later) to expand the range of Conceptual Dependency structures a given pattern may match. Such a heavy burden, when attempting to deal with an utterance with which the script system is in principle unable to deal considerably lessens the apparent advantages of using scripts in language analysis.

Non-script-based Inference

The SAM system also has a set of processes which attempt to recognise input Conceptual Dependency structures which, whilst they do not correspond exactly to a pattern the system was expecting, are nevertheless functionally equivalent to some pattern the script is expecting. There are nine classes of inference rule discussed by Cullingford. They include Immediate-Result Inferences, which, for example, allow a pattern which expects a description of an object being physically moved from point A to point B to match an utterance which states that the object is at point B. Another example is Transitivity of Proximity which allows a pattern expecting A to be near B to match a statement that B is near A.[8]

There are two shortcomings in the Cullingford's discussion. First, whilst it is mentioned that these processes are intimately associated with patterns, no discussion is made of *when* these inference rules are applied. There are at least two possible places the rules may be applied. The first is immediately a pattern is tried but fails, the

---

[8]The rules are rather more complicated than I have indicated. I do not want to imply that the rules are completely naive.

[9]Cullingford mentions this list without actually describing how it is managed in any detail. [DeJong79], in his discussion of SAM, describes a search list, which seems to correspond to the high-priority search list of Cullingford, and describes briefly how script headers are added to it, on the basis of predictions from active scripts and the mention of what might be called suggestive objects in the text. An example of the latter would be that mention of an ambulance would cause the addition of the hospital script. The source of this information is not clear.

second is at the end of the high-priority search list[9] . A third possibility is at the end of all the patterns, but as I mentioned earlier, there are likely to be a large number of script headers available in any practical system, thus making this approach unattractive. However, the process of extended matching is likely to further aggravate the system's problems with dealing with an utterance which can neither act as a header for one of the systems scripts nor does it match the expectations of the currently active scripts.

This difficulty with inferencing might be bypassed if the "discrimination net which decides which inferences, if any, should be made"[10] could very rapidly determine whether a match could be synthesised. However, it seems rather unlikely (to say the least) that there exists an algorithm to do this which is faster than applying rules and testing for a match directly.

Cullingford acknowledges that there are many texts which SAM could not analyse even in principle. The point here is: what is appropriate behaviour if the system is presented with such a text, as will inevitably happen if SAM were used practically. It seems clear that, whatever else it should do, it should not use large amounts of computational resources on efforts to apply scripts to a text when such efforts are bound to end in failure.

The Representation of Nominal Concepts

My final point concerns the representation of nominals used in the SAM system. Consider the example of what is referred to as the PP[11] class PERSON on page 165 of [Cullingford78]:

---

[10][Cullingford78], page 120.

[11]Picture Producer: the Conceptual Dependency name for nominal-type concepts.

PERSON:

PERSNAME	personal name
SURNAME	family name
AGE	age
RESIDENCE	the place where this person lives
GENDER	sex
TITLE	a title such as "Mr." or "Premier"
OCCUPATION	a long term occupation such as "medical doctor"
EMPLOYER	the organisation the person works for
FUNCTION	a short-term function such as driver
POLITY	a political unit the person serves in an official capacity
FAMILY	a pointer to the persons family group
HUSBAND/WIFE/SON etc.	a pointer to another person related to this one

It seems to me that, for example, a political unit a person serves in an official capacity is an unlikely candidate for a general purpose representation of facts about people. However, it does seem a perfectly reasonable thing to include in a role filler specification of a script about official visits. I see this representation of nominal concepts as a mistaken attempt to remove overmuch of the analysis of nominals into the non-scriptal part of the system.

Summary

To summarise, there are a number of serious problems with SAM. Amongst the most serious are the lack of a tolerably expensive script activation algorithm; restrictions on the ways scripts may interact which cause unnecessary difficulties when processing certain texts; and the use and control of non-scriptal inference.

2.7 FRUMP - DeJong

The FRUMP system of [DeJong79] is rather different from the other systems described in this chapter. It has a parser as an integral part of the script application and inference mechanism so it does not fall into the three box model of section 2.2. Its other distinctive property is that it does not attempt to analyse the whole of the text presented to it, it only skims (to use DeJong's word) its input.

In outline the operation of FRUMP is as follows. The system begins by scanning the input story left-to-right for a so-called structure building word, typically the main verb of the first sentence. The structure associated with the word, which will be a skeletal Conceptual Dependency structure, is processed using a data-structure

called a Sketch Script Initiator Discrimination Tree. This structure contains predictions about the possible form of the input and depending on the match between the words of the input and these predictions one of the systems Sketch Scripts will be selected, provided, of course, the system has a Sketchy Script which is appropriate for the analysis of the text. Once such a Sketchy Script has been activated it is used to provide predictions of the form of the remainder of the text. The predictions are compared with the input text by the Substantiator, a combined parser and general inference mechanism, which uses crude heuristics to try to find input material which satisfies the Sketchy Scripts predictions. The system simply skips over parts of the text which do not satisfy its predictions. Although the system is crude in many ways, and often misanalyses input texts (for example, by using inappropriate Sketchy Scripts to analyse them), FRUMP analysed many more different texts than any previous natural language understanding system, and operated with great computational efficiency.

I want to concentrate on FRUMP as a script applier so I will not dwell on it as a parser. However, it would be quite impossible to pass over this aspect of the system because it undoubtedly profoundly influenced the design of the system as a whole. DeJong emphasises the advantages of his system in terms of robustness, that is, its ability to understand, or at least deal with, texts for which it has inadequate vocabulary, or which are ill-formed in some way. It is not clear to me that a system organised around the three box model of section 2.2would do a great deal worse than FRUMP provided that it had a well developed vocabulary and it had some means of error recovery[12]. A stronger argument for this architecture, which DeJong points out, is that the inferencer's[13] understanding of context and the expectations and preferences associated with it can be used at the lowest level of language processing, for example to prefer particular word senses during text analysis. Though the communication of such information between the parser and inferencer is acknowledged to be important, no system prior to FRUMP had possessed a natural means of passing it over. Thus the SAM system of [Cullingford78] and [Schank75c] actually modified the semantic parsers initial dictionary entries, whilst processing the text, in order to express preferences based on high-level pragmatic information.

The main issues FRUMP deals with as a script applier are: script selection, the role of inference when attempting to satisfy predictions, and what I will call the problem of hallucinated matches. They will be discussed in this order.

---

[12]In any case [Granger77] has given some indications of how this might be done for missing vocabulary in such a system.

[13]In the context of FRUMP, the PREDICTOR.

Script Selection

DeJong describes three different script activation techniques which are used in FRUMP. The first of these is Explicit Reference Activation, which appears to correspond to activation on the mention of a relevant key word in the text. Unfortunately there is no clear, technical, example of this form of activation, and it is not obvious how it differs from Event Induced Activation, to which I will come in a moment. The second is Implicit Reference Activation, in which the activation of one script is caused by the activation of another. For example, mention of a major earthquake causes the activation of a script to deal with text about corresponding relief operations. Finally there is Event Induced Activation.

To describe this last form of activation it is necessary to spend a few moments reviewing in more detail how the text analysis in FRUMP works initially. If there is no current context, which means here that there are no active scripts, the input text is scanned from left to right until what is known as a ***structure building*** word is found. A structure building words appears to be any word which has a word sense associated with a particular Conceptual Dependency act or state. The main verb in the sentence is almost always a structure building word, but many nouns, in particular nouns which describe actions, such as "earthquake", and "riot", will also have this property. This structure selection operation appears to me to be the only bottom-up operation in the whole system, but DeJong rather hedges this point in his conclusion.

The skeletal Conceptual Dependency structure associated with the structure building word is then passed to the so-called Sketchy Script Initiator Discriminator Tree (SSIDT), whose nodes are tests and whose leaves are the names of scripts to be activated should all the tests above them be true. The first set of tests to be applied determine whether the structure built is a STATE or an EVENT, the next set determines which Conceptual Dependency primitive the structure contains, and lower tests examine particular role fillers. If there is insufficient information available to satisfy a test a prediction that it is satisfied is made. The SUBSTANTIATOR then attempts to confirm or deny this hypothesis by using top-down text parsing and inference as it does for predictions from any source. The tree is traversed on the basis of the results of the tests, and eventually either the system comes to a leaf indicating that a particular script should be activated, or to a point at which it is clear no script can be associated with the text segment. In the latter case the system resumes scanning the text from left to right until it finds another structure building word.

There are, however, some technical problems with this approach. DeJong says that information gained during the process of selection should be available to the initiated script. Precisely what information is retained and how it is passed to the process which initiates scripts is not made entirely clear.

Consider:

(15) Tokyo was shaken yesterday by an earthquake which measured 6.3 on the Richter scale.

It appears that to deal with this, FRUMP will select the earthquake script on the basis of "Tokyo was shaken by an earthquake". To do so it must determine that Tokyo is a piece of ground[14]. This is not necessarily a cheap operation as it essentially involves an inferential process.

Let us assume, for the moment, that in order to save storing redundant information the dictionary entry for "Tokyo" is only marked with a feature like CITY. Then, if one wanted to determine whether or not Tokyo is a piece of ground, inference rules might have to be applied which could deduce that usually cities are polities, and, in the absence of information to the contrary, polities are associated with pieces of ground.

DeJong says that any partial Conceptual Dependency structure built during the traversal of the SSIDT is retained. But there is probably no need to retain the fact that Tokyo is a polity if the earthquake script is activated. Equally if, say, an election script is activated, there is probably no need to store the fact that Tokyo is a piece of ground. Thus, it seems, to deal with quite a straightforward example, DeJong must either carry around information which might not be needed by the script activated; abandon his principle that conceptual role fillers should only be examined once; or mark in the SSIDT nodes which information should be retained if a given script is activated. None of these seem satisfactory.

It should be pointed out that the Sketchy Script Discriminator Tree technique is essentially equivalent to the script selection method of [Cullingford78], but replaces the linear search used there by a fast parallel search which minimises computation. DeJong acknowledges this. Though DeJong's approach to the selecting scripts on the basis of incoming text or its representation has a number of drawbacks, it constitutes an answer to many of my criticisms of [Cullingford78]. In particular, it has the property that it can rapidly determine that no script can be initiated from a given structure, if this is the case.

There is another aspect of SSIDT which I want to bring out. I think it is rather doubtful whether the particular tests around which they are organised are as useful as they could be. DeJong points out that they are fundamentally associated with a primitive decomposition of word meanings. However, he does not discuss the advantages and disadvantages the particular scheme he uses, namely Conceptual

---

[14]See [DeJong79] figure 3.1. Note that the word "earthquake" will only be seen by the system after it has determined that it is the ground which is shaking, as it tries to determine whether the cause is an explosion or an earthquake.

Dependency. Given that the intention of SSIDTs is to find a script which is likely to apply to a text segment with the minimum of computation, it seems rather strange to use as one's highest level indices the primitives of a representation system primarily fitted to the discrimination of the meaning of verbs and adjectives; or, at least, actions and states of nominal concepts. For example, DeJong discusses in detail the script initiation process for the sentence:

(16) Israel has sent troops into Lebanon.

It seems to me that the word "troops" is a much better place to start looking for scripts to initiate than "sent", De Jong's departure point. Its meanings are much more specific than sent and it has a sense very closely associated with the script eventually initiated, $FIGHTING.

However given the processing cycle of FRUMP, "sent" has virtues as a place to start the script initiation process. Its very generality means that there is almost bound to be some (probably large) set of predictions which can be made as to the possible surrounding words. That is to say, it is a good place to start analysing the sentence in which it occurs. But to say this is to make FRUMP look much more like many other more conventional semantic parsers which are based, implicitly or explicitly, on the notion that the verb is central to the clause. These include [Riesbeck75], [Wilks75b], [Boguraev79] and [Cater80].

Taking these two features into account, the script initiation technique described in [DeJong79] is rather ambivalent between the demands of accurate script selection with the minimum of computation on the one hand, and the attempt to process text without a conventional semantic parser on the other.

Non-scriptal Inference

The role of non-scriptal inference in FRUMP is also problematical. As in SAM, if a match cannot be made between a script's predictions and the text, a rule-based inference system may be invoked. In fact in FRUMP there is no firm distinction between the parsing of the text and the application of inference rules. Whether an inference rule or a parsing rule is applied in the attempt to satisfy a prediction is decided on the basis of the pre-specified cost of the application of the rule and the certainty of its result, should one be produced. In practice, it seems, parsing rules are normally applied before inference rules during the processing of a given text segment.

In FRUMP there are two distinct sorts of inference rule application process. One is carried out by the Conceptualization Inferencer, and the other by the Role Inferencer. Unfortunately, the discussion of the Conceptualization Inferencer in [DeJong79] is too brief and vague for useful comment.

The function of the Role Inferencer is to attempt to satisfy a prediction that a Conceptual Dependency role is filled with a particular kind of filler. Although the Role Inferencer has access to many rules, they are indexed by the conceptual primitives to which they apply (it appears such information is always available when the Role Inferencer is called), and the case role which they fill. This, claims DeJong, allows the rapid selection of rules which may be able to satisfy a prediction: typically these are few in number. Rules have associated with them a crude means of calculating the cost of their application, and FRUMP may avoid selecting a particular rule because it is too expensive. This is , in general, an attractive scheme for organising inference in this context. It has, however, one serious drawback. If FRUMP is presented with an unsatisfiable prediction, the cost of recognising it as such is very high compared with satisfying any satisfiable prediction. In particular, the total time spent in recognising it as unsatisfiable is not bounded: every rule, both parsing and inference, which might possibly satisfy it and which has a cost estimate less than the current maximum permissible cost will be applied in the attempt to satisfy it. This is a product of limiting the maximum cost for a given rule. A technique which limited the maximum total cost spent attempting to infer a given proposition, like that proposed in [Norman75], would not have such a defect. DeJong's argument that such a scheme would make it difficult to adjust the maximum permissible cost is unconvincing.

Hallucinated Matches

I now want to turn to what I will call the hallucination of matches. I mean by hallucinated matches the misassignment of a text to a script and the subsequent misanalysis of the text on the basis of that assignment. The DeJong's examples contain several instances of FRUMP completely missing the point of a story. He is rather dismissive about them, pointing out that they are produced because FRUMP either lacks a script or because its vocabulary lacks a particular word sense. But both of these possibilities are important. DeJong admits that not all texts are scripty, even in the news story domain. Consequently there are bound to be some stories for which the system does not have an appropriate script. In addition, it is in the nature of language that it changes constantly, so no natural language system can have a complete, static lexicon. Now if DeJong did not claim that FRUMP was a robust system these would be unimportant problems. I cannot see that a system can be called robust when it produces complete misanalyses and does so without leaving any indication that it has had difficulty processing the text. It would probably be better if the system produced no analysis at all and indicated that it could not process the text.

I have called this problem the hallucination of matches because in effect what FRUMP has done is to see in the incoming text a text which fits its expectations, regardless of what the input text actually says.

This problem arises because FRUMP simply skims over sections of text which do not match its predictions. It thus effectively fails to process in any way things which are unusual in the text. The direct inclusion of material which was not expected by the system would at least give some indication in the summary that the text had been misanalysed, but at the price of losing the interlingual representation of summaries. It would also have the the effect of relegating the pragmatic prediction-based parsing section of the program from its central role in the extraction of summaries to the position of merely identifying the most straightforward parts of the text.

Before leaving FRUMP, one final point needs to be made. DeJong provides very little discussion of the generation of summaries. The output of FRUMP is a Conceptual Dependency structure, apparently generated by a Conceptual Dependency template plus the bindings of script variables during the analysis. This structure is then handed over to programs which can, apparently, convert it into any one of a number of different languages. However there is no discussion of how summaries are generated when more than one script has been used to analyse a text[15], or any one of a great many other complex cases which one can envisage occurring.

2.8 Ms Malaprop - Charniak

The Ms Malaprop program of [Charniak79], [Charniak78a], and [Charniak77], is not a full-blown natural language processing system. It takes as its input, and generates as its output, predicate-argument formal language expressions which Charniak asserts would be fairly straightforward to extract from and map into natural language. Thus it could form the central inference box of the three box idealised natural language processing system described in 2.2. However, it has been influential in the development of Scrabble in a number of ways, so some discussion of it is in order.

The system is based on a formal and quite well specified frame representation scheme. Unfortunately, almost all Charniak's discussion of it is based an a single example frame describing how to paint some everyday object, like a wall or a chair. All the example inputs (which are expressed in English in the papers) take the form of two or three sentence stories followed by a question and answer. Also there is no discussion of how one would select an appropriate frame for a text segment, nor what would happen if a statement in the text could not be integrated with any frame instance. All of this is quite deliberate

---

[15]De Jong does consider texts requiring the use of more than one Sketchy Script, but only texts requiring the use of Sketchy Scripts linked by so-called Issue Skeletons are discussed. These Issue Skeletons are equivalent to single sketchy scripts from the point of view of summary generation.

on the part of Charniak, but nevertheless makes it difficult to analyse Ms Malaprop in terms of its implications for building systems capable of seriously performing some language processing task.

Even so, there are a number of criticisms which may be legitimately levelled at this work.

[Charniak78a] dwells on the problem of capturing common-sense knowledge about painting in his frame representation. It is never made clear why the frame representation is more useful for this purpose than another representation scheme. In particular, there is no attempt to describe why starting with the frame representation gives better results than starting with the same information expressed in any other form of representation, for example an unclustered semantic network.

Secondly, it seems to me that Charniak takes for granted what [Wilks76] calls the Do It Hypothesis, that:

> "Computer understanding of language is essentially connected with our ability to perform certain physical actions: those discussed by the language. Computers only understand language about certain activities if they have performed, or are able to perform, them."

This hypothesis certainly remains unproven, so it is not at all clear that the kind of description of painting proposed by [Charniak78a] has a central place in a natural language understanding system, as Charniak seems to have believed.

However, disregarding the type of information Charniak chooses to express in his frame representation, there is one aspect of Charniak's frame representation which has been very influential on the design of Scrabble. This is the use of precisely one kind of frame variable which can have associated with it very complex descriptions of its required and likely properties: this idea is carried directly over to Scrabble, since it neatly deals with the problems I have associated with the script independent PP-memory of SAM described in section 2.6.

2.9 Some Conclusions

There seem to be three main conclusions to be drawn from an examination of previous attempts to build natural language processing systems whose inference components were based around strong expectations about the content of text about particular topics.

First, the use of general, non-prediction-based inference in such systems appears expensive and cumbersome in comparison with its advantages.

Second, none of the schemes used in previous systems to identify the topic of the text (and hence its associated expectations) seem entirely satisfactory.

Third, there are serious dangers in ignoring parts of the text during analysis.

2.10 A Review of Summary Production

This section is included here because the material it presents forms part of the background to Scrabble, despite the fact that it covers somewhat different ground to the rest of the chapter. The previous sections of the chapter has focused on the inferential analysis of text. This section concerns one of the uses which may be made of the products of such inferential analysis: summary production. Four projects will be considered here. Two are computer systems whose inferential analysis processors were described earlier in this chapter: they are Schank's group's SAM and DeJong's FRUMP. The third project was described in [Rumelhart75], describes an unimplemented scheme for story summarisation. The fourth project, described in [Lehnert81], is oriented towards the construction of a computer system, although there is, as yet, no detailed published account of the implementation of Lehnert's ideas.

The SAM system described in section 2.6 could be run in a summary generation mode, whilst the FRUMP of 2.7 was exclusively a summary generation system. Both systems operated as summarisers by associating a pre-specified outline with each of their script prototypes. To generate a summary for a particular input text, elements extracted from the input material whilst it was being read by the system were inserted into the outline summaries for the scripts used to analyse the input. The filled-in summary outlines (which were Conceptual Dependency structures in both cases) were strung together and then converted back into English, or one of the systems other output languages. The SAM system also included mechanisms for varying the length of the summary produced for a given story, by attaching procedures for specifying the relative interest of various items in the input story, so that less interesting items could be omitted if a shorter summary was required. For example, the system had the information that the number of people killed in a road accident is more important to include in the summary than the number of people injured. Thus if SAM was instructed to produce a short summary of a text about a road accident which included both a figure for the number of people killed and the number of people injured, in its summary it would include the number of dead in preference to the number of injured.

Although the attachment of pre-specified summaries to scripts is fairly simple-minded technique, the SAM and FRUMP systems showed that it worked well for quite a wide variety of texts. It forms the main background to the Scrabble summariser.

[Rumelhart75] proposes the use of story grammars to analyse texts to produce summaries. In essence Rumelhart proposes a sophisticated analysis of the supra-sentential syntactic and semantic form of an input text. This form of representation of the input facilitates the extraction of summaries by well-defined rules. In outline the theory is a good one and is, perhaps, capable of producing very good summaries indeed. However, serious problems are presented by the automatic extraction of the necessary analysis of the input text. It is not clear how to go about by producing a program to perform the necessary analysis. Therefore Rumelhart's proposals will not be pursued any further in this thesis.

[Lehnert81] proposes a system based on so-called plot units as the basis of text analysis for summary generation. It is a technique primarily orientated towards the analysis of text about human interactions and emotions, and is thus, in a sense, less general than the techniques used by SAM and FRUMP. However, within its subject domain the plot unit technique relies on less specific information than would be used by either SAM or FRUMP. However, it is primarily a representation device, rather than a inferential text analysis technique, and in fact Lehnert proposes the use of a predictive inference system for the extraction of plot units from input text. Thus in proposing a representation device which facilitates the production of summaries, rather than a technique for the extraction of the representation from an input text, it is somewhat to one side of the main concerns of this thesis.

The next chapter discusses various of the problems of previous predictive inference and summary generation systems brought out in this chapter, and introduces the solutions to them used in Scrabble.

3. An Overview of Scrabble

Chapter 2 presented a class of problems which must be faced by any computer system capable of performing any but the shallowest natural language processing. The class of problems discussed were those requiring inference. A model structure for a computer system to tackle these problems was then proposed. This was also used to delimit the role of inference in the system. Two strategies, prediction and explanation, were then introduced which could form the basis of a computer system to tackle the problem of inference in a natural language processing system. One strategy, prediction, was picked up for more detailed examination, and several previous attempts to build or design natural language processing based on predictive inference were discussed. Three conclusions about the shortcomings of these previous systems were reached. First, the use made of non-predictive inference was unsatisfactory. Second, the mechanisms for the identification of an input text's topic were inadequate. Third, techniques for dealing with unexpected material in the input by simply ignoring it were likely to lead to misanalysis of input texts. The chapter ended with a consideration of previous attempts to construct or design summary production systems.

This chapter is an introduction to the more technical parts of the thesis. Subsequent chapters will describe in detail how various parts of the Scrabble program work. Essentially this chapter describes *why* the program works the way it does.

It begins with a brief description of some technical terms which will be used and expanded on in the remainder of the thesis. This is followed by an outline description of the ideas behind Scrabble, as distilled from the problems of previous systems. The structure of the complete Scrabble ***system*** is then described. The description includes a brief account of of the two programs which perform the input and output functions of the system not performed by the Scrabble core itself. The three programs which form the Scrabble core, the stereotype manager, the stereotype applier, and the summariser are also discussed. Finally an annotated example of the Scrabble system in operation is presented.

3.1 Some Technical Terms

This section introduces some of the technical terms used to refer to various key concepts in Scrabble. Their interpretation will be expanded during this chapter, and, in fact, throughout the remainder of the thesis.

Two sorts of term are introduced here. One sort is oriented towards the descriptions of texts and their properties. The other sort

is oriented towards the representation of knowledge about texts.

The first two are text-oriented terms related to the notion of ***topic***, a term widely used in the linguistic and natural language processing literature, unfortunately with several different meanings and sets of implications. Two terms are used in this thesis to refer to aspects of the notion of topic:

Topic-type. The topic-type of a piece of text is its general subject. Although a text may have only one topic-type, many have more than one. Different segments of a text may have different topic-types. They may either be independent, or form a sub-topic-type in the context of the topic-type of a longer text segment.

Topic-manifestation. What a particular text segment says about its topic-type is a topic-manifestation.

Consider, for example, the following text:

(1) John and Mary went to the zoo. They saw the lions and tigers. They felt hungry so they went to the cafe and had a hamburger. They could see the zebras and giraffes from their table. After they had seen the elephant they left. They went home by train but it took a long time because there was a signal failure.

The topic-type of "John and Mary went to the zoo. They saw the lions and tigers." is a visit to the zoo. That it was John and Mary who visited the zoo, and that the particular wild animals they saw were lions and tigers are properties of the topic-manifestation. The text then moves to a new topic-type, which is a sub-topic-type of the visit to the zoo. This topic-type is eating at a cafe: it is subordinate to the visit to the zoo topic-type because one does not normally see zebras and giraffes from ones table. Again, who went to the cafe and the fact that they ate hamburgers are properties of the topic-manifestation. The text then returns to the topic-type of a visit to the zoo. Finally, an independent topic-type is introduced. "They went home by train but it took a long time because there was a signal failure" is only very weakly associated with a visit to a zoo: the topic-type here is a train journey.

The following terms are used to refer to what might loosely be called the knowledge structures of the Scrabble system.

Stereotype. A stereotype is an abstract representation of the expressions which might be expected to occur in texts which possess a given topic-type. I will refine this notion further below.

Script. A script is a stereotype whose topic-type is associated with some event sequence which possesses a strong temporal ordering and is action oriented.

Frame. A frame is a stereotype whose topic-type is associated with a static description of some kind. The expressions are not strongly temporally related, except, perhaps, they are all expected to be true more or less simultaneously.

In my terms the so-called supermarket frame of section 2.4 (from [Charniak76b]) is a script since its expressions occur in some particular order in time, as is the school-day frame of [Minsky74]. However, the birthday party frame of [Minsky74] is a frame in my terms, since it is a description of things one might expect to be true about (American) birthday parties, and gives no information about the order of events at them. This distinction is, of course, rather crude. The two terms really define two ends of a spectrum, along which lie topic-types which might be more or less temporally ordered. However, the two extremes provide useful reference points for the description of the behaviour of the Scrabble program when processing certain kinds of texts.

I have introduced the word stereotype to cover both the kinds of representations I have called scripts and frames in order to provide a subsuming term for other writers very varied uses of these terms, and also to emphasise an important distinction between my uses of the words script and frame and that of some others. [Schank75c], [Cullingford78] and [Charniak79] write of their frames and scripts as representations of events or objects in the ordinary real world. I make no such claims. The stereotypes Scrabble uses are intended to be abstract representations of the likely contents of a piece of text with a given topic-type. Of course, texts often discuss the real world, and human readers will often believe themselves to see a relationship between a text and some occurrence in, or state of, their real world. They might well exhibit this understanding by some form of language behaviour.

If Scrabble's stereotypes represent anything in the real world it is only through their capacity to allow the program to exhibit language behaviour apparently connected with it. They are at least one step removed from anything in the real world. They are objects associated with text processing.

In this sense Scrabble's stereotypes are related to story grammars of, for example, [Rumelhart75]. However, there are a number of important differences. Firstly, story grammars are usually considered to be general descriptions of stories, whereas stereotypes are associated with particular topic-types. Secondly, stereotypes provide a much more detailed set of norms about their texts (this is, of course, to some extent associated with their being topic-type-specific). Thirdly, stereotypes are more descriptions of the normal content for texts, whereas story grammars are more descriptions of *form*.

Finally I want to introduce some more precise terms related to the concepts of stereotype, script and frame, as they are exploited by the Scrabble program. The two terms reflect the distinction made between topic-type and topic-manifestation above, but relate to the knowledge structures manipulated by Scrabble rather than texts and their contents.

Expectation is my term for the word "prediction" in Chapter 2. It will be used to describe the predictions about texts' contents manipulated by the Scrabble program.

Stereotype-prototype. A stereotype-prototype, in Scrabble, is the fundamental representation of the expectations associated with a particular topic-type. A stereotype-prototype does not correspond to any particular text, and so is fairly abstract. When I introduced the term topic-type I used the rather vague phrase "general subject" to describe its intended meaning. Whilst I believe this is sufficient to convey how I would wish to distinguish different topic-types, rather more precision is required to produce stereotype-prototypes. In particular, one of the requirements implicit in the general strategy adopted is that the expectations should form a fairly small set. This, in turn, implies that when a stereotype-prototype for a particular topic-type is written, firm and inextensive boundaries have to be drawn around what is to be considered a normal or usual utterance about that topic-type. Ultimately this has meant that the practical definition of a topic-type within the Scrabble program is both closely associated with the expectations generated from a particular stereotype-prototype and is fairly narrow in scope. I will return to this later.

Stereotype-instance. A stereotype-instance is created when it is tentatively recognised that the program possesses an appropriate stereotype-prototype for the text. A stereotype-instance represents the system's knowledge of a particular topic-manifestation. Initially a stereotype-instance consists of just the set of expectations taken from the stereotype-prototype, but as processing of the text proceeds and some of the expectations are fulfilled, the remaining expectations will in general become more specific. This will occur, for example, if two of the expectations refer to the same object, if one of them is satisfied by some specifically referring object, like a proper name, the other will then expect the same name or a compatible object reference.

In story (17) above, the stereotype-prototype for a visit to a zoo expects, amongst other things, that the same people will arrive at the zoo as look at the animals. Hence, once the sentence "John and Mary went to the zoo" is input to the system, the newly created stereotype-instance for this visit to the zoo contains the specific expectation that "John and Mary" will look at animals, rather than the vague expectation that some people will look at animals.

The distinction made here between stereotype-prototypes and stereotype-instances is based on [Charniak75].

The process of forming a stereotype-instance from a stereotype-prototype is called ***stereotype-instantiation***.

The system distinguishes two sorts of expectation. The first kind are ***variable-expectations***. For example, some of the variable-expectations one would expect to find in stereotype-prototype about the topic-type of eating at a restaurant, are that the actor of ordering, eating and so on, will be human, that which is eaten will be food, etc. Variable-expectations also express the notion that, say, in a restaurant the people sat at a table usually eat the food, and not, for example, the person who serves it. In more text oriented terms, variable-expectations specify a set of co-referring nominals which are expected to occur in an input text and a set of features the nominals are expected to be able to possess.

Utterance-expectations are what the stereotype-prototype expects to be written about the variable-expectations. For instance, one expects to read that food is eaten in a normal text about eating in a restaurant.

The mechanism by which fulfilled expectations are propagated to cause other expectations to be made more specific is that when an input utterance satisfies an utterance-expectations it may also fulfil some variable-expectations. Other utterance-expectations which depend on these variable-expectations will then be further specified.

When an expectation of either sort is fulfilled a modified copy of the semantic representation for the utterance or the relevant fragment of it is associated with the expectation. A representation of the topic-manifestation is formed by these data-structures associated with satisfied predictions.

By analogy the terms ***script-prototype***, ***script-instance***, ***frame-prototype*** and ***frame-instance*** are used to describe stereotype-instances and stereotype-prototypes of the frame- and script-like kind.

To summarise the main technical terms introduced in this section are:

Topic-type	text-oriented term: the general topic of an input text
Topic-manifestation	text-oriented term: what a particular text segment says about its topic-type.
Stereotype-prototype	knowledge representation structure: the systems expectations about texts of a particular topic-type.
Stereotype-instance	knowledge represenation structure: a partially constructed representation of a particular text - corresponds to a topic-manifestation.
Variable-expectation	knowledge representation structure: the characters and objects expected to occur in texts of particular topic-types.
Utterance-expectation	knowledge representation structure: the abstract representation of the individual text segments expected by a stereotype-prototype.

3.2 The Ideas behind the Scrabble Program.

Scrabble is based on a number of distinctive ideas which which were developed to overcome some of the disadvantages and shortcomings of previous prediction-based natural language processing systems. These ideas fall into two separate classes.

The first class concerns the process of text analysis. That is, the process by which expectations are selected as being applicable to a particular text, matches are discovered between portions of the text and particular expectations, and success and failure of matching is recorded. Such ideas would be applicable to any expectation-based natural language processing system, regardless of its task domain.

The second class are specific to summarising, Scrabble's task domain. The next section describes why summarising was chosen as the test task domain, so I will not do that here. I will, however, describe the distinctive features of the Scrabble summariser very briefly here.

All the ideas presented here as distinctive of Scrabble spring from one overall view of the strength of predictive or expectation-based inference and how that strength relates to more general problems of language analysis. A major problem of inference in natural language processing is that it is usual for there to be very large numbers of inferences which could be made from any input natural language utterance, in fact, more than can be made in the time available to the system. There is therefore a requirement to select which inferences to make. What is needed is, essentially, a mechanism for identifying the most interesting, or salient, inferences and material in the text. It is suggested here that predictive inference provides a rapid (if sometimes crude) method of identifying, and performing shallow inference on, the least salient portions of the text. If a text segment is entirely predictable from the topic-type of the section of text in which it occurs it is unlikely to be the crux of the authors message. Therefore it is unlikely to be useful to apply a more intensive or extensive, and either way very expensive, inference engine to it. At the same time, the recognition of topic which can be achieved through the use of predictive inference is likely to provide a useful context for a more general form of inference engine to operate in. The essence of this work is therefore an attempt to construct a predictive inference system which attempts to maximise the ability of the system to recognise and process commonplace segments in a text at the minimum computational cost, but which achieves this goal without introducing the possibility that elements of the input material will be ignored by the system: all the input material will remain available for processing by other parts of the system.

This overall view has four major implications for the text analysis portion of Scrabble. They are:

1) no inference should be performed when matching expectations against sentence representations;

2) it is better to invoke expectations which turn out to be irrelevant than to fail to invoke expectations which *are* relevant to the text;

3) the system should not skip over sections of text which it does not expect: they should be marked and integrated, in so far as possible, with the expected material;

4) it is not necessary to associate segments of a text with a particular topic-type.

Point 1: that no inference should be done when matching expectations against the text, is adopted as a principle for three reasons. First, the primary virtue of a predictive inference system is that commonplace material may be processed with great computational efficiency. The approach adopted here is intended to maximise the

efficiency of the processing of such material. It has the disadvantage that the system may fail to recognise a match between input material and one of its expectations. This problem could be ameliorated by the introduction of complex inference into the process of matching itself; but the introduction of such inference would lead to a significant decrease in the efficiency of the matching process (which is important because the process of matching expectations to the text is the central process of predictive inference); and also might lead to the misclassification of unusual material as commonplace, since the range of material which may match an expectation will be much increased.

Point 2, that is better to invoke expectation which turn out to be irrelevant than to fail to invoke expectations which *are* relevant to the input text, is adopted as an attempt to overcome the difficulty of discovering whether a particular section of text is or is not about a particular topic-type without invoking the expectations associated with that topic-type. At a certain level a text section may be a described as *only* being an instance of a topic-type *if* the majority of its material is expected in a text about that topic. From the point of view of implementing a computer system, it is impractical to compare incoming text segments with all the predictions the system has in order to identify the topic-type of the text. Therefore, a computationally cheaper solution must be found, even at the expense of failing to recognise the topic-type correctly. Previous systems were designed to use quite sophisticated techniques to try to obtain very high accuracy of invocation of expectations with very little information. Because the matching of expectations and the text in Scrabble is relatively inexpensive, it is possible to postpone firm identification of the topic-type and invoke sets of expectations which turn out to be irrelevant. If all plausible candidates for the topic-type of a text are considered initially, greater accuracy of the identification of the topic-type may be achieved subsequently; and this may be done without crippling computational expense in Scrabble.

Point 3, that the system should not skip over sections of text which is does not expect: it seems unlikely that any practical predictive inference system will ever have enough predictions to expect all the input material with which it might be presented, so a motivated way of dealing with such unpredicted material is required. The system of [DeJong79] merely skips unpredicted material: this is clearly unsatisfactory. What is required is a scheme which exploits the advantages of predictive inference when processing commonplace material whilst not obscuring the (at least potential) interest of unpredicted, and therefore unusual, material. This work assumes that the text being processed is coherent in some sense, so that unpredicted text segments in an input text may be taken to have something to do with predicted segments which occur before or after them. This assumption is sufficient to allow a predictive inference system to do some simple integration of unpredicted and predicted material from the input text. In general, the representations produced will allow any subsequent processor to focus on those parts of the

input material requiring the most expensive processing whilst minimising the cost of recognising and processing, for example, scene-setting information. This is likely to be of value whether the application is, as here, summarising, where all that is required is fairly shallow processing to construct a reasonably coherent text from the representations passed forward, or question answering, in which case the subsequent processing may be very deep inference.

It should be noted that there are important implications for the architecture of the system if any but the most simple-minded processing of the unpredicted material is done. To facilitate processing it seem likely that the unpredicted material will need to be translated into some form which can be easily manipulated by computer. This, in turn, implies that the system must be able to construct representations of parts of the input on the basis syntactic and semantic information alone: that is without any information from the predictive inference system. Thus the system requires a decoupled analysis, or semantic parsing component, unlike DeJong's system.

Point 4, that it is not necessary to associate segments of texts with a particular topic-type: previous predictive inference systems have been based on the assumption that texts, in general, move progressively from one topic-type to another, dealing with only one at a time. This may well lead to perfectly ordinary utterances being classified as unusual because the systems assume, incorrectly, that their topic-type has been finished with. Scrabble is based on the observation that a text segment may be associated with more than one topic-type. For example, if the topic-type of one portion of an input text is eating in a restaurant, and then the sentence:

(2) Then they went to the cinema.

occurs in the input, the sentence can occur both as the beginning of a section whose topic-type is going to the cinema, and also as the (temporal) end of the description of eating in a restaurant. More complex cases can occur. For example, it is unusual to see wild animals from the window of a restaurant unless the restaurant is in a zoo. I feel such a case is more elegantly handled by allowing the predictions associated with the zoo topic-type and the restaurant topic-type to be compared with the text simultaneously, rather than, say, having the system recognise a separate restaurant-in-a-zoo topic-type. In addition, to construct representations for the purposes of Point 3 above it is more important to recognise whether or not a text segment is usual than to uniquely assign it to a topic-type.

Leaving the text analysis program, the central idea behind the Scrabble summariser is that portions of text which were expected may be summarised in a straightforward, template-driven way, providing a framework into which unexpected utterances may be fitted.

Now, there is nothing novel about the simple proposal that, if one front-ends a summariser with a predictive analysis process, reasonably

good summaries of the commonplace portions of an input text can be produced by template driven methods. FRUMP and SAM in summary generation mode operate in precisely this way. In Scrabble the mapping onto template summaries of the predictable parts of the text is the main means by which reduction of the volume of the input material is achieved: what is different is the role of the filled-in templates in the summary representation produced. In Scrabble, in contrast to other systems, the unpredicted material is treated as the most salient part of the input text: it is therefore the most important material to include in the summary. However, because Scrabble operates without a non-predictive inference component unpredicted material may be processed only shallowly by the system. Thus in order to construct a reasonably coherent summary some processing of the unpredicted material must be done: in particular the unpredicted material from the input must be fitted into some kind textual context in order to be comprehensible. The filled-in summary templates are seen here as providing both a textual context into which unpredicted material may be fitted, and more general contextual information which may be used to process the unpredicted material so that the summary forms a coherent whole.

There are two important consequences of adopting this design framework for the summariser. First, it is unlikely that misleading summaries will be produced. If the text contains large amounts of unexpected material, or the system cannot recognise its topic-type (which may occur if the system has not been primed to deal with texts of that topic-type) the system will produce an unreduced rendering of the original text rather than a summary as such. Alternatively, if the unpredicted material in the text violates, in some way, the assumptions made about the relations between unpredicted and predicted material in the input text it is likely that the summary will be obviously ill-formed in some way. If we consider circumstances in which the original text is unavailable to the reader, for example if the summary text were produced in a different language from the original, I believe the production of obviously ill-formed, or, at least, unreduced, texts is better than the production of apparently good summaries which misrepresent the original text, as happened at times with DeJong's FRUMP.

The second consequence of the system design proposed here is that the summariser cannot manipulate unpredicted input material as undigested natural language strings. If the summariser was to manipulate text strings very little integration of unpredicted material and filled in summary templates could be done. Even a simple form of integration, like straightforward cross-sentence pronominalisation, is difficult to achieve when dealing with straightforward text strings. For example, without having the text strings in some sort of parsed representation, it is difficult to convert:

(3) John threw fish to the seals. The seals barked joyfully.

into

(4) John threw fish to the seals. They barked joyfully.

as one would probably wish to, without running the risk of converting:

(5) The king attached his seals to the wildlife preservation order. The seals barked joyfully.

into

(6) The king attached his seals to the wildlife preservation order. They barked joyfully.

In order to overcome the problems presented by integration in this style, it was decided that the summariser should take in unpredicted material from the input text in some semantic representation which facilitated manipulations like the identification of semantically plausible co-referents. In turn the decision to use a semantic representation language with these properties has implications for the text analysis portion of the program. In particular the text analysis portion of the program must be capable of producing reasonably deep representations of the meaning of unexpected segments of the input text.

3.3 A Task for a Predictive Language Analyser

The previous section discussed some novel features of the Scrabble text analysis system, and attempted to suggest why they were improvements on the approaches adopted in previous predictive inference systems. It also made some, largely separate, claims about the novelty and utility of the way in which Scrabble produces summaries. This section explains why the task of summarising was chosen as the test bed for the ideas about predictive inference-based text analysis presented in the last section.

I will take it as read that any claims about a natural language processing system should in essence be based on its ability to perform either some language manipulating task, like summarising, or to perform some other language stimulated physical task, for example a robot responding to spoken commands. Other positions are quite tenable, but a comparison of methodologies at this level is beyond the scope of this work.

There are three externally visible properties which should be possessed by a predictive inference system for text analysis constructed on the basis outlined in section 3.2: it should be capable of identifying the topic-type or topic-types of a segment of an input text with very high accuracy; it should produce as its output representations which demonstrably contain any unexpected material in the input text; and it should be efficient.

A good summary is accepted as being one which contains the most salient points in the original text. However, it cannot contain just

the salient points: they must be placed in a context in which they will be comprehensible. Furthermore it cannot contain just the original textual expression of the salient material: it is unlikely that the summary will form an acceptable natural language text if it does. (Of course, it may be that in the summary text the most acceptable way to express the salient material is in its original form, but other possibilities must be allowed for.)

It is an assumption of this work that unexpected material for the topic-type of the section of the input text in which it occurs, is precisely the material which is most likely to be salient or interesting. It is proposed that given such an assumption a good summary may be described as one which contains an indication of the major topic-type of each piece of the input text together with both an indication of the specifics of the topic-manifestation and what is unusual about it. In more implementation oriented terms this would require that there appeared in the summary an indication of a suitable stereotype-prototype for the analysis a piece of text, some indication of the bindings of the variable parts of the corresponding stereotype-instance, and, suitably integrated, the unpredicted material which occurred in that section.

So, to furnish the summary generation process with these data-structures the predictive inference mechanism must be capable both of identifying the topic-type of a section of text so that it can access the corresponding stereotype-prototype and of passing over the unpredicted material in the original text in a form which will allow it to be integrated into the summary. But, disregarding efficiency for the moment, these are precisely the properties required to demonstrate that the predictive inference mechanism has achieved the goals set out for it. If the summarising system is reasonably efficient as a whole, clearly the inference mechanism will have to be efficient. Therefore it should be easy to determine whether the criterion of efficiency has been satisfied.

Thus it seems that the generation of summaries, as well as any intrinsic utility it may have, is a good test bed on which to place a predictive inference system.

Of course, the view of what constitutes a good summary and how to generate one presented here is too simplistic. In particular the definition of a good summary adopted here does not include all the factors which are likely to be taken into account by a human judge of summary quality, although it probably includes some of the properties a human produced summary would be expected to have. Some limitations of the Scrabble approach will be presented at the end of this thesis

3.4 An Overview of Scrabble

Chapter 1 introduced the gross structure of Scrabble: this section expands on the material presented there, particularly in section 1.1. The reader may find it useful to refer to the diagram in that section.

The ***Scrabble system*** essentially works as a five module system, an English semantic parser, a combined stereotype management module, a stereotype application module, a text representation summariser and an English generator[1]. As mentioned earlier, together the stereotype management module, the stereotype application module and the text representation summariser form the ***Scrabble core*** itself. I will sometimes refer collectively to the programs of the Scrabble core as the Scrabble program. The Scrabble core is the primary concern of this work: it is entirely my work. The English semantic parser and the English generator are modified and extended versions of those described in [Cater82].

The requirement for a separable component capable of producing some form of easily manipulable representation of unpredicted material in the input text was described in section 3.2. In order to satisfy this requirement, Scrabble had to have access to a semantic parser which operated (at least) on the material which the predictive inference system did not expect. In fact, in Scrabble, it was decided that if any form of semantic parser was to be used in the system it should operate on all incoming text. Severe problems, primarily of control, would be presented if it were only used on unpredicted material. Additionally, front-ending the predictive inference system with a semantic parser greatly simplifies the inference system: the inference system need take much less account of the vagaries of natural language than if it has to deal with text directly.

Equally, as consequence of the decision to integrate the different parts of the summary by manipulating unexpected material in a semantic representation language, it was necessary to use a generator to convert the summary material back into English text. Again it was decided that the summary generation system would be simplified if the Scrabble text representation summariser dealt only with whatever representation the generator took as input, and never with English directly.

Having made the decision to use a semantic parser and a generator, and consequentially a representation language or languages to be input and output by the Scrabble program itself, it was of course necessary to choose a particular semantic parser, generator and representation language to use. This choice will be discussed in

---

[1]The terms "semantic parser" and "generator" were introduced in section 2.2.

the next section.

Turning to the remaining three highest-level modules of the Scrabble system, the stereotype management module, the stereotype application module and the text representation summarising system, the stereotype management and stereotype application modules are implemented so that they are a general purpose predictive inference system, whilst the text representation summarising system is, in the main specific to the task of generating summaries. I will henceforth call the stereotype management module and stereotype application module the ***text analysis*** module, and text representation summarising system the ***summariser***. The section after next contains an outline description of these two top-level modules of the Scrabble program. Selecting the stereotype-prototypes which are appropriate for the analysis of a particular text is one of the primary functions of the stereotype management module. I will, on occasion, use the term stereotype selection module to refer to what has been called here the the stereotype management module, particularly when concentrating on the modules role in selecting stereotypes.

3.5 The Choice of an Analysis and Generation Mechanism

This section gives the reason for the choice of the English front- and back-ends used by the Scrabble program. The reasons for adopting the use of a semantic parser as a front-end to the Scrabble program and a generator as the back-end were given in the earlier part of this chapter. However, there remained the problem of choosing the particular semantic parser and generator, and their associated semantic representation languages.

It seems clear that the semantic parser and generator should use the same semantic representation language. If they did not, a sophisticated program would be required to translate unpredicted material in the input text between the two representation languages. The construction of such a program seemed an unnecessary burden, unlikely to contribute much to the primary goals of the project: demonstrating the capabilities of the model of predictive inference embodied in the Scrabble program and its utility in summary production.

Three possible strategies presented themselves for the provision of a representation language and a semantic parser and generator for it.

First, a new representation language might be designed, and a semantic parser and generator written for it.

Second, an existing representation language might be adopted and new programs written to translate between it and English.

Third, an existing representation language, for which a semantic parser and generator existed, might be taken over.

I was fortunate in having the third strategy open to me: in fact I had the choice of two complete representation language, semantic parser and generator sets available to me, those of [Boguraev79] and [Cater82]. Clearly, adopting one of these sets would involve less work than either the first or second strategy, unless there was evidence that they were very inappropriate for the task in hand. It was hoped that by avoiding work on the front-end and back-ends of the Scrabble system a more effective core Scrabble program might be constructed in the time available. It therefore remained to choose between the systems of Boguraev and Cater. The primary basis for making the decision was the suitability of the representation languages for the purposes of predictive inference and summary generation. As there was no intention to experiment with, for example, the relation between inference and the process of semantic parsing itself, all that was required was that the semantic parser and generator chosen should translate between the semantic representation language and English: the internal architecture of the system was of no great concern.

Boguraev's system uses a semantic representation scheme based on that of [Wilks73], [Wilks75b] and [Wilks77]. The problem with the representation scheme of [Boguraev79] is that, unlike that of Wilk's, it is intentionally not strongly ***canonical***. I mean by strongly canonical that all classes of utterances deemed to have the same meaning (out of context) and hence mapped into the same representation are typically large. The representation used by Boguraev quite closely reflects the surface text so equivalence classes are small. In particular, it contains identifiable vestiges of almost all the surface lexical items in the original text:

> "A semantic component is [thus] assembled from its subcomponents, which obviously implies that the semantic formulas of the surface words ***always*** appear in the final representation."[2]

For example,

(7) The man walked to the park.

is represented by:

---

[2][Boguraev79] page 3.44. Note that this does not imply that the representation scheme of [Boguraev79] is in no respects canonical. In particular, syntactic variants with little or no associated semantic variation, for example, the passive and active versions of a sentence, are represented identically.

```
((clause
    (type declarative)
    (tns past)
    (v
       (walk1
          ((*hum subj) ((self obje) (move cause)))
          (@@ agent (man1 (mal man)))
          (@@
             destination
             (park1
                ((((plant change)
                      cause)
                   goal)
                  (where point)))) ))))
```

Structure A

whilst

(8) The man went to the park on foot.

is represented by:

```
((clause
    (type declarative)
    (tns past)
    (v
       (go1
          ((man subj) ((self obje) ((*pla to) move)))
          (@@ agent (man1 (mal man)))
          (@@
             destination
             (park1
                ((((plant change)
                      cause)
                   goal)
                  (where point))))
          (@@ instrument (foot1 (man part))) ))))
```

Structure B

The representations produced for the two sentences are quite different, although the preferred readings for the sentences have the same import.

This lack of canonicality presents serious problems for a stereotype-based inference system, and indeed predictive inferences systems in general. The fundamental process of stereotype application is the recognition that an utterance in the text satisfies an expectation of a stereotype-instance, or alternatively, that an utterance satisfies no such expectations. In other words it is essentially a pattern matching process. Weak canonicality in the semantic representation leads to difficulties with this recognition process. Some of these might be tackled by organising the pattern matching processes so that they ignored the most obvious traces of the surface in the representation, for example word sense names. However, this cannot provide a complete solution, because, for example, in examples Structure A and Structure B it is difficult to extract the

close relationship between the two sentences. In particular the extraction of this relationship would require extremely sophisticated manipulations of the surface orientated case frames which form the backbone of the representation structures and the semantic formula of verbs and case frame fillers. Manipulation so sophisticated that it may well be best classified as inference. Thus it seemed that the adoption of Boguraev's representation language ran the danger of opening the trapdoor to unconstrained computation when matching expectations against the text, the reason given in section 3.2 for avoiding the use of rule-based inference when performing this matching. Thus it seemed Boguraev's system was unsuitable for my purpose.

It might be objected that these problems with Boguraev's system are problems of practice, rather than principle, and should therefore be discounted. For example, within Wilks' system it would be possible to write formulae for "go1" and "walk1" which facilitated the recognition of the underlying similarity of the two sentences. However the work involved with overcoming the problems would have been significant, considerably lessening the advantages of taking over an existing system.

The other analyser/generator and representation language system available to me, the relevant parts of the AD-HAC system of [Cater80] and [Cater82] used a representation system explicitly designed to be strongly canonical in the above sense. The representation scheme is a development of the Conceptual Dependency system of [Schank75b]. I will refer to Cater's system as CD, to distinguish it from the Schank system, which I will refer to as Conceptual Dependency.

As an example, consider again examples (7) and (8). They are represented identically by:

```
((EVENT (ACTOR MAN1)
        (ACT PTRANS)
        (OBJECT MAN1)
        (FROM DUMMY-PLACE2)
        (TO PARK1)
        (INST (EVENT (ACTOR MAN1)
                     (ACT MOVE)
                     (OBJECT (FOCUS (STATE THING))
                             (STATE (STATENAME PART)
                                    (THING LEGS1)
                                    (VAL MAN1)))
                     (TIME (NAMED TIMEPOINT1)
                           (COMPARISON
                              (BEFORE *NOW*)))))
        (TIME (NAMED TIMEPOINT1)
              (COMPARISON
                 (BEFORE *NOW*)))))
```

Structure C

In general Cater's system produces the same structure for those sentences, deemed synonymous, which differ either in the verbs and adjectives used or their syntactic form. It is thus highly canonical in this direction.

However, it is only very weakly canonical in the representation of nominals. Consider, for example, the token PARK1 in the above structure. Although there is an orthogonal data structure describing the features of parks, and containing links to individual nominals derived from the word "park" (like PARK1), there is no mechanism for linking "park" with its synonyms and near synonyms. The existence of the latter mechanism would provide an effective substitute for genuinely canonical structures to represent nominals.

In principle the CD primitives and their arrangements in the case labelled structures appeared very well adapted to pattern matching between structures derived from the input text and the kind of expectations likely to be derived from stereotype-instances. Certainly it appeared there would be fewer problems with Cater's representation language than with Boguraev's from this point of view. Thus Cater's system seemed adequately well adapted to the task of forming a front-end to a predictive inference system. It was therefore chosen to form the front- and back-ends for Scrabble.

The form of Cater's representation language and the programs which translate between it and English is not crucially important to any of the arguments and techniques of this thesis. Therefore no detailed exposition of Cater's system will be presented here. Lengthy descriptions of all parts of his system used in the Scrabble project may be found in [Cater82].

3.6 The Scrabble Program

This section gives a brief introduction to the two remaining modules of the Scrabble system: the two which form the Scrabble program itself. They are the text analysis module and the summariser.

The text analysis module, which builds the internal representation of the text, in turn consists of two major sub-modules, the stereotype management module and the stereotype application module. Both work from representations of the input text provided by the AD-HAC analyser. They each are activated at the end of every sentence in the text and may be handed, in addition to CD-structures derived from the input text, information from either of the other two sub-modules of the text analysis module.

The stereotype suggestion module processes information associated with nominal tokens in the CD-structure to find stereotype-instances which *may*be appropriate to the current sentence. Its primary goal is to implement the principle of section 3.2that it is

better to invoke expectations that are irrelevant to the text than to fail to invoke expectations which are irrelevant. In practice this principle requires that a stereotype-prototype should be instantiated if there is the faintest hint in the text that it may be useful in the analysis of the text. It is then left to the management module to reject suggestion which are clearly irrelevant.

At the same time, the management module monitors the set of active instances to determine if it appears that any of them has either been spuriously activated, in the sense that it has turned out to be unable to predict the contents of the text, or is no longer relevant to the text. These two modules will be described together in Chapter 4.

The stereotype application module takes the predictions of the currently active instances and matches them against the incoming CD-structures. It is responsible for removing predictions which have been satisfied from the set of predictions which will be matched against the next text segment, and for identifying the propositions in the text which were not predicted. It also continuously makes the predictions of a given stereotype-instance more specific, by passing on information about the way in which previous predictions were satisfied. This usually has the effect of constraining the utterances another prediction can match. For example, if a human being[3] is predicted in the text, if it is, say matched against a male, a subsequent attempt to match it against "she" will fail, whereas "he" will succeed. Alternatively, if it has never been matched both will succeed. The stereotype application module is described in Chapter 5.

The summary generation component, which takes as input the text representation produced by the analysis phase and produces as output CD-structures suitable for conversion into English by the AD-HAC generator, consists of three main components. The first takes canned CD-structures[4], associated with stereotype-prototypes, and the particulars of instances of those prototypes which were used in the analysis of the text in order to produce what I will call single stereotype summaries. These canned CD-structures have associated with them an indication of what are likely to be the most significant variable bindings in an instance of their prototype. If these variables are not tightly bound, for example to a particular proper name, the second section of the summariser program makes some strong assumptions about the coherence of the text to produce more specific referents than could be generated in the single stereotype summary. Finally, a third process organises the set of single stereotype summaries into a reasonable narrative order. The CD-structures produced by this are then passed over to the AD-HAC English generator in turn. The summariser is described in Chapter 6.

---

[3]I mean, of course, a text segment which could be interpreted as referring to a human being.

[4]Actually CD-like structures containing stereotype variables.

The next section consists of an annotated example Scrabble run which it is hoped will clarify the points made in this chapter.

3.7 An Example Scrabble Run

For the sake of brevity, this section contains a fairly simple example of the kind of text which Scrabble can summarise. However it illustrates in outline the processing done to produce the summary. The example text concerns buying a present. It is:

(9) Mary wanted to give Susan a present. She thought Susan would like to have a computer so she went to the shop which sold them. The computers looked exciting. Mary walked over to one and tried it by writing a little program. She enjoyed it and decided it would be a good present for Susan. She paid for it and took it away with her.

Selecting and buying presents is a single topic-type, and so the text is analysed using just one stereotype. The form in which stereotype-prototypes are stored for use by Scrabble is quite complicated, so rather than attempt to introduce it here, an informal description of the information contained in the prototype will be substituted. Examples of the representation of stereotype-prototypes used by the program will be presented in Chapter 5. When studying the illustration stereotype-prototype it should be remembered that because a canonical semantic representation language is used when linking utterance-expectations to propositions in the input text, any paraphrase of the expected utterance will be acceptable. Informally, the expectations contained in the stereotype-prototype which are used in the analysis of the input text are as follows.

Present buying concerns two people or sets of people, a 'giver' and a 'receiver', something which is bought, the 'present', and a place, the 'purchase-place'. Before anything else happens the 'giver' decides that he or she wants give a present to the 'receiver'. Then the 'giver' goes to a 'purchase-place' which sells things the 'giver' thinks will be appropriate as a 'present'. Then the 'giver' looks for things which the 'receiver' likes and which he or she thinks the 'receiver' will also like. The 'giver' may do things to determine the quality or suitability of the potential 'present'. The looking and trying may be repeated. A 'present' may eventually be paid for and perhaps taken away.

The other information contained in the stereotype-prototype for present giving concerns the generation of summaries. Here it is that texts whose material follows the above form should be summarised by sentences of the form "'giver' bought 'present' for 'receiver'".

In what follows blocks of text indented at the right and left hand sides and printed in this font are the output from the computer program; annotations for the output are unindented.

```
Run of AD-HAC with Scrabble version 15.13
in 950K at 12.53 on 23 SEP 82
[]
```

Square brackets mark the beginning and end of the processing of the AD-HAC semantic parser: usually they will enclose messages about the performance of the system, but all such messages were switched off for this run.

```
Entering Scrabble for sentence :
 (MARY WANTED TO GIVE SUSAN A PRESENT)
```

Although the program prints out a copy of the text of sentence, the Scrabble program does not have access to the text as such. It can only access the CD-representation of the input. The text of the sentence is printed solely as a debugging aid.

The word "present" in the input suggests that the present buying stereotype the system possesses which looks as if it may be useful here. Of course in a completely realistic system many other stereotypes might be suggested, and certainly others would be during the processing of the text. Buying presents is the only stereotype suggested at any time during the processing of this simple story. Because wanting to give presents is a predictable beginning to a text for which the stereotype is suitable, a new instance of the buying presents prototype is formed.

```
Stereotypes suggested:
((PRESENT))
The following Stereotype Instances have been activated
(PRESENT2)
Spent 575 ms in Scrabble
[]
Entering Scrabble for sentence :
 (SHE THOUGHT SUSAN WOULD LIKE TO HAVE A COMPUTER SO
    SHE WENT TO THE SHOP WHICH SOLD THEM)
```

Neither "shop" nor "computer" suggest any new stereotypes in the current system, so no new instances are activated. The absence of any printed messages from PRESENT2, the only active stereotype-instance, indicates that it expected an input sentence like the one which arrived. PRESENT2 provisionally narrows its expectations about the 'present' to it being a computer.

```
No new stereotype instances were activated
Spent 140 ms in Scrabble
[]
Entering Scrabble for sentence :
 (THE COMPUTERS LOOKED EXCITING)
```

A predictable thing to say about a potential present.

```
No new stereotype instances were activated
Spent 63 ms in Scrabble
[]
Entering Scrabble for sentence :
 (MARY WALKED OVER TO ONE AND TRIED IT BY WRITING A
    LITTLE PROGRAM)
```

This sentence presents a number of difficulties for the program. First it contains two pronouns "one" and "it": they are presented to

Scrabble as distinctive markers with few specific features associated with them. Scrabble assumes, in general, that pronouns correspond to its variable-expectations. Two of the systems utterance-expectations are that the 'giver' may move close to a potential 'present', and may try it. In this case Scrabble is already expecting that the present will be a computer: both "one" and "it" can refer to a (specific) computer, so the system creates a new variable-expectation that a single specific computer is the 'present'.

The second difficulty is presented by this input sentence is that "writing a little program" is not something Scrabble expects as such: it knows nothing about computers and how they are tried. This is dealt with by causing the program to accept any instrument for "try": not a complete solution, but one which produces acceptable results for most input texts.

```
No new stereotype instances were activated
Spent 113 ms in Scrabble
[]
Entering Scrabble for sentence :
 (SHE ENJOYED IT AND DECIDED IT WOULD BE A GOOD PRESENT
   FOR SUSAN)
```

Again a predictable input sentence if the potential 'present' has been identified. The first "it" is not identified with the computer which is expected to be the 'present'. The AD-HAC analyser marks it as referring to some action through expectation associated with "enjoy". It may therefore be distinguished from an "it" which should be associated with a concrete object.

```
Stereotypes suggested:
((PRESENT))
No new stereotype instances were activated
Spent 83 ms in Scrabble
[]
Entering Scrabble for sentence :
 (SHE PAID FOR IT AND TOOK IT AWAY WITH HER)
```

The final sentence in the text is again quite predictable. Again the two occurences of the word "it" have their referents resolved on the basis of the systems prior expectations.

```
No new stereotype instances were activated
Spent 46 ms in Scrabble
```

At the end of processing the text the system has the following three pieces of information to fit into the summary. First the 'giver' is Mary; second the 'receiver' is Susan; third the present is one specific computer, which has not been identified by name. Therefore the output of the system is:

```
English summary is:
  MARY BOUGHT A COMPUTER FOR SUSAN.
```

The next three chapters describe in detail the processing illustrated by this example.

4. Topic-type Identification and Stereotype Management

This chapter describes the means by which Scrabble identifies the topic-type of a section of the original text and controls the more detail analysis of input material. In the Scrabble program topic-type identification involves the selection of those of the system's stereotype-prototypes which can be used to analyse some part of the input text and the assignment of the input ***text segments*** (or at least most of them) to particular instances of those stereotype-prototypes.

The use of the phrase ***text segment*** here deserves some further explanation. Typically within Scrabble the utterance-expectations of a stereotype-instance will correspond to a clause in the input. However it must be remembered that the input material is presented to the Scrabble program as CD-structures: thus features of the original text like clause boundaries cannot be discovered by the Scrabble program. In practice, an utterance-expectation may be for as simple a CD-structure as a single STATE construct or it may be arbitrarily complicated. Therefore a text segment, within the Scrabble program, may range from part of a noun phrase, containing a single adjective and a noun, to a complete sentence.

Of the principles described as being fundamental to Scrabble in section 3.2 there are two of relevance here.

First, that it is better to invoke expectations which turn out to be irrelevant to the text than to fail to invoke expectations which are relevant.

Second, that it is not necessary to associate segments of a text with a particular topic-type.

In practice, in order to be certain that expectations which are relevant to the text are always invoked it is necessary to instantiate a stereotype-prototype on the faintest suggestion that it will be relevant to the input text. For example, the occurrence of a word in the text which in some rarely occurring contexts is connected with the topic-type for the stereotype-prototype must provoke the formation of a corresponding stereotype-instance, unless there is strong evidence that it will be inappropriate. The process detecting these hints in the input text is called ***stereotype suggestion***.

Distinguished from this process of stereotype suggestion is the ***elimination*** of suggested candidate stereotype-instances which are not, in fact, appropriate for the analysis of the input text, and the subsequent ***identification*** of the topic-types of various parts of the input on the basis of the stereotype-instances which are not

eliminated. This elimination and identification is part of ***stereotype management***.

The other part of stereotype management concerns the delimitation of the parts of the input text which are associated with a particular topic-manifestation. In other words, identifying those text segments which should be associated with a given stereotype-instances. Of course, this association is intimately connected with the topic-type identification process. In particular only those stereotype-instances which succeed in predicting significant parts of the text will not be eliminated, and, in the end, text segments should only be assigned to stereotype-instances whose corresponding topic-types have been identified as being the topic-type of part of the input text.

The primary responsibility for the association of text segments with stereotype-instances falls to the stereotype application module, which is described in the next chapter. However it is a stereotype management responsibility to decide when a stereotype-instance which has been identified as being appropriate for some part of the input text is no longer appropriate. This occurs when, for instance, the topic-type of the text changes. It is necessary to suspend stereotype-instances in this way in order to minimise the computational cost of comparing predictions with incoming material in CD form.

In accordance with section 3.2 the delimitation of the parts of the text which are associated with a particular stereotype-instance does not require that a particular text segment should be uniquely assigned to a stereotype-instance. Nor, in fact, is there any requirement that the areas of the original text which are associated with a particular stereotype-instances should be contiguous.

The remainder of this chapter is organised as follows. It begins with a detailed description of the stereotype suggestion mechanism. A detailed overview of the Scrabble stereotype manager follows. This, in turn, is followed by sections which describe in some detail the implementation of the mechanisms used to provide topic-type identification, and to suspend stereotype-instances which have been useful in the analysis of the text, but are no longer useful. Next there is a description of some of the ancillary information collected by the manager whilst the input text is processed: this information is used by the summariser. Finally a detailed example is presented.

4.1 The Stereotype Suggestion Mechanism

As has been pointed out in Chapter 3 and in the introductory part of this chapter, there are a number of properties one would like programs which implement the topic-type identification sections of a stereotype-based inference system to possess. They include ***precision***, by which I mean that the system should never associate a text segment with topic-types whose stereotypes cannot be appropriate for that text segment; ***comprehensiveness***, by which I mean that the

system should always recognise that a text segment is associated with a topic-type for which the system has stereotype-prototypes[1]; and ***economy***, by which I mean that the computational costs of associating stereotype-prototypes with text segments, and of recognising that the topic-type of a piece of text is not one for which the system possesses stereotype-prototypes, should be low. I believe the Scrabble system is strong on comprehensiveness and precision simultaneously, but could be improved as regards computational cost, although it is not disastrously expensive at present.

The Scrabble system achieves both high precision and comprehensiveness at the same time by exploiting the separation of the stereotype suggestion mechanism from the general management mechanism. The requirements for the suggestion mechanism are much weaker than the requirements for the identification mechanism as a whole. In particular, the suggestion mechanism does not need to be precise: provided it is comprehensive, the general management system can (or at least should) obtain high precision by examining the suggested stereotype-prototypes together with the current context, as represented by the set of currently active stereotype-instances and their success at predicting the contents of the text entering the system . It is, however, all the more important that it is computationally cheap, because it provides only a limited part of the stereotype identification mechanism, much of the work being pushed onto the stereotype management process proper.

There are four specific properties of the current Scrabble stereotype suggestion mechanism which are intended to overcome problems of previous systems. First, semantically rich or suggestive words (or their residual representations) giving good clues to stereotype-prototypes which might be appropriate for the analysis of a text segment are processed early in the suggestion procedure. Secondly, no processing effort is expended on information which gives little clue as to the topic-type of the current text segment: for example processing of CD EVENT and STATE primitives is avoided. Thirdly, there is no complex exploration of the CD structure representing the text segment when attempting to find the representations of suggestive words. Fourthly, the system is capable of finding and applying a suitable stereotype for a text segment (if it has any) even if that text segment is not connected to any previous text segment in a predictable way. Fifthly, the suggestion mechanism should be capable of dealing with the problem which I will call ***low precision suggestion***, as follows.

---

[1]The interested reader might like to compare my terms precision and comprehensiveness with the terms precision and recall used in the literature of Reference, alias Document, Retrieval Systems.

Consider the following text fragments, which are adapted from [Charniak78b]:

(1) Jack picked a can of tuna off the shelf. He switched on the light.

and

(2) Jack picked a can of tuna off the shelf. He put it in his basket.

In a system which has stereotypes for preparing meals in the kitchen at home and for shopping at the supermarket, it is clear that the first sentence in examples (1) and (2) could be associated with either: but, given the context supplied by the first sentence, in example (1) the second sentence can only be associated with the kitchen and in example (2) the second sentence can only be associated with the supermarket. The crucial point about the first sentence in both fragments is not that it can be associated with either the supermarket or the kitchen, but that it can not be associated with, say, stereotypes for warfare or railway journeys. I have christened this phenomenon low-precision-suggestion because the first sentence *does* contain enough information to do some selection amongst the systems stereotypes, but not enough to select precisely one. The examples of [DeJong79] do not contain a text in which this occurs, and it is not clear how he would handle it. He might argue that it does not occur in real texts and whilst this might be true for short news stories, it seems to me that such gradual narrowing of possible topic-type is quite a common device, especially in fiction. If the reader is doubtful I suggest that the first few paragraphs of the story "Reflections of a Kept Ape" in [McEwan79] are examined.

What is required to deal with examples (1)and (2) is a general mechanism capable of recognising that the first sentence in each case has something to do with food. Then those stereotypes whose topic-types are most closely associated with food may be activated, and further processing done to see whether they are, in fact, appropriate for this text, and, crucially, to ascertain that those stereotype whose topic-types have no connection with food need not be so processed. Of course, this is very tentative: John might have been picking up a can to use as a weapon, but the most one can hope for from stereotypes is the rapid analysis of the usual, and it seems safe to interpret the usual meaning of the sentence as having something to do with food, provided other mechanisms allow for later correction. This can be described as drawing a very low precision conclusion, but, again, the requirements on the suggestion mechanism are such that it should err on the side of imprecision in order to achieve greater comprehensiveness. This argument leads to the outline of a scheme as follows: the stereotypes for say, eating in a restaurant, shopping at the supermarket, and preparing a meal in the kitchen are all activated when a word[2] which

---

[2]Or, more particularly, a word sense.

can be associated with the concept of food is mentioned. Restaurant eating can be immediately eliminated because the restaurant stereotype will not predict the picking of anything off shelves, never mind tins. Both the kitchen and supermarket stereotypes will have predicted the first sentence of (1) and (2), and so will be allowed to continue. Only the kitchen stereotype will predict the second sentence of (1), and so the supermarket stereotype will be eliminated, and vice versa for text fragment (2).

The essential requirement for the stereotype suggestion mechanism is, then, that it should provide an efficient and comprehensive means of connecting together the lexicon entries for word senses with concepts associated with stereotype-prototypes.

In principle the links between word-senses and the corresponding stereotype-prototypes could be made automatically. If the system had access to a lexicon containing, for each word sense, semantic information expressed in, say, the semantic primitive language of [Wilks77], the links could be constructed as follows. Each stereotype-prototype would have associated with it a set of patterns of primitives describing (probably quite crudely) its topic-type. The stereotype manager would simply be informed when a word-sense whose semantic entry contained a pattern which was associated with a stereotype-prototype was used in an input sentence.

The current Scrabble system uses a crude approximation to this scheme, largely because Cater's representation language was not at all well adapted to this scheme. In particular the AD-HAC analyser employs very simple lexicon entries for nouns. Very commonly, it was found, it is nouns which are the words whose lexicon entries would, ideally, contain the kind of primitive patterns which would trigger the suggestion of stereotype-prototypes. Large scale modification of the parser could not be undertaken in the time available, so it was impossible to test automated pre-processing of the lexicon to identify what might be called suggestive primitive patterns. So the simpler course of manually extending the lexicon to include a new set of primitives for use by the stereotype suggestion process. The current (pitifully small) set of such primitives is:

Primitive	**Example words**
-PRESENT-	present
-TRAVEL-	car,train
-FOOD-	tuna,supermarket,restaurant
-WILD-ANIMALS-	lion,monkey,zoo

I will call these objects ***suggestive primitives***. Of course a serious system would require many, many more such patterns.

On the other hand, the structure of the AD-HAC analyser was well adapted for the construction of a reasonably efficient implementation of the idealised scheme. In particular it facilitated the process of separating from the primary semantic representation of an input

sentence, the word-senses whose semantic entries contained suggestive primitives. In many other parsing frameworks it would be difficult to devise a computationally inexpensive scheme which both avoided taking the primary sentential semantic representation to pieces to discover from which word senses it had been constructed, whilst ensuring the stereotype manager was informed only of the senses of words which actually occurred in the input.

The two devices in the AD-HAC analyser which were used for this purpose were ***requests*** and ***registers***. It is not appropriate to provide a detailed description of either device here: the interested reader should see [Cater82] for such a description. Briefly the most relevant aspects of the devices are as follows.

The request device is fundamental to the operation of Cater's parser. Its relevant property here is that it provides (amongst many other things) a means by which arbitrary actions may be performed when an attempt is made to use a given sense of a word in the input stream in the construction of a semantic representation of a complete input sentence.

The register mechanism provides a means by which arbitrary information may be associated with, and carried along by, particular partial parses, so that the information may be discarded if the parse does not pan out in some way. Again the register mechanism can be used in many other ways.

The implementation of the mechanism by which the stereotype suggestion mechanism is informed when use is made of a word sense associated with a suggestive primitive is as follows. To each such word sense is attached a request which has just one action: to add the relevant primitive to a register used for accumulating suggestive primitives during the construction of a reading of part of a sentence. Then, if this partial parse forms part of a complete parse of the input sentence, the contents of the register are passed to the stereotype suggestion mechanism. That is, the stereotype suggestion mechanism is informed whenever a suggestive primitive is associated with a word sense which was used to form the semantic representation of an input sentence is being passed to the Scrabble program.

The stereotype suggestion mechanism looks up all the suggestive primitives passed to it in its map from suggestive primitives to stereotype-prototype names, and passes on the names it obtains to the stereotype management process. If more than one stereotype-prototype is associated with a one of the suggestive primitives the data structure will group the stereotype-prototypes together. The groupings are known as ***stereotype suggestion packets***. They are used by the stereotype manager, as will be described in section 4.3.

The stereotype suggestion process is applied to all incoming text, and not just, say, when the system finds that it has no active

stereotype-instance capable of predicting the contents of the current text segment. It is this feature of implementation which allows the system to recognise, and deal with, unexpected changes of topic-type of the text. Such changes of topic-type will, of course, require the suggestion and possible instantiation of new stereotype-prototypes.

To summarise, I will return to the four principles underlying the Scrabble stereotype suggestion mechanism mentioned earlier in this chapter. The implementation both processes suggestive word senses early and avoids the need for sophisticated manipulation of CD EVENT and STATE primitives or other properties of complete sentence representations, because it has no need to examine the representations of complete sentences: rather it gets the information necessary for the suggestion process in a separate data structure. Because it attempts to suggest stereotypes for every new sentence entering the system, it deals with the problem of unexpected changes of topic-type. Furthermore, since it deals with small but fairly loose conceptual patterns, rather than either specific words or larger, specific, propositions, it deals with the problem of low precision suggestion.

4.2 Stereotype Management

The stereotype management module has two primary roles. Firstly it filters and monitors the stereotype-prototypes thrown up by the suggestion mechanism, discarding those which are unlikely to be useful in the analysis of the text and forming instances of the rest. Secondly it attempts to determine when a stereotype-instance, hitherto appropriate to the text, ceases to be relevant to it, and hence should be suspended. The manager also has a number of subsidiary roles, the most important of which is to record the textual order in which stereotype activation, suspension and unpredicted utterances occur.

The filtering of suggested stereotype-prototypes primarily relies on the context provided by the success or failure of existing instances at predicting the parts of the text prior to the current sentence. The process of filtering out suggested prototypes will be described in more detail in the next section.

In the implementation, monitoring instances of suggested prototypes, once such instances are formed, is closely intertwined with the process of deciding whether or not to ***suspend*** an instance. To ***suspend*** an instance is to retain its variable bindings and the contents of those parts of the text which it has successfully predicted, so that it may be incorporated in the representation of the whole text. Clearly only instances which have been identified as being appropriate to at least some parts of the text will be suspended.

If an instance is identified as being inappropriate for the current text segment it is rejected, and most records of it having been active are destroyed, leaving notes of when it was formed and of the other instances which were formed on the basis of the same suggestion. This process is called ***killing***. It is the selection of stereotype-instances to kill which constitutes the high precision part of stereotype identification.

The monitoring is described in section 4.4.

Before beginning a detailed description of stereotype management a brief overview of the ***order*** in which the activities of the manager are performed is necessary. When control enters the Scrabble program after the AD-HAC analyser has processed a sentence, the first thing that happens is that all existing, active stereotype-instances are applied to the new CD representation. (The process of ***stereotype application*** is described in the next chapter). This will yield (of concern to the manager) a set of messages describing the degree of success or failure of each instance at predicting the contents of the new CD representation. The manager then examines these messages to determine if any of the active instances should be killed or suspended. Then any suggestions thrown up by the stereotype suggestion mechanism are examined, filtered as necessary, and instances formed of any prototypes which are not filtered out. Next, these new stereotype-instances (if any have been formed) are applied to the CD representation of the current sentence. Their success or failure is monitored, and they may be added to the list of (genuinely) active stereotype-instances if this seems appropriate.

4.3 The Activation of Stereotype-Instances

The central part of the activation of stereotype-instances is selecting out from the suggested stereotype-prototypes those which should have new instances formed. The suggested stereotype-prototypes are all processed in the same way: in particular prototypes which already have instances which have been used to analyse some preceding part of the text are not processed in a fundamentally different way from those which have never had an instance activated. The selection is done by weeding out those of the suggested stereotype-prototypes for which there already exists an active instances which is successfully predicting the input material, or which have had instances killed in circumstances which suggest that a new instance will not be able to predict the text's contents.

The former case is quite straightforward: if a new sentence of a text continues to have the same topic-type as the preceding section it is quite likely that the new section will use word senses suggestive of that, existing, topic-type. In the absence of contrary information it seems safe to assume that the new sentence concerns a continuation of the same manifestation of the suggested topic-type. Hence, there is

no need to form a new instance of the stereotype-prototype appropriate for that topic-type. An example of contrary information in this case would be the failure of an instance to predict the contents of the current sentence. (Remember that the predictions of existing active instances are compared with the representation of the text before the weeding process is initiated.)

The latter case is more complicated. In the current implementation the interpretation of stereotype-prototypes "which have had instances killed in circumstances which suggest that a new instance will not be able to predict the text's contents" is quite narrow. Let us restrict our attention to one of the stereotype-prototypes suggested for the current sentence. If, whilst processing some preceding sentence, this prototype was suggested, and an instance was formed but subsequently killed, an examination is made of any instances previously also formed on the basis of that suggestion packet. If any of those instances is still active, it is assumed that these are more appropriate for the analysis of the current text segment than the stereotype-prototype under consideration. Hence it is not worth forming a new instance.

For example, consider the following (rather artificial) text fragment:

(3) Jack picked a can of tuna off the shelf. He switched on the light. He opened the can. He got some lettuce from the fridge.

Suppose, for the moment, that the only word in the first sentence whose semantic entry contains a suggestive primitive is "tuna". Further suppose that the primitive pattern represents "food" and is associated with two stereotype-prototypes, shopping-at-the-supermarket and making-a-meal-in-the-kitchen. If instances of both are formed when the first sentence is read, they will both successfully predict this first sentence[3]. However, when both the second and third sentences are read the kitchen instance will predict them, but the supermarket instance will not. Such failure to match will, presumably, lead to the supermarket instance being killed. When the fourth sentence is read, "lettuce" will again cause the suggestion of the making-a-meal-in-the-kitchen and the shopping-at-the-supermarket prototypes. The kitchen prototype will not be instantiated afresh because there is an active instance which has matched all the preceding text. The supermarket prototype will not be instantiated because there was an instance of it which was activated with the successful kitchen instance, but has

---

[3]It may seem odd to use the word "predict" here, but the point is that although the Scrabble system has already read and to some extent processed the sentence, this is the first time any attempt is made to determine whether the input sentence is an utterance either stereotype-instance predicts could occur in a text about its topic-type.

[4]Note that for the sake of simplicity this description has ignored all suggestions made from words in the input other than "tuna" and "lettuce".

since been killed[4].

The process of instantiation is a relatively expensive operation in terms of computation, but not so expensive that inappropriate instantiation seriously degrades the performance of the system. It consists of loading a prototype into main storage if it is not already present there, and then operating on it to convert what might be called its rather declarative form of expression into a pattern matching program and a framework to hold the results of the matching. This process is described fully in the next chapter.

One of the problems with the scheme used is the amount of copying required. Although this is a major expense in the current system, an organisation using something akin to the virtual copy mechanism of [Fahlman79] would considerably reduce the expense and could be implemented without much difficulty.

Once the new instances are formed they are handed over to the stereotype application module, so that their predictions about the contents of the text may be compared with the complete representation of the current sentence. They are then handed over to the main stereotype management module for processing in the normal way. This main stereotype management module is described in the next section.

4.4 Managing Active Stereotype-Instances

There are two distinct but related roles taken on by the portion of the program concerned with managing active stereotype-instances. The first is the refinement of the topic-type identification done by the processes described in section 4.3. The second is the deactivation of stereotype-instances which no longer appear to be relevant to the text.

The heart of the program is the examination of the results of the application of both the active and newly activated stereotype-instances to the representation of the current sentence. This application process is described in Chapter 5. Almost as central is the comparison of these results with those of the stereotype suggestion process described in the last section.

The results of the application process are stored in a data-structure which indicates, for each stereotype-instance, whether it predicted the whole, part or none of the current sentence. If the contents of the data-structure indicate that an instance predicted the whole of the current sentence, the manager takes no particular action. If the data-structure indicates that there was no part of the sentence the instance predicted then the action depends on the instance's earlier success at predicting the contents of the previous sentence and whether the prototype of which it is an instance has been suggested again. If that prototype has been thrown up by the

stereotype suggestion mechanism during the processing of the current sentence it is merely recorded that it has predicted no part of the current sentence. If it predicted no part of the previous sentence, it is considered a candidate for ***killing*** or ***suspension***. If it did have some success at predicting the contents of the previous sentence, then again it is merely recorded that it could predict no part of the current sentence. Of course, the use of this preset number (two) of sentences for inspection is not entirely satisfactory: I will return to this point at the end of the thesis. In addition, if an instance failed to predict part of the current sentence a count of the number of propositions[5] it has failed to predict is incremented.

If an instance becomes a candidate for killing or suspension, then it is examined to determine if it appears that it was activated inappropriately, or was appropriately activated but the text has shifted its topic-type so that the instance is no longer relevant. This is done by comparing the number of the instance's predictions which have been satisfied with the number of propositions which occurred in the text which it failed to predict. If it has predicted less than it failed to predict then it is killed and all trace of it having been activated is removed from the system with the exception of the information used by the stereotype activation mechanism, as described in section 4.3. However even if it has predicted more of the input text than it has failed to predict, it may still be killed. This will happen if there is another active and more effective stereotype-instance which was activated on the basis of the same suggestive primitive which caused the activation of the instance under consideration. However the instance under consideration will not be killed if the other instance has failed to predict something in the input text which was predicted by the current instance. In other words, an instance is killed even if it has been moderately successful at predicting the contents of the current text section if there is an instance associated with a similar topic-type which has been more successful at predicting the contents of this text section.

For example, consider the following text:

(4) John picked a can of tuna off the shelf. He put it in his basket. He got some tomatoes and some lettuce. He found the rice and picked up a bag. He went to the checkout and paid. He went home.

Let us assume that the first sentence causes the activation of instances of the making-a-meal-in-the-kitchen and the shopping-at-the-supermarket stereotype-prototypes (and no others). Suppose they both predict the first sentence, but the instance of the kitchen prototype

---

[5]I mean by proposition here a representation which contains exactly one EVENT, STATE, or CAUSE. Such propositions may correspond to a single sentence: but all but the simplest sentences will be represented by CD-structures containing several such propositions.

fails to predict the second sentence. The kitchen instance will not be considered for killing or suspension yet because it succeeded in predicting the previous sentence. Presumeably both stereotypes will predict both the third and fourth sentences. However, the fourth, and then the fifth, sentence describes things one would expect to happen when shopping at the supermarket, but not when making a meal in a kitchen. Since the kitchen stereotype has failed to predict two successive sentences it is considered a candidate for killing or suspension. Because the supermarket instance was activated by the same suggestive primitive (*-FOOD-* from "tuna") as the kitchen instance, and it has predicted all the items the kitchen instances successfully predicted (and more), the kitchen instance is killed.

If an instance survives these tests to see whether it should be killed it is suspended; that is, its state is preserved, including such information as how its utterance- and variable-predictions were satisfied , but its predictions will not be matched against any further text presented to the system. The preserved states of stereotype-instances are the prime part of the text representation which is passed to the summariser. But this preservation of stereotype-instance states could, in principle allow suspended instances to be reactivated. It would possibly extend the range of texts with which the system could deal if such suspended instance could be reactivated, but this is not done in the present system.

The killing of instances is the final process by which the precision of stereotype identification is improved by the stereotype manager. Equally, the suspension of an instance is, in effect, definite confirmation that the stereotype-prototype from which the instance was formed was appropriate for the topic-type of the text section which the instance was used to process.

4.5 Other Information Recorded by the Manager

Although the text representation constructed by Scrabble has nothing to do stereotype management, since stereotype management is concerned with the control of the process of text analysis rather than its output, a brief mention of the text representation is appropriate here because some important elements of it are constructed by the stereotype management program whilst it is performing its primary task of controlling text analysis.

Although the most important part of the text representation used by Scrabble is derived from the suspended states of stereotype-instances, there are two data-structures which are constructed by the manager itself which also form parts of the text representation. They are a record of all the items in the input text which some instance has failed to predict, and a related record of the textual order of activation and suspension (if applicable) of all instances which have

not been killed.

The former data-structure is extended each time an active instance fails to predict some item in the text. In it the stereotype manager records the name of the instance, the CD-structure it failed to match, and any instances which succeeded in predicting the CD-structure. This is done whilst the manager is determining whether an instance is a candidate for killing or suspension.

The latter data-structure is extended at two different times. First, whenever the processes described in 4.3result in a new stereotype being activated, a note to this effect is added to the data-structure by the manager. Second, whenever an instance is suspended a note is added recording this event. The data-structure is always maintained so that these notes occur in the textually order of the items which triggered their being added.

These two data-structures are not examined by the manager, except when housekeeping after an instance has been killed. However they are central to the summariser's processing, so further discussion of them will be left until chapter 6.

4.6 A Detailed Example of Suggestion and Management

To illustrate the stereotype suggestion and management process in Scrabble the following story will be considered:

(5) JACK PICKED A CAN OF TUNA OFF THE SHELF. HE PUT IT IN HIS BASKET. HE PAID FOR IT AND WENT HOME.

In the remainder of this section, once again computer output generated whilst processing the input text is indicated by right and left indentation, and by this font. During this run more detailed debugging output was produced than during the run shown in section 3.7. As an aid to clarity some of the less relevant material has been removed.

We begin as Scrabble is entered for the first sentence:

```
Entering Scrabble for sentence :
 (JACK PICKED A CAN OF TUNA OFF THE SHELF)
```

The first thing that happens is that the suggestion mechanism is handed a list of the primitives associated with stereotype-prototypes. In this case there is just one:

```
Primitives suggested: (*-FOOD-*)
```

The associated stereotype-prototypes are looked up and suggested:

```
Stereotypes suggested:

((RESTAURANT KITCHEN SUPERMARKET))
```

In this case the three prototypes are for eating in a restaurant,

making a meal in the kitchen and shopping at the supermarket. Because there are no active instances of the suggested prototypes, instances of each of them are formed. They are applied to the text by mechanisms described in the next chapter. The results of the comparison of their expectations and the input material are passed back to the manager. In this case the instance of the restaurant prototype does not expect the sentence; but the instances of the cooking in the kitchen and shopping in the supermarket prototypes do. For successful matches the messages include the CD-structure which matched the stereotype's expectation, the CD-path by which it was extracted from the original structure, and the name of the stereotype slot it fitted.

```
(Message RESTAURANT1 NO-MATCH NIL)

(Message
     KITCHEN1
     MATCHED
     ((Match-message
          ((EVENT
               (ACTOR JACK)
               (ACT PTRANS)
               (FROM SHELF1)
               (INST
                   (EVENT
                       (ACTOR JACK)
                       (ACT GRASP)
                       (OBJECT
                           (FOCUS (STATE THING))
                           (STATE
                               (STATENAME QUANTITY)
                               (THING TUNA3)
                               (VAL *CANFUL*1)
                               (TIME
                                   (NAMED TIMEPOINT1)
                                   (COMPARISON (BEFORE *NOW*)))) )
                       (TIME
                           (NAMED TIMEPOINT1)
                           (COMPARISON (BEFORE *NOW*)))) )
               (OBJECT
                   (FOCUS (STATE THING))
                   (STATE
                       (STATENAME QUANTITY)
                       (THING TUNA3)
                       (VAL *CANFUL*1)
                       (TIME
                           (NAMED TIMEPOINT1)
                           (COMPARISON (BEFORE *NOW*)))) )
               (TIME
                   (NAMED TIMEPOINT1)
                   (COMPARISON (BEFORE *NOW*)))) )
          NIL
          . get-food1)))
```

```
(Message
    SUPERMARKET1
    MATCHED
    ((Match-message
        ((EVENT
            (ACTOR JACK)
            (ACT PTRANS)
            (FROM SHELF1)
            (INST
                (EVENT
                    (ACTOR JACK)
                    (ACT GRASP)
                    (OBJECT
                        (FOCUS (STATE THING))
                        (STATE
                            (STATENAME QUANTITY)
                            (THING TUNA3)
                            (VAL *CANFUL*1)
                            (TIME
                                (NAMED TIMEPOINT1)
                                (COMPARISON (BEFORE *NOW*)))) )
                    (TIME
                        (NAMED TIMEPOINT1)
                        (COMPARISON (BEFORE *NOW*)))) )
            (OBJECT
                (FOCUS (STATE THING))
                (STATE
                    (STATENAME QUANTITY)
                    (THING TUNA3)
                    (VAL *CANFUL*1)
                    (TIME
                        (NAMED TIMEPOINT1)
                        (COMPARISON (BEFORE *NOW*)))) )
            (TIME
                (NAMED TIMEPOINT1)
                (COMPARISON (BEFORE *NOW*)))) )
        NIL
        . getitems1)))
```

On the basis of these messages the manager decides that the eating in a restaurant stereotype-instance is not appropriate for the analysis of the input text, and so it should be killed, rather than added to the list of properly active instances. On the other hand, both the instances of the supermarket and kitchen stereotypes successfully predicted the material of the first sentence: therefore they are properly activated. The manager outputs some messages to confirm that this has happened.

```
No match by stereotype : RESTAURANT1

The following stereotype Instances have been activated
(KITCHEN1 SUPERMARKET1)

Spent 862 ms in Scrabble
```

Processing of the next sentence begins.

```
Entering Scrabble for sentence :
 (HE PUT IT IN HIS BASKET)
```

No new stereotypes are suggested by this sentence. From the active stereotype-instances, KITCHEN1 and SUPERMARKET1, the manager gets back messages indicating that the KITCHEN1 does not expect things to

be put in baskets but SUPERMARKET1 does. Because KITCHEN1 expected the previous sentence it is not considered for killing or suspension: it remains active in the hope that later in the text the manager will be in a better position to judge whether it is relevant to the topic-type of the text or not. The message from SUPERMARKET1 contains the complete CD-representation of the input sentence, indicating that it expected all of it.

```
(Message KITCHEN1 NO-MATCH NIL)
(Message
    SUPERMARKET1
    MATCHED
    ((Match-message
        ((EVENT
            (ACTOR DUMMY-MALE1)
            (ACT PTRANS)
            (OBJECT DUMMY-THING2)
            (TO
                (FOCUS (STATE THING))
                (STATE
                    (STATENAME POSS)
                    (THING BASKET2)
                    (VAL DUMMY-MALE2)
                    (TIME
                        (NAMED TIMEPOINT2)
                        (COMPARISON (BEFORE *NOW*)))) )
            (TIME
                (COMPARISON (AFTER TIMEPOINT1))
                (NAMED TIMEPOINT2)
                (COMPARISON (BEFORE *NOW*)))) )
        NIL
        . basketitems1)))
```

Once again, before beginning processing of the next sentence, the manager prints some confirmatory messages.

```
No match by stereotype : KITCHEN1

No new stereotype instances were activated

Spent 116 ms in Scrabble
Entering Scrabble for sentence :
 (HE PAID FOR IT AND WENT HOME)
```

Once again KITCHEN1 does not expect the input material, but SUPERMARKET1 does. KITCHEN1 has now failed to recognise anything for two successive sentences, and hence becomes a candidate for killing or suspension. Because it has failed to predict more of the input than it has predicted it is killed, leaving SUPERMARKET1 as the sole remaining stereotype-instance. Note that the current sentence matches two of the expectations of the supermarket stereotype in different ways. The first is that the shopper will pay for some goods, and the second is that he or she will leave the supermarket. This is reflected in the messages passed back to the manager. The message indicating that the input sentence matches one of the instance's expectations contains the two CD-structures, corresponding to the two clauses of the sentence, separated.

```
(Message KITCHEN1 NO-MATCH NIL)
(Message
     SUPERMARKET1
     MATCHED
     ((Match-message
          ((CAUSE
               (ANTECEDENT
                    (EVENT
                         (ACTOR DUMMY-HUMAN4)
                         (ACT DO)
                         (TIME
                              (NAMED TIMEPOINT4)
                              (COMPARISON
                                   (BEFORE
                                        (NAMED TIMEPOINT3)
                                        (COMPARISON
                                             (BEFORE *NOW*)))) )))

               (RESULT
                    (EVENT
                         (ACTOR DUMMY-MALE3)
                         (ACT ATRANS)
                         (OBJECT DUMMY-THING3)
                         (FROM DUMMY-MALE3)
                         (TO DUMMY-HUMAN4)
                         (TIME
                              (NAMED TIMEPOINT3)
                              (COMPARISON (BEFORE *NOW*)))) )))
           (CONJUNCT FIRST)
           . pay2)
          (Match-message
             ((EVENT
                  (ACTOR DUMMY-MALE3)
                  (ACT PTRANS)
                  (OBJECT DUMMY-MALE3)
                  (FROM DUMMY-PLACE3)
                  (TO
                       (FOCUS (STATE THING))
                       (STATE
                            (STATENAME OWN)
                            (THING HOME1)
                            (VAL DUMMY-MALE3)
                            (TIME
                                 (NAMED TIMEPOINT6)
                                 (COMPARISON (BEFORE *NOW*))
                                 (COMPARISON (AFTER TIMEPOINT3)))) )

                   (TIME
                       (COMPARISON (AFTER TIMEPOINT2))
                       (NAMED TIMEPOINT6)
                       (COMPARISON (BEFORE *NOW*))
                       (COMPARISON (AFTER TIMEPOINT3)))) )
                (CONJUNCT SECOND)
                . getitems2)))


     No match by stereotype : KITCHEN1

     No new stereotype instances were activated

     Killing stereotype instance KITCHEN1
```

Since SUPERMARKET1 is the only remaining stereotype-instance, the topic-type of the text has now been firmly identified as shopping at the supermarket. If the summariser was entered at this point, as it would be during a full Scrabble run, it would be passed as data the

summary template and variable bindings of SUPERMARKET1, which expected all the input. The summary would be constructed from this data.

The next chapter describes the process by which CD-structures extracted from the input text are matched against the expectations of each stereotype. It contains a detailed discussion of the representation of the stereotype-prototype for shopping at the supermarket, which was used when processing this example.

5. Stereotype Application

Stereotype application is the process by which the set of expectations associated with each currently active stereotype-instance are compared with incoming CD-structures representing sentences of the input text. The results of the comparison may be that the stereotype-instance expected the contents of the sentence; that it expected part, but not all, of the sentence's contents; or that it expected none of the contents of the sentence. These results are passed back to the stereotype manager, as described in chapter 4.

The main issues to be considered here are: how is it recognised that a particular CD-structure matches an expectation; how are such successful matches recorded and what effects do they have on the system; what happens when there is no match between an incoming text segment and any of the system's expectations; what happens when an incoming CD-structure only partially matches the systems expectations? The way these questions are answered determines the operational basis of the whole text understanding process.

The strategy adopted for the recognition of expected material necessarily follows from the representation of Scrabble's stereotype-prototypes and stereotype-instances. The chapter therefore begins with a description of the representation of stereotype-prototypes. The description is based on a detailed example. This is followed by a description of the representation of stereotype-instances used within Scrabble, motivated by a preliminary discussion of the use made of the representations when examining incoming CD-structures. The chapter then moves to a detailed discussion of the mechanism by which individual utterance-expectations are compared with CD-structures extracted form input text sentences: this is followed by a description of the preprocessing which occurs when an utterance-expectation of the system is satisfied by part of the input text. Next there is a detailed description of the way the results of stereotype application are communicated to the stereotype manager. Finally a detailed example is presented.

Much of the material in this chapter is extremely detailed. It is hoped that the detailed material will provide some insight into the internal working of the program.

5.1 An Annotated Example of Stereotype-Prototype.

This section introduces the representation for stereotype-prototypes used in Scrabble. It is illustrated by a discussion of Scrabble's shopping-at-the-supermarket stereotype. This is, in essence, Scrabble's encoding of the Supermarket frame of [Charniak76b] described in section 2.4.

Another example stereotype-prototype, for texts about visits to zoos, may be found in Appendix B.

A stereotype-prototype is divided into three main sections. The first introduces the set of variable-expectations; the second describes is the set of utterance-expectations; finally there is some information connected with the generation of summaries.

The terms variable-expectation and utterance-expectation were introduced in section 3.1. Variable-expectations describe co-referring nominals which are expected to occur in an input text which shares the prototype's topic-type and the nominals expected properties. Utterance-expectations describe text segments which are expected to occur in an input text with the prototype's topic-type. The scope of the individual text segments expected by a Scrabble stereotype may be anything between a single noun adjective pair and a complete complex sentence.

Illustrative material from the program is indented at the right and left hand sides of the page, and is printed in `this font`. All the illustrative material has been taken from the data-structure used by the program. However, for the sake of brevity, the list of utterance-expectations has been slightly shortened, and some variables have been omitted.

Prototype name

The first (minor) section of a stereotype-prototype is its name.

```
(SUPERMARKET
```

This short name has no great importance: it is merely a convenient handle to use when manipulating the prototype.

Variables-expectations

The word VARIABLES indicates that what follows is a description of the variable-expectations which occur in the stereotype prototype's utterance-expectations.

```
(VARIABLES
```

By convention the first set of variable-expectation declarations are those describing the CD-nominals which are expected to occur in (the CD-structures constructed to represent sentences of) texts about the prototype's topic-type. They are followed by declarations which, in essence, describe the expected temporal interrelationships of the events described by the prototypes utterance-expectations.

Variables Expecting CD-nominals

Variables describing CD-nominals which are expected to occur in the representations of texts about the stereotype-prototypes topic-type have four parts.

The first is a name, which again is merely a handle facilitating easy reference to the variable elsewhere.

The second is the sorting[1] predicate of the variable. The predicate will determine whether a CD-nominal in a representation extracted from an input text sentence may be bound to this variable. The precise sense in which "bound" is being used here will be detailed later in this chapter. The predicate consists of two parts: a function and a value. The value typically belongs to the set of nominal "features" used in Cater's representation language (see section 2.3.6 [Cater82]), although some belong to the set of so-called "functions" in Cater's system. Cater's "features" are a set of about 25 classificatory primitives. They include such things as MALE,CONCRETE and BEAST. The "functions" (which are not described in any detail in [Cater82]) are not hierarchically organised, and cover such things as FOOD and STORE. Scrabble's predicate's (mathematical) function is expected to be a two-place predicate which can test a CD-nominal to determine whether it possess the specified quality.

Third there is a variable type. This specifies the circumstances in which a new instance of a variable may be created. A variable may expect only one object of the type to which it refers to occur in a particular text section, or it may expect a group of connected objects, or several unconnected objects. Examples of each kind will be given below.

Fourth there is a default. The default specifies a CD-nominal which should be used when generating a summary CD-structure for a text whose topic-type has been identified, in this case as shopping-at-the-supermarket, but which does not contain a specific mention of any object expected by the variable.

The first variable in the SUPERMARKET stereotype is called 'shopper'. It expects a CD-nominal with the feature HUMAN. (The name SCRIPT-FEATURE for nominal feature testing functions is a historic hangover.) As its type is CAN-BE-GROUP, it can be bound to groups or to more than one different person. Its default is DUMMY-HUMAN, which might appear in an English summary as "some people".

---

[1] "to sort" is used here in the sense of to place according to kind, rather than to put in, for example, numerical order.

```
(shopper
    (SCRIPT-FEATURE HUMAN)
    CAN-BE-GROUP
    DUMMY-HUMAN)
```

The next variable is 'purchase-item' which expects any CONCRETE object. Within the semantic feature markers used there is no means of identifying those objects which can be bought at a supermarket, as opposed to other kinds of concrete objects, say rocks, since this is an essentially pragmatic distinction. The variable is marked as REPEATABLE so there may be as many 'purchase-items' as are needed to match the story. Like DUMMY-HUMAN in shopper, DUMMY-THINGS indicates what the default CD-nominal is for purchase-items.

```
(purchase-item
    (SCRIPT-FEATURE CONCRETE)
    REPEATABLE
    DUMMY-THINGS)
```

The variable 'basket' has the typing predicate (MUST-BE-A BASKET). This indicates that any object bound to this variable must be an instance of the CD generic token[2] BASKET, or something which could refer to a basket. Of course this is not an entirely accurate way of representing the actual constraints in a real supermarket, for example it would be better if the constraint also allowed trolleys but disallowed shopper-owned baskets. This is not possible with the present system. Equally the current Scrabble representation language is not sufficiently expressive to represent the idea that whilst two or even three baskets is a reasonable number to expect the shopper to use, a hundred is not a number of baskets which should be classified as usual.

```
(basket
    (MUST-BE-A BASKET)
    REPEATABLE
    BASKET)
```

For 'fridge-shelf' the typing predicate is (SCRIPT-FUNC-OF STORE). SCRIPT-FUNC-OF tests to see whether a CD-nominal is marked as having the specified function. For example, the CD-nominals derived from the words "banana", "biscuit" and "tuna" have the function FOOD. The words "shelf", "fridge" and so on produce tokens marked with the function STORE.

```
(fridge-shelf
    (SCRIPT-FUNC-OF STORE)
    REPEATABLE
    SHELF)
```

---

[2]I mean by "CD generic token" the CD nominal token which underlies all instances of particular CD token. For example, when the AD-HAC analyser reads the word "BASKET" in the input text it will create a unique token, say BASKET27, to represent this occurrence of the word "BASKET". This token, BASKET27, is linked to another internal token with, by convention, the same name as the input word, in this case BASKET. This last token, BASKET, which I call a CD generic token, represents the meaning of "BASKET", in the sense that it is linked to all the information the system possesses about baskets. See [Cater82] Chapter 2 for more details.

The variable 'market' which describes the place where the shopping takes place has the type code UNIQUE which indicates that there can be only one such place in a text segment analysed by a single instance of this prototype.

```
(market .
    (MUST-BE-A SUPERMARKET)
    UNIQUE
    SUPERMARKET)
```

The remaining variables in the shopping-at-the-supermarket stereotype-prototype use similar sorting predicates and types to those introduced already. The defaults should be self explanatory.

```
(cashier
    (SCRIPT-FEATURE HUMAN)
    UNIQUE
    CASHIER)
(checkout
    (MUST-BE-A CHECK-OUT)
    UNIQUE
    CHECK-OUT)
```

Variables expecting CD time tokens

Variables which expect CD time tokens specify variable-expectations about the temporal interrelationships between utterances, as expressed in input texts by the tenses and aspects of various clauses, and by explicit conjunctions of time, like "before","after" and "until". The expected interrelations are specified using Cater's primitives BEFORE, AFTER, and DURING, described in section 2.3.5 of [Cater82], which should be referred to for a description of the time representation scheme which underlies Scrabble. Although Cater's scheme is quite elaborate it has many holes and cannot be regarded as a real solution to the many difficult problems of time representation.

Variables which expect CD time tokens have three parts.

As for variables which expect CD-nominals, the first element in the variable characterisation is a name.

The second element specifies the relationship between time tokens which may be bound to this variable and those which may be bound to other variables within this stereotype. It consists of a list of pairs of Cater's CD time relation primitives (BEFORE, AFTER, DURING) and the name of another time variable. It should be pointed out that these expected relations between time variables also serve as defaults for the time interrelations of the utterance-expectations in the stereotype, since they form a basic expected temporal framework for events described by the stereotype.

The third element is a 'type code, which is always TIME, allowing this class of variables to be distinguished from those which expect CD-nominals. this is necessary because of the different meanings and

forms of the two sorts of variables.

The set for "SUPERMARKET" is:

```
(gotime
   ((BEFORE getbaskettime))
   TIME)
(getbaskettime
    ((BEFORE getitemtime))
    TIME)
(getitemtime
    ((BEFORE paytime)
        (AFTER getbaskettime))
    TIME)
(basketitemtime
    ((BEFORE paytime)
        (AFTER getbaskettime))
    TIME)
(checkouttime
    ((BEFORE
         leavetime)
        (AFTER gotime)
        (AFTER getbaskettime)
        (AFTER getitemtime)
        (AFTER basketitemtime))
    TIME)
(paytime
    ((BEFORE leavetime)
        (AFTER checkouttime))
    TIME)
(leavetime
    ((AFTER gotime)
        (AFTER getbaskettime)
        (AFTER getitemtime)
        (AFTER paytime))
    TIME))
```

Utterance-expectations

The utterance-expectations of a stereotype-prototype are introduced by WITH-SLOTS. WITH-SLOTS refers to the locations in which input material which satisfies the expectations will be put.

```
(WITH-SLOTS
```

Each utterance-expectation consists of three parts. The first is again a name. The second is a pattern, which essentially specifies those utterances which satisfy the expectation. The third are demons, actions which should be performed if the expectation is satisfied by part of the input. Demons are not necessarily included in the specification of utterance-expectations.

Patterns are introduced by the word PATTERN. They are followed by either a single CD-derived structure, or a set of such structures introduced by the keyword ALTERNATE-PATTERNS. Each CD-derived structure consists of a legitimate CD-structure with some of its case slot fillers replaced by variables declared in the previous section.

Demons are introduced by the word DEMONS. The most common demon causes a new instance of the current slot and its pattern to be formed.

The first slot in the SUPERMARKET stereotype-prototype is named 'gotomarket'. It expects that the input text may contain a reference to the person doing the shopping going to the supermarket. In CD all verbs of motion cause the production of representations using the primitive PTRANS. This pattern expects (through the TIME variable 'gotime') that such a movement to the supermarket will occur before any of the other actions expected by the prototype.

```
(gotomarket
    (PATTERN
        ((EVENT
            (ACT PTRANS)
            (OBJECT shopper)
            (TO market)
            (TIME gotime) ))))
```

Next the 'shopper' is expected to pick up a 'basket'.

```
(getbasket
    (PATTERN
        ((EVENT
            (ACT PTRANS)
            (ACTOR shopper)
            (OBJECT
                (GROUP
                    (1 shopper)
                    (2 basket)))
            (TIME getbaskettime) ))))
```

Next is the slot 'getitems'. The expectation associated with 'getitems' is that the input text will contain utterances describing the shopper moving things ('purchase-items') from places where they are stored in the supermarket (variable 'fridge-shelf') to a basket (variable 'basket'). The demon REPEAT is attached to 'getitems'. REPEAT allows a stereotype-prototype to have an expectation that an input text may contain arbitrarily many utterances following the same prototypical pattern. In this case the stereotype will accept as usual as many occurences of the 'shopper' moving goods from shelves and fridges to the 'basket' as appear in the input text. REPEAT operates by creating a new CD-derived and associated slot when the most recently created version of the utterance-expectation is satisfied. In general new instances of all the variables in the pattern will be created, but the creation of such new variables may be inhibited by giving arguments to REPEAT specifying the variables which should be the same as in the original. Thus, in detail, this utterance-expectation might be read as: the same shopper may be expected to repeatedly move new concrete items from new storage places into the same one basket.

```
(getitems
    (PATTERN
        ((EVENT
            (ACT PTRANS)
            (ACTOR shopper)
            (OBJECT purchase-item)
            (FROM fridge-shelf)
            (TO basket)
            (TIME getitemtime))))
    (DEMONS
        (REPEAT shopper basket)))
```

The slot 'checkout' uses the ALTERNATE-PATTERNS device to specify the different ways that the expectation that the shopper will go to the check-out with the purchase items may be satisfied. The two expressions correspond to "John took his purchases to the checkout" and "John went to the checkout". Perhaps the necessity of expressing both possibilities is a weakness in the system. It is a consequence of the system not using a non-stereotype-based inferencer.

```
(checkout
    (PATTERN
        (ALTERNATE-PATTERNS
            ((EVENT
                (ACT PTRANS)
                (ACTOR shopper)
                (OBJECT
                    (GROUP
                        (1 shopper)
                        (2 item)))
                (TO checkout)
                (TIME checkouttime)))
            ((EVENT
                (ACT PTRANS)
                (ACTOR shopper)
                (OBJECT shopper)
                (TO checkout)
                (TIME checkouttime))))))
```

Finally there are expectations that the shopper will pay and leave.

```
(pay
    (PATTERN
        ((CAUSE
            (ANTECEDENT
                (EVENT
                    (ACTOR cashier)
                    (ACT DO)))
            (RESULT
                (EVENT
                    (ACTOR shopper)
                    (ACT ATRANS)
                    (OBJECT *MONEY*)
                    (TO cashier)
                    (TIME paytime)))))))
(leave
    (PATTERN
        ((EVENT
            (ACT PTRANS)
            (ACTOR shopper)
            (FROM market)
            (TIME leavetime)) ))))
```

Information for use when generating summaries

The final element of a stereotype-prototype in Scrabble is the template summary representation for texts about the prototype's topic-type. It is introduced by the word SUMMARY. The summary template has only been included here for the sake of completeness: its contents and their use will be discussed in Chapter 6.

```
(SUMMARY
    (TIME-DURATION gotime leavetime)
    (MAINCHARS shopper)
    ((EVENT
          (ACTOR shopper)
          (ACT PTRANS)
          (OBJECT shopper)
          (TO market)
          (TIME . gotime)))))
```

This section has described how stereotype-prototypes used by the Scrabble system are represented. The representation used is primarily intended to facilitate the construction of stereotype-instances. This construction is discussed in the next section.

5.2 The Internal Representation of Expectations

This section describes the process by which stereotype-instances are formed from stereotype-prototypes. It focuses on the form of the representation of stereotype-instances, since the representation determines, to a large extent, the shape of the stereotype application process. In turn the form of representation used is intended to facilitate the recognition of input CD-representations which satisfy utterance-expectations of stereotype-instances. This recognition process will be described in the next section.

In essence the formation of a stereotype-instance from a prototype consists of the construction of an identifiable distinctive copy of prototype. It is necessary to make the copy identifiable and distinctive so that it can be distinguished from other instances, and in particular different instances of the same prototype. In addition, the existence of the REPEATABLE variable-expectation type and the REPEAT demon necessitate the formation of distinctive identifiable copies of variable-expectations and utterance-expectations.

The initial formation of a stereotype-instance has four parts.

First there is the construction of a unique name for the instance: this an entirely straightforward matter, based on the use of a primitive LISP feature. It will, therefore, not be discussed further.

Next the variable-expectations specified in the stereotype-prototype are processed to form a set of uniquely identified variables. Each has the information in the associated prototype converted into an easily manipulable form. In addition, each has storage allocated which will be used to hold information collected whilst processing the input text.

Third, the 'slots' of the stereotype-prototype are dealt with. They are given unique names; the unique names of any variables in their PATTERN or DEMON sections are converted to the unique names assigned to those variables in the instance being formed; and they are converted into a form which both provides receptacles in which to store any input CD-structure satisfying the utterance-expectation they express, and facilitates the process of recognising such CD-structures.

Finally, the summary template is processed to bind into it the unique names given to the variables it contains.

The remainder of this section will discuss the processing of variable-expectations, utterance-expectations and the summary templates, although little attention will be given to the last of these, as it is a fairly straightforward process.

Processing variable-expectations

I will deal first with variables expecting CD-nominals.

As was pointed out in the previous section the specification of a variable expecting a CD-nominal in a stereotype-prototype has four parts: a name, a sorting predicate, a type and a default.

The first stage of processing a variable which expects a CD-nominal is to create a unique name for it. A record is kept relating these unique names to the original names in the stereotype-prototype. This record is used when processing the utterance-expectations of the prototype.

Next the sorting predicate is extracted from the prototype. The first part of the sorting predicate is expected to be a function of two arguments. A currying technique[3] is used to produce a single argument lambda-expression which calls the specified function with, as its first argument, the argument of the lambda-expression, and as its second argument the second element in the sorting-predicate specification in the stereotype-prototype. The lambda-expression is then attached to the unique name created for the instance of the variable.

Once the sorting-predicate has been processed the type and default in the variable specification are attached to the unique name.

---

[3]See, for example, [Brady77].

Two further slots are attached to the variables name at this point. The first will provide a means, when processing the text, of associating this variable instance with any other instances of the same variable specification extracted from the prototype. Such instances may be created for a variable whose type is CAN-BE-GROUP or REPEATABLE and are called SISTER-VARIABLES. Secondly a slot is created through which any CD-nominal extracted from the text may be bound to the variable instance.

Variables of type TIME are processed in a more straightforward manner. Each is given a unique name, and a record is kept of the correspondence between the unique name and the original specified in the stereotype variable. When all the variables have been so named, the time constraints between variables are processed. Each original variable name associated with one of the relation primitives (BEFORE, AFTER, and DURING) is replaced by the corresponding newly created name. The resulting list of relationships between a given variable and others in the stereotype is then attached to the unique name. Finally slots are attached to the variable name giving the type (TIME in this case), and for a binding.

Two technical points are worth making. First, the variables are stored as an association list on the LISP property list of their stereotype-instance. This means that finding a variable involves searching, but the implementation was chosen because it was considered that the number of variables associated with given instance would be small. It is not clear to what extent, if at all, the use of, say, hash addressing techniques (see, for example, [Bornat79]) would improve the speed of the system, but it is a matter worthy of investigation. Second, the various attributes of a given variable, i.e. whether, and to what, it is set, its type, its sorting predicate or time constraints and so on, are also stored as association lists. It would be computationally more efficient, both in time and in space, to store this information as fixed format records, say of the style proposed by [Charniak80].

Processing utterance-expectations

The processing done to convert the description of the utterance-expectations in a stereotype-prototype will be illustrated by means of an example. I will consider just one typical slot of the prototype for shopping at a supermarket, introduced in section 5.1. The slot is 'get-items':

```
(getitems
    (PATTERN
        ((EVENT
              (ACT PTRANS)
              (ACTOR shopper)
              (OBJECT purchase-item)
              (FROM fridge-shelf)
              (TO basket)
              (TIME getitemtime))))
    (DEMONS
        (REPEAT shopper basket)))
```

The first point to be made is that two separate data-structures represent such a slot in the stereotype-instance. One is a framework in which an utterance which satisfies the expectations represented by the slot description will be recorded. I will refer to this data-structure as the ***slot instance***. When a new stereotype-instance is formed it contains only the information that it is unset. The other data-structure is a pattern/action pair which in Scrabble, following the Schankian tradition, is called a ***request***. A data-structure is also set up linking slot-instances to the corresponding requests.

The requests are programs which are intended to recognise the CD-representations of the set of utterances in an incoming text which satisfy a given expectation of the system, and to act appropriately if this occurs. Acting appropriately consists of recording the utterance in the slot instance and recording any relevant information about the way the utterance satisfied any variable-expectations. Additional actions which should be performed if an expected structure is recognised in the text are indicated explicitly in the DEMONS section of the description of the slot in the prototype. The sorts of thing these demons do was indicated in the previous section.

The request patterns consist of CD-derived structures with parts replaced by stereotype variables. Typically these parts are those where a CD-nominal representation should occur. The patterns are extracted from the PATTERN section of an utterance-expectation specification by replacing all variable names in the description in the prototype by the corresponding unique variable name created when processing the variable-expectations.

At the same time any variable names associated with demons are replaced by the unique versions of their name.

To summarise this section on the construction of stereotype-instances from stereotype-prototypes, from the point of view of the analysis of incoming text a stereotype-instance is represented by three data-structures, each linked to a unique name created for the instance. First there is a set of variables; second there is a set of slot-instances, initially set to an unset marker; third there is a set of

requests. This section has concentrated on the way these data-structures are extracted from a stereotype-instance: the remainder of this chapter concerns the way the expectations represented by these data-structures are manipulated whilst processing an input text.

5.3 Marrying up Expectations and the Text

As noted in Chapter 3, one of the essential features of the Scrabble system is that it never employs inference rules in the attempt to match its expectations against the representation of incoming text. This means that the process of recognising that an utterance is expected is purely one of pattern matching, unlike, say, [DeJong79] where it is also a process of selecting rules which will assist or allow recognition.

The pattern matching technique used is rather similar to that of [Cullingford78]. It has two stages, one of which essentially corresponds to Cullingford's Backbone match and the other to his Rolemerge stage. In Scrabble the first stage matches request's patterns against incoming CD-structures without regard for previous context. The second stage merges the results of the first stage into the stereotype-instance, disregarding any apparent first stage match which conflict with any previous context, as represented by the stereotype-instance's current state. The matching process is repeated for each new sentence read whilst the stereotype-instance is active.

In the first stage, the system iterates through all the active stereotype-instances and through each request currently associated with each of these instances, and records the results of the requests' patterns attempts to match the representation of the current sentence. The form in which these results are recorded will be described shortly. This process is then repeated for any newly activated stereotype-instances (see section 4.3) I will describe how the pattern matching process proceeds by consideration of an example.

The example pattern is from an instance of the visiting-a-zoo stereotype-prototype. It express the expectation that the people who go to the zoo will look at animals.

```
((EVENT
    (ACT ATTEND)
    (ACTOR goer1)
    (OBJECT *EYE*)
    (TO animal1)
    (TIME looktime1)))
```

The objects in lower case in the pattern are variables. They are instances of the following:

```
(goer (SCRIPT-FEATURE HUMAN) CAN-BE-GROUP DUMMY-HUMAN)
(animal (SCRIPT-FEATURE BEAST) REPEATABLE ANIMAL)
(looktime ((BEFORE leavetime) (AFTER arrivetime)) TIME)
```

where 'leavetime' and 'arrivetime' are the variables associated with the

times at which the 'goer' arrived at and left the zoo, should these be identified while processing the input text. The numerical suffixes on the variables in the pattern are the product of the process by which variable instances are give unique identifying names during their construction.

We will consider how this pattern will match the CD-structure:

```
((EVENT
    (ACTOR HUMAN-JOHN)
    (ACT ATTEND)
    (OBJECT *EYES*)
    (TO MONKEYS1)
    (TIME
        (NAMED TIMEPOINT4)
        (COMPARISON
            (BEFORE *NOW*)))))
```

The first step in the pattern matching process is the examination of the top-level structuring primitives of the pattern and CD-structure respectively. In this case they match, of course. For EVENTs, STATEs and CONJUNCTs no further processing is done at this stage, and matching proceeds on the rest of the structure. For CAUSEs, ANTECEDENT and RESULT paths are followed, and the process recurses on the results.

The next step is to iterate through the case-slots of the pattern, extracting the associated filler from the pattern and the corresponding one from the CD-structure. If there is no corresponding case slot the match fails. What happens next depends on the nature of the fillers extracted both from the pattern and from the CD-structure. In our example, the first to be extracted will be the primitive ATTEND: since it is not a variable it must match the corresponding case slot filler in the CD-structure exactly, and indeed it does. Next we extract 'goer1' from the pattern. Since it is a variable, the process of checking it is more complex. The variable sorting predicate is applied to the CD-nominal[4] to determine if it is acceptable. Again in this case it is, since HUMAN-JOHN possess the feature HUMAN.

Because the CD-nominal, HUMAN-JOHN, is a possible match for the variable 'goer1', the name of the nominal and the variable instance are recorded.

Continuing through the example, *EYE* is the next case slot filler in the pattern. Like ATTEND it is not a variable, and hence must match the corresponding item in the CD-structure exactly. Again there is a match. Next 'animal1' is extracted from the pattern, and is

---

[4]In this simple case: if the case slot filler extracted from the CD-structure was not a simple CD-nominal more complex processing (which will be described below) would follow.

processed in the same way as 'goer1'.

Finally, we come to the TIME case role. Unlike other types of variable, TIME variables are matched in a way which takes account of previous textual context. In particular, the variables' time constraints are compared with any bindings of other time variables.

In the example the variable 'looktime1' must occur AFTER an instance of 'arrivetime' and BEFORE an instance of 'leavetime'. The particular variable instances will have been specified when creating unique names for variables when first setting up the stereotype-instance. In order to check the constraints, first the instance 'leavetime ' is looked up. If this is bound it is checked that the specified relationship holds between the token to which it is bound and the prospective binding for 'looktime1'. If the specified relationship does not hold the whole match fails. If the instance of 'leavetime' is bound but no relationship can be discovered between it and TIMEPOINT4 (the prospective binding for 'looktime1' in the example) it is assumed that the constraining relationship holds. If the instance of 'leavetime' is unbound its time constraints are examined to try to check that acceptance of TIMEPOINT4 as the binding for 'looktime1' will not make it impossible to subsequently satisfy all the constraints associated with 'leavetime1'. This would occur if some other variable was bound to a value BEFORE TIMEPOINT4, but was constrained to be AFTER the 'leavetime' instance. The process is then repeated for 'arrivetime'.

Checks are also made with any variables of type TIME which have been bound whilst trying to match the current pattern, but in practice most patterns have only one variable of this type.

There are five features of the pattern matcher which are not exploited whilst processing this simple example. Two concern the processing of more complicated CD-structures. Two allow patterns to be constructed which will match a wider range of CD-structures than those illustrated so far in this section. The fifth allows the possibility of successful matches after certain kinds of match failure.

First, if there is another CD-derived structure, which would be complete if it occurred alone, embedded in the pattern, the pattern matcher simply recurses on the embedded lower-level CD-derivative and filler of the corresponding case-slot in the input CD-structure.

Second, if a CD GROUP occurs anywhere in the CD-structure it is converted into a composite CD-nominal which indirectly references the members of the GROUP, before matching begins. If there is an attempt to match a pattern which contains a GROUP against a composite nominal token of this kind, every combination of text-derived and pattern group members is tried. Obviously if there is a GROUP in the pattern any attempt to match it against anything other than a composite token or a dummy filler which is compatible with a composite filler will fail.

Thirdly, the primitive act DO is taken to match any primitive act whether it occurs in the pattern or the incoming CD-structure.

Fourthly, if a request has ALTERNATE-PATTERNS (for example requests constructed from the 'checkout' slot in the example stereotype-prototype of section 5.1), each pattern is matched against the current CD-structure until one succeeds or there are none left.

Finally, if there is a failure to match at any point, an attempt is made to find a CD-structure satisfying the prediction embedded in the input CD-structure. This is done by iterating through the case slots of the CD-structure looking for a slot filled by a CD-structure, as opposed to a single nominal. If such a CD-structure is found, the pattern matcher is called on it and on the original, failed, pattern.

To conclude this section, the form in which the results of the pattern matching process are passed back is as follows. If none of the set of patterns associated with a request matches any part of the representation of the sentence the result is simply null. However, if some part of the representation does match one of the patterns, the result consists of the parts of the representation which matched the text together with a list which indicates to which objects in the matched CD-structure variables were bound in order to achieve the match. This device of only passing back those parts of the representation which have been matched effectively performs the function of subproposition analysis.

5.4 The Collation of Satisfied Expectations

If an incoming CD-structure matches the pattern of a request, the next stage is to integrate the results of the match with the request's stereotype-instance, or, perhaps, reject the match if its integration would cause the stereotype-instance to become inconsistent in some way. The process of integration and checking which achieves this is called ***collation***.

The process of collation is fundamental to the operation of Scrabble because it is the primary means by which the referents of anaphora in the input text are determined by the program.

If an expectation of a stereotype-instance is apparently satisfied three actions are taken. First, the request representing the satisfied expectation is removed from the instances's active request list. Second, any demons associated with the request are executed. Third, the results of the match are recorded. This section deals with each of these actions in order.

The modification of the active request list caused by the recognition that the current sentence satisfies the expectation

represented by a particular request is quite straightforward. The relevant request is merely removed from the list of unsatisfied requests. No attempt is made, for example, to order the requests so those whose expectations are most likely to be satisfied are examined first or remove any which correspond to causal or temporal precursors of the request whose expectations were satisfied. There are two reasons for this. Firstly, it is quite common for the order of sentences in a text to be different from the temporal order of the events which they describe. [Cullingford78], for example, cites newspaper stories as a type of text where it is rare for the order of the text as a whole to correspond to the temporal order of the events described. In consequence, any attempt to remove requests may result in a perfectly ordinary utterance being classified as unusual because the request which corresponds to its expectation has been removed. Secondly, the comparison between request patterns and the incoming CD-structure is organised in such a way that the order of examination of requests has little effect on processing time.

Immediately after it has been recognised that some part, or all, of the CD-representation of the current sentence satisfies an expectation represented by a request, the request is examined to determine if it has any demons associated with it. If it has, they are evaluated at this point. Typically they cause a copy of the current slot instance and its associated request to be created, perhaps causing the generation of new instances of some of its associated variables. For example, consider the slot 'getitems' of section 5.1:

```
(getitems
    (PATTERN
        ((EVENT
             (ACT PTRANS)
             (ACTOR shopper)
             (OBJECT purchase-item)
             (FROM fridge-shelf)
             (TO basket)
             (TIME getitemtime))))
    (DEMONS
        (REPEAT shopper basket)))
```

There are no particular restrictions on the effects of the demons: they are simply LISP functions. At present REPEAT and ADD-SLOT are the only demons used by the system. REPEAT causes a new instance of the current slot and its associated request to be created (A fuller description of REPEAT was given in section 5.1.) ADD-SLOT causes a completely new slot to be introduced when the expectation of the current slot is satisfied.

The final phase of processing satisfied expectations, the recording of results, is performed after all the requests derived from all the active and newly-activated stereotype-instances have been compared with the CD-representations constructed from the text sentences. The needs of summarising, together with the lack of any subproposition

analysis, as such, of incoming representations, provide the rationale for the postponement of result recording until this late stage. A particular requirement of summarising is that we do not want to repeat any information: i.e. a good summary is as short as possible[5]. When several stereotype-instances have been active during the processing of a given sentence, if they merely record information as if no other stereotype-instance were active, there is a danger that information which is recorded as expected in one instance will be reported as unexpected by another stereotype-instance. Difficulties arise when generating a summary from the latter instance, particularly in the more complex cases where, for example, both instances have matched part, but not all, of the representation of a particular sentence. This problem can be circumvented if the results of all the matches (or failures to match) are collated before any propositions are recorded in an instance.[6] This part of the collation process proceeds by comparing all satisfied expectations with a copy of the complete CD-representation of the current sentence. The parts of the sentence representation which satisfied each expectation are removed from the copy. When all the satisfied expectations have been processed, the remaining parts of the CD-structure are marked as not expected by any instance.

Suppose we are progressing through a text and the only active stereotype-instance is one of the zoo prototype, and we encounter the sentence:

(1) After he saw the lions, he went to a restaurant.

From the point of view of the instance of the zoo stereotype, the whole of the sentence is expected with the second clause corresponding to an expectation that the zoo will be left at some point, it being immaterial for where. However, from the point of view of the newly activated instance of the restaurant stereotype (whose activation will be triggered by the word "restaurant" in the sentence) "After he saw the lions" is unexpected. By postponing collating the successfully matched expectations until the end of sentence processing it can be recorded that this proposition was successfully predicted by another instance. Then, assuming neither of the instances has been ***killed*** (see section 4.4) before summarising has begun, when a summary is constructed it can easily be determined that the proposition about the lions should not be included among the unexpected utterances in the section of text which the restaurant stereotype was used to analyse.

---

[5]I do not mean by this that needless repetition would be acceptable in another domain; rather that when producing summaries the need to avoid tautologies is felt particularly acutely.

[6]In fact only partial failures to match are dealt with by the collation process. Total failures to predict any part of a sentences are handled in a rather more straightforward way, as described below.

The remainder of the processing done to accomplish the recording of the results of the matching of the text against the different instances' expectations is fairly straightforward, but is crucial because of the role it plays in determining the referents of anaphoric expressions.

The first stage is to update, if necessary, the bindings of any variable instances which occurred in the patterns of any requests which correspond to satisfied expectations. Usually such variables will merely have been bound to an individual CD-nominal during pattern matching, and in such cases all that needs to be done is to check that the previous binding of the variable and the nominal could co-refer. If they could, the variable is bound to the most specific of the two. I will describe what happens if they cannot co-refer below. This selection of the most specific CD-nominal is Scrabble's primary mechanism for associating anaphoric expressions, in particular pronouns, in the input text with their referents.

If only those parts of the input text which are recognised as usual by a particular stereotype-instance are considered, this part of the collation process in effect produces a representation of the input in which all pronouns associated with a particular variable-expectation are replaced by the most specific object associated with that variable-expectation in the topic-manifestation. For example consider:

(2) John went to the zoo. He saw a lion.

Both 'he' and 'John' are associated with the same variable-expectation in the same topic-manifestation (of the visit-to-the-zoo topic-type). Therefore in the corresponding stereotype-instance the text is represented as if the anaphoric construct, "he", was replaced by the more specific proper name "John". That is it is represented as if the input text was:

(3) John went to the zoo. John saw a lion.

The present mechanism for determining whether two nominals can co-refer is quite crude however. The mechanism checks to see whether either CD-token has a semantic features (like MALE,HUMAN,PLURAL etc.) which could *not* be associated with the other. The AD-HAC semantic feature scheme, whilst sufficiently powerful to allow this mechanism to operate for most straightforward cases involving pronouns, is not sufficiently powerful to handle very complex cases involving definite noun phrases. For example, the present mechanism could not be used to tie together,say, "Ronald Reagan" and "the President of the United States on the first of October 1982". Hence Scrabble is incapable of resolving such complex anaphora.

Slightly more complex processing occurs if a variable instance has either been previously bound to a CD GROUP or a binding to a group token has been set up during pattern matching (see section 5.3). In the latter case, if the variable instance has previously been bound to

a single nominal, a set of interlinked variable instances is created referring to both the group token and separately, to each of its members. However, this will not be done if either the variable instance is of type UNIQUE or if the previous binding could not be a member, or the whole, of the same GROUP. Conversely, if the new binding derived from the text is a single CD-nominal, and the previous binding was a group, the set of variable instances created to refer the group's members is scanned, and if necessary and appropriate, extended until a suitable one to bind to the new CD-nominal is found.

The processing of new and old bindings derived from plural indefinite anaphors such as the word "they" proceeds in a rather similar way. The token representing "they" is treated as a single group token, and then processed as before. However, the actual token representing "they" may not be recorded if its assumed co-referent is more specific, for example if it is derived from a complete definite noun phrase, because the information provided by "they" is redundant.

There are some special complexities concerned with the processing of variables of type TIME. Full CD-propositions may occur embedded in time tokens, so that during their processing manipulations corresponding to the collation of pattern matches, described above, must be performed to avoid the misclassification of such embedded propositions as expected or unexpected. In particular, such a proposition which was expected should be removed so that only information which might be useful for the summariser is recorded in the variable binding. Ideally this information would be confined to that needed to determine the tense and aspect of summary sentences, and the temporal interrelatedness of different instances, in so far as this is relevant to the construction of the summary; it should not determine the temporal relationships between individual propositions in the text, where they have been linked by conjunctions of time and so on. This compression is to a large measure achieved in a fairly natural way by the use of the collation mechanism. However some redundant information often remains after collation because of the difficulty of calculating exactly which information will be necessary to compute a tense and aspect for the summary.

At the same time, the time constraints between the various time variables of the stereotype-instance are checked.

Checks, like those on time constraints and on the compatibility of new and old variable bindings may, of course, cause an utterance which appeared to satisfy an expectation if only local constraints are taken into account to fail because it does not satisfy more global constraints. Such failure may occur if, for example, the old and new bindings are both derived from definite noun phrases which cannot co-refer, but which share the features tested by the typing predicate of the variable (see section 5.1). The effect of such a failure is to cause the abandonment of the whole of the attempt to record the successful matches, followed by the recommencement of the collation process

with the offending match removed. This rather heavy-handed approach is necessary because of the difficulty of designing a mechanism which will unwrap the effects of processing such mismatches without excessive cost in the cases where no such unwrapping is necessary. In fact in the present implementation difficulties may still arise with the reinstatement of previous variable bindings, but this is largely a problem with the lowest level of the implementation, and could be cured by a fairly straightforward redesign.

The result of integrating successful matches into the variable bindings of the active stereotype-instances is a data structure which pairs, for each match, the nominals and time tokens which occurred in the CD-structure with the variables to which they are bound[7]. This structure is used to generate the final record of the satisfaction of utterance-expectations. It consists of the structure with all the bound objects replaced by the variable instances to which they are bound. This latter structure is not, in fact, used in the present project other than as a debugging aid, but in an implementation which included a non-stereotype-based inferencer it would provide the best means of indicating, for example, how the stereotype-based system had referred anaphoric expressions in the original text.

5.5 Communication with the Stereotype Manager

This section describes how the results of the processes treated in sections 5.3 and 5.4 are communicated to the stereotype management process discussed in Chapter 4. The stereotype manager, from this point of view, requires the information to perform two primary tasks. The first is to remove from further consideration any instances whose expectations no longer appear to fit the text. The second is the recording for subsequent processing by the summariser of utterances which were not expected by any instance.

The communication is achieved by what might be called a 'message passing' system.

Messages may be dispatched at two different times. Firstly, at the end of the examination of the requests associated with a stereotype-instance, if no request has been found which predicts any part of the current sentence a message is dispatched which indicates this. If there is a match, even if it is partial, no messages are dispatched until right at the end of the application process. Message passing must be done right at the end of the processing cycle for a sentence because the nature of the messages to be sent can be affected by failures which

---

[7]This binding can, of course, be indirect: if the nominal in the CD-structure is anaphoric in nature the variable may be bound to some more specific CD-nominal with which it is thought to co-refer.

occur when a match is being integrated into an instance during the collation process. Whether such a failure has occurred cannot be determined until this stage.

When the collation process has finished, messages are dispatched to the manager which indicate, for each instance which registered a partial or complete match at the end of the first phase of pattern matching, the parts of the sentence's CD-structure which were genuinely expected. For partial matches, that is those cases where, after complete integration, it has been determined that an instance had successfully predicted some parts, but not all, of the current sentence, an indication is passed over of how the part which was predicted may be extracted from the complete representation of the current sentence. Also at this stage, the manager is informed if, as a result of the collation process, any instance has joined the class of those which predicted no part of the current sentence. Finally, if some parts of the CD-structure were not matched by any instance, the manager is sent a message for each instance which did predict some part of the structure, informing it of this totally unpredicted part.

It may seem more natural to inform the manager of those parts of the CD-structure which *each* instance failed to predict, and indeed this is probably the best information to communicate to a non-stereotype-based inference system. However, within the context of summarising, if this were done there would be problems, similar to those described in section 5.4, with ensuring that the summariser does not repeat itself. The manager essentially takes messages which indicate that a text segment was unexpected and records them and their textual relation to other, expected, parts of the text. Recording parts of the text which were predicted by one instance but not by another along with those which were not predicted by any instance only complicates the task of the summariser. This is because any utterance which has been recognised as expected by any instance which is active or suspended at summarising time need to be considered as unexpected material which merits incorporation into the summary produced. The separation of utterances expected by some, but not all, of the active stereotype-instances from utterances which were not expected by any instances is not totally trivial, because of modifications which may be be made to the original structures during the collation process. Additionally, such messages about partial match failure are of little use to the management process because the criteria for the killing or suspension of instances are based on complete failures to predict the content of sentences.

The disadvantage of the scheme adopted here is the additional complexity of killing instances it entails. Suppose some instance is killed at a particular point in the processing of a text. Suppose further that it was the only instance, among several active ones, to have matched part of a previous sentence. The CD-structure representing the part of the sentence this instances predicted must be re-classified as not having been predicted by any instance. The present

implementation is incapable of this.

5.6 Concluding Remarks

In essence the process of stereotype application is a fairly simple operation. It is primarily a process of pattern matching. The elaboration of the process in Scrabble arise form two sources. First the need to consider matches between anaphoric expressions in the input text and variables in stereotype-instances already bound to specific nominals. Second, the need to allow for partial matches between the expectations of an instance and the input material, and the communication of the results of such partial matches to the stereotype manager.

6. The Construction of Summaries

This chapter describes how Scrabble converts the set of active and suspended stereotype-instances and the other data structures formed by the stereotype management process into a summary representation and how this representation is passed over to the AD-HAC generator for conversion into the output summary text.

The summary generation program differs crucially from any of the programs of the analysis phase, perhaps reflecting a general difference between the processes of language analysis and generation. During the analysis phase all decisions are, or at least would ideally be, tentative. That is, at any point the decision to assign a particular utterance to the set of usual ones about a given topic-type, or to activate a particular stereotype-instance, and so on, may be undone when the context has been extended by subsequent utterances. This is rather similar to the intra-sentential processing of the AD-HAC analyser, which, in turn is derived from the Preference Semantics approach of [Wilks73], [Wilks75a], [Wilks75b] etc. In contrast, when the summariser is called, no further information will become available, and the emphasis is thrown onto making definite decisions with the highest probability of being correct given the information available. Because the summariser is always called after the whole of a text has been analysed, there is no point in keeping options open in the hope of having better information later. This is also reflected, to some extent, in the structure of the AD-HAC generator.

6.1 Task and Structure of the Summariser

The Scrabble summariser takes as input three data-structures. The first is a list of active and suspended stereotype-instances. The second is a list of CD-structures derived from the input text which were not expected by an active instance when they entered the Scrabble analysis system. The third is a data-structure which represents the textual interrelationships of the propositions in the input text which gave rise to the elements of the other two data-structures. The task of the summariser is to construct a ordered linear sequence of CD-structures (the ***summary representation***) which can be converted into an English text.

The summarising program consists of four main parts.

The first essentially extends part of the work of the stereotype management process. It attempts to chose between related instances used to analyse the same text segment by extending the techniques which are used by the manager to decide whether to suspend or kill stereotype-instances during the analysis phase. In particular it attempts to reduce the number of instances in the set to be considered to one from each suggestion packet, if this has not already been done.

It differs from the equivalent process in the stereotype manager in that it operates by making stronger assumptions so the number of instances to be examined during summary representation construction is minimised. Much of the processing of the manager follows the general strategy of postponing firm decisions as long as is practicable in the hope that more reliable decisions may be made when more information is available. Once summarising has commenced no further information will become available from the input text. Hence the summarising process cannot postpone a decision about which stereotype-instances to use in the summary. Rather it must attempt to select the a best set of instances given the information available at the end of the text analysis process.

Once this weeding of instances has been done, the elements which will form the summary representation can be identified, even though their exact content has not been computed. The summary representation will consist of a CD-structure derived from each of the remaining stereotype-instances plus any input CD-structure which was not expected by any of the stereotype-instances used in the formation of the summary representation. The next task undertaken by the summariser is the construction of the ordering of the elements which will be used in the summary representation. The order is computed using both any temporal information which may be derived from the original text and the data-structure representing the textual order of the input material.

Thirdly, each stereotype-instance is processed to produce a CD-structure summarising that part of the input which it expected, and these CD-structures are linked with the ones derived from the unexpected parts of the text to produce CD-structures which will form sentences in the output summary text.

The final part of the summariser employs some strong assumptions about the coherence of the input text to do additional resolution of anaphoric references in the input. The additional anaphora resolution extends the summary representation so that it contains sufficient information to allow a coherent output summary text to be produced.

The product of the processing is an ordered set of CD-structures, each of which corresponds to what will become a sentence in the final natural language summary.

Each of the four parts of the summariser will be dealt with in turn in the first part of this chapter. This description will be followed by some fairly brief notes on the use made of Cater's generator (see [Cater82]) to convert the summary representation into English. The chapter ends with a detailed example run of the summariser.

6.2 Stereotype Selection

One of the problems with the management techniques described in chapter 4 is that the selection between different stereotype-instances activated by the same suggestion packet will not take place until at least two sentences have been read following the sentence in which the word which triggered the suggestion occurred. For once a stereotype has been activated, only when it has failed to predict anything for two successive sentences is any processing done to determine whether the topic-type of a text section is, in fact, the topic-type of this particular stereotype-instance. The details of this mechanism were given in section 4.4.

This scheme works for the middle of a text but from the point of view of summarising, problems are presented in that if the discussion of a topic in the text lasts less than three sentences, and there is no subsequent text, no topic-type selection will have taken place, and all the instances which were activated by the first of the sentences will remain on the list of active instances. Furthermore, if an instance has failed to predict some utterances, but has never been applied to two successive sentences which either it failed to predict or which did not re-suggest it, it will not have undergone scrutiny for topic-type selection. Admittedly this will only occur at the end of the text, but it is none the less serious for that.

The technique used to force topic-type selection in these limit cases extends the method used by the stereotype manager to determine, when considering whether to kill or suspend an instance, whether there was any instance activated by the same suggestion packet which has had more success at predicting the contents of the text than the instance under consideration. Thus if there are two stereotype-instances A and B, whose activation was triggered by the same suggestion packet[1], this processing will remove A from the list of active instances to be used to summarise the text if:

1) there is some utterance in the text which B predicted but A did not;

and:

2) there is no utterance in the text which A predicted but B did not.

Once this weeding process has been completed the set of stereotype-instances which will be used in the construction of the summary representation is complete. The next three sections describe how the summary representation is constructed from the remaining stereotype-instances, the representations of unexpected utterances in the text, and the representation of the overall structure of the input text.

---

[1]see section 4.1.

6.3 Producing an Order for the Elements of the Summary

Once the processing of section 6.2 has been completed, all the elements which are to be used to make up the summary are known. They are the remaining set of active and suspended filled-in stereotype-instances, the CD-representations of the utterances which various instances failed to predict, and the textual and temporal relations between all of these. The next two tasks of the summary construction program are to form from this information the CD-structures whose content will eventually be expressed in the English of the summary, and the computation of the order in which the natural language sentences will be output.

The two tasks can essentially proceed independently, that is on a sequential machine, either task can be performed first. The work of ordering can proceed independently from the construction of the CD-structures because each of the elements which will form the summary representation has been identified and named, in the sense that each has a unique identifying property. Either they will be generated from a particular filled-in stereotype-instance and its associated data-structures, or they will consist of the substantially unmodified representation of some segment of the original text. The latter case corresponds to an unpredicted text segment and will have a unique name generated for it. Stereotype-instances also have a unique name assigned to them when they are first constructed (see section 5.2), and as the ordering process works on names, rather than on the CD-structures themselves, the two tasks can be pursued independently.

The term ***summary element*** will be used to refer to the objects ordered during this process; that is the named stereotype-instances and unexpected utterance representations.

In general the summariser attempts to produce its summary in ***narrative order***. That is, it tries to construct a textual order for the output which follows the temporal ordering of events described in the text. Of course this cannot always be achieved, either because the text has no natural temporal order, for example a description of a static scene, or because the text describes events which happen in parallel. I will describe these more complicated cases later, but I will begin with a description of how the system deals with an input which is in simple narrative order.

The simplest case for the ordering process occurs when all the utterances in the input text have been associated with a stereotype-instance and text segments which the various instances were used to analyse overlap in such a way that there are explicit indications of their temporal relationships. For example, consider:

(1) John and Mary went to the zoo. They saw the elephant and John fed the monkeys peanuts. After they saw the lions they went to a restaurant and ate some hamburgers.

The conjunction "after" in the third sentence allows the two sections

of text, one of which will have been analysed using an instance of the zoo stereotype and the other using an instance of the restaurant stereotype-prototype, to be placed in a definite temporal order: 'ZOO' followed by 'RESTAURANT'. Hence the ordering produced will be the summary of the section of the text dealing with the visit to the zoo, followed by the summary of the restaurant text section.

Of course this strategy will only generate an order for summary elements where there is explicit temporal information in the text. If there is insufficient temporal information in the input text to order all the summary elements an order is chosen which reflects the order in which the material was presented in the original text. In essence this is done by merging together explicit temporal information in the input text and information about the textual order of input material collected by the stereotype manager during the analysis process.

The manager keeps track of the textual order in which stereotype-instances are activated and suspended, and also where unmatched propositions occur within the framework set up by this. If no instances or propositions have been placed in order using temporal information, the summary elements are produced in the order in which their precursor utterances occurred in the input text. This might happen, for example, in a description of a static scene. More typically, in the corpus of texts considered by this project, some but not all of the summary elements will have been placed in some order by the first phase of the ordering process. This entails more complex processing to integrate the two information sources. For example, let us assume that there are three summary elements, two of which have been ordered using temporal information. The processes of the second phase will then place the remaining summary element before, after or between the others if this will cause all three to be in the order of their antecedents in the original text. Of course, this cannot be done if the temporal order of the first two elements is different from their textual ordering. In this case the third element will be placed immediately before the element which was its precursor in the textual order.

At the end of this process, then, the order in which the various elements which will form the summary representation has been decided.

6.4 Generating Summaries from Stereotype-Instances

At this stage of processing the partially constructed summary representation consists of an ordered list of named summary elements. One class of summary element, corresponding to the CD-representations of unexpected utterances in the input text, is in a form which may be converted more or less directly into English by a ***generator*** (see section 2.2). The other class, corresponding to the remaining stereotype-instances, requires more complex processing before it can be transformed into CD-structures and thence into English text. This processing, to accomplish the transformation from

stereotype-instances to CD-structures in the summary representation, is absolutely crucial to Scrabble's summarising process since it is the sole means by which the input text is compressed during a Scrabble run. It is achieved by extracting from the stereotype-instance the information most likely to provide a brief indication of the material in the input text which was expected by the instance. Thus Scrabble's fundamental text compression mechanism is the filling out of a pre-specified outline summary representation for each topic-type the system has recognised in the input text.

The transformation is achieved by associating with each prototype and hence with its instances a ***summary template***. A summary template looks very much like the patterns which represent the expectations of sections 5.1 and 5.2. It consists of CD-derived structures containing the unique names of stereotype variable instances. The construction of the summary CD-structure then simply consists of the replacement of the stereotype-instance variables by the values of their bindings, or failing, that the defaults specified in the description of the variables in the stereotype-prototype (as described in section 5.2). In the case of variables of type TIME defaults are implicitly created by using the time constraints of the variable.

A detailed description of the way templates are manipulated will be postponed until the example at the end of this chapter.

It must be acknowledged that this template-driven technique, in effect providing standard summaries for text about a given topic, is very simple minded; and it might be thought odd that such a central part of a summarising system, that is the component which actual condenses the input material, relies on such a straightforward method. However, previous summarising systems, like SAM [Schank75c] and FRUMP [DeJong79] have shown its usefulness for condensing predictable parts of a text.

6.5 Expanding Summary Elements

Once filled-in summary templates have been extracted from the stereotype-instances, the partially constructed summary representation consists of an ordered list of CD-structures. These CD-structures could be converted into a series of sentences by application of an English generator to each in turn, but the resultant summary of the input would be unlikely to form an acceptable English text. There are two reasons why the list of CD-structures may not form a good English text.

First, the CD-structures correspond to unnaturally short sentences: for example filled-in summary templates from different stereotype-instances will not be linked by conjunction markers, even where this would produce superior English. The techniques used to deal with this problem will be described in the next section.

Secondly, the CD-representations of portions of the text which were not expected by a stereotype-instance occur at suitable places in the partially-built summary representation, but in their original form. Thus, for example, they may contain residual anaphoric tokens corresponding exactly to, say, pronouns in the input text. Without additional processing these tokens will cause the production of exactly the same pronouns in the output summary text as occurred in the input text. This may well be inappropriate if the pronoun refers to some object which was mentioned earlier in the input text, but the prospective order of presentation of the summary text is such that the referent is first mentioned after the use of the pronoun.

Similar problems with the inappropriate use of anaphoric references may also occur in connection with the filled-in summary templates produced from stereotype-instances. It may well not have been possible, whilst processing the input text, to bind anything but residual CD anaphora markers to a stereotype variable which was subsequently used when filling in the summary template. This can lead to the production of pronouns or other anaphoric expressions in which do not, in fact, refer to any definite nominal which occurs in the output summary text.

The technique used in Scrabble to overcome the problems presented by residual anaphora markers in the summary representation is as follows. Its goal is to eliminate all anaphora markers from the summary representation. Thus, if the summary representation can be expressed better in English if anaphoric devices are used in the output summary text, any conversion from specific nominals to corresponding anaphoric expressions is done only whilst generating English from the summary representation.

The technique used to identify the referents of residual anaphora markers relies purely on knowledge of the structure of texts: it does not require any deep understanding of unexpected portions of text. It is based on a particular assumption about the ***coherence*** of the input text.

It should be pointed out that the word "coherence" is being used here in a different sense from the usual use of the word by the text grammarians, for example [vanDijk77]. Van Dijk would use "coherent" to refer only to propositions which concern a single topic-manifestation (see section 3.1). Hence Van Dijk would use the word "coherence" to refer to the relationships between parts of the input text predicted by a single stereotype-instance. ***Coherence*** is defined here for a multi-topic (multi-stereotype) text as that which links the different topics together so that they from a complete whole. The fact that a text does not usually contain sets of characters which do not interact with each other is a manifestation of coherence.

The specific way in which Scrabble's assumption about coherence of the input text is implemented is as follows. If an object in a

segment of the input text is referred to loosely, for example by means of a pronoun, then, if there is a closely preceding text segment with an identified topic, the object will co-refer with the central object of that topic, unless, of course, there is contrary information.

To facilitate exploitation of this rule, each summary template has one or more of its variables marked as being ***central***. In other words it is expected that any nominals bound to those variables will occur elsewhere in the input text. When the summary template is filled-in the CD-nominals which were bound to the central variables are marked as central.

For the anaphora resolution, all the summary elements are scanned in an order which reflects the order of the input text. If the first element is a filled-in summary template it is examined to find the CD-nominals which are marked as central. If it corresponds to an unexpected utterance in the input text it is left as it is. Next the second summary element (in the order of the input text) is picked up. If this second element contains any residual CD anaphora markers their semantic features are compared with the CD-nominals marked as central in the first summary element, and any anaphora markers which match the CD-nominals are replaced by the specific CD-nominals. Obviously, if either the CD-nominals do not match the semantic features (like MALE, PLURAL, ANIMATE, and so on), or are less specific, the replacement is not done. Equally, if there is more than one way the central CD-nominals of the first element could match the anaphora markers of the second element the replacement is not done. The latter constraint is applied in order to minimise the possibility of generating a summary which misrepresents the input text. As was noted in Chapter 3, it is probably better to risk producing a semantically disconnected summary text than to produce one which misrepresents the input material.

If the second summary element is a filled-in summary template the process is repeated with the second and third elements. If it is not the same initial, marked, central CD-nominals are carried forward for processing on the third summary element. The procedure is applied to each summary element in turn.

Once this process has been completed all that remains to form the summary representation is to block the summary elements up into units which will form sentences in the output text, and to convert the representation into English. These two tasks will be discussed in the next two sections.

6.6 Constructing Sentential CD-structures

Once the anaphora resolution of the last section has been completed, the partially constructed summary representation contains the complete CD-structures from which the English text of the summary will be produced. One further algorithm is, however, applied to it to improve its readability.

The CD-structures which compose the summary representation at this point correspond, in general, to single clause sentences. A text could be constructed directly from this partial summary representation, but its appearance would usually be improved if its parts were linked by conjunctions at suitable points. Scrabble therefore applies some simple techniques, in a final pass over the summary representation, so that the English generator can produce output with better flow.

Two simple rules are used to link summary elements produced from unexpected CD-structures in the input to summary elements produced from stereotype-instances. The first is that if an unexpected utterance occurs between two utterances predicted by the same stereotype-instance, the corresponding summary elements should be linked by the representation for a contrastive conjunction (for example "but"). The second is that if the input text contains two consecutive unexpected sentences, and neither of their representations contains a conjunction, they may be linked be a non-contrastive conjunction (for example "and"). Both rules contain checks to ensure that they do not produce CD-structures which correspond to sentences containing unacceptable combinations of conjunctions, for example "and" linking three successive clauses.

Once these conjunction representations have been inserted in the input the summary representation is complete. All that remains is to hand it over to the English generator for conversion into English text.

6.7 Generating English from the Summary Representation

In the final stage of a Scrabble run the summary representation is converted by the AD-HAC generator into English text. A technical description of the AD-HAC generator is inappropriate here: from the point of view of this thesis the generator is a black box whose internal working are of little or no interest. A full technical description of the generator is given in [Cater82].

However, the generator used by Scrabble is slightly different from Cater's original, and a few words on the different properties of the version used by Scrabble (rather than details of their implementation) seems appropriate here.

Cater's generator was designed to produce single sentences in isolation, rather than running text. The major modification made to

the program for use by Scrabble was to carry over information about noun phrases output during the generation of a sentence from one sentence to the next in order to facilitate cross-sentence pronominalisation. The algorithm used by the Scrabble version of the generator is to use a pronoun for the second and subsequent times a nominal token appears in the summary representation, provided there is no possibility of ambiguity. Although this algorithm is extremely crude it represents a significant improvement over Cater's original when generating running text.

There are a number of other minor differences. They essentially provide mechanisms through which the generator can communicate with the summarising system. The most significant of they will be pointed out in the example of the next section.

6.8 An Example of Summary Generation

The example text of this section is rather more complex than those considered elsewhere in this thesis. It is intended to illustrate a number of the more powerful features of the Scrabble system. From the point of view of the summariser, the synthesis of the output summary text requires the full power of the algorithms which resolve the referents of anaphoric tokens left after the text analysis process and the full power of the algorithms which are used to block up summary material into sentences. From the point of view of the text analysis process, it illustrates the systems ability to deal with two distinct topic-types which are related in the text in a way which is quite unexpected, so far as Scrabble is concerned, and the system's ability to deal with entirely unexpected textual material. The input text is:

(2) MIKE AND MARY WENT TO THE ZOO. THEY SAW THE ELEPHANTS AND FED THE MONKEYS PEANUTS. AFTER THEY HAD LOOKED AT THE LIONS, THEY WENT TO THE ZOO'S RESTAURANT. THEY COULD SEE THE ZEBRAS AND GIRAFFES FROM THEIR TABLE. AFTER THEY HAD EATEN THEIR MEAL THEY REALISED THEY DIDN'T HAVE ANY MONEY. THEY HAD TO WASH DISHES BEFORE THEY COULD LEAVE.

The discussion of the construction of the summary will be prefaced by some brief notes on the way the story was analysed. Much of the processing follows a similar course to previous examples, but notes will be added to the computer output to explain some of the distinctive processing required to deal with this story. Once again, program output is indented at the left and right hand sides and is marked with vertical bars ("|") at the right hand side of the page.

```
Run of AD-HAC with Scrabble version 15.14
 in 850K at 14.24 on 24 SEP 82
[]
Entering Scrabble for sentence :
 (MIKE AND MARY WENT TO THE ZOO)
Stereotypes suggested:
((ZOO))
The following Stereotype Instances have been activated
(Z002)
Spent 150 ms in Scrabble
[]
Entering Scrabble for sentence :
 (THEY SAW THE ELEPHANTS AND FED THE MONKEYS PEANUTS)
No new stereotype instances were activated
Spent 57 ms in Scrabble
[]
```

The "they" in the previous sentence is identified with "Mike and Mary" because the zoo stereotype contains the prediction that the same people who went to the zoo will look at and feed animals. The first "they" in the next sentence is similarly resolved. However, the second clause is not expected by the zoo stereotype, except that it has the information that going to places belonging to, or in, the zoo, do not involve leaving the zoo. Scrabble may mis-interpret the sentence because it has, at this point, an expectation that the zoo's visitors will eventually leave the zoo and by default it assumes that all places which are not obviously in the zoo are outside it. (The summary of a broadly similar text to the one considered here in which leaving the zoo is the correct interpretation is contained in Appendix A). At the same time, the word "restaurant" triggers the activation of instances of all Scrabble's stereotypes associated with food. A more sophisticated semantics would, of course, immediately recognise that certainly the supermarket stereotype, and possibly the kitchen stereotype, are inappropriate here, but the present system cannot do this. Therefore, at the end of processing the sentence Scrabble has four active stereotype-instances, Z002, RESTAURANT2, KITCHEN1, and SUPERMARKET1.

```
Entering Scrabble for sentence :
 (AFTER THEY HAD LOOKED AT THE LIONS , THEY WENT
    TO THE ZOO'S RESTAURANT)
Stereotypes suggested:
((RESTAURANT KITCHEN SUPERMARKET))
The following Stereotype Instances have been activated
(RESTAURANT2 KITCHEN1 SUPERMARKET1)
Spent 788 ms in Scrabble
[]
Entering Scrabble for sentence :
 (THEY COULD SEE THE ZEBRAS AND GIRAFFES FROM THEIR TABLE)
```

Between them Z002 and RESTAURANT2 expect the whole of this sentence: Z002 expects people to able to see animals, and RESTAURANT2 for them to be at a table of their own. Note, however, whilst on one hand "they" is assumed to refer to "Mike and Mary" by Z002, RESTAURANT2 is only able to infer that the people at the table are the same as the people who went to the restaurant. No attempt is made at this stage to collate the pronoun resolution done by different instances.

```
No match by stereotype : KITCHEN1
No match by stereotype : SUPERMARKET1
No new stereotype instances were activated
Spent 159 ms in Scrabble
[]
Entering Scrabble for sentence :
 (AFTER THEY HAD EATEN THEIR MEAL THEY REALISED THEY
    DIDN'T HAVE ANY MONEY)
```

The restaurant stereotype expects that people will eat their meal. It does not expect the second clause, but this failure is not commented on by the manager.

```
No match by stereotype : Z002
No match by stereotype : KITCHEN1
No match by stereotype : SUPERMARKET1
No new stereotype instances were activated
Killing stereotype instance SUPERMARKET1
Killing stereotype instance KITCHEN1
Spent 76 ms in Scrabble
[]
Entering Scrabble for sentence :
 (THEY HAD TO WASH DISHES BEFORE THEY COULD LEAVE)
No new stereotype instances were activated
Spent 90 ms in Scrabble
```

The first clause of the final sentence is not expected by either of the active instances. However, the second clause "they could leave" is expected by both Z002 and RESTAURANT2. They both expect that one can leave at any time, which in many ways is inappropriate here, but its inclusion amongst the expected material for both the topic-types of the text does not cause the production of a misleading summary.

The Scrabble summariser is now invoked.

The summariser is entered with six elements from which the summary material will be formed. Four are the associated with the remaining active stereotype-instances, RESTAURANT2 and Z002. They are the instance's summary templates and variable bindings. The remaining two are the CD-representations of the two unexpected propositions in the input text, "they realised they didn't have any money" and "they had to wash dishes".

In addition the summariser takes as data the records of activation and suspension of the instances which will be used when forming the summary, and the records of the textual positions of the unpredicted material in the input text.

For the example text there is no need to reduce the number of stereotype-instances at the beginning of summarising. In Scrabble's terms the two active instances concern disjunct topic-types, so both should be included in the summary.

The construction of an order for the summary is a fairly straightforward matter for this example: the summariser cannot produce a particular linear order for any of the elements which will form the summary. It therefore presents them in the order of

activation of the stereotype instances, and then follows with the two unpredicted propositions, again in their textual order. Informally, then the summary will be ordered: summary for Z002, summary for RESTAURANT2, "they realised they didn't have any money", and finally "they had to wash dishes".

The first substantial piece of processing undertaken by the program concerns the extraction and the filling in of the summary templates for Z002 and RESTAURANT2.

The summary template for Z002 is:

```
((MAINCHAR goer1)
   ((EVENT (ACTOR goer1) (ACT PTRANS) (OBJECT goer1)
       (TO zoo1)
       (TIME . gotozootime1))))
```

The CD-nominal token which replaces the MAINCHAR variable in the summary template will by marked so that it can used in subsequent processing.

The variable bindings to be substituted into the summary template are:

```
(goer1
   (SET . GROUP#1))
(zoo1
   (SET . ZOO1))
(gotozootime1
   (SET (NAMED TIMEPOINT1)
      (COMPARISON (BEFORE *NOW*))))
```

GROUP#1 is marked as being best expressed in English as "MARY AND MIKE". The summarising structure[2] for Z002 is then:

```
((EVENT (ACTOR GROUP#1) (ACT PTRANS) (OBJECT GROUP#1)
     (TO ZOO1)
     (TIME
        (NAMED TIMEPOINT1)
        (COMPARISON (BEFORE *NOW*)))))
```

In English this CD-structure could be expressed as "Mike and Mary went to a zoo".

Next the summariser constructs the summary CD-structure for the input text material predicted by RESTAURANT2. The summary template and associated variable bindings for RESTAURANT2 are:

---

[2]Note that the time token is much abbreviated here: the ones actually produced by the program contain all the temporal information which may be inferred from all the time variables of the stereotype-instance.

```
((MAINCHARS customer1)
   ((EVENT (ACTOR customer1) (ACT PTRANS)
        (OBJECT customer1)
        (TO restaurant1)
        (TIME . gotime1))))
   (customer1
      (SET . DUMMY-UNKNOWNS11))
   (restaurant1
      (SET . RESTAURANT1))
   (gotime1
      (SET (COMPARISON (AFTER TIMEPOINT6))
         (NAMED TIMEPOINT14)
         (COMPARISON (BEFORE *NOW*)))))
```

This produces the following CD-structure:

```
((EVENT
     (ACTOR DUMMY-HUMANS)
     (ACT PTRANS)
     (OBJECT DUMMY-HUMANS)
     (TO RESTAURANT1)
     (TIME
        (COMPARISON (AFTER TIMEPOINT6))
        (NAMED TIMEPOINT14)
        (COMPARISON (BEFORE *NOW*)))))
```

Note the token DUMMY-HUMANS ("some people") in the above structure. The variable 'customer1' which appeared in the ACTOR and OBJECT roles of the summary template was bound to DUMMY-UNKNOWNS11. DUMMY-UNKNOWNS11 was, in turn, derived from the word "they" in the input text. "They" does not, in general, refer to human beings: however the restaurant stereotype has the information that customers in restaurants are usually human. When constructing the summary, the system selects the default for the variable if the default is more specific than the actual binding of the variable extracted from the input text, as is the case here.

These two filled in CD-structures form part of the input to the next stage of processing, that described in section 6.5. The object of this processing, it will be remembered, is to eliminate, where possible, anaphoric tokens (like DUMMY-UNKNOWNS) from the summary representation. The other two CD-structures which are handed over to that process correspond to the two unexpected propositions in the input text. They are:

```
((STATE
   (STATENAME MLOC)
   (MOBJECT
      (STATE
         (STATENAME POSS)
         (THING MONEY1)
         (VAL DUMMY-UNKNOWNS9)
         (TIME
            (NAMED TIMEPOINT8)
            (COMPARISON (BEFORE *NOW*)))
         (TRUTH FALSE)))
   (CERTAINTY 8.1E-1)
   (INCP DUMMY-UNKNOWNS8)
   (TIME
      (NAMED TIMESPAN1)
      (COMPARISON (BEFORE *NOW*))
      (TS
         (COMPARISON (AFTER TIMEPOINT3))
         (NAMED TIMEPOINT8)
         (COMPARISON (BEFORE *NOW*)))) ))
```

In English "somethings started to know that some other things possessed no money". Note the tokens DUMMY-UNKNOWNS9 and DUMMY-UNKNOWNS8: they indicate that at this stage the system does not identify the people who do not have any money with the people who are doing the thinking.

```
((CAUSE
    (ANTECEDENT
       (EVENT
          (ACTOR DUMMY-HUMANS)
          (ACT DO)
          (TIME
             (NAMED TIMEPOINT13)
             (COMPARISON (BEFORE *NOW*)))) )
    (RESULT
       (CAUSE
          (ANTECEDENT
             (EVENT
                (ACTOR DUMMY-UNKNOWNS10)
                (ACT PTRANS)
                (OBJECT *WATER*)
                (FROM DUMMY-PLACE4)
                (TO DISHES1)
                (TIME
                   (COMPARISON (AFTER TIMEPOINT8))
                   (NAMED TIMEPOINT13)
                   (COMPARISON (BEFORE *NOW*)))) )
          (RESULT
             (STATE
                (STATENAME CLEANNESS)
                (THING DISHES1)
                (VAL (HIGHERBY 3))
                (TIME
                   (NAMED TIMEPOINT15)
                   (COMPARISON
                      (AFTER
                         (NAMED TIMEPOINT13)
                         (COMPARISON
                            (BEFORE *NOW*)))) ))) ))) )
```

This structure is derived from "They had to wash dishes".

The system picks up the main character in the first summary element, that is GROUP#1 (representing "Mike and Mary") in ZOO2 and percolates it forward through the other summary elements. In this example it replaces all instances of DUMMY-HUMANS and DUMMY-UNKNOWNS with GROUP#1, since they all may co-refer with the specific token GROUP#1.

The final process is to block the summary representation into sentences. Following the rules of section 6.6, the two unexpected propositions in the input text are conjoined, and then conjoined to the filled-in summary template for RESTAURANT2 by a conjunction marked as contrastive. The contrastive conjunction "but" is always used to conjoin sections of the summary derived from expected material in the input to sections derived from unexpected material, whilst "and" is used to conjoin both paired sections of unexpected material (as here) and paired sections of unexpected material. Although this algorithm is extremely simple-minded, it appears to work quite well in practice.

The use of a contrastive conjunction, rather than the non-contrastive "and", is achieved by opening an extra channel of communication between the generator and the summariser. This is an extension to Cater's system.

The final summary representation consists, then, of the original filled-in summary template for Z002; the complex CD-structure, below, which represents the filled-in template for RESTAURANT2 and also the unexpected propositions in the input.

```
((CONJUNCT
     (FIRST
        (EVENT (ACTOR GROUP*1) (ACT PTRANS)
           (OBJECT GROUP*1)
           (TO RESTAURANT3)
           (TIME
              (COMPARISON (AFTER TIMEPOINT6))
              (NAMED TIMEPOINT14)
              (COMPARISON (BEFORE *NOW*)))))

     (SECOND
        (CONJUNCT
           (FIRST
              (STATE
                 (STATENAME MLOC)
                 (MOBJECT
                    (STATE
                       (STATENAME POSS)
                       (THING MONEY1)
                       (VAL GROUP*1)
                       (TIME (NAMED TIMEPOINT8)
                          (COMPARISON (BEFORE *NOW*)))
                       (TRUTH FALSE)))
                 (CERTAINTY 8.1E-1)
                 (INCP GROUP*1)
                 (TIME
                    (NAMED TIMESPAN1)
                    (COMPARISON (BEFORE *NOW*))
                    (TS
                       (COMPARISON (AFTER TIMEPOINT3))
                       (NAMED TIMEPOINT8)
                       (COMPARISON (BEFORE *NOW*)))) ))
```

```
(SECOND
   (CAUSE
      (ANTECEDENT
         (EVENT
            (ACTOR DUMMY-HUMANS)
            (ACT DO)
            (TIME
               (NAMED TIMEPOINT13)
               (COMPARISON (BEFORE *NOW*)))) )
      (RESULT
         (CAUSE
            (ANTECEDENT
               (EVENT
                  (ACTOR GROUP#1)
                  (ACT PTRANS)
                  (OBJECT *WATER*)
                  (FROM DUMMY-PLACE4)
                  (TO DISHES1)
                  (TIME
                     (COMPARISON
                        (AFTER TIMEPOINT8))
                     (NAMED TIMEPOINT13)
                     (COMPARISON
                        (BEFORE *NOW*)))) )
            (RESULT
               (STATE
                  (STATENAME CLEANNESS)
                  (THING DISHES1)
                  (VAL (HIGHERBY 3))
                  (TIME
                     (NAMED TIMEPOINT15)
                     (COMPARISON
                        (AFTER
                           (NAMED TIMEPOINT13)
                           (COMPARISON
                              (BEFORE *NOW*))))
                  ))) ))) ))) )))
```

Note the occurences of GROUP#1 (the token which represents "Mike and Mary") in this structure. GROUP#1 has been inserted in the structure as a replacement for a variety of anaphoric tokens, as was noted above.

Finally, the summary representation is converted back into English. In this example all the occurences of "they" in the input text occur as "they" in the output summary text. However this only happens because given Scrabble predictions about the text, and its assumption about the coherence of the input they are all recognised as co-referring with "Mike and Mary". If they were not recognised as co-referring with this definite reference the English generator would be at pains to ensure the output text reflected the fact that the nominal tokens did not co-refer with "Mike and Mary".

The final output is therefore:

```
English summary is:
  MARY AND MIKE WENT TO A ZOO. THEY WENT TO A RESTAURANT
  BUT THEY REALISED THAT THEY DIDN'T HAVE ANY MONEY
  AND THEY HAD TO WASH SOME DISHES.
```

6.9 Concluding remarks

The example summary produced in the previous section is, of course, far from perfect. It is typical of the output from Scrabble: some summaries the system has produced are better, some worse.

The main problem with it is a certain lack of continuity. The program has only the crudest model of what constitutes a well-formed text, and such a model is probably necessary for the generation of really good running text. The construction of such a model is well beyond the scope of the present project.

On the other hand, the processing necessary to generate this summary text illustrates that it is possible to integrate unexpected material from an input text with summary material condensed from the input text by expectation-driven analysis and template-driven generation. The output summary text has all the salient points of the input text, even though not all the input textual material was expected by Scrabble.

More example summaries and the texts from which they were extracted are presented in Appendix A.

7. Conclusions

The conclusions to this thesis can be divided into four main areas. First are conclusions about predictive inference in general as a mechanism for use in natural language understanding systems, and in particular summarising systems. Second are conclusions about the approach adopted when building the Scrabble program as opposed to the approaches adopted for other predictive inference systems. Third are some relatively low level remarks about how the Scrabble program could be improved whilst sticking to the current outline design. Fourth is a discussion of the problems with the current front and back ends to Scrabble, the AD-HAC analyser and generator. The remainder of this chapter presents these four groups of conclusions in this order. The chapter closes with a summary of the results of this work.

All the conclusions drawn here must be regarded as tentative. Scrabble has only processed about twenty texts from half a dozen different subject domains, and this does not constitute a real test of many of the ideas presented in the thesis.

There are three main reasons Scrabble has been tested with such a small sample of texts. First, it has proved to be difficult to extend the vocabulary of the AD-HAC semantic parser. Second, design problems with the Scrabble system, the most important of which will be described in section 7.3, have led to software maintenance problems when adding new subject domains to the system. Third, severe practical problems have been presented by the difficulty of maintaining a large suite of LISP programs in the Cambridge environment, in particular without the aid of interactive debugging facilities. The first two major problem areas will be discussed in more detail later in this chapter, but the third area, the difficulties presented by the limitations of the local computing environment in which Scrabble was developed, will not be discussed further.

It should, however, be pointed out that whilst the testing of Scrabble must be regarded as an inadequate basis for any firm conclusions, it is no less adequate than the published testing of other, strictly comparable systems. Although the SAM system ([Cullingford78]) appears to have been tested on upwards of a dozen different stories, and Cullingford claims that it has used eighteen different scripts at one time or another, it is not at all clear that any version of the system had all the scripts available when processing a particular story. The systems of [DeJong79] and [Lebowitz80] have processed much larger numbers of texts (about 300 in the case of Lebowitz) but are not comparable with Scrabble in that they do not process the whole of an input text: whilst this allows the system to handle comparatively large numbers of texts, as was pointed out in Chapter 2, this skimming approach carries with it the danger of seriously mis-analysing input material in some circumstances. Mellish (private communication) has commented that the natural language front-end to the Edinburgh

MECHO project (see [Byrd80] and [Mellish80]) only succeeded in processing about 15 texts completely. Most other natural language processing systems which do deep inferential analysis of their input have either been restricted to some very limited domain of discourse, restricting the scope of any tests performed on them, or have little or no published indication of the scale of any tests performed on them, or both[1].

7.1 Predictive Inference and Summarising

Experience with Scrabble has pointed out a number of serious problems with current predictive inference systems for use by summarisers.

The first is ***inextsibability***. It is difficult to see how to extend the kind of technology used by Scrabble to the summarising of longer texts. Like its precursors FRUMP ([DeJong79]) and SAM ([Cullingford78]), Scrabble's summaries always have some mention of each of the topics of the input text. They lack any mechanism for selecting amongst the topics covered by the input text on the basis of their salience. This makes the mechanism inapplicable to any but the shortest input texts. If a current predictive inference system were applied to even a text of half a dozen pages, like a short story, it could not select what one might call the crucial twist: it would, in all probability, merely condense the supporting matter. Furthermore, it seems unlikely that this problem can be overcome by giving the system more complex and higher level predictions of the present type.

The way forward for summarising, it seems to me, lies more with mechanisms which embody an understanding of the likely structure of texts, rather than the likely content, which is the focus of current predictive inference systems. [Lehnert81] describes a system which moves some way in the structure oriented direction, although there are, as yet, no published results for the effectiveness of her technique when used by a computer system.

However, structure oriented systems will undoubtedly require knowledge about, for example, the normal occurences on a visit to a zoo, if they are to deal with the domain of zoo visits. Therefore structure oriented techniques are to be regarded as an necessary extention to predictive inference systems, rather than a replacement for them. Second, again considering longer input texts, the use of the techniques used in Scrabble will cause all unexpected material in the

---

[1]A few days before this thesis was completed I received a copy of [Dyer82]. An initial examination of this document indicates that the system it describes has been tested on just three texts in two knowledge domains. (However, it must be admitted that the length of the texts and the depth of the systems understanding of them is extremely impressive.)

input, no matter how trivial, to appear in the output summary text. Overcoming this problem probably requires the introduction of general purpose inference mechanisms capable of deeper understanding of unexpected material.

Of course computer systems capable of processing a story of half a dozen pages are still science fiction, so perhaps criticism of predictive inference on these grounds is premature.

The final problem, however, is very much a gap in the capabilities of current predictive understanding systems. Consider, for example, the text:

(1) John is really crazy. Yesterday, he walked into a restaurant, ordered a meal, then got up and walked out.

It is difficult to see how Scrabble can be prevented from producing as a summary for this text:

(2) John is really crazy. Yesterday he went to a restaurant.[2]

From within the current Scrabble model one possible approach to this problem text is to demand that some key expectation of the eating-in-a-restauarant stereotype is satisfied before it is confirmed as being relevant to the incoming text. Here, for example, the construction of a misleading summary might be prevented by insisting that the restaurant summary's appropriateness is only confirmed when there has been some mention of a meal being eaten. However, the use of such key predictions would inevitably complicate Scrabble's stereotype-prototypes, which would also complicate the processes of stereotype management and stereotype application. In all probability such additional complexity would slow the analysis of text sections containing only commonplace material. This, in addition, would run contrary to primary argument for using predictive inference in a system like Scrabble: that is the argument that the necessary speed of processing of common place material in an input text can be achieved only be the use of a predictive inference system.

Problem texts like example (1) might be better dealt with by a system which uses both stereotype-based, predictive, inference, and more general, non-predictive inference mechanisms. In outline, I imagine by default such a system would process incoming texts by using techniques similar to those adopted in building Scrabble. If, however, some item in an input text did not correspond to the systems expectations, a non-predictive inference system would be invoked. The non-predictive inference system, amongst other things, would have the capacity to suspend the normal processing of the predictive inference system. Implicitly predictive inference systems assume predictions about the contents of an input text are assumed to be true about the world the input text describes unless they are explicitly contradicted. In other words, such systems do not really

---

[2]I am indebted to Arthur Cater for pointing out this problem.

distinguish between what they expect to read and what they actually read. To deal with examples like (1), my hypothetical integrated inference system might suspend this assumption about the text whilst using predictions, in this case about restaurants, to process the input and to perform tasks like pronoun resolution. Then the summariser would at least have the opportunity to examine the text representation to determine which parts were constructed directly from the text, and which parts could only be constructed by implication , and hence are suspect in this case.

On the other hand, it is not at all clear how to construct such an integrated inference system, and the computational cost of using such an integrated system might well be prohibitive. The construction of such a system remains a problem for future research.

7.2 Scrabble Compared with other Predictive Inference Systems

This section compares Scrabble with its two primary precursors, FRUMP ([DeJong79]) and SAM ([Schank75c], [Cullingford78]).

There are three main differences between Scrabble and FRUMP or SAM of concern here.

First, the mechanisms for the identification of stereotypes which are suitable for the processing of a particular text, as described in Chapter 4, are quite different from the mechanisms used by DeJong or Cullingford. As well as texts containing the sort of direct clues to their topic-type, which DeJong and Cullingford could handle, Scrabble can handle texts with only very indirect indication of their topic-type (see for example section 4.6). Cullingford and DeJong cannot handle such texts. It could be argued that the computational expense entailed by Scrabble's mechanisms, in particular the frequency with which inappropriate stereotype instances are activated, outweighs the advantages of the Scrabble scheme for dealing with this problem. To be fair, FRUMP has been shown to be capable of operating with many more sketchy scripts than Scrabble has stereotypes. However until Scrabble has been tested with large numbers of stereotypes this remains a matter of argument. A direct comparison of Scrabble and FRUMP in this area must wait until some of the more serious problems with semantic parsing have been overcome.

Second Scrabble presents a solution the problems presented by texts whose topic-type is one for which the system is prepared, but which contain unexpected text segments. FRUMP dealt with such texts by completely ignoring the unexpected material; SAM could not really deal with such material at all. The technique used in Scrabble of integrating unexpected input material with condensed versions of the predictable part of the input represents a significant advance. Of course, the present Scrabble summarising system is naive in many

respects. The summaries it produces could be improved by, for example, a model of what constitutes a complete text, and the provision of facilities for improving the continuity of the generated text. However, compared with FRUMP, the techniques used in Scrabble very considerably reduce the possibility of producing an output summary text which misrepresents the input.

Third, Scrabble's ability to apply predictions from two different stereotypes to the same piece of text, as illustrated by the example of section 6.8, provides a considerable increase in power over DeJong and Cullingford's approach at no significant increase in computational cost. Scrabble also operates without the introduction of additional mechanisms, like DeJong's Issue Skeletons, to handle multiple topic-type stories.

Together these three properties of Scrabble considerably enhance the power of a predictive inference mechanism.

7.3 The Scrabble Program

There are three main problem areas within the current implementation of the Scrabble inferential analysis and summarising programs. They are, first, ad hoc features of the stereotype management process; second, limitations of the stereotype application process; and third, some extreme inefficiencies in the summarising program.

The current stereotype manager relies on a count of sentences to determine when a stereotype instance becomes a candidate for killing or suspension. The examples in this thesis were run with this count set to two, so that an instance is killed or suspended if it fails to predict anything for two sentences, and neither of the sentences suggest the instances stereotype again. This is an entirely ad hoc solution to a serious problem for text analysis systems based on a notion of topic: what constitutes a change of topic, and how is it signalled in the text? Both FRUMP and SAM had a very limited ability to detect changes of topic, essentially by using predictions that the close of one script would be followed by some other specific event. For example, FRUMP possesses a prediction that the end of an earthquake will be followed by the beginning of a rescue operation. However, the use of specific predictions about how scripts are likely to fit together does not seem sufficiently powerful to deal with a wide range of real texts. It is not at all easy to see how to deal with the problem of topic changes, but it clearly presents an interesting subject for future research.

The pattern matching system used by the stereotype application module leaves much to be desired. Scrabble spends much of its time in this module, so its performance crucially effects the performance of the system is a whole. The present two pass approach in which

patterns are compared one at a time against the CD-structures extracted from the input text is by no means as efficient as it might be. For example it is almost certainly less efficient than a single pass approach, in which the information about variable bindings in stereotype instances, representing the context set up by preceding text segments is used at the earliest phase of matching a CD-structure. Fast parallel techniques in which all patterns for all active stereotypes are compared with input CD-structures at the same time, might also provide significant improvements in performance.

Finally, the techniques used by the summariser to determine the tense and aspect of a summary are extremely unwieldy, and greatly slow down the summary generation process. Essentially the summariser produces all the inferences it can about time relationships between elements in the story and between events the systems assumes occurred in the world of story but were not mentioned, so that it can ensure the English generator has enough information to choose a suitable tense and aspect for its output. The number of such inferences may be very large, especially because stereotypes often have a great deal of information about likely temporal interrelations between events only implicit in a particular input text. An algorithm which only did enough to, say, ensure that the whole output summary could be produced in the simple past tense, would greatly improve the efficiency of summary generation. However the construction of an algorithm which did this but would still produce correct output for more complex cases is extremely difficult.

7.4 Semantic Parsing and Language Generation

Although there are many ways in which the Scrabble program, that is the Scrabble predictive inferential analysis mechanism and summary representation construction system could be improved, the most obvious limitations of the Scrabble system as a whole come from the limitations of the AD-HAC semantic parser and generator. Many of these limitations moreover stem from quite fundamental difficulties and present, to my mind, some of most pressing problems for future research in natural language processing.

The AD-HAC semantic parser has three main drawbacks for Scrabble's purposes. They are: the difficulty of extending the parser's vocabulary; the parser's handling of genuine ambiguity in the input; and the systems robustness, in the sense of its ability to handle ill-formed inputs, missing vocabulary and the like.

Many dictionary entries used by the AD-HAC parser are quite simple, for example many nouns' dictionary definitions are a single line of text. However some dictionary entries, notably for polysemous verbs and conjunctions, are upwards of a hundred or even two hundred lines long. This extreme complexity arises because in Cater's system the dictionary entries for some classes of words contain not only a

description of the word's syntactic categories and meanings but also a description of how to parse text segments in which the word occurs. This, in turn, often leads to clashes between old words and new when extending the dictionary. Such clashes have presented significant problems when attempting to test Scrabble in domains other than those of Cater's original AD-HAC program.

At the same time, widening the vocabulary has, in general, increased the number of possible readings of an input sentence to be considered by the AD-HAC semantic parser. The parser always parses sentences as if they are out of context. The performance of the system could, I believe, often be improved by making Scrabble's expectations about the input accessible to the AD-HAC parser, although the implementation of such a scheme would require significant changes to both.

The final point concerning the AD-HAC semantic parser here concerns its lack of robustness, that is its inability to parse texts for which it has not been prepared. In particular, problems are caused by the requirement that all words in all input sentences must be represented in the systems dictionary, and the requirement that the input text must consist of well-formed sentences. Together these two requirements mean that the Scrabble system cannot operate on much real text. To analyse almost any text a certain amount of preparatory work must be done, putting proper names in the dictionary, ensuring the input matches the AD-HAC parser's idea of correct sentences, and so on. A practical natural language system would need to run without such hand preparation.

Of these three problem areas for semantic parsing, the latter two, robustness and the integration of parsing and inference (of whatever style) are beyond the scope of any current natural language processing system which attempts to analyse all its input. The first, the complexity of dictionary entries, is a problem for all parsers which have followed the approach of [Riesbeck75]. They present, I believe, the most important problems for future researchers in natural language processing. Without solutions to them, systems like Scrabble are difficult to evaluate, because of the inevitable restrictions on the form of their input material.

There are two serious shortcomings of the AD-HAC generator when it is used to produce running texts, like summaries.

First, the generator lacks any model of what constitutes a coherent text, and is incapable of producing significantly different expressions of sentences when they are embedded in text and when they stand alone, as say, one sentence answers to questions. To some extent this problem is a product of the interface to the generator: it takes a single CD-structure and produces a single sentence from it. [Cater82]

suggests that better English would be produced if the generator was capable of taking large CD-structures and producing many sentences from them. I concur with this view, but the construction of such a generator remains an open research problem.

Second, the lack of any significant communication between the generator and the summarising system, combined with the limitations of the CD representation language, prevents the system ever producing English which adequately conveys linguistic devices like focus. However the construction of a generator with this property also remains an open research problem.

Whilst discussing difficulties encountered when using the AD-HAC semantic parser and English generator, it is appropriate to make a few remarks about Cater's CD representation language. CD, of course, provides the interface between both the semantic parser and Scrabble and between Scrabble and the language generator.

The main virtues of CD, from the point of view of the text analysis subprograms in Scrabble, stem from its properties as a so-called semantic primitive-based language. In particular, the property of semantic primitive-based languages that many different natural language strings which are deemed synonymous are represented by identical data structures. This greatly eases the task of matching expectations against incoming text.

However, the use made of semantic primitives in Cater's system is extremely limited. There is no serious attempt to use a primitive system to represent nominal concepts, and the treatment of CD STATEs (the representations of state verbs and adjectives) is messy, employing an open ended set of primitives with ill-defined boundaries between them.

Furthermore, where the set of semantic primitives is well-defined and limited, that is the representation of action-oriented verbs, other problems present themselves. In particular, the notion of synonymy embodied in two different word forms being translated into the same primitive is utterly fixed and cannot be affected by any wider context. Although this problem has not affected the current Scrabble system future difficulties can be forseen. For example, consider the problems which might be presented to Scrabble by:

(3) John went from the restaurant to his car.

in contrast to the rather strange:

(4) John travelled from the restaurant to his car.

The two sentences are identically represented in Cater's CD, but I suspect the second sentence is not really best analysed as a normal exit from eating a meal at a restaurant.

The difficulty of overcoming this problem are perhaps best illustrated by considering the pair:

(5) John went to the Himalayas from Heathrow.

(6) John travelled to the Himalayas from Heathrow.

These two sentences are probably strictly synonymous.

It is difficult to envisage a scheme which overcomes this problem whilst retaining the advantages of semantic primitives when examining representations of input text for specific patterns.

I believe that at present the most fruitful approach to overcoming the problems of CD as a representation language is to consider abandoning the semantic primitive notion. A possible successor is a case-labelled dependency structure differing from CD in that it would contain explicit markers representing the senses of words used in the input. Such word sense markers would replace all STATE and ACT primitives in Cater's current scheme whilst providing a better motivated mechanism for the inclusion of markers closely associated with specific words in the representations produced for input sentences. To facilitate matching between expectations and parts of the input text this representation scheme would need to be backed up by some kind of thesaurus relating different word senses to, for example, their possible synonyms and antonyms. The work of [Fahlman79] and [Alshawi82] amongst others, holds out hope that a sophisticated flexible multi-dimensional thesaurus of the kind required here can be manipulated with acceptable computational efficiency.

Once again, however, the construction of a representation scheme like this, or for that matter any serious replacement for CD for the Scrabble system is a matter for future research.

7.5 Closing Remarks

Scrabble represents, I believe, a significant advance over previous attempts to build summarising systems based on the principles of predictive inference. New techniques in a number of different areas have been tested, and many of them appear to offer important improvements over techniques employed in previous systems. In particular, new techniques have been developed for the identification of the topic or topics of an input text; for the incorporation of unexpected material in an input text into the system's summaries; and for the exploitation of interactions between expectations associated with different stereotypes. Together these advances have brought the goal of a practical natural language summarising system based on the principles of predictive inference much closer.

However, like much work in natural language processing, this work has thrown up more problems than it has found solutions. In particular, it has brought into focus the need for better semantic parsers and

language generators before work on inference can properly be evaluated.

Having said that, it is my hope that Scrabble represents some small steps along the road towards computer systems which genuinely and demonstrably understand natural language.

Appendix A

This appendix contains a large number of additional summaries produced by Scrabble, together with the corresponding input texts. The set presented here constitutes the majority of texts processed by Scrabble. The only texts which Scrabble has processed which are not included are even more straightforward than the simplest ones presented here. The omitted texts were, in general, constructed as test for specific features of the system, or to allow the debugging of new stereotype-prototypes.

The first two texts concern uneventful visits to the zoo. In both cases the only complexity is that the visitors do not do everything together. They both illustrate the English generator's penchant for translating definite articles into indefinite articles.

1. Input text:

MARY AND JOHN WENT TO THE ZOO. THEY SAW THE ZEBRA AND FED THE MONKEYS PEANUTS. JOHN SAW THE ELEPHANT. THEY WENT HOME.

English summary is:

JOHN AND MARY WENT TO A ZOO.

2. Input text:

MARY AND JOHN WENT TO THE ZOO. THEY SAW THE MONKEYS AND JOHN FED THE ELEPHANT A BANANA. THEY WENT HOME.

English summary is:

JOHN AND MARY WENT TO A ZOO.

The next text also has a visit to the zoo as its central topic. However there is an irrelevant (and therefore unexpected) sentence in the middle of the input text. This text was the first one in which Scrabble identified an unexpected part of the input and successfully incorporated it into a summary.

3. Input text:

JOHN WENT TO THE ZOO. HE SAW THE MONKEYS AND FED THE ELEPHANT. DALE LOOKED AT SOME CARS. JOHN WENT HOME.

English summary is:

JOHN WENT TO A ZOO BUT DALE LOOKED AT SOME CARS.

The text number 4 is about eating a meal in a restaurant (the real home ground of predictive inference). It is similar to part of a story which Cullingford's SAM could process, but is noteworthy because it illustrates similar results being obtained by different techniques. In order to process texts like this SAM needed to have the information that one may run out of money in a restaurant and that one may be required to wash dishes if one does. Scrabble uses only information about the normal course of events in a restaurant. Scrabble' summary is, however, imperfect because the system never realises that the two mentions of "the waiter" refer to the same person. Therefore the second occurrence of "waiter" is not pronominalised in the output summary. Note also the slightly eccentric translation of "furious" into "livid". This is caused by a clash between the dialects of CD used by the semantic parser and by the generator.

4. Input text:

MIKE AND MARY WENT TO A RESTAURANT. THEY ASKED FOR STEAKS. AFTER THEY HAD EATEN THEM , THEY TOLD THE WAITER THAT THEY HAD NO MONEY. THE WAITER WAS FURIOUS. THEY RAN AWAY.

English summary is:

MARY AND MIKE WENT TO A RESTAURANT BUT THEY TOLD A WAITER THAT THEY DIDN'T HAVE ANY MONEY AND A WAITER WAS LIVID.

The next set of stories illustrates the two stereotypes used so far interacting in different ways to analyse each text.

All illustrate two different aspects of the Scrabble's processing. First they all require the introduction of a new stereotype for eating a meal in a restaurant part way through the input text. In fact, the introduction of the restaurant stereotype requires the elimination of some irrelevant stereotypes: the system does not have access to the relation between the word "restaurant", which indicates the change of topic in each case, and the restaurant stereotype. It uses the information that "restaurant" is associated with food, and it must, therefore, examine the possibility that other stereotypes associated with food (in practice for shopping at a supermarket and making a meal in the kitchen) are not appropriate for the analysis of the given input text. For these particular texts the other food stereotypes can be eliminated immediately, because the individual stereotypes have access to particular CD-nominals derived form the input, and hence the word "restaurant", which is inappropriate for them.

The first text is quite straightforward, the only complexity being presented by the necessity to carry forward the main characters from the visit to the zoo to allow determination of the referent of the word "they" in the latter part of the text. It is a separate matter that, after resolution, all the resolved anaphoric expressions for this text are converted back into "they" for a smooth output summary text.

The system assumes that "Mike and Mary" left the zoo to go to the restaurant, but it has no means of expressing this in the summary.

5. Input text:

MIKE AND MARY WENT TO THE ZOO. THEY SAW THE ZEBRAS AND MIKE FED THE ELEPHANT. AFTER THEY HAD SEEN THE MONKEYS , THEY WENT TO A RESTAURANT. THEY ASKED FOR STEAKS. AFTER THEY HAD EATEN THEM , THEY TOLD THE WAITER THAT THEY HAD NO MONEY. THE WAITER WAS FURIOUS. THEY RAN AWAY.

English summary is:

MARY AND MIKE WENT TO A ZOO. THEY WENT TO A RESTAURANT BUT THEY TOLD A WAITER THAT THEY DIDN'T HAVE ANY MONEY AND A WAITER WAS LIVID.

Example 6 differs from example 5 only slightly:

6. Input text:

MARY AND JOHN WENT TO THE ZOO. THEY FED THE MONKEYS SOME PEANUTS. AFTER THEY SAW THE ZEBRA , THEY WENT TO A RESTAURANT. JOHN TOLD THE WAITER THAT HE WANTED A STEAK. BECAUSE THEY DIDN'T HAVE MONEY THE WAITER WAS ANGRY. THEY RAN AWAY.

English summary is:

JOHN AND MARY WENT TO A ZOO. THEY WENT TO A RESTAURANT BUT A WAITER WAS FURIOUS BECAUSE THEY DIDN'T HAVE SOME MONEY.

Now a further variation. This time the restaurant is in the zoo, and the system succeeds in recognising that all the action takes place within the zoo, although this does not affect the summary produced, because the the summariser has no means of using this part of the text representation.

7. Input text:

MIKE AND MARY WENT TO THE ZOO. THEY SAW THE ELEPHANTS AND FED THE MONKEYS PEANUTS. AFTER THEY HAD LOOKED AT THE LIONS , THEY WENT TO THE ZOO'S RESTAURANT. THEY COULD SEE THE ZEBRAS AND GIRAFFES FROM THEIR TABLE. AFTER THEY HAD EATEN THEIR MEAL THEY REALISED THEY DIDN'T HAVE ANY MONEY. THEY HAD TO WASH DISHES BEFORE THEY COULD LEAVE.

English summary is:

MARY AND MIKE WENT TO A ZOO. THEY WENT TO A RESTAURANT BUT THEY REALISED THAT THEY DIDN'T HAVE ANY MONEY AND THEY HAD TO WASH SOME DISHES.

Example 8 is a more complex story in which there is only an indirect indication that the zoo was not left when the restaurant was entered.

8. Input text:

MIKE AND MARY WENT TO THE ZOO. THEY SAW THE ELEPHANTS AND FED THE MONKEYS PEANUTS. AFTER THEY HAD LOOKED AT THE LIONS , THEY WENT TO THE RESTAURANT. THEY COULD SEE THE ZEBRAS AND GIRAFFES FROM THEIR TABLE. AFTER THEY HAD EATEN THEIR MEAL THEY REALISED THEY DIDN'T HAVE ANY MONEY . THEY HAD TO WASH DISHES BEFORE THEY COULD LEAVE.

English summary is:

MARY AND MIKE WENT TO A ZOO. THEY WENT TO A RESTAURANT BUT THEY REALISED THAT THEY DIDN'T HAVE ANY MONEY AND THEY HAD TO WASH SOME DISHES.

The next text requires the use of a different stereotype, that for train journeys. In fact almost none of the text is predictable, or rather, the basic stereotype for train journeys covers very little of the story. The summary, although quite reasonable in content, is rather longwinded. This longwindedness comes from two interacting sources. First, the AD-HAC generator selects a rather uneconomic way to express the material. Second, because the generator uses more clauses than is strictly necessary, the summariser's mechanisms for controlling the length of sentences produced fails, because they are based on crude heuristics intended to identify which CD-structures the generator will convert into complete clauses.

9. Input text:

MIKE AND LESLEY WENT FROM CAMBRIDGE TO LONDON BY TRAIN. ON THE WAY TO LONDON THE TRAIN STOPPED AND THE GUARD SAID THE SIGNALS HAD FAILED. THE FAMILY HAD BOOKS WHICH THEY READ UNTIL THE TRAIN STARTED.

English summary is:

A TRAIN CONVEYED LESLEY AND MIKE FROM CAMBRIDGE TO LONDON BUT A GUARD STATED THAT SOME SIGNALS BECAME BROKEN AND A FAMILY HAD THE BOOKS,WHICH THEY WERE READING.

The next four stories are concerned either with shopping at the supermarket or with making a meal at home, two new stereotypes.

The processing of all these stories primarily illustrates the power of Scrabble's stereotype identification algorithms when operating with little information, rather than Scrabble's power as a summariser.

10. Input text:

JACK PICKED A CAN OF TUNA OFF THE SHELF. HE PAID FOR IT AND WENT HOME.

English summary is:

JACK WENT TO A SUPERMARKET.

The summaries produced for stories about making meals are extremely longwinded because the English generator lacks the appropriate sense of "to make" and is, therefore, reduced to producing a literal, step by step, translation of the summary CD structure into English. A more comprehensive generator is, of course, desirable, but noted in the body of the thesis, improving the AD-HAC generator was not deemed an essential part of the research reported here. Example 11 text also illustrates another problem with the current summariser. For a single topic story the summariser essentially introduces the topic first and only then produces unexpected material. In this case following the original textual order more closely would give better results. In addition, the crudity of the algorithm for selecting a conjunction to join the genuinely reduced material to the unexpected material is exposed here.

11. Input text:

JOHN WAS HUNGRY SO HE SENT MARY TO THE KITCHEN. SHE GOT SOME STEAK FROM THE FRIDGE AND MADE A PIE. JOHN LOVED IT.

English summary is:

MARY,WHO WAS DOING SOMETHING CAUSING A PIE TO EXIST,WAS IN A KITCHEN BUT JOHN WAS HUNGRY AND HE LIKED IT.

The next two texts and summaries are taken from [Charniak78b], where they are used as examples of texts representing extremes of difficulty in identifying the appropriate frame (to use Charniak's word), or at least, to distinguish which text is associated with supermarket shopping and which is associated with making a meal in the kitchen.

The summaries, therefore, should therefore be taken as evidence that Scrabble has succeeded, in spite of the extremely limited information available, in identifying the most probable of its stereotypes for the text rather than as illustrating Scrabble's powers as a summariser.

12. Input text:

JACK PICKED A CAN OF TUNA OFF THE SHELF. HE SWITCHED ON THE LIGHT.

English summary is:

JACK,WHO WAS DOING SOMETHING CAUSING SOME MEAL TO EXIST,WAS IN A KITCHEN.

13. Input text:

JACK PICKED A CAN OF TUNA OFF THE SHELF. HE PUT IT IN HIS BASKET.

English summary is:

JACK WENT TO A SUPERMARKET.

The final four stories in this appendix use a sixth stereotype-prototype, for buying presents.

The first present buying text is entirely straightforward.

14. Input text:

MARY WANTED TO GIVE SUSAN A PRESENT. SHE THOUGHT SUSAN WOULD LIKE TO HAVE A COMPUTER SO SHE WENT TO THE SHOP WHICH SOLD THEM. THE COMPUTERS LOOKED EXCITING. MARY WALKED OVER TO ONE AND TRIED IT BY WRITING A LITTLE PROGRAM. SHE ENJOYED IT AND DECIDED IT WOULD BE A GOOD PRESENT FOR SUSAN. SHE PAID FOR IT AND TOOK IT AWAY WITH HER.

English summary is:

MARY BOUGHT A COMPUTER FOR SUSAN.

The next two stories require the use of different stereotypes at different times during story processing. They both contain a word, "monkey" in one case and "tiger" in the other which causes the suggestion of the zoo stereotype at an irrelevant place, while the text's topic-type is present-buying. In both cases, therefore, the system must explicitly recognise that the zoo stereotype is irrelevant.

Note in both cases the possible implication in the summary that the animal was bought from the zoo. This false implication can be only be identified by deep inference on the summary representation. In particular, this problem can only be dealt with if the system possesses some means of recognising, and dealing with, actual or potential ambiguity in it summary. Thus this summary illustrates a limitation of the techniques used in Scrabble.

15. Input text:

JOHN WENT TO THE ZOO. HE FED THE MONKEYS PEANUTS. HE WANTED TO GIVE MARIEL A PRESENT AND HE THOUGHT SHE WOULD LIKE A MONKEY. HE WENT TO A SHOP WHICH SOLD THEM AND GOT ONE. SHE HATED IT.

English summary is:

JOHN WENT TO A ZOO. HE BOUGHT A MONKEY FOR MARIEL BUT SHE HATED IT.

The present-buying stereotype does not have any predictions about the actual reactions of the recipients of presents, although it does expect that the person buying the present will buy a present they think the recipient will like. "She loved it" in the next story is thus not expected and so is included in the summary. This slight misclassification (of "she loved it" as unexpected) is probably more the source of the slightly unnatural summary text, rather than the inappropriate use of "but" responsible for inadequacies in other summaries.

16. Input text:

JOHN WENT TO THE ZOO. HE SAW THE LIONS AND TIGERS. HE WANTED TO GIVE MARIEL A PRESENT AND HE THOUGHT SHE WOULD LIKE A TIGER. HE WENT TO A SHOP WHICH SOLD THEM AND GOT ONE. SHE LOVED IT UNTIL IT ATE HER.

English summary is:

JOHN WENT TO A ZOO. HE BOUGHT A TIGER FOR MARIEL BUT SHE LIKED IT UNTIL IT ATE HER.

Finally a story similar to the previous two, this time illustrating a rather naive paraphrase of "he gave her his present".

17. Input text:

SPUD WENT TO A SHOP. HE DECIDED A ZEBRA WOULD BE A GOOD PRESENT FOR MURIEL. HE BOUGHT ONE. SHE LOVED HIM AFTER HE GAVE HER HIS PRESENT.

English summary is:

SPUD BOUGHT A ZEBRA FOR MURIEL BUT SHE LIKED HIM AFTER HE GAVE HIS OWN PRESENT TO HER.

Appendix B

This appendix contains the stereotype-prototype which encodes the systems knowledge of visits to zoos. It is included here as supporting material for section 5.1. In particular, it is included so the reader can see the information in a stereotype-prototype in a rather more concise form than the presentation of section 5.1, which of neccessity included substantial annotations in addition to the material from the stereotype-prototype.

```
(ZOO
    (VARIABLES
        (goer (SCRIPT-FEATURE HUMAN) CAN-BE-GROUP DUMMY-HUMAN)
        (zoo  (MUST-BE-A ZOO) UNIQUE ZOO)
        (animalsfood (SCRIPT-FUNC-OF FOOD) REPEATABLE FOOD)
        (animal (SCRIPT-FEATURE ANIMAL) REPEATABLE ANIMAL)
        (gotozootime
            ((BEFORE looktime)
                (BEFORE feedtime)
                (BEFORE leavezootime))
            TIME)
        (looktime ((AFTER gotozootime)) TIME)
        (feedtime ((AFTER gotozootime)) TIME)
        (leavezootime
            ((AFTER looktime)
             (AFTER feedtime)
             (AFTER gotozootime))
            TIME))
    (WITH-SLOTS
        (gotozoo
            (PATTERN
                ((EVENT
                    (ACT PTRANS)
                    (OBJECT goer)
                    (TO zoo)
                    (TIME gotozootime)))))
        (lookatanimals
            (PATTERN
                (ALTERNATE-PATTERNS
                    ((EVENT
                        (ACTOR goer)
                        (ACT MTRANS)
                        (TOCP goer)
                        (FROM *EYE*)
                        (INST
                            (EVENT
                                (ACTOR goer)
                                (ACT ATTEND)
                                (OBJECT *EYE*)
                                (TO animal)
                                (TIME looktime)))
                        (TIME looktime)))
                    ((EVENT
                        (ACTOR goer)
                        (ACT ATTEND)
                        (TO animal)
                        (OBJECT *EYE*)
                        (TIME looktime)))))
            (DEMONS
                (REPEAT goer)))
```

```
(feedanimals
    (PATTERN
        ((CAUSE
            (ANTECEDENT
                (EVENT
                    (ACTOR goer)
                    (ACT DO)
                    (TIME feedtime)))
            (RESULT
                (EVENT
                    (ACTOR animal)
                    (ACT INGEST)
                    (OBJECT animalsfood)
                    (TIME feedtime)) )
            (TIME feedtime)))))
 (leavezoo
     (PATTERN
         (ALTERNATE-PATTERNS
             (EXCEPT
                 ((EVENT
                     (ACT PTRANS)
                     (OBJECT goer)
                     (TO
                         (FOCUS (STATE THING))
                         (STATE
                             (STATENAME POSS)
                             (VAL zoo)) ))))
             (EXCEPT
                 ((EVENT
                     (ACT PTRANS)
                     (OBJECT goer)
                     (TO
                         (FOCUS (STATE THING))
                         (STATE
                             (STATENAME
                                 (STATE LOC)
                                 (SPATEL *INSIDE*)
                                 (VAL zoo))) ))))
             ((EVENT
                 (ACT PTRANS)
                 (OBJECT goer)
                 (FROM zoo)
                 (TIME leavezootime)))))) )
(SUMMARY
    (TIME-DURATION
        gotozootime
        leavezootime)
    (MAINCHAR goer)
    ((EVENT
        (ACTOR goer)
        (ACT PTRANS)
        (OBJECT goer)
        (TO zoo)
        (TIME . gotozootime)))))
```

Bibliography

[Alshawi82]
Alshawi, H.
A Clustering Technique Semantic Network Processing.
European Conference on Artificial Intelligence,
Orsay, France. 1982.

[Bobrow77a]
Bobrow, D. G. et al
GUS, A Frame-Driven Dialog System.
Artificial Intelligence Vol 8 pp 155-173. 1977.

[Bobrow77b]
Bobrow, D. G., Winograd, T. and the KRL research group
Experience with KRL-0: one cycle of a knowledge representation language.
5th International Joint Conference on Artificial Intelligence,
Cambridge, Massachusetts. 1977.

[Boguraev79]
Boguraev, B. K.
Automatic Resolution of Linguistic Ambiguities.
Technical Report No. 11,
University of Cambridge Computer Laboratory,
Cambridge. 1979.

[Brady77]
Brady, J.M.
The Theory of Computation.
Chapman and Hall, London. 1977.

[Byrd80]
Byrd, L., and Borning, A.
Extending MECHO to Solve Statics Problems.
AISB-80 Conference Proceedings,
Society for the Study of Artificial Intelligence and the Simulation of Behaviour. July 1980.

[Cater80]
Cater, A.W.S.
Analysing English Text: A Non-deterministic Approach with Limited Memory.
AISB-80 Conference Proceedings,
Society for the Study of Artificial Intelligence and the Simulation of Behaviour. July 1980.

[Cater82]
Cater, A.W.S.
Analysis and Inference for English.
Technical Report No. 19,
University of Cambridge Computer Laboratory,
Cambridge. 1982.

[Charniak72]
Charniak, E.
Toward a Model of Childrens Story Comprehension.
Technical Report AI-TR-266,
Massachusetts Institute of Technology
Artificial Intelligence Laboratory,
Cambridge, Massachusetts. December 1972.

[Charniak75]
Charniak, E.
Organisation and Inference in a Frame-like System of Common Sense Knowledge.
Theoretical Issues in Natural Language Processing,
Cambridge, Massachusetts. 1975.

[Charniak76a]
Charniak E.
Inference and Knowledge 1.
in Computational Semantics,
E. Charniak and Y. Wilks (eds.)
North Holland, Amsterdam. 1976.

[Charniak76b]
Charniak, E.
Inference and Knowledge 2.
in Computational Semantics,
E. Charniak and Y. Wilks (eds.)
North Holland, Amsterdam. 1976

[Charniak77]
Charniak, E.
A Framed PAINTING: the Representation of a Common Sense Knowledge Fragment.
Cognitive Science, vol. 1 pp. 355-394. 1977.

[Charniak78a]
Charniak, E.
On the Use of Framed Knowledge in Language Comprehension.
Artificial Intelligence, vol 11 pp 225-265. 1978.

[Charniak78b]
Charniak, E.
With Spoon in Hand This Must be the Eating Frame.
Theoretical Issues in Natural Language Processing-2,
Association for Computing Machinery and The
Association for Computational Linguistics. 1978.

[Charniak79]
Charniak, E.
Ms. Malaprop, a Language Comprehension Program.
in Frame Conceptions and Text Understanding,
D. Melzing (ed.)
De Gruyter, Berlin. 1979.

[Charniak80]
Charniak, E., Riesbeck, C., and McDermott, D.
Artificial Intelligence Programming.
Lawrence Erlbaum Associates, Hillsdale, New Jersey. 1980.

[Charniak81]
Charniak E.
Context Recognition in Language Comprehension.
To appear in:
W. Lehnert and M. Ringle
Knowledge Representation for Language Processing Systems.
Lawrence Erlbaum Associates,
Hillsdale, New Jersey. Forthcoming.

[Cullingford78]
Cullingford, R. E.
Script Application:
Computer Understanding of Newspaper Stories.
Research Report #116.
Yale University Department of Computer Science,
New Haven, Connecticut. January 1978.

[DeJong79]
De Jong, G.F.
Skimming Stories in Real Time: an Experiment in Integrated Understanding.
Research Report #158.
Yale University Department of Computer Science,
New Haven, Connecticut. May 1979.

[Dyer82]
Dyer, M. G.
In-depth Understanding.
Research Report #219.
Yale University Department of Computer Science,
New Haven, Connecticut. May 1982.

[Fahlman79]
Fahlman, S. E.
NETL: A System for Representing and
Using Real-World Knowledge.
The MIT Press, Cambridge, Massachusetts. 1979.

[Granger77]
Granger, R.H.
FOUL-UP: A Program that Figures out Meaning
of Words from Context.
5th International Joint Conference
on Artificial Intelligence,
Cambridge, Massachusetts. 1977.

[Hayes79]
Hayes, Patrick J.
The Logic of Frames.
in Frame Conceptions and Text Understanding,
D. Meltzing (ed.)
De Gruyter, Berlin. 1979.

[Lebowitz80]
Lebowitz, M.
Generalization and Memory
in an Integrated Understanding System.
Report #186.
Yale University Department of Computer Science,
New Haven, Connecticut. October 1980.

[Lehnert78]
Lehnert, W. G.
The Process of Question Answering.
Lawrence Erlbaum Associates, Hillsdale, New Jersey. 1978.

[Lehnert79]
Lehnert, W. G.
The Role of Scripts in Understanding.
in Frame Conceptions and Text Understanding,
D. Meltzing (ed.)
De Gruyter, Berlin. 1979.

[Lehnert81]
Lehnert, W. G.
Plot Units and Narrative Summarisation.
Cognitive Science, Vol 5 pp 293-331. 1981.

[Lyons77]
Lyons, J.
Semantics, Volume 2.
Cambridge University Press, Cambridge. 1977.

[McEwan79]
McEwan, I.
In Between the Sheets.
Picador, London. 1979.

[Mellish80]
Mellish, C. S.
Some Problems in Early Noun Phrase Interpretation.
AISB-80 Conference Proceedings,
Society for the Study of Artificial Intelligence and
the Simulation of Behaviour. July 1980.

[Minsky74]
Minsky, M.
A Framework for Representing Knowledge.
Artificial Intelligence Memo No. 306,
Massachusetts Institute of Technology,
Cambridge, Massachusetts. June 1974.

[Norman75]
Norman, D.A. and Bobrow, D.G.
On Data-limited and Resource-limited Processes.
Cognitive Psychology, Vol. 7, pp. 44-64. 1975.

[Rieger75]
Rieger, C. J.
Conceptual Memory and Inference.
in [Schank75b].

[Riesbeck75]
Riesbeck, C. K.
Conceptual Analysis.
in [Schank75b].

[Rumelhart75]
Rumelhart, D. E.
Notes on a Schema for Stories.
in Representation and Understanding
D.G. Bobrow and A. Collins (eds.)
Academic Press, London. 1975.

[Small80]
Small, S.
Word Expert Parsing: A Theory of Distibuted
Word-based Natural Lnaguage Understanding.
TR-954, Computer Science Technical Report
Series, University of Maryland, College Park,
Maryland. September 1980.

[Schank75a]
Schank, R. C. and Abelson, R. P.
Script, Plans and Knowledge.
Advanced Papers of the Fourth International Conference
on Artificial Intelligence.
Tblisi, Georgia, USSR. 1975.

[Schank75b]
Schank, R. C.
Conceptual Information Processing.
North Holland, Amsterdam. 1975.

[Schank75c]
Schank R. C. and the Yale A.I. Project
SAM -- A Story Understander.
Research Report #43.
Yale University Department of Computer Science,
New Haven, Connecticut. 1975.

[Schank77]
Schank, R. C. and Abelson, R. P.
Scripts Plans Goals and Understanding.
Lawrence Erlbaum Associates, Hillsdale, New Jersey. 1977

[Schank78]
Schank, R. C.
Inference in the Conceptual Dependency Paradigm:
a Personal History.
Research Report #141.
Yale University Department of Computer Science,
New Haven, Connecticut. September 1978.

[Schank79]
Schank, R. C.
Interestingness: Controlling Inferences.
Artificial Intelligence, Vol 12 pp 273-279. 1979.

[Sidner79]
Sidner, C. L.
Towards a Computation Theory of Definitie
Anaphora Comprehension in English Discourse.
Artificial Intelligence Laboratory Technical
Report 537.
Massachusetts Institute of Technology,
Cambridge, Massachusetts. May 1979.

[vanDijk77]
van Dijk, T. A.
Complex Semantic Information Processing.
in Natural Language in Information Science
D.E. Walker, H. Karlsgren and M. Kay (eds.)
Skirptor, Stockholm. 1977.

[Wilensky78]
Wilensky, R.
Understanding Goal-based Stories.
Research Report #140.
Yale University Department of Computer Science,
New Haven, Connecticut. 1978.

[Wilks73]
Wilks, Y. A.
An Artificial Intelligence Approach to Machine Translation.
in Computer Models of Thought and Language
Colby, K. and Schank R. C. (eds.)
Freeman, San Francisco. 1973.

[Wilks75a]
Wilks, Y.A.
A Preferential Pattern Seeking Semantics for Natural
Language Inference.
Artificial Intelligence Vol 6 pp 53-74. 1975.

[Wilks75b]
Wilks, Y.A.
An Intelligent Analyser and Understander of English.
Communications of the ACM, Vol 18 pp 264-274, 1975.

[Wilks76]
Wilks, Y. A.
Frames, Scripts, Stories and Fantasies.
International Conference on Psychology of Language,
University of Stirling. June 1976.

[Wilks77]
Wilks, Y. A.
Good and Bad Arguments about Semantic Primitives.
Communication and Cognition Vol 10 pp 181-221. 1977.

[Winograd72]
Winograd, T.
Understanding Natural Language.
Edinburgh University Press, Edinburgh. 1972.